

INFORME MANTENIMIENTO Y DESARROLLO ESPECIALIZADO SISTEMA DE INFORMACIÓN DE GESTIÓN DE PROYECTOS – SIGP

SEXTO PAGO CONTRATO 237-2025

TABLA DE CONTENIDO

INFORME MANTENIMIENTO Y DESARROLLO ESPECIALIZADO SISTEMA DE INFORMACIÓN DE GESTIÓN DE PROYECTOS - SIGP 1

10. Incluir en la interfaz de usuario una etiqueta que muestre el nombre del archivo y su tamaño (en bytes, KB o MB). El objetivo de esta etiqueta es brindar información clara al usuario sobre los archivos cargados en la plataforma SIGP. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6) 1

Aplicativo de Gestión 3

Componentes 3

Codificación 4

Formulario de Beneficios Tributarios 5

Componentes 5

Codificación 6

11. Registrar la fecha y hora de carga del documento en el sistema. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6) 7

Aplicativo de Beneficios Tributarios 9

Componentes 9

Codificación 10

Formulario de Proyectos y Programas 11

Componentes 11

Codificación 12

12. Fijar el mensaje de notificación de validación y envío en la interfaz de usuario de forma permanente. Eliminar la opción de cerrar, descartar o eliminar dicho mensaje, asegurando que

permanezca visible para el usuario en todo momento. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6) 13

Formulario de Proyectos y Programas	14
Componentes	14
Codificación	15
Formulario de Beneficios Tributarios	15
Componentes	15
Codificación	16
Formulario de Reconocimiento de Actores	16
Componentes	16
Codificación	16
13. Registrar la fecha y hora en el campo de “Progreso”. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)	17
Aplicativo de Gestión	19
Componentes	19
Codificación	19
14. Validación de campos al generar la evaluación en la sección de Beneficios Tributarios. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)	20
Formulario de Beneficios Tributarios	22
Componentes	22
Codificación	23
15. Límite de palabras: Implementar una funcionalidad que permita configurar un límite mínimo y máximo de palabras para las secciones de criterios y subcriterios en el formulario de evaluación de beneficios tributarios. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6).....	28
Formatos de Evaluación	31
Componentes	31
Codificación	32
Evaluación de Criterios Beneficios Tributarios.....	35
Componentes	35

Codificación	36
16. Para la configuración de convocatoria crear un campo que restrinja la contrapartida con respecto al solicitado en la financiación, en la actualizada está la restricción a la sumatoria del total del proyecto contrapartida + financiación. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6) ..	40
17. Dar soporte, capacitación y actualizar la documentación de usuario en cuanto a la configuración de Gran Áreas por Convocatoria y Disciplina OCDE proyectos y beneficios tributarios. Registrar Código BPIN y Número de Contrato en proyectos de BT, configuración de Cantidad Máxima de Evaluaciones de Proyectos Asignadas que un Evaluador puede tener por Convocatoria, configuración de acceso a funcionarios a la Evaluación de Proyectos por Convocatoria, reclamaciones Panel Evaluación Proyectos, configuración de Matriz evaluación BT tamaño Empresa y a Resoluciones de Ingresos No Constitutivos de Renta. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)	45
Ajustes Resoluciones Beneficios Tributarios	47
Crédito Fiscal por Inversiones en CTel	47
Deducción y Descuento por Inversiones en CTel	84
Ajustar Validación visualización de evaluación de proyectos (fecha de cierre).....	123
Implementar que por convocatoria se habilite si los funcionarios internos pueden modificar la calificación evaluación, siempre y cuando estos pertenezcan a un grupo específico	126
Habilitar funcionarios por Convocatoria	127
Validaciones para habilitar modificación de evaluación a funcionarios internos.....	130
Parametrizar una cantidad máxima y mínima de proyectos que pueden presentarse por programa	136
Aplicativo de Gestión	136
Formulario de Proyectos y Programas	141
Implementar Reporte Certificados Evaluadores BT	144
Aplicativo de Gestión	145
Configurar Gran Áreas Por Convocatoria y Disciplina OCDE proyectos y beneficios tributarios	205
Aplicativo de Gestión	205
Formulario de Proyectos y Programas	243
Formulario de Beneficios Tributarios	261

Configurar Cantidad Máxima de Evaluaciones Asignadas por Evaluador en una Convocatoria.	270
Configurar cantidad máxima de evaluadores por proyecto	271
Validaciones al asignar evaluaciones en los proyectos	273
Configurar Acceso Funcionarios Evaluación Proyectos por Convocatoria.....	276
Creación pantalla de Configuración de Funcionarios por Convocatoria.....	277
Ajuste Acceso Evaluación de Proyectos/Programas por Funcionarios Internos.....	345
Ajuste Acceso Evaluación de Beneficios Tributarios por Funcionarios Internos	352
Reclamaciones Panel Evaluación Proyectos.....	358
Configuración de Matrices de Evaluación de Beneficios Tributarios por Tamaño Empresa ..	374
Matrices de Evaluación por Convocatoria	375
Ajuste de Lógica para obtener y mostrar la matriz de evaluación calificación según el tamaño de la empresa	386
18. Soporte y capacitación para nuevos requerimientos en el uso de los módulos del SIGP al personal de OTSI de MinCiencias. Y revisión de problemas encontrados al usar la aplicación. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)	390
a. Anonimizar Reportes de Evaluación	390
Módulo de Proyectos y Programas	392
Módulo de Beneficios Tributarios	420
b. Desarrollo Nuevo Informe de Evaluación Tercer Evaluador	433
Codificación	435
c. Implementar Registro de Múltiples Líneas Temáticas y Múltiples Sublíneas Temáticas	457
Formulario de Líneas y Sublíneas Temáticas.....	458
d. Soporte de Asignación de Evaluadores de Proyectos en la Convocatoria 963-2025 Orquídeas.....	489
e. Implementar Separación Técnica de Funciones en el Módulo de Administración del SIGP	491
Gestión de Usuarios	494
Funcionarios Evaluación.....	496

f.	Inconsistencias en el reporte de proyectos – Convocatoria 968.....	528
g.	Reporte de Evaluación de la Convocatoria 963-2025 Orquídeas	531
h.	Apoyo Renovación Certificado SSL del SIGP	532
i.	Migración del Módulo HÉRCULES a Nuevo Servidor	540
	Configurar DADs WebTier	541
	Actualizar objetos y permisos	542
	Reiniciar servicios de WebTier	547
j.	Renderizar Campos Formulario Beneficios Tributarios	547
k.	Apoyo Parametrización Rubrica Conv 965-2025 Movilidades Académicas con Europa	552
	INFORME DE ACTIVIDADES DE SOPORTE EN EL USO DEL SIGP EN LA CREACION DE CONVOCATORIAS Y TRANSFERIR EL CONOCIMIENTO DEL SIGP A LOS FUNCIONARIOS DE MINCIENCIAS, (No. Obligación Contrato: 2.2.3, 2.2.4)	554
	Introducción	554
	Resumen de Actividades	554
	Detalle de Casos Atendidos.....	554
	Solución de Errores	554
	Conclusiones.....	557

INFORME MANTENIMIENTO Y DESARROLLO ESPECIALIZADO SISTEMA DE INFORMACIÓN DE GESTIÓN DE PROYECTOS - SIGP

- 10. Incluir en la interfaz de usuario una etiqueta que muestre el nombre del archivo y su tamaño (en bytes, KB o MB). El objetivo de esta etiqueta es brindar información clara al usuario sobre los archivos cargados en la plataforma SIGP. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)**

Solicitud de requerimiento No. 164117

Descripción y alcance del Requerimiento

En el módulo Requisitos del formulario de Beneficios Tributarios del Sistema de Gestión de Proyectos (SIGP), se implementará una mejora en la interfaz de usuario que permitirá mostrar, junto al nombre de cada archivo cargado, su tamaño en una unidad legible (bytes, KB o MB).

Esta información se presentará mediante una etiqueta visible inmediatamente después de que el usuario adjunte un archivo, permitiendo validar de forma rápida el peso del documento antes y después de ser cargado.

El ajuste será aplicado en dos entornos del sistema:

- Formulario en Línea de Beneficios Tributarios donde los proponentes cargan los documentos solicitados por convocatoria, visualizando en la lista de requisitos el nombre del archivo cargado y su tamaño.
- Aplicativo de Gestión (interfaz interna): donde los funcionarios de MinCiencias revisan y gestionan los requisitos radicados. En esta interfaz, junto al nombre de cada archivo cargado, se deberá incluir también la información del tamaño, para garantizar coherencia con la vista del usuario externo.

El alcance contempla la modificación de la lógica de carga de archivos, la actualización de la interfaz gráfica en ambas aplicaciones, y la validación para que los valores se muestren de forma dinámica y acorde con el formato más adecuado (bytes, KB o MB) según el tamaño del archivo.

Justificación

Actualmente, el sistema SIGP permite la carga de archivos para cumplir con los requisitos de una convocatoria, pero no muestra de manera visible ni el tamaño del archivo cargado ni su peso después de adjuntarlo. Esto genera incertidumbre en los usuarios externos, quienes no pueden verificar de inmediato si el documento cumple con los límites establecidos de peso, y en los funcionarios internos, quienes no tienen referencia del tamaño de los documentos que revisan.

Desarrollo e Implementación

Se realizaron ajustes en la aplicación SIGP WEB, específicamente en el módulo de requisitos de proyectos para el aplicativo de Gestión y Beneficios Tributarios. El desarrollo se orientó a mejorar la experiencia del usuario al cargar y visualizar documentos, incorporando en la interfaz una columna que muestra el tamaño del archivo adjunto a cada requisito.

El cambio se aplicó exclusivamente en la pantalla de requisitos para proyectos de beneficios tributarios, tanto en el aplicativo de Gestión como en el de Beneficios Tributarios, e incluyó los siguientes elementos:

1. Nueva columna "Tamaño" en la tabla de requisitos:

- Se agregó una columna que presenta el tamaño del archivo cargado al requisito, calculado dinámicamente en kilobytes (KB) a partir de los bytes almacenados en la propiedad archivo del objeto DocumentoDTO.

2. Método de cálculo en el controlador de sesión:

- Se implementó el método `getTamanoDocumentoEnKB(DocumentoDTO documento)` en el controlador correspondiente. Este método convierte el tamaño del arreglo de bytes (`byte[]`) a una unidad legible (KB) y lo expone para su uso en la interfaz.

3. Integración con JSF:

- La nueva columna fue incorporada en el componente `<p:dataTable>` de PrimeFaces, utilizando EL Expression para invocar el método mencionado y renderizar el valor en tiempo real.

4. Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- **Java EE (JSF + EJB):** Se utilizó JSF para la incorporación de la nueva columna en la tabla de requisitos, y EJB para mantener encapsulada la lógica de cálculo en el backend.
- **PrimeFaces:** Para renderizar dinámicamente la nueva columna utilizando expresiones EL con componentes `<p:column>` y `<h:outputText>`.
- **Java:** Implementación del método auxiliar en el controlador de sesión para obtener el tamaño del archivo en KB.

Aplicativo de Gestión

Componentes

Con el fin de brindar mayor claridad al usuario sobre los documentos cargados en los requisitos de una convocatoria, se integraron y adaptaron componentes dentro del aplicativo de Gestión. Esta mejora permite visualizar el tamaño del archivo adjunto directamente desde la interfaz, facilitando la verificación del contenido subido por el proponente. A continuación, se detallan los principales componentes utilizados:

- **Vista (View):** La interfaz de usuario está representada por el archivo `requisitos.xhtml`. Estas páginas XHTML define la estructura visual del formulario de requisitos, incluyendo los botones y secciones relacionadas con requisitos.
- **Controlador (Controllers de Conversación):** La lógica de presentación está gestionada por la clase `RequisitoProyectoConvController`. Este controlador maneja el ciclo de vida del formulario, controla la renderización de botones, realiza validaciones, y coordinan las acciones del usuario con los servicios del backend.
- **Servicio EJB (EJB Service):** Representado por la clase `ProyectosServiceBean`, que es el EJB (Enterprise JavaBeans) para usar los valores de la base de datos.

- **Base de Datos (Database):** Representada por la entidad DocumentoDTO, un data transfer object que hace referencia a la tabla SGP_DOCUMENTOS indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de Usuario (requisitos.xhtml)

Columna "Tamaño"

En la tabla principal de requisitos del módulo de Beneficios Tributarios, se incorporó una nueva columna denominada "Tamaño", que permite visualizar el tamaño del archivo adjunto en kilobytes (KB). Esta funcionalidad proporciona mayor transparencia al usuario sobre los documentos cargados y facilita la validación visual del contenido.

```
<p:column width="10%">
  <f:facet name="header">
    <h:outputText
value="#{msg['gestion.beneficiosTributarios.requisitos.tamano.label']}" />
  </f:facet>
  <h:outputText
value="#{requisitoProyectoConvController.calcularTamañoKb(item.documentoDTO.docSize)}"
rendered="#{item.documentoDTO ne null}"
styleClass="pull-right"/>
</p:column>
```

A continuación, se explican las propiedades más relevantes del componente:

- **width="10%":** Define el ancho de la columna dentro de la tabla.
- **<f:facet name="header">:** Configura el encabezado de la columna, utilizando una etiqueta internacionalizada (**msg[...]**).
- **value="#{requisitoProyectoConvController.calcularTamañoKb(item.documentoDTO.docSize)}":** Invoca el método encargado de calcular y retornar el tamaño del documento en KB, recibiendo como parámetro el tamaño en bytes desde la propiedad docSize del objeto DocumentoDTO.
- **rendered="#{item.documentoDTO ne null}":** Controla la visibilidad del valor solo cuando existe un documento asociado al requisito.

Controlador de Conversación (RequisitoProyectoConvController)

Método calcularTamañoKb

Este método se encuentra en el controlador y es invocado desde la interfaz gráfica para mostrar el tamaño de un archivo en kilobytes dentro de la columna "Tamaño" de la tabla de requisitos del módulo de Beneficios Tributarios.

```
public String calcularTamañoKb(Long sizeDoc) {  
    String str = "";  
    if (sizeDoc != null) {  
        Double tamañoKb = sizeDoc.doubleValue() / 1024;  
        str = String.format("%.2f", tamañoKb) + " KB";  
    }  
    return str;  
}
```

El método recibe como parámetro un valor de tipo Long correspondiente al tamaño del archivo en bytes (**sizeDoc**). Verifica si el valor no es nulo y, en caso afirmativo, lo convierte a kilobytes dividiéndolo entre 1024 mediante **doubleValue()**. Luego, formatea el resultado con dos decimales utilizando **String.format("%.2f", tamañoKb)** y le agrega la unidad "KB" para componer la cadena final. Si el parámetro es nulo, retorna una cadena vacía. Esta lógica es utilizada directamente en el atributo value de un componente **<h:outputText>** en la vista.

Formulario de Beneficios Tributarios

Componentes

Con el fin de brindar mayor claridad al usuario sobre los documentos cargados en los requisitos de una convocatoria, se integraron y adaptaron componentes dentro del aplicativo de Beneficios Tributarios. Esta mejora permite visualizar el tamaño del archivo adjunto directamente desde la interfaz, facilitando la verificación del contenido subido por el proponente. A continuación, se detallan los principales componentes utilizados:

- **Vista (View):** La interfaz de usuario está representada por el archivo requisitos.xhtml, una página XHTML que define la estructura visual del formulario de requisitos, incluyendo los botones y secciones relacionadas con la gestión de requisitos.
- **Controlador (Controller de Conversación):** La lógica de presentación está gestionada por la clase RequisitoConvController. Este controlador maneja el ciclo de vida del formulario,

controla la renderización de botones, realizan validaciones, y coordina las acciones del usuario con los servicios del backend.

- **Servicio EJB (EJB Service):** Representado por la clase `ProyectosServiceBean`, que es el EJB (Enterprise JavaBeans) para usar los valores de la base de datos.
- **Base de Datos (Database):** Representada por la entidad `DocumentoDTO`, un data transfer object que hace referencia a la tabla `SGP_DOCUMENTO` indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de Usuario (requisitos.xhtml)

Columna "Tamaño"

En la tabla principal de requisitos, se incorporó una nueva columna denominada "Tamaño", que permite visualizar el tamaño del archivo adjunto en kilobytes (KB). Esta funcionalidad proporciona mayor transparencia al usuario sobre los documentos cargados y facilita la validación visual del contenido.

```
<p:column width="10%">
  <f:facet name="header">
    <h:outputText
      value="#{msg['formulario.estimulos.proyecto.requisitos.tamano.label']}" />
    </f:facet>
    <h:outputText
      value="#{requisitoConvController.calcularTamañoKb(item.documentoDTO.docSize)}"
      rendered="#{item.documentoDTO ne null}"
      styleClass="pull-right"/>
  </h:outputText>
</p:column>
```

A continuación, se explican las propiedades más relevantes del componente:

- **width="10%":** Define el ancho de la columna dentro de la tabla.
- **<f:facet name="header">:** Configura el encabezado de la columna, utilizando una etiqueta internacionalizada (**msg[...]**).

- **value="#{requisitoConvController.calcularTamañoKb(item.documentoDTO.docSize)}":**
Invoca el método encargado de calcular y retornar el tamaño del documento en **KB**, recibiendo como parámetro el tamaño en bytes desde la propiedad **docSize** del objeto **DocumentoDTO**.
- **rendered="#{item.documentoDTO ne null}":** Controla la visibilidad del valor solo cuando existe un documento asociado al requisito.

Controlador de Conversación (RequisitoConvController)

Método calcularTamañoKb

Este método se encuentra en el controlador y es invocado desde la interfaz gráfica para mostrar el tamaño de un archivo en kilobytes dentro de la columna "Tamaño" de la tabla de requisitos.

```
public String calcularTamañoKb(Long sizeDoc) {  
    String str = "";  
    if (sizeDoc != null) {  
        Double tamañoKb = sizeDoc.doubleValue() / 1024;  
        str = String.format("%.2f", tamañoKb) + " KB";  
    }  
    return str;  
}
```

El método recibe como parámetro un valor de tipo **Long** correspondiente al tamaño del archivo en bytes (**sizeDoc**). Verifica si el valor no es nulo y, en caso afirmativo, lo convierte a kilobytes dividiéndolo entre 1024 mediante **doubleValue()**. Luego, formatea el resultado con dos decimales utilizando **String.format("%.2f", tamañoKb)** y le agrega la unidad "**KB**" para componer la cadena final. Si el parámetro es nulo, retorna una cadena vacía. Esta lógica es utilizada directamente en el atributo **value** de un componente **<h:outputText>** en la vista.

11. Registrar la fecha y hora de carga del documento en el sistema. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)

Solicitud de requerimiento No. 163312

Descripción y alcance del Requerimiento

Se requiere implementar en el formulario en línea de Beneficios Tributarios del SIGP, específicamente en la sección de Requisitos, la visualización de la fecha y hora en la que un documento fue cargado en el sistema.

La información mostrada deberá indicar según sea el caso:

- Fecha de creación: cuando el documento es cargado por primera vez.
- Fecha de actualización: cuando el documento existente es reemplazado por una nueva versión.

Este desarrollo aplica únicamente para el formulario en línea de Beneficios Tributarios y no afectará otros módulos o formularios del SIGP. La visualización deberá realizarse en la misma interfaz donde actualmente se listan los requisitos, junto a los demás datos del archivo.

Justificación

La inclusión de la fecha y hora de carga y/o actualización del documento permitirá:

- Brindar mayor trazabilidad y control sobre la gestión documental del proyecto.
- Facilitar a los usuarios la identificación de la versión más reciente del documento.
- Mejorar el seguimiento de los tiempos de entrega de requisitos establecidos por la convocatoria.
- Apoyar los procesos de auditoría y verificación al contar con un registro visible y confiable de las acciones realizadas sobre cada documento.

Desarrollo e Implementación

Se realizaron ajustes en la aplicación SIGP WEB, específicamente en el módulo de Requisitos de Proyectos para los aplicativos de Beneficios Tributarios y Proyectos y Programas. El objetivo del desarrollo fue mejorar la trazabilidad de los documentos cargados por los proponentes, incorporando en la interfaz una etiqueta que muestra la fecha y hora de carga del archivo asociado a cada requisito.

El cambio fue implementado únicamente en la pantalla de requisitos de los aplicativos mencionados e incluyó los siguientes elementos funcionales:

1. Visualización de la fecha y hora de carga del documento:

- Se incorporó un componente visual que muestra la fecha y hora exactas en las que fue cargado el archivo, utilizando la propiedad fechaRegistro de la entidad DocumentoDTO.
- Esta información se presenta mediante un h:outputText, ubicado junto al label de la columna “Progreso”.

2. Formateo de fecha en el controlador:

- Para mostrar la información con el formato adecuado, se utilizó el componente estándar f:convertDateTime de JSF, aunque en algunos casos también se aplicó lógica de formateo desde el backend.

3. Integración con JSF:

- La etiqueta fue incorporada en la vista utilizando h:outputText, obteniendo el valor directamente desde la propiedad fechaRegistro del objeto o a través de un método en el backing bean.

4. Tecnologías Utilizadas

El desarrollo se apoyó en las siguientes tecnologías:

- **Java EE (JSF y EJB):** JSF fue utilizado para el manejo de la vista y presentación de datos, mientras que EJB permitió encapsular la lógica de negocio relacionada con el manejo de documentos y requisitos.
- **PrimeFaces:** Se aprovechó la integración con JSF para ubicar la nueva etiqueta dentro de la tabla de requisitos o junto a los botones existentes, utilizando componentes visuales estándar.
- **Java:** Se empleó para el tratamiento de fechas y la posible conversión de formatos desde el backend, como complemento al formateo en la capa de presentación.

Aplicativo de Beneficios Tributarios

Componentes

Con el fin de brindar mayor claridad al usuario sobre los documentos cargados en los requisitos del proyecto de una convocatoria, se integraron y adaptaron componentes dentro del aplicativo de Beneficios Tributarios. Esta mejora permite visualizar la fecha y hora de carga del archivo

directamente desde la interfaz, facilitando la trazabilidad y validación del momento en que el proponente realizó el registro documental. A continuación, se detallan los principales componentes utilizados:

- **Vista (View):** La interfaz de usuario está representada por el archivo requisitos.xhtml. Esta páginas XHTML define la estructura visual del formulario de requisitos, incluyendo los botones y secciones relacionadas con la gestión y subsanación de documentos.
- **Servicio EJB (EJB Service):** Representado por la clase ProyectosServiceBean, que es el EJB (Enterprise JavaBeans) para usar los valores de la base de datos.
- **Base de Datos (Database):** Representada por la entidad DocumentoDTO, un data transfer object que hace referencia a la tabla SGP_DOCUMENTOS indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de Usuario (requisitos.xhtml)

Columna “Progreso”

En la tabla principal de requisitos del módulo de Beneficios Tributarios, se integró una etiqueta visual (label) que indica la fecha de subida del documento. Esta información se muestra dentro de la columna “Progreso”, justo debajo del estado del requisito, cuando el documento ha sido cargado por el usuario.

```
<ui:fragment rendered="#{item.documentoDTO.nombre != null}">
  <br/>
  <div class="badge badge-info" style="white-space: initial;font-size: 9px;">
    <b>#{msg['formulario.estimulos.proyecto.requisitos.progreso.fechaSubida']} </b>
    <h:outputText value="#{item.documentoDTO.fechaModificacion != null
      ? item.documentoDTO.fechaModificacion
      : item.documentoDTO.fechaCreacion}"
      >
    <f:convertDateTime pattern="dd/MM/yyyy HH:mm" timeZone="EST"/>
    </h:outputText>
  </div>
</ui:fragment>
```

A continuación, se explican las propiedades más relevantes del componente:

- **<ui:fragment rendered="#{item.documentoDTO.nombre != null}">**: Renderiza el contenido únicamente si existe un documento cargado. Evita mostrar la etiqueta si no hay archivo asociado al requisito.
- **<div class="badge badge-info">**: Se usa una clase visual para mostrar el texto dentro de un recuadro estilo badge (etiqueta). La clase badge-info le da un color azul claro, manteniendo coherencia visual con el resto de la interfaz.
- **white-space: initial; font-size: 9px; (CSS inline)**: Permite que el contenido se acomode en múltiples líneas si es necesario. El tamaño de fuente pequeño asegura que la información no sobrecargue visualmente la celda.
- **#{msg['formulario.estimulos.proyecto.requisitos.progreso.fechaSubida']}**: Permite internacionalizar el texto del label. El mensaje es cargado desde el archivo de propiedades del sistema.
- **<h:outputText>**: Muestra el valor de la fecha obtenida desde el documentoDTO. Se evalúa primero fechaModificacion, y si no existe, se usa fechaCreacion.
- **<f:convertDateTime pattern="dd/MM/yyyy HH:mm">**: Convierte el objeto Date al formato día/mes/año hora:minuto (ej. 30/07/2025 14:35). Permite que la fecha sea más comprensible para los usuarios.
- **timeZone="EST"**: Se define la zona horaria para asegurar que la fecha/hora se muestre correctamente según la localización del sistema.

Formulario de Proyectos y Programas

Componentes

Con el fin de brindar mayor claridad al usuario sobre los documentos cargados en los requisitos del proyecto de una convocatoria, se integraron y adaptaron componentes dentro del aplicativo de Proyectos y Programas. Esta mejora permite visualizar la fecha y hora de carga del archivo directamente desde la interfaz, facilitando la trazabilidad y validación del momento en que el proponente realizó el registro documental. A continuación, se detallan los principales componentes utilizados:

- **Vista (View)**: La interfaz de usuario está representada por el archivo requisitos.xhtml. Esta página XHTML define la estructura visual del formulario de requisitos, incluyendo los botones y secciones relacionadas con la gestión y subsanación de documentos.
- **Servicio EJB (EJB Service)**: Representado por la clase ProyectosServiceBean, que es el EJB (Enterprise JavaBeans) para usar los valores de la base de datos.

- **Base de Datos (Database):** Representada por la entidad DocumentoDTO, un data transfer object que hace referencia a la tabla SGP_DOCUMENTOS indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de Usuario (requisitos.xhtml)

Columna "Progreso"

En la tabla principal de requisitos del módulo de Proyectos y Programas, se integró una etiqueta visual (label) que indica la fecha de subida del documento. Esta información se muestra dentro de la columna "Progreso", justo debajo del estado del requisito, cuando el documento ha sido cargado por el usuario.

```
<ui:fragment rendered="#{item.documentoDTO.nombre != null}">
  <br/>
  <div class="badge badge-info" style="white-space: initial;font-size: 9px;">
    <b>#{msg['formulario.proyectos.requisitos.progreso.fechaSubida']}</b>
    <h:outputText value="#{item.documentoDTO.fechaModificacion != null
      ? item.documentoDTO.fechaModificacion
      : item.documentoDTO.fechaCreacion}"
      >
    <f:convertDateTime pattern="dd/MM/yyyy HH:mm" timeZone="EST"/>
    </h:outputText>
  </div>
</ui:fragment>
```

A continuación, se explican las propiedades más relevantes del componente:

- **<ui:fragment rendered="#{item.documentoDTO.nombre != null}">:** Renderiza el contenido únicamente si existe un documento cargado. Evita mostrar la etiqueta si no hay archivo asociado al requisito.
- **<div class="badge badge-info">:** Se usa una clase visual para mostrar el texto dentro de un recuadro estilo badge (etiqueta). La clase badge-info le da un color azul claro, manteniendo coherencia visual con el resto de la interfaz.
- **white-space: initial; font-size: 9px; (CSS inline):** Permite que el contenido se acomode en múltiples líneas si es necesario. El tamaño de fuente pequeño asegura que la información no sobrecargue visualmente la celda.

- **#{msg['formulario.proyectos.requisitos.progreso.fechaSubida']}: Permite internacionalizar el texto del label. El mensaje es cargado desde el archivo de propiedades del sistema.**
- **<h:outputText>: Muestra el valor de la fecha obtenida desde el documentoDTO. Se evalúa primero fechaModificacion, y si no existe, se usa fechaCreacion.**
- **<f:convertDateTime pattern="dd/MM/yyyy HH:mm">: Convierte el objeto Date al formato día/mes/año hora:minuto (ej. 30/07/2025 14:35). Permite que la fecha sea más comprensible para los usuarios.**
- **timeZone="EST": Se define la zona horaria para asegurar que la fecha/hora se muestre correctamente según la localización del sistema.**

12. Fijar el mensaje de notificación de validación y envío en la interfaz de usuario de forma permanente. Eliminar la opción de cerrar, descartar o eliminar dicho mensaje, asegurando que permanezca visible para el usuario en todo momento. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)

Solicitud de requerimiento No. 163308

Descripción y alcance del Requerimiento

Implementar un ajuste en la interfaz de usuario del sistema para los formularios de Proyectos/Programas y Beneficios Tributarios, consistente en fijar de forma permanente el mensaje de notificación que se muestra en el proceso de validación y envío del proyecto. Este mensaje no deberá contar con opciones para cerrarlo, descartarlo o eliminarlo, asegurando que permanezca visible para el usuario durante toda la sesión en la que esté interactuando con la pantalla.

El desarrollo aplicará tanto al formulario en línea de Proyectos/Programas como al formulario en línea de Beneficios Tributarios, en la sección de validación previa a la radicación del proyecto.

Justificación

Actualmente, las notificaciones y alertas que se presentan al usuario en la etapa de validación y envío pueden ser cerradas u ocultadas manualmente, lo que genera el riesgo de que el proponente no cuente con información clave antes de radicar el proyecto. Mantener este mensaje visible de

forma fija asegura que los usuarios siempre dispongan de las instrucciones y advertencias necesarias, reduciendo errores de radicación y garantizando el cumplimiento de los requisitos establecidos para la presentación del proyecto.

Desarrollo e Implementación

Se realizaron ajustes en la aplicación SIGP WEB, específicamente en las pantallas de validación de los formularios de Beneficios Tributarios, Proyectos y Programas, y Reconocimiento de Actores. El desarrollo se centró en mantener visibles de forma permanente los mensajes de validación mostrados al usuario, eliminando cualquier opción de cerrarlos manualmente.

- Se eliminó manualmente la etiqueta <button> de cierre (<button class="close">x</button>) dentro de los div con clase alert, en las pantallas de validación de los formularios mencionados.
- Con este cambio, los mensajes informativos permanecen visibles durante todo el proceso de validación y envío, sin posibilidad de ser descartados por el usuario.

Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- **JSF (JavaServer Faces):** Para el manejo de las vistas y plantillas donde se muestran los mensajes.
- **PrimeFaces:** Para la integración con componentes de interfaz que mostraban alertas o mensajes al usuario.

Formulario de Proyectos y Programas

Componentes

Para fijar los mensajes de alerta en la pantalla de validación del formulario de Proyectos y Programas, se emplearon los siguientes elementos:

- **Vista (View):** Archivo de interfaz que define la estructura visual de la pantalla de validación del proyecto. En esta vista se ubican los mensajes de alerta que deben permanecer fijos para el usuario.

- **PrimeFaces:** Framework utilizado para construir los componentes gráficos de la interfaz. En este caso, permite renderizar los mensajes y formularios de forma dinámica, asegurando una experiencia de usuario fluida y estructurada.

Codificación

Interfaz de Usuario (validación.xhtml)

Modificación de Alertas Permanentes

Para asegurar que los mensajes de alerta permanezcan visibles de forma permanente en la pantalla de validación, se eliminó manualmente la etiqueta **<button>** utilizada para cerrar dichos mensajes. En particular, se removió la siguiente línea de código de cada div con clase alert:

```
<button aria-hidden="true" data-dismiss="alert" class="close" type="button">×</button>
```

La etiqueta **<button>** anterior activa la funcionalidad de cierre del mensaje al hacer clic sobre la "X". Esta acción está ligada al atributo **data-dismiss="alert"**, propio de Bootstrap o integraciones similares, que permite ocultar el div correspondiente. Al eliminar esta etiqueta, se evita que el usuario pueda cerrar el mensaje, fijando así su visibilidad de manera permanente en la interfaz y asegurando que la notificación cumpla su propósito informativo hasta que el usuario realice alguna acción en el sistema.

Formulario de Beneficios Tributarios

Componentes

Para fijar los mensajes de alerta en la pantalla de validación del formulario de Beneficios Tributarios, se emplearon los siguientes elementos:

- **Vista (View):** Archivo de interfaz que define la estructura visual de la pantalla de validación del proyecto. En esta vista se ubican los mensajes de alerta que deben permanecer fijos para el usuario.
- **PrimeFaces:** Framework utilizado para construir los componentes gráficos de la interfaz. En este caso, permite renderizar los mensajes y formularios de forma dinámica, asegurando una experiencia de usuario fluida y estructurada.

Codificación

Interfaz de Usuario (validación.xhtml)

Modificación de Alertas Permanentes

Para asegurar que los mensajes de alerta permanezcan visibles de forma permanente en la pantalla de validación, se eliminó manualmente la etiqueta **<button>** utilizada para cerrar dichos mensajes. En particular, se removió la siguiente línea de código de cada div con clase alert:

```
<button aria-hidden="true" data-dismiss="alert" class="close" type="button">×</button>
```

La etiqueta **<button>** anterior activa la funcionalidad de cierre del mensaje al hacer clic sobre la "X". Esta acción está ligada al atributo **data-dismiss="alert"**, propio de Bootstrap o integraciones similares, que permite ocultar el div correspondiente. Al eliminar esta etiqueta, se evita que el usuario pueda cerrar el mensaje, fijando así su visibilidad de manera permanente en la interfaz y asegurando que la notificación cumpla su propósito informativo hasta que el usuario realice alguna acción en el sistema.

Formulario de Reconocimiento de Actores

Componentes

Para fijar los mensajes de alerta en la pantalla de validación del formulario de Reconocimiento de Actores, se emplearon los siguientes elementos:

- **Vista (View):** Archivo de interfaz que define la estructura visual de la pantalla de validación de la solicitud. En esta vista se ubican los mensajes de alerta que deben permanecer fijos para el usuario.
- **PrimeFaces:** Framework utilizado para construir los componentes gráficos de la interfaz. En este caso, permite renderizar los mensajes y formularios de forma dinámica, asegurando una experiencia de usuario fluida y estructurada.

Codificación

Interfaz de Usuario (validación.xhtml)

Modificación de Alertas Permanentes

Para asegurar que los mensajes de alerta permanezcan visibles de forma permanente en la pantalla de validación, se eliminó manualmente la etiqueta **<button>** utilizada para cerrar dichos mensajes. En particular, se removió la siguiente línea de código de cada div con clase alert:

```
<button aria-hidden="true" data-dismiss="alert" class="close" type="button">x</button>
```

La etiqueta **<button>** anterior activa la funcionalidad de cierre del mensaje al hacer clic sobre la "X". Esta acción está ligada al atributo **data-dismiss="alert"**, propio de Bootstrap o integraciones similares, que permite ocultar el div correspondiente. Al eliminar esta etiqueta, se evita que el usuario pueda cerrar el mensaje, fijando así su visibilidad de manera permanente en la interfaz y asegurando que la notificación cumpla su propósito informativo hasta que el usuario realice alguna acción en el sistema.

13. Registrar la fecha y hora en el campo de "Progreso". (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)

Solicitud de requerimiento No. 163311

Descripción y alcance del Requerimiento

En el módulo de Requisitos de Beneficios Tributarios del aplicativo de Gestión de MinCiencias, se requiere incorporar la funcionalidad que permita registrar y mostrar la fecha y hora en el campo "Progreso" de la pantalla de requisitos de los proyectos.

El sistema deberá identificar si el documento asociado a un requisito fue cargado por primera vez (fecha creación) o si corresponde a una modificación posterior (fecha actualización), almacenando esta información en la base de datos.

La interfaz de usuario se actualizará para visualizar estos datos en tiempo real, de forma que los usuarios internos puedan verificar fácilmente cuándo se realizó la carga o actualización de cada documento.

Justificación

Actualmente, el módulo no permite identificar de manera precisa la fecha y hora de carga o modificación de los documentos de requisitos, lo que dificulta el seguimiento del avance y control de los proyectos.

La incorporación de esta funcionalidad mejorará la trazabilidad de la información, optimizará la labor de verificación por parte de los funcionarios y garantizará un mayor control sobre los

cambios realizados, fortaleciendo la transparencia y la gestión documental dentro del proceso de evaluación y seguimiento.

Desarrollo e Implementación

Se realizó un ajuste en la aplicación SIGP WEB, específicamente en el módulo de requisitos de proyectos para el aplicativo de Gestión. El desarrollo se orientó a mejorar la trazabilidad de los documentos cargados por los proponentes, incorporando en la interfaz una etiqueta que muestra la fecha y hora de carga del archivo asociado a cada requisito en la columna “Progreso”.

El cambio se aplicó exclusivamente en la pantalla de requisitos para proyectos de beneficios tributarios, dentro del aplicativo de Gestión, e incluyó los siguientes elementos:

1. Etiqueta con la fecha y hora de carga del documento:

- Se incorporó un componente visual que muestra la fecha y hora exacta en que fue registrado el documento, utilizando el campo fechaRegistro de la entidad DocumentoDT”.
- Esta información se presenta en un h:outputText, ubicado junto a la descripción de la columna “Progreso”

2. Integración con JSF:

- La etiqueta fue incorporada en la vista utilizando el componente h:outputText, que obtiene el valor de la fecha directamente desde la propiedad fechaRegistro o a través de un método específico del controlador.
- Se usó f:convertDateTime para dar formato consistente al valor, permitiendo su interpretación en tiempo real por parte del usuario.

3. Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- **Java EE (JSF + EJB):** Se utilizó JSF para representar la fecha en la vista, y EJB para mantener la lógica de negocio encapsulada.
- **PrimeFaces:** Integración visual dentro de la tabla de requisitos o cerca de los botones existentes utilizando componentes estándar (h:outputText).

- **Java:** Se incorporó lógica de formateo de fecha en el backend, en caso de no usarse directamente la conversión desde la vista JSF.

Aplicativo de Gestión

Componentes

Con el fin de brindar mayor claridad al usuario sobre los documentos cargados en los requisitos de una convocatoria, se integraron y adaptaron componentes dentro del aplicativo de Gestión. Esta mejora permite visualizar la fecha y hora de carga del archivo directamente desde la interfaz, facilitando la trazabilidad y validación del momento en que el proponente realizó el registro documental. A continuación, se detallan los principales componentes utilizados:

- **Vista (View):** La interfaz de usuario está representada por el archivo requisitos.xhtml. Esta páginas XHTML define la estructura visual del formulario de requisitos, incluyendo los botones y secciones relacionadas con la gestión y subsanación de documentos.
- **Servicio EJB (EJB Service):** Representado por la clase ProyectosServiceBean, que es el EJB (Enterprise JavaBeans) para usar los valores de la base de datos.
- **Base de Datos (Database):** Representada por la entidad DocumentoDTO, un data transfer object que hace referencia a la tabla SGP_DOCUMENTOS indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de Usuario (requisitos.xhtml)

Columna “Progreso”

En la tabla principal de requisitos del módulo de Beneficios Tributarios, se integró una etiqueta informativa dentro de una nueva columna denominada "Progreso". Esta etiqueta permite al usuario visualizar de forma clara si un documento ha sido cargado al requisito, junto con la fecha y hora exacta de dicha carga. El dato se muestra utilizando un formato legible con ayuda del componente **convertDateTime**.

```
<p:column width="12%">  
  <f:facet name="header">
```

```
<h:outputText value="#{msg['gestion.proyectosProgramas.requisitos.progreso.label']}"
/>
</f:facet>
<h:outputText styleClass="#{item.documentoDTO.nombre != null ? 'badge badge-primary' :
''}"
value="#{item.documentoDTO.nombre != null ?
msg['gestion.proyectosProgramas.requisitos.progreso.ok'] : ''}"
style="white-space: initial;"/>
<ui:fragment rendered="#{item.documentoDTO.nombre != null}">
<br/>
<div class="badge badge-info" style="white-space: initial;font-size: 9px;">
<b>#{msg['gestion.beneficiosTributarios.requisitos.subidoElDia.label']} </b>
<h:outputText value="#{item.documentoDTO.fechaModificacion != null
? item.documentoDTO.fechaModificacion
: item.documentoDTO.fechaCreacion}"
>
<f:convertDateTime pattern="dd/MM/yyyy HH:mm" timeZone="EST"/>
</h:outputText>
</ui:fragment>
</p:column>
```

A continuación, se explican las propiedades más relevantes del componente:

- **<p:column width="12%">**: Define una columna en la tabla con un ancho fijo del **12%**. Su encabezado es dinámico, obtenido desde el archivo de propiedades (**gestion.proyectosProgramas.requisitos.progreso.label**).
- **<h:outputText styleClass="..." value="...">**: Muestra una etiqueta tipo badge (badge-primary) que indica si el documento ha sido cargado. Esta se renderiza condicionalmente si **item.documentoDTO.nombre** es distinto de **null**, lo que sugiere que existe un archivo asociado.
- **<ui:fragment rendered="...">**: Se utiliza para mostrar de forma condicional la fecha de carga del documento solo si existe un documento adjunto.
- **<div class="badge badge-info">**: Renderiza visualmente la fecha y hora como un badge informativo con estilo personalizado para mejor legibilidad.
- **<h:outputText> + <f:convertDateTime>**: Muestra la fecha de modificación (**fechaModificacion**) si está disponible, o en su defecto la fecha de creación (**fechaCreacion**). Se aplica un convertidor de fecha con formato **dd/MM/yyyy HH:mm** y zona horaria **EST**, garantizando coherencia con el resto del sistema.

14. Validación de campos al generar la evaluación en la sección de Beneficios Tributarios. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)

Descripción y alcance del Requerimiento

Se requiere implementar una validación integral al momento de grabar una evaluación en los proyectos de Beneficios Tributarios.

Esta validación debe activarse al presionar el botón "Grabar" y deberá verificar lo siguiente:

1. Diligenciamiento de todos los campos obligatorios (concepto, criterios, subcriterios).
2. Las restricciones de caracteres en cada campo.
3. La sumatoria de puntajes sea correcta (criterios, subcriterios).
4. El sistema debe validar que todos los campos obligatorios del formulario de evaluación estén completos antes de permitir al usuario finalizar la evaluación. Es necesario mostrar un mensaje de error indicando los campos que faltan por completar y resaltar los campos obligatorios vacíos en el formulario.

Justificación

Se requiere implementar una validación automática de puntajes y campos obligatorios al finalizar la evaluación, con el fin de evitar el registro de evaluaciones incompletas o con inconsistencias, situación que actualmente afecta la calidad del proceso y genera reprocesos tanto para los evaluadores como para el equipo técnico. Esta validación permitirá garantizar la integridad y completitud de la información, reducir errores humanos, mejorar la trazabilidad de las decisiones tomadas y optimizar los tiempos de revisión, asegurando así la calidad de los datos registrados en el sistema.

Desarrollo e Implementación

Se realizó un ajuste implementado en la aplicación SIGP WEB, enfocado en fortalecer el proceso de evaluación de proyectos del módulo de Beneficios Tributarios dentro del aplicativo de Gestión.

El objetivo de este desarrollo fue garantizar que todas las evaluaciones realizadas por los funcionarios cumplan con los criterios mínimos de completitud, estructura y consistencia antes de ser finalizadas y registradas en el sistema.

El cambio se aplicó exclusivamente en el formulario de evaluación del módulo de Beneficios Tributarios, y comprende validaciones clave que aseguran la calidad de la información ingresada. Las funcionalidades incluidas fueron:

- Se implementó una verificación automática al presionar el botón “Grabar” que impide guardar la evaluación si existen campos obligatorios sin diligenciar, tales como: concepto general, criterios y subcriterios evaluativos.

- El sistema genera un mensaje de error detallando los campos faltantes y realiza un enfoque automático (focus) en el primer campo obligatorio vacío para guiar al evaluador en la corrección.
- Cada vez que se modifica un puntaje a nivel de subcriterio, el sistema recalcula automáticamente las sumatorias correspondientes en los niveles superiores (criterios y total general).
- Esta lógica evita inconsistencias numéricas y asegura que el puntaje refleje con precisión la evaluación ingresada.

Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- **Java EE (EJB):** Encargado de gestionar la lógica de negocio, incluyendo la validación de criterios evaluativos al guardar la evaluación.
- **JSF + PrimeFaces:** Usado para controlar la lógica del botón “Grabar” y mostrar mensajes dinámicos al evaluador.
- **JPA (Criteria API):** Utilizado para la consulta de los subcriterios evaluados versus los pendientes, según el proyecto.
- **PL/SQL:** Complemento en la capa de base de datos para obtener información precisa de los subcriterios pendientes.
- **Archivos .properties:** Para la internacionalización de mensajes de error y advertencia al evaluador.

Formulario de Beneficios Tributarios

Componentes

Para permitir la validación de puntajes y campos vacíos al finalizar la evaluación de proyectos en el módulo de Beneficios Tributarios se utilizaron los siguientes componentes en la aplicación:

- **Vista (View):** Representada por los archivos evaluacion.xhtml y proyectosEvaluador.xhtml, que son páginas XHTML asociadas a la interfaz de usuario.
- **Controlador (Controller de Conversación):** Representado por la clase ProyectoBeneficiosEvaluadoresConvController, que es un controlador de la capa de presentación.
- **Servicio EJB (EJB Service):** Representado por la clase BeneficiosServiceBean, que es el EJB (Enterprise JavaBeans) para gestionar los valores de la base de datos.

- **Base de Datos (Database):** Representada por la entidad ProyectoEvaluacionDTO, un data transfer object que hace referencia a la tabla PRY_EVALUACIONES indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **JPA (Java Persistence API):** Representada por el uso de **CriteriaBuilder** y otras clases del API de persistencia, utilizadas para la gestión de entidades, incluyendo operaciones de consulta, actualización y almacenamiento de los datos.
- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de Usuario (evaluación.xhtml)

Cambios necesarios para el enfoque automático en campos obligatorios no diligenciados.

Para lograr que el sistema enfoque (focus) automáticamente los campos obligatorios que no han sido diligenciados al intentar guardar una evaluación, se realizaron los siguientes ajustes en la interfaz de usuario:

- Evitar duplicación de mensajes de validación:

```
skipDetailIfEqualsSummary="true"
```

- Se agregó la propiedad **skipDetailIfEqualsSummary="true"** al componente **<p:messages>**.
- Esto permite que, si el resumen y el detalle del mensaje son iguales, solo se muestre uno, evitando duplicados innecesarios en la interfaz.

- Actualización dinámica de mensajes de error en campos seleccionables:

```
<p:ajax event="change" update="estadoEvaluador, proyectodtoFormDetalle:msg" process="@this"/>
```

- En los componentes **<p:selectOneMenu>** se incorporó un **<p:ajax>** con el evento **change**, y la instrucción **update="estadoEvaluador**,

proyectedtoFormDetalle:msg" para forzar la actualización del área de mensajes en tiempo real cada vez que el usuario modifica una selección.

- Esto permite mostrar o eliminar validaciones requeridas sin necesidad de recargar toda la vista.
- Refresco completo de componentes al crear o editar una evaluación:
 - En los botones que abren el formulario de edición o creación de evaluación, se añadió un update que incluye los selectores, la tabla de evaluadores y los mensajes:

```
update="msg, listadoEvaluadoresTabla, tipoEvaluador, estadoEvaluador, tipoEvaluacion"
```

- Esta actualización asegura que todos los componentes necesarios estén renderizados y actualizados correctamente, lo que permite una validación efectiva y habilita el comportamiento de enfoque sobre los campos vacíos al intentar guardar.

Controlador de Conversación ProyectoBeneficiosEvaluadoresConvController)

Método editar

Este método es el encargado de procesar la acción de edición de las evaluaciones de un proyecto, y es invocado directamente desde el botón "Grabar" en la interfaz gráfica (archivo **evaluacion.xhtml**) correspondiente a la pantalla de evaluación de proyectos. En el proceso de edición se actualizan los datos de modificación (usuario y fecha), y se incorporan validaciones clave para garantizar la completitud del proceso evaluativo antes de permitir su persistencia.

```
@Override
public void editar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));

    try {
        ProyectoEvaluacionDTO proyectoEvaluacionDTO =
        proyectoBeneficiosEvaluadoresSessController.getProyectoEvaluacionDTO();
        proyectoEvaluacionDTO.setFechaModificacion(new Date());

        proyectoEvaluacionDTO.setUsuarioModificador(seguridadSessController.getSelected().getUsuario());
        //... más validaciones

        if (proyectoEvaluacionDTO.getConcepto() != null &&
        !proyectoEvaluacionDTO.getConcepto().isEmpty()) {
            Map<String, Object> filters = new HashMap();
```

```

        filters.put("proyectoId",
proyectoEvaluacionDTO.getProyectoId().getProyectoId());
        List<ProyectoEvaluacionDTO> lstEvaluaciones =
beneficiosServiceFacade.proyectosEvaluacionList(-1, -1, "evaluacionProyectoId", "A",
filters);
        if (lstEvaluaciones != null && !lstEvaluaciones.isEmpty()) {
            ProyectoEvaluacionDTO antes = lstEvaluaciones.stream()
                .filter(t
t.getEvaluacionProyectoId().equals(proyectoEvaluacionDTO.getEvaluacionProyectoId()))
                .findFirst()
                .orElse(null);
            if (antes != null) {
                if ((antes.getConcepto() == null || antes.getConcepto().isEmpty())
||
!antes.getConcepto().equals(proyectoEvaluacionDTO.getConcepto())) {
beneficiosServiceFacade.verificarCamposVacios(proyectoEvaluacionDTO);
                }
            }
        }
    }
}

```

Una de las validaciones más relevantes implementadas en este método es la relacionada con el campo **"Concepto Final"**. Si el concepto ha sido modificado (o fue ingresado por primera vez), el sistema procede a verificar si todos los subcriterios de evaluación han sido calificados. Esta verificación se realiza mediante la llamada al método **verificarCamposVacios(...)** del **beneficiosServiceFacade**. La lógica garantiza que el evaluador no pueda grabar una evaluación parcial (con subcriterios sin calificar) una vez haya redactado o cambiado su concepto final, promoviendo así la calidad y completitud del proceso.

Otro fragmento relevante dentro del flujo de evaluación es la invocación al método que recalcula automáticamente el puntaje total luego de realizar modificaciones sobre la evaluación.

```

if (proyectoEvaluacionDTO.getConcepto() != null &&
!proyectoEvaluacionDTO.getConcepto().isEmpty()) {
beneficiosServiceFacade.callSBProyectosCalcularPuntaje(proyectoEvaluacionPr
oyectoId());
}

```

Se valida que el campo **"concepto"** no esté vacío antes de proceder. Luego, se invoca un método del **ServiceFacade** que recalcula el puntaje total asociado a la evaluación, garantizando que cualquier cambio en subcriterios se refleje inmediatamente en la sumatoria final.

Método editarEvaluacion

El método `editarEvaluacion()` es invocado desde la interfaz gráfica **proyectosEvaluador.xhtml**, específicamente al presionar el botón **“Grabar”**, cuya funcionalidad es modificar la evaluación de un proyecto. Antes de persistir cualquier cambio, el método realiza una serie de validaciones orientadas a garantizar la integridad del proceso evaluativo.

```
public void editarEvaluacion(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ProyectoEvaluacionDTO proyectoEvaluacionDTO = evaluacionSessController.getSelected();
        proyectoEvaluacionDTO.setFechaModificacion(new Date());

        proyectoEvaluacionDTO.setUsuarioModificador(seguridadSessController.getSelected().getUsuario());

        //... otras validaciones

        //verificar campos vacíos evaluación
        beneficiosServiceFacade.verificarCamposVacios(proyectoEvaluacionDTO);
    }
}
```

Una de las validaciones más importantes incorporadas en este método es la verificación de completitud en la calificación de todos los subcriterios de evaluación. Esta validación se realiza siempre que se intente modificar una evaluación, sin necesidad de que el concepto final haya sido modificado previamente. Para ello, se invoca el método **verificarCamposVacios(...)**, el cual valida que no existan subcriterios pendientes por calificar. En caso de encontrar subcriterios sin evaluación, el sistema evita la grabación de la evaluación y despliega un mensaje informando la cantidad de campos faltantes. Esto garantiza que las evaluaciones se realicen de manera completa antes de ser almacenadas.

Servicio EJB (BeneficiosServiceBean)

Método Verificar Campos Vacíos

El método **verificarCamposVacios** forma parte del servicio EJB **BeneficiosServiceBean** y se encarga de validar que todos los subcriterios evaluables de una evaluación hayan sido calificados antes de permitir su persistencia. Se ejecuta como una validación previa al guardado de la evaluación, garantizando la integridad y completitud del proceso.

```
@Override
public void verificarCamposVacios(ProyectoEvaluacion proyectoEvaluacion) throws
SystemException {
    if (proyectoEvaluacion != null) {
        if (proyectoEvaluacion.getProyectoId() != null &&
            proyectoEvaluacion.getEvaluacionProyectoId() != null) {
            int numberMissingEvaluation = this.getStoreManager(ProyectosStoreManager.class)
                .validarCamposPorEvaluar(proyectoEvaluacion);
            if (numberMissingEvaluation != 0) {
                // ...
            }
        }
    }
}
```

```

        SystemException error = new
        SystemException("#{msg['gestion.beneficiosTributarios.evaluadores.error.camposSinEvaluar']}");
        error.getParameters().add(String.valueOf(numberMissingEvaluation));
        throw error;
    }
}
}
}

```

El método verifica que el objeto **ProyectoEvaluacion** recibido contenga los identificadores necesarios (**proyectoId** y **evaluacionProyectoId**). Si los datos son válidos, invoca al método **validarCamposPorEvaluar(...)** del store manager **ProyectosStoreManager**, el cual retorna la cantidad de subcriterios aún sin calificación. Si el número retornado es distinto de cero, se lanza una excepción **SystemException**, adjuntando como parámetro el número de campos faltantes, e interrumpiendo así el proceso de modificación.

```

@Override
public int validarCamposPorEvaluar(ProyectoEvaluacion proyectoEvaluacion) {
    Query query = em.createNativeQuery("SELECT COUNT(*)\n" +
        " FROM PRY_EVALUACIONES_CALIFICACION p\n" +
        " WHERE p.proyecto_id = ?\n" +
        " AND p.evaluacion_proyecto_id = ?\n" +
        " AND p.calificacion IS NULL\n" +
        " AND p.entenable = 'S' ");
    query.setParameter(1, proyectoEvaluacion.getProyectoId().getProyectoId());
    query.setParameter(2, proyectoEvaluacion.getEvaluacionProyectoId());
    Object o = query.getSingleResult();
    int result = 0;
    if (o instanceof BigDecimal) {
        return ((BigDecimal) o).intValue();
    } else if (o instanceof Long) {
        return ((Long) o).intValue(); // Correcto: obtienes el valor de Long y lo conviertes
    }
    return result;
}

```

Este método realiza la validación directa sobre la base de datos utilizando una consulta SQL nativa, cuyo propósito es contar los registros de la tabla **PRY_EVALUACIONES_CALIFICACION** que correspondan a una evaluación específica (**proyecto_id** y **evaluacion_proyecto_id**), y que no tengan calificación asignada (**calificacion IS NULL**) y estén marcados como entrables (**entenable = 'S'**).

La consulta es ejecutada mediante un **EntityManager (JPA)** usando **createNativeQuery()**. Una vez obtenida la respuesta (**getSingleResult()**), se convierte el valor a **int**, tomando en cuenta que el tipo devuelto por la base de datos puede variar (**BigDecimal** o **Long**, dependiendo del proveedor **JDBC**).

Este método retorna el número de subcriterios faltantes por evaluar, lo cual es utilizado por la capa de servicio para determinar si se permite o no guardar la evaluación.

Método Proyectos Calcular Calificación

El método **callSBProyectosCalcularCalificacion** hace parte del servicio EJB **BeneficiosServiceBean** y tiene como propósito actualizar automáticamente el puntaje total de la evaluación de un proyecto cuando esta es modificada.

```
@Override
public void callSBProyectosCalcularCalificacion(Long evaluacionProyectoId) {
    em.createNativeQuery("call SB_PROYECTOS.calcular_calificacion(?)")
        .setParameter(1, evaluacionProyectoId).executeUpdate();
}
```

Este método ejecuta una consulta nativa que invoca el procedimiento almacenado **SB_PROYECTOS.calcular_calificacion**, al cual se le pasa como parámetro el ID de la evaluación del proyecto. La lógica de cálculo del puntaje reside en la base de datos, lo que garantiza que las sumatorias de criterios y subcriterios se mantengan consistentes cada vez que se actualiza una evaluación.

15. Límite de palabras: Implementar una funcionalidad que permita configurar un límite mínimo y máximo de palabras para las secciones de criterios y subcriterios en el formulario de evaluación de beneficios tributarios. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)

Solicitud de requerimiento No. 163307

Descripción y alcance del Requerimiento

Se requiere la implementación de una funcionalidad configurable que permita establecer límites mínimos y máximos de palabras en las secciones de criterios y subcriterios del formulario de evaluación de proyectos de Beneficios Tributarios utilizado por los pares evaluadores o comités técnicos.

La funcionalidad debe cumplir con los siguientes lineamientos técnicos:

1. Configuración de parámetros:
 - Agregar una opción en el panel de administración del sistema que permita definir los siguientes valores:

- Límite mínimo de palabras por sección.
 - Límite máximo de palabras por sección.
 - La configuración debe permitir definir los límites de manera independiente para cada criterio y subcriterio, si así se requiere.
2. Validación en tiempo real:
- Si el texto ingresado no cumple con los valores configurados, el sistema debe generar una alerta informativa y bloquear el envío del formulario hasta que se corrija.
3. Mensajes de validación personalizados:
- Mostrar mensajes claros como:
 - “Debe escribir al menos X palabras.”
 - “Ha excedido el límite máximo de Y palabras.”
4. Persistencia y trazabilidad:
- Almacenar en la base de datos los valores configurados.
 - Registrar en el historial del sistema quién y cuándo modificó los límites.
5. Compatibilidad:
- La funcionalidad debe ser compatible con navegadores modernos y adaptarse a dispositivos móviles.

Justificación

Se requiere que el sistema valide los límites de palabras en secciones clave del formulario de evaluación de beneficios tributarios —como los criterios, subcriterios y el concepto final— con el fin de garantizar la consistencia, calidad y extensión adecuada de la información diligenciada por los evaluadores. Esta funcionalidad permitirá estandarizar las evaluaciones, facilitar su análisis y asegurar que los aportes se mantengan dentro de los parámetros definidos por la entidad. Asimismo, contribuye a preservar la integridad de la información registrada, evitando excesos o carencias que puedan afectar la objetividad y utilidad de los resultados del proceso evaluativo.

Desarrollo e Implementación

El desarrollo realizado en la aplicación SIGP WEB, estuvo orientado a la implementación de una nueva funcionalidad de configuración de límites de palabras para los campos de concepto en los subcriterios de evaluación dentro de las matrices asociadas a las convocatorias de proyectos. Esta mejora tiene como propósito establecer un control sobre la extensión textual de los conceptos registrados por los evaluadores, permitiendo definir un mínimo y un máximo de palabras, según lo requiera cada convocatoria.

El desarrollo abarcó la modificación e integración de componentes tanto en el módulo de gestión como en el formulario de evaluación, permitiendo la configuración y aplicación de los límites definidos por cada matriz. Las funcionalidades incluidas fueron:

1. Configuración de límites de palabras en la matriz de evaluación

- Inclusión de campos para el valor mínimo y máximo de palabras por subcriterio evaluable.
- Visualización y edición de estos valores desde la pantalla de gestión de matrices de evaluación.

2. Validación en el formulario de evaluación

- Contador dinámico de palabras en el campo de concepto por subcriterio.
- Validación del cumplimiento de los valores mínimos y máximos configurados.
- Mensajes de retroalimentación al evaluador en caso de incumplimiento.

3. Persistencia y lógica de negocio

- Almacenamiento de los valores configurados en la base de datos.
- Aplicación de la lógica de validación durante la creación o edición de evaluaciones.

4. Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- **Java EE (Java EE (EJB)):** Gestión de la lógica de negocio para la obtención y validación de los parámetros configurados por matriz.

- **JSF + PrimeFaces:** Construcción de las pantallas de configuración y validación en tiempo real de los conceptos evaluativos.
- **JPA (Criteria API):** Integración con la capa de persistencia para el almacenamiento de los valores de límite por matriz.
- **PL/SQL:** Modificaciones en la base de datos para incluir los nuevos campos de configuración.
- ***Mensajes .properties:** Para la internacionalización de los mensajes de validación y retroalimentación en el formulario de evaluación.

Formatos de Evaluación

Componentes

Para permitir la configuración del límite mínimo y máximo de palabras en los conceptos de subcriterios evaluativos desde la pantalla de Formatos de Evaluación, se utilizaron los siguientes componentes en la aplicación:

- **Vista (View):** Representada por el archivo `formatosEvaluacion.xhtml`, que es una página XHTML asociada a la interfaz de usuario.
- **Controlador (Controller de Vista y Conversación):** Representado por las clases `FormatoEvaluacionViewController` y `FormatoEvaluacionConvController`, que son los controladores de la capa de presentación.
- **Servicio EJB (EJB Service):** Representado por la clase `FormatoEvaluacionServiceBean`, que es el EJB (Enterprise JavaBeans) para gestionar los valores de la base de datos.
- **Base de Datos (Database):** Representada por la entidad `ProyectoDTO`, un data transfer object que hace referencia a la tabla `SGP_PROYECTOS` indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **JPA (Java Persistence API):** Representada por el uso de **CriteriaBuilder** y otras clases del API de persistencia, utilizadas para la gestión de entidades, incluyendo operaciones de consulta, actualización y almacenamiento de los datos.

- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de usuario (proyectosAntiguos.xhtml)

Campos "Mínimo de Palabras" y "Máximo de Palabras"

Este componente forma parte de la vista de gestión de formatos de evaluación. Permite configurar de manera visual los límites mínimo y máximo de palabras para los subcriterios entrables (aquellos donde se requiere ingresar un concepto evaluativo). La visualización de estos campos se condiciona únicamente a los subcriterios marcados como entrables (**entrable = 'S'**).

```
<!-- Fila Mínimo de Palabras - Máximo de Palabras -->
<p:outputPanel id="minMaxWords">
    <!-- Solo aplica para subcriterios donde entrable es "S" -->
    <ui:fragment rendered="#{formatoEvaluacionViewController.selected.entrable eq 'S'}">
        <div class="row">
            <div class="col-lg-6">
                <div class="form-group">
                    <p:outputLabel
value="#{msg['sb.menu.gestion.administracion.formatoEvaluacion.maximoPalabras.label']}"
                    styleClass="col-lg-6 control-label"
                    style="text-align: right"
                    for="maximoPalabras"/>
                    <div class="col-lg-6">
                        <p:inputNumber id="maximoPalabras"
maxValue="99999"
decimalPlaces="0"

value="#{formatoEvaluacionViewController.selected.maximoPalabras}"
                    style="width:100%"
                    required="false"

placeholder="#{msg['sb.menu.gestion.administracion.formatoEvaluacion.maximoPalabras.placehol
der']}">
                        </p:inputNumber>
                    </div>
                </div>
            </div>
            <div class="col-lg-6">
                <div class="form-group">
                    <p:outputLabel
value="#{msg['sb.menu.gestion.administracion.formatoEvaluacion.minimoPalabras.label']}"
                    styleClass="col-lg-6 control-label"
                    style="text-align: center"
                    for="minimoPalabras"/>
                    <div class="col-lg-6">
                        <p:inputNumber id="minimoPalabras"
maxValue="99999"
decimalPlaces="0"

value="#{formatoEvaluacionViewController.selected.minimoPalabras}"
                    style="width:100%"
```

```

required="false"

placeholder="#{msg['sb.menu.gestion.administracion.formatoEvaluacion.minimoPalabras.placehol
der']}">
        </p:inputNumber>
    </div>
</div>
</div>
</div>
</ui:fragment>
</p:outputPanel>

```

A continuación, una explicación más detallada:

Contenedor General

- **<p:outputPanel id="minMaxWords">**: Panel de PrimeFaces que encapsula los campos y puede ser actualizado dinámicamente mediante **AJAX**.
- **<ui:fragment rendered="#{formatoEvaluacionViewController.selected.entenable eq 'S'}">**: Fragmento condicional que renderiza el contenido solo si el subcriterio es entrable, es decir, si requiere un campo para ingresar concepto.

Campo "Máximo de Palabras"

- **<p:outputLabel>**: Etiqueta asociada al campo numérico. Su texto se toma desde el archivo de mensajes internacionalizado.
- **<p:inputNumber id="maximoPalabras">**:
 - **maxValue="99999"**: Establece el valor máximo permitido.
 - **decimalPlaces="0"**: Solo acepta números enteros.
 - **value="#{formatoEvaluacionViewController.selected.maximoPalabras}"**: Valor vinculado al atributo del subcriterio actualmente seleccionado.
 - **placeholder**: Texto guía dentro del campo, también internacionalizado.
 - **style="width:100%"**: Se ajusta al ancho disponible del contenedor.

Campo "Mínimo de Palabras"

- **<p:outputLabel>**: Etiqueta asociada al campo numérico. Su texto se toma desde el archivo de mensajes internacionalizado.
- **<p:inputNumber id="minimoPalabras">**:
 - **maxValue="99999"**: Establece el valor máximo permitido.
 - **decimalPlaces="0"**: Solo acepta números enteros.
 - **value="#{formatoEvaluacionViewController.selected.minimoPalabras}"**: Valor vinculado al atributo del subcriterio actualmente seleccionado.
 - **placeholder**: Texto guía dentro del campo, también internacionalizado.

- **style="width:100%"**: Se ajusta al ancho disponible del contenedor.

Servicio EJB (FormatoEvaluacionServiceBean)

Durante las operaciones de creación y edición de un formato de evaluación, se integró la invocación al método **verifyMinAndMaxWords**, encargado de validar que los valores ingresados para el número mínimo y máximo de palabras sean coherentes y cumplan ciertas reglas de negocio definidas.

Esta validación se realiza desde el servicio EJB (**FormatoEvaluacionServiceBean**), y se ejecuta antes de persistir los datos en la base de datos. El método se invoca dentro de las funciones **guardarFormatoEvaluacion** y **editarFormatoEvaluacion**, como se muestra a continuación:

```
@Override
public FormatoEvaluacion guardarFormatoEvaluacion(FormatoEvaluacion formatoEvaluacion) throws
SystemException {
    verifyMinAndMaxWords(formatoEvaluacion);
    return this.crear(formatoEvaluacion, FormatoEvaluacion.class);
}

@Override
public void editarFormatoEvaluacion(FormatoEvaluacion formatoEvaluacion) throws
SystemException {
    verifyMinAndMaxWords(formatoEvaluacion);
    this.modificar(formatoEvaluacion, FormatoEvaluacion.class);
}

private void verifyMinAndMaxWords(FormatoEvaluacion formatoEvaluacion) throws SystemException
{
    if (formatoEvaluacion.getMaximoPalabras() != null &&
formatoEvaluacion.getMinimoPalabras() != null) {
        if (formatoEvaluacion.getMaximoPalabras() < formatoEvaluacion.getMinimoPalabras()) {
            throw new
SystemException("#{msg['sb.menu.gestion.administracion.formatoEvaluacion.minMaxPalabras.exce
ption']}");
        }
        else if (formatoEvaluacion.getMaximoPalabras() != null) {
            if (formatoEvaluacion.getMaximoPalabras() < 60) {
                throw new
SystemException("#{msg['sb.menu.gestion.administracion.formatoEvaluacion.maxPalabras.excepti
on']}");
            }
        }
    }
}
```

El método **verifyMinAndMaxWords** valida los campos **minimoPalabras** y **maximoPalabras** del objeto **FormatoEvaluacion** antes de guardarlo en base de datos, asegurando coherencia lógica y cumplimiento de las reglas de negocio. Primero, si ambos valores están definidos, verifica que el

mínimo no sea mayor que el máximo; de lo contrario, lanza una **SystemException** con un mensaje internacionalizado. Adicionalmente, si solo se define el **maximoPalabras**, se valida que no sea inferior a 60, ya que este es el valor mínimo aceptado por defecto. Estas validaciones previenen configuraciones inconsistentes y garantizan una entrada de datos sólida y controlada.

Evaluación de Criterios Beneficios Tributarios

Componentes

Para permitir la validación del límite mínimo y máximo de palabras en los conceptos de subcriterios evaluativos desde la pantalla de evaluación del módulo de Beneficios Tributarios, se utilizaron los siguientes componentes en la aplicación:

- **Vista (View):** Representada por el archivo `evaluacionCalificacion.xhtml` que es una página XHTML asociada a la interfaz de usuario.
- **Controlador (Controller de Vista y Conversación):** Representado por las clases `ProyectoBeneficiosEvaluacionCalificacionSessController` y `ProyectoBeneficiosEvaluacionCalificacionConvController`, que son los controladores de la capa de presentación.
- **Servicio EJB (EJB Service):** Representado por la clase `ProyectoServiceBean`, que es el EJB (Enterprise JavaBeans) para gestionar las calificaciones de evaluación.
- **Base de Datos (Database):** Representada por la entidad `EvaluacionCalificacionDTO`, un data transfer object que hace referencia a la tabla `PRY_EVALUACIONES_CALIFICACION` indicando que hay alguna interacción con la base de datos para manejar estos tipos.
- **JPA (Java Persistence API):** Representada por el uso de **CriteriaBuilder** y otras clases del API de persistencia, utilizadas para la gestión de entidades, incluyendo operaciones de consulta, actualización y almacenamiento de los datos.
- **PrimeFaces:** Utilizado para la construcción de la interfaz gráfica, proporcionando componentes como formularios, campos de entrada, tablas y validaciones, que permiten una interacción fluida y validada con el usuario final.

Codificación

Interfaz de usuario (evaluacionCalificacion.xhtml)

Campo "Concepto"

La interfaz de evaluación —**evaluacionCalificacion.xhtml**— presenta la matriz de evaluación y contiene un campo denominado **"Concepto"**. Este campo permite al evaluador ingresar una justificación escrita para cada subcriterio evaluable, siempre que esté marcado como "entrable".

```
<p:inputTextarea id="concepto"
                 style="resize: none; overflow-y: scroll;"
                 rows="7"

value="#{proyectoBeneficiosEvaluacionCalificacionSessController.aspectoCalificacionSeleccion
ado.evaluacionCalificacionDTO.concepto}"
                 styleClass="form-control"
                 cols="30"                 counter="displayConceptoFinal"                 maxLength="8000"
counterTemplate="{0} caracteres restantes."
                 autoResize="false"
                 required="true"

requiredMessage="#{msg['gestion.beneficiosTributarios.evaluadores.calificacion.conceptoArg.r
equired']}"

placeholder="#{proyectoBeneficiosEvaluacionCalificacionSessController.obtenerPlaceHolderConc
epto(proyectoBeneficiosEvaluacionCalificacionSessController.aspectoCalificacionSeleccionado)
}"

readonly="#{proyectoBeneficiosEvaluacionCalificacionSessController.evaluacionCerrada
or
!proyectoBeneficiosEvaluacionCalificacionSessController.usuarioHabilitadoEvaluar}"
                 onkeyup="updateWordCount()"
/>
<h:outputText id="displayConceptoFinal" style="font-style: italic" />
<h:outputText id="wordCount" style="font-style: italic; display: block;" value="0 palabras"
/>
```

A continuación, una explicación más detallada:

- **<p:inputTextarea id="concepto">**: Área de texto donde el evaluador redacta su concepto o justificación.
- **value="#{proyectoBeneficiosEvaluacionCalificacionSessController.aspectoCalificacionSeleccionado.evaluacionCalificacionDTO.concepto}"**: El contenido ingresado se enlaza al DTO correspondiente al aspecto de calificación actualmente seleccionado.
- **rows="7" y cols="30"**: Define el tamaño visual del área de texto (alto y ancho base).
- **style="resize: none; overflow-y: scroll;"**: Se desactiva el redimensionamiento manual del campo, permitiendo el scroll vertical si el contenido excede el área.
- **styleClass="form-control"**: Aplica estilos de diseño consistentes con el resto de los campos del formulario.

- **maxlength="8000"**: Limita el número total de caracteres que se puede ingresar.
- **counter="displayConceptoFinal" y counterTemplate="{0} caracteres restantes."**: Muestra un contador dinámico de caracteres restantes asociado al componente **h:outputText** con ID **displayConceptoFinal**.
- **required="true" y requiredMessage**: Hace obligatorio el ingreso de texto. Si el campo está vacío, se mostrará un mensaje de error personalizado.
- **placeholder="#{...obtenerPlaceHolderConcepto(...)}"**: Se muestra un texto guía dentro del campo, determinado dinámicamente según el aspecto de calificación.
- **readonly="#{...evaluacionCerrada or /usuarioHabilitadoEvaluar}"**: Controla si el campo debe estar deshabilitado, dependiendo del estado de la evaluación y los permisos del usuario.
- **onkeyup="updateWordCount()"**: Evento que actualiza dinámicamente el número de palabras ingresadas en el campo.
- **<h:outputText id="wordCount" value="0 palabras" />**: Muestra el conteo actual de palabras (actualizado por JavaScript según el contenido escrito).

Función JavaScript updateWordCount()

Se implementó la función **updateWordCount()** con el objetivo de contar en tiempo real el número de palabras que el evaluador ha ingresado en el campo **"Concepto"**. Esta funcionalidad permite brindar retroalimentación inmediata al usuario sobre la extensión de su texto, especialmente útil cuando se establecen límites mínimos o máximos de palabras.

```
<script type="text/javascript">
  function updateWordCount() {
    const textArea = document.getElementById('dlgConceptoArgForm:concepto');
    const wordCountDisplay = document.getElementById('dlgConceptoArgForm:wordCount');
    const words = textArea.value.trim().split(/\s+/).filter(word => word.length > 0);
    const wordCount = words.length;
    wordCountDisplay.innerText = wordCount + " palabras";
  }
</script>
```

A continuación, una explicación más detallada:

- **getElementById('dlgConceptoArgForm:concepto')**: Obtiene una referencia al campo **<p:inputTextarea>** donde el usuario escribe el concepto.
- **getElementById('dlgConceptoArgForm:wordCount')**: Obtiene la referencia al elemento donde se mostrará el número actual de palabras.
- **textArea.value.trim().split(/\s+/)**: Elimina espacios al inicio y final, y luego divide el texto por cualquier secuencia de espacios o saltos de línea.
- **.filter(word => word.length > 0)**: Asegura que no se cuenten espacios vacíos como palabras.

- **wordCountDisplay.innerText = wordCount + " palabras":** Actualiza el texto visible en la pantalla con el número actual de palabras ingresadas.

Controlador de Vista (ProyectoBeneficiosEvaluacionCalificacionSessController)

Se incorporó el método **obtenerPlaceholderConcepto(CalificacionEvaluadorVO aspectoCalificacionSeleccionado)**, el cual permite construir dinámicamente el texto de ayuda (placeholder) que se muestra dentro del campo **"Concepto"** en el formulario de evaluación.

```
public String obtenerPlaceholderConcepto(CalificacionEvaluadorVO
aspectoCalificacionSeleccionado) {
    ResourceBundle bundle =
    FacesContext.getCurrentInstance().getApplication().getResourceBundle(FacesContext.getCurrent
Instance(), "msg");

    if (aspectoCalificacionSeleccionado != null &&
aspectoCalificacionSeleccionado.getFormatoEvaluacionDTO() != null) {

        Integer minimo =
aspectoCalificacionSeleccionado.getFormatoEvaluacionDTO().getMinimoPalabras();
        Integer maximo =
aspectoCalificacionSeleccionado.getFormatoEvaluacionDTO().getMaximoPalabras();

        String pattern;

        if (minimo != null && maximo != null) {
            pattern =
bundle.getString("gestion.beneficiosTributarios.evaluadores.calificacion.conceptoArg.minMaxW
ords.placeholder");
            return MessageFormat.format(pattern, minimo, maximo);
        } else if (minimo != null) {
            pattern =
bundle.getString("gestion.beneficiosTributarios.evaluadores.calificacion.conceptoArg.minWord
s.placeholder");
            return MessageFormat.format(pattern, minimo);
        } else if (maximo != null) {
            pattern =
bundle.getString("gestion.beneficiosTributarios.evaluadores.calificacion.conceptoArg.maxWord
s.placeholder");
            return MessageFormat.format(pattern, maximo);
        }
    }
    return
bundle.getString("gestion.beneficiosTributarios.evaluadores.calificacion.conceptoArg.placeho
lder");
}
```

Este método consulta los valores configurados para el mínimo y máximo de palabras del subcriterio seleccionado dentro de la matriz de evaluación, y con base en ello devuelve un mensaje internacionalizado acorde a la configuración. Si ambos valores están definidos, el placeholder indica el rango esperado; si solo uno de los dos está presente, se ajusta el mensaje de forma

correspondiente. En ausencia de configuración, retorna un texto por defecto. La plantilla del mensaje se toma del archivo de propiedades **msg** y se formatea utilizando **MessageFormat**.

Controlador *de* *Conversación*
(ProyectoBeneficiosEvaluacionCalificacionConvController)

Durante el proceso de calificación de un subcriterio en la matriz de evaluación, se invoca el método **editar()**, encargado de procesar y validar la información ingresada por el evaluador. Entre las validaciones realizadas se encuentra la verificación del número de palabras ingresadas en el campo **"Concepto"**, asegurando que cumpla con los límites mínimo y/o máximo definidos previamente en la configuración del formato de evaluación. Esta validación se aplica antes de permitir el guardado de la calificación y tiene como objetivo garantizar la calidad y completitud del concepto argumentativo.

```
// Validamos que el documento de verificacion tenga como minimo 60 palabras o cumpla con la
// configuración
if (calificacionEvaluadorVO.getEvaluacionCalificacionDTO().getConcepto() != null &&
!calificacionEvaluadorVO.getEvaluacionCalificacionDTO().getConcepto().isEmpty()) {

    StringTokenizer stringTokenizerConcepto = new
StringTokenizer(calificacionEvaluadorVO.getEvaluacionCalificacionDTO().getConcepto());

    if (calificacionEvaluadorVO.getFormatoEvaluacionDTO() != null) {

        Integer minimo =
calificacionEvaluadorVO.getFormatoEvaluacionDTO().getMinimoPalabras();
        Integer maximo =
calificacionEvaluadorVO.getFormatoEvaluacionDTO().getMaximoPalabras();

        SystemException excepcion = new SystemException("");

        if (minimo != null && maximo != null) {
            if (stringTokenizerConcepto.countTokens() < minimo ||
stringTokenizerConcepto.countTokens() > maximo) {

                excepcion.setMessage(resolveExpression("#{msg['gestion.beneficiosTributarios.evaluadores.cal
ificacion.error.conceptoArg.minMaxWords']}"));
                excepcion.setParameters(new ArrayList<>());
                excepcion.getParameters().add(minimo.toString());
                excepcion.getParameters().add(maximo.toString());
                modifyAlert = "S";
                throw excepcion;
            }
        } else if (minimo != null) {
            if (stringTokenizerConcepto.countTokens() < minimo) {

                excepcion.setMessage(resolveExpression("#{msg['gestion.beneficiosTributarios.evaluadores.cal
ificacion.error.conceptoArg.minWords']}"));
                excepcion.setParameters(new ArrayList<>());
                excepcion.getParameters().add(minimo.toString());
                modifyAlert = "S";
                throw excepcion;
            }
        }
    }
}
```



```
    }
    } else if (maximo != null) {
        if (stringTokenizerConcepto.countTokens() < 60 ||
            stringTokenizerConcepto.countTokens() > maximo) {

            excepcion.setMessage(resolveExpression("#{msg['gestion.beneficiosTributarios.evaluadores.calificacion.error.conceptoArg.maxWords']}"));
            excepcion.setParameters(new ArrayList<>());
            excepcion.getParameters().add(maximo.toString());
            modifyAlert = "S";
            throw excepcion;
        }
    } else {
        if (stringTokenizerConcepto.countTokens() < 60) {
            throw new
            Exception("#{msg['gestion.beneficiosTributarios.evaluadores.calificacion.error.conceptoArg']}");
        }
    }
}
}
```

Las validaciones sobre el campo concepto se realizan utilizando la clase **StringTokenizer** para contar la cantidad de palabras ingresadas. Las validaciones dependen de los valores configurados en el objeto **FormatoEvaluacionDTO** asociado al subcriterio evaluado. Si se encuentran definidos tanto el mínimo como el máximo de palabras, se valida que el número de palabras esté dentro del rango permitido; en caso contrario, se lanza una **SystemException** con el mensaje correspondiente.

Si solo se encuentra definido el valor mínimo, se verifica que el número de palabras no sea inferior a ese umbral. Por el contrario, si solo se define el valor máximo, se exige además un mínimo fijo de 60 palabras. Finalmente, si no existe ninguna configuración, se aplica una validación por defecto que exige un mínimo de 60 palabras. En todos los escenarios, los mensajes de error son internacionalizados y, cuando corresponde, parametrizados para reflejar los valores esperados. Esta validación evita el ingreso de conceptos incompletos o fuera de los rangos permitidos.

16. Para la configuración de convocatoria crear un campo que restrinja la contrapartida con respecto al solicitado en la financiación, en la actualizada está la restricción a la sumatoria del total del proyecto contrapartida + financiación. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)

Descripción y alcance del Requerimiento

Implementar en la configuración de convocatorias un nuevo parámetro que permita definir si el porcentaje de contrapartida debe calcularse sobre:

- El monto total del proyecto (financiación solicitada + contrapartida), o
- Únicamente sobre el monto de la financiación solicitada.

Este parámetro será configurable desde el módulo de gestión, de forma independiente para cada convocatoria. La lógica asociada a este parámetro debe aplicarse en el formulario de proyectos, de manera que, al registrar los rubros de financiación y contrapartida, el sistema valide y calcule los porcentajes según la configuración establecida, impidiendo el registro de valores que no cumplan la regla definida.

Justificación

Actualmente, el cálculo del porcentaje de contrapartida está fijado sobre un criterio único, lo que no permite adaptarse a los diferentes lineamientos de cada convocatoria. Al incorporar este parámetro, se otorga flexibilidad al sistema para ajustarse a las condiciones particulares de cada proceso, garantizando que las validaciones en el formulario de proyectos respondan a la configuración definida y evitando errores de registro que afecten el cumplimiento de los requisitos.

Desarrollo e Implementación

El ajuste busca permitir que, en la configuración de cada convocatoria, se defina si el porcentaje de contrapartida en dinero debe aplicarse sobre el **Monto Total del Proyecto** o sobre el **Monto Total Financiado**, y que esta definición se refleje posteriormente en la **validación de los proyectos** antes de su radicación en **Minciencias**.

El desarrollo contempla:

- Modificación de la estructura de la tabla SGP_CONVOCATORIAS.
- Creación de restricciones de integridad para garantizar valores válidos.
- Incorporación del nuevo campo en la capa de presentación (pantalla de administración de convocatorias).
- Integración del nuevo atributo en el controlador de sesión y las listas de configuración (convocatoriaSessController, configApplController).

- **Extensión del impacto en los formularios de proyectos:** ajuste de validaciones previas al envío y radicación.

Cambios en Base de Datos

Script de Creación del Campo

```
-- Add/modify columns
ALTER TABLE SGP_CONVOCATORIAS
ADD aplicar_porc_contrap_monto VARCHAR2(1);

-- Add comments to the columns
COMMENT ON COLUMN SGP_CONVOCATORIAS.aplicar_porc_contrap_monto
IS 'Indicar a qué monto total debe aplicarse el porcentaje de contrapartida dinero (Monto
Total Proyecto, Monto Total Financiado)';

-- Create/Recreate check constraints
ALTER TABLE SGP_CONVOCATORIAS
ADD CONSTRAINT SGPCONVOC_APLICAR_PORC_CK
CHECK (aplicar_porc_contrap_monto IN (
    'P' /* Monto Total Proyecto */,
    'F' /* Monto Total Financiado */
));
```

Detalles Técnicos

- **Nombre del campo:** aplicar_porc_contrap_monto
- **Tipo de dato:** VARCHAR2(1)
- **Valores permitidos:**
 - 'P': Monto Total Proyecto
 - 'F': Monto Total Financiado
- **Restricción:** SGPCONVOC_APLICAR_PORC_CK asegura que solo se permitan valores válidos.

Cambios en la Capa de Presentación (JSF + PrimeFaces)

Código HTML/JSF

```
<!-- ¿Aplicar % Contrapartida Dinero al monto de? -->
<div class="col-lg-6">
    <div class="form-group">
        <p:outputLabel
            value="#{msg['gestion.configuracion.convocatorias.aplicarPorcentajeContrapartida.label']}"
            styleClass="col-lg-5 control-label"
            style="text-align: right"/>
```

```
for="aplicarPorcContrapMonto"/>
<div class="col-lg-5">
  <p:selectOneMenu id="aplicarPorcContrapMonto"
value="#{convocatoriaSessController.selected.aplicarPorcContrapMonto}">
    <f:selectItems
value="#{configApplController.listaAplicarPorcContrapartidaConvMonto}"/>
    </p:selectOneMenu>
  </div>
</div>
</div>
```

Descripción

- **Etiqueta de campo:** "¿Aplicar % Contrapartida Dinero al monto de?".
- **Componente usado:** p:selectOneMenu de PrimeFaces.
- **Controlador de sesión:** convocatoriaSessController → asocia el valor seleccionado al objeto selected.aplicarPorcContrapMonto.
- **Lista de opciones:** configApplController.listaAplicarPorcContrapartidaConvMonto (contiene los valores 'P' y 'F' con sus descripciones).

Cambios en la Capa de Negocio

- Se agregó la propiedad aplicarPorcContrapMonto en el objeto **Convocatoria**.
- La lógica de negocio podrá validar el valor para determinar sobre qué monto aplicar el cálculo de contrapartida.

Cambios en el Formulario de Proyectos

Contexto

Cuando un usuario diligencia un **proyecto** para una convocatoria:

- Se debe verificar que el porcentaje de contrapartida ingresado cumpla con la parametrización definida en la convocatoria (aplicar_porc_contrap_monto).
- El sistema debe rechazar la radicación si el cálculo no cumple con lo exigido.

Lógica de Validación

1. **Obtener configuración de la convocatoria:**

- Valor de aplicar_porcentaje_contrapartida_monto desde la tabla SGP_CONVOCATORIAS.

2. Calcular contrapartida según configuración:

- Si P: aplicar % sobre **Monto Total Proyecto**.
- Si F: aplicar % sobre **Monto Total Financiado**.

3. Comparar con los datos del proyecto ingresado:

- Validar que el monto de contrapartida en dinero cumple con el porcentaje mínimo exigido.

4. Resultado:

- Si cumple → permitir continuar con el envío.
- Si no cumple → mostrar mensaje de validación al usuario e impedir radicación.

Mensajes de Validación (ejemplo)

- Para P:
“El porcentaje de contrapartida en dinero debe calcularse sobre el Monto Total del Proyecto. Revise los valores ingresados.”
- Para F:
“El porcentaje de contrapartida en dinero debe calcularse sobre el Monto Total Financiado. Revise los valores ingresados.”

Impacto Técnico

- Ajustes en el **servicio de validación de proyectos** previo al radicado.
- Inclusión del nuevo atributo en los DTO/Entities que participan en la validación.
- Ajustes en la capa de negocio que determina la elegibilidad del proyecto.

- 17. Dar soporte, capacitación y actualizar la documentación de usuario en cuanto a la configuración de Gran Áreas por Convocatoria y Disciplina OCDE proyectos y beneficios tributarios. Registrar Código BPIN y Número de Contrato en proyectos de BT, configuración de Cantidad Máxima de Evaluaciones de Proyectos Asignadas que un Evaluador puede tener por Convocatoria, configuración de acceso a funcionarios a la Evaluación de Proyectos por Convocatoria, reclamaciones Panel Evaluación Proyectos, configuración de Matriz evaluación BT tamaño Empresa y a Resoluciones de Ingresos No Constitutivos de Renta. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)**

Solicitud de requerimiento No. 163316

Descripción y alcance del Requerimiento

Se debe brindar soporte, capacitación y actualización de la documentación de usuario sobre los siguientes procesos del módulo de convocatorias:

- Configuración de Gran Áreas por Convocatoria y Disciplina OCDE para proyectos y beneficios tributarios.
- Registro de Código BPIN y Número de Contrato en proyectos de beneficios tributarios (BT).
- Configuración de la Cantidad Máxima de Evaluaciones de Proyectos Asignadas que un evaluador puede tener por convocatoria.
- Configuración de acceso a funcionarios para la Evaluación de Proyectos por Convocatoria.
- Gestión de reclamaciones en el Panel de Evaluación de Proyectos.
- Configuración de la Matriz de Evaluación de BT según el tamaño de la empresa.
- Configuración asociada a Resoluciones de Ingresos No Constitutivos de Renta.
- El alcance incluye:
- Capacitación a usuarios funcionales sobre el uso de las nuevas y existentes funcionalidades.
- Actualización de manuales y guías de usuario en formato digital, asegurando que reflejen la operativa vigente.
- Ajustes y verificación de la correcta parametrización en el sistema para cada una de las configuraciones mencionadas.

Justificación

El impacto de los requerimientos radica en la mejora de la gestión y operación del sistema SIGP y de los módulos asociados a beneficios tributarios. En el caso de los ajustes de interfaz y funcionalidad, estos cambios permiten optimizar la experiencia del usuario, asegurando una mayor eficiencia en los procesos de consulta, registro y seguimiento de información. Por otra parte, las modificaciones en las resoluciones de beneficios tributarios generan un impacto directo en la claridad y pertinencia de los documentos emitidos, lo que fortalece la transparencia y la confiabilidad de los trámites ante los usuarios externos. Adicionalmente, los ajustes técnicos en la plataforma contribuyen a la estabilidad del sistema, reducen la posibilidad de errores operativos y garantizan un mejor desempeño general. En conjunto, los requerimientos implementados impulsan la modernización del sistema, mejorando tanto la usabilidad como la calidad de la información generada y gestionada.

Desarrollo e Implementación

El desarrollo de este requerimiento está asociada al soporte, capacitación y actualización de la documentación de usuario en el módulo de convocatorias y beneficios tributarios del Sistema de Gestión de Proyectos (SIGP).

En este manual se describen los aspectos técnicos implementados, la lógica de negocio aplicada y los ajustes realizados en el sistema, con el propósito de garantizar la correcta operación de los procesos funcionales asociados a convocatorias, evaluaciones y beneficios tributarios.

El desarrollo contempla la implementación y ajustes técnicos necesarios para soportar los siguientes procesos:

- Configuración de Gran Áreas por Convocatoria y Disciplina OCDE para proyectos y beneficios tributarios.
- Registro de Código BPIN y Número de Contrato en proyectos de beneficios tributarios.
- Configuración de la Cantidad Máxima de Evaluaciones de Proyectos Asignadas que un evaluador puede tener por convocatoria.
- Configuración de acceso a funcionarios para la Evaluación de Proyectos por Convocatoria.
- Gestión de reclamaciones en el Panel de Evaluación de Proyectos.
- Configuración de la Matriz de Evaluación de beneficios tributarios según el tamaño de la empresa.
- Configuración asociada a Resoluciones de Ingresos No Constitutivos de Renta.

El alcance incluye la verificación de la correcta parametrización en el sistema, la actualización de manuales y guías de usuario y la capacitación a usuarios funcionales sobre el uso de las funcionalidades descritas.

Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- **Java EE (Java EE (EJB)):** Gestión de la lógica de negocio para la obtención y validación de los parámetros configurados por matriz.
- **JSF + PrimeFaces:** Construcción de las pantallas de configuración y validación en tiempo real de los conceptos evaluativos.
- **JPA (Criteria API):** Integración con la capa de persistencia para el almacenamiento de los valores de límite por matriz.
- **PL/SQL:** Modificaciones en la base de datos para incluir los nuevos campos de configuración.
- ***Mensajes .properties:** Para la internacionalización de los mensajes de validación y retroalimentación en el formulario de evaluación.

Ajustes Resoluciones Beneficios Tributarios

El requerimiento consistió en ajustar las resoluciones de BT (Beneficios Tributarios) para tramites de beneficios tributarios por inversión, los cuales eran:

1. Crédito Fiscal por Inversiones en CTel
2. Deducción y Descuento por Inversiones en CTel

Crédito Fiscal por Inversiones en CTel

Las **resoluciones de crédito fiscal por inversiones en CTel** son documentos emitidos por las autoridades tributarias o entidades regulatorias que certifican el cumplimiento de los requisitos necesarios para acceder a este crédito fiscal. Estas resoluciones cumplen las siguientes funciones:

1. **Aprobar el crédito:** Certifican que el proyecto de inversión en CTel cumple con los criterios y estándares necesarios y autorizan la deducción o reducción del impuesto.

2. **Rechazar el credito:** En caso de no cumplir con los criterios, la resolución también puede denegar el crédito o solicitar información adicional para que se revise el proyecto.

Resolucion APROBADO

Sección de Parámetros

```
<parameter name="PROYECTO_ID" class="java.lang.String"/>
<parameter name="IMG" class="java.lang.String">
  <parameterDescription><![CDATA[]]></parameterDescription>
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="SUBREPORT_DIR" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_BASE64" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP_Completo" class="java.lang.String"/>
<parameter name="VIGENCIA" class="java.lang.Integer">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_SECUNDARIA_BASE64" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_vto_bno" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso2" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_proyecto" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
```

En esta sección se definen varios parámetros (<parameter>) que permiten la configuración dinámica del reporte. Algunos parámetros incluyen:

1. **PROYECTO_ID:** Identificador del proyecto, utilizado en la consulta para filtrar información específica.
2. **IMG, IMG_BASE64, IMG_SECUNDARIA_BASE64:** Campos para imágenes que pueden ser usadas en el reporte (base64 es un formato de codificación de imágenes).

3. **nombre_entidad_SIGP y nombre_entidad_SIGP_Completo:** Nombres de la entidad, que podrían usarse en encabezados o pies de página.
4. **VIGENCIA:** Año o periodo de vigencia del proyecto.
5. **func_vto_bno, func_reviso, func_reviso2:** Nombres de funcionarios que participaron en la revisión del proyecto.
6. **SUBREPORT_DIR:** Directorio donde se guardan los subreportes, permitiendo una estructura modular para el reporte principal.

Cada parámetro es de tipo **java.lang.String** o **java.lang.Integer** y se puede referenciar en el cuerpo del reporte.

Sección de Consulta (Query)

```
<queryString>
<![CDATA[SELECT (SELECT LENGTHB(detalle) AS length_in_bytes
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON') size_considerando
, NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
1),
'') AS considerando_parte1,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
4001),
'') AS considerando_parte2,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
8001),
'') AS considerando_parte3,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
12001),
'') AS considerando_parte4,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro =
'CREDINVOA1')),
```

```
'[[NUMERO_RESOLUCION]]',
p.numero_resolucion_exencion),
'[[ANIO_RESOLUCION]]',
p.ano_resolucion_exencion),
'[[TIPO_ENTIDAD]]',
lower(tent.descripcion_valor_definicion)),
'[[ENTIDAD]]',
e.nombre_entidad),
'[[NIT]]',
e.nit || '-' ||
e.digito_verificacion),
'[[TITULO]]',
upper(p.titulo)),
'[[CODIGO_PROYECTO]]',
p.codigo_proyecto),
'[[TIPO_PROYECTO]]',
vdtip.descripcion_valor_definicion),
'[[CONCEPTO_EVALUACION]]',
p.concepto_evaluacion),
'[[DIA_REUNION_ACTA]]',
to_char(acta.fecha_acta,
                                'DD')),
                                '[[MES_REUNION_ACTA]]',
to_char(acta.fecha_acta,
                                'Month',
                                'nls_date_language=spanish')),
                                '[[ANO_REUNION_ACTA]]',
to_char(acta.fecha_acta,
                                'YYYY')),
                                '[[NUMERO_ACTA]]',
acta.numero),
                                '[[ANO_ACTA]]',
acta.ano),
                                '[[PROGRAMA]]',
lower(d.nombre_dependencia)),
4000,
1) AS considerando_aprobado1,
```

```

dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle

FROM adm_parametros_sistema

WHERE codigo_parametro =

'CREDINVCOA2'),

'[[NUMERO_RESOLUCION]]',

p.numero_resolucion_exencion),

'[[ANIO_RESOLUCION]]',

p.ano_resolucion_exencion),

'[[TIPO_ENTIDAD]]',

lower(tent.descripcion_valor_definicion)),

'[[ENTIDAD]]',

e.nombre_entidad),

'[[NIT]]',

e.nit || '-' ||

e.digito_verificacion),

'[[TITULO]]',

upper(p.titulo)),

'[[CODIGO_PROYECTO]]',

p.codigo_proyecto),

'[[TIPO_PROYECTO]]',

vdtip.descripcion_valor_definicion),

'[[CONCEPTO_EVALUACION]]',

p.concepto_evaluacion),

'[[DIA_REUNION_ACTA]]',

to_char(acta.fecha_acta,

'DD')),

'[[MES_REUNION_ACTA]]',

to_char(acta.fecha_acta,

'Month',

'nls_date_language=spanish')),

'[[ANO_REUNION_ACTA]]',

```

```

to_char(acta.fecha_acta,
        'YYYY'),
        '[[NUMERO_ACTA]]',
        acta.numero),
        '[[ANO_ACTA]]',
        acta.anio),
        '[[PROGRAMA]]',
        lower(d.nombre_dependencia)),
        '[[PUNTAJE_PROMEDIO_EVALUACION]]',
        (SELECT nvl(AVG(eval.puntaje),
                     0)
         FROM pry_evaluaciones eval
         WHERE eval.proyecto_id = p.proyecto_id)),
        4000,
        1) AS considerando_aprobado2,
nvl(dbms_lob.substr(p.concepto_evaluacion,
                   4000,
                   1),
    '') concepto_evaluacion,
nvl(dbms_lob.substr(p.concepto_evaluacion,
                   8000,
                   4001),
    '') concepto_evaluacion2,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle
                                                             FROM adm_parametros_sistema
                                                             WHERE codigo_parametro =
                                                             'CREDINVAR1'),
                                                             '[[TITULO]]',
                                                             upper(p.titulo)),
                                                             '[[CODIGO_PROYECTO]]',
                                                             p.codigo_proyecto),
                                                             '[[TIPO_PROYECTO]]',
                                                             vdtip.descripcion_valor_definicion),
                                                             '[[ENTIDAD]]',
                                                             e.nombre_entidad),
                                                             '[[NIT]]',
                                                             e.nit || '-' || e.digito_verificacion),
                   4000,
                   1) AS articulo1,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle
                                                             FROM adm_parametros_sistema
                                                             WHERE codigo_parametro =
                                                             'CREDINVART2'),
                                                             '[[TITULO]]',
                                                             upper(p.titulo)),
                                                             '[[CODIGO_PROYECTO]]',
                                                             p.codigo_proyecto),
                                                             '[[TIPO_PROYECTO]]',
                                                             vdtip.descripcion_valor_definicion),
                                                             '[[ENTIDAD]]',
                                                             e.nombre_entidad),
                                                             '[[NIT]]',
                                                             e.nit || '-' || e.digito_verificacion),
                   4000,
                   1) AS articulo2,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT
detalle
                                                             FROM
adm_parametros_sistema
                                                             WHERE
codigo_parametro =

```

```
'CREDINVART'),

                                                                 '[[TITULO]]',

upper(p.titulo)),

'[[CODIGO_PROYECTO]]',

                                                                 p.codigo_proyecto),
                                                                 '[[TIPO_PROYECTO]]',

vdtip.descripcion_valor_definicion),

                                                                 '[[ENTIDAD]]',
                                                                 e.nombre_entidad),
                                                                 '[[NIT]]',
                                                                 e.nit || '-' ||
                                                                 e.digito_verificacion),
                                                                 '[[ANIO_ACTA]]',
                                                                 to_char(p.fecha_acta_decision,
                                                                 'yyyy'),
                                                                 '[[ENTIDAD_IDENTIFICADA_NIT]]',
                                                                 f_ent_proyecto_rubros_aprob(p.proyecto_id,
                                                                 ${VIGENCIA}
                                                                 )),

                                                                 4000,
                                                                 1) AS articulos,
dbms_lob.substr((SELECT detalle
                  FROM adm_parametros_sistema
                  WHERE codigo_parametro = 'NOM_PERS'),
                  4000,
                  1) AS nombre,
dbms_lob.substr((SELECT detalle
                  FROM adm_parametros_sistema
                  WHERE codigo_parametro = 'CARGO_PERS'),
                  4000,
                  1) AS cargo,
vdtip.descripcion_valor_definicion tipo_proyecto,
to_char(acta.anio,
        '9999') anio_acta,
acta.numero numero_acta,
(SELECT v.descripcion_valor_definicion
 FROM adm_valores_definiciones v
 WHERE v.tipo_definicion = 'CODIF_EST'
       AND v.valor_definicion = '070') formato_version,
dbms_lob.substr((SELECT detalle
                  FROM adm_parametros_sistema
                  WHERE codigo_parametro = 'CREDINVNORA'),
                  4000,
                  1) AS normatividad
FROM sgp_proyectos      p,
   ent_entidades        e,
   sb_dependencias      d,
   adm_valores_definiciones tent,
   adm_valores_definiciones vdtip,
   sgp_actas_decision   acta
WHERE e.entidad_id(+) = p.entidad_n_id
     AND d.dependencia_id(+) = p.dependencia_id
     AND tent.valor_definicion_id(+) = e.tipo_entidad_id
     AND vdtip.valor_definicion_id(+) = p.tipo_proyecto_id
     AND acta.acta_id(+) = p.acta_consejo_id
     AND p.proyecto_id = ${PROYECTO_ID}]]>
</queryString>
```

La consulta (<queryString>) está escrita en SQL y contiene varias subconsultas y funciones de Oracle para extraer y procesar datos específicos. Ejemplo de algunos puntos clave:

1. **Subconsulta de Texto de Considerando:** Utiliza dbms_lob.substr() para obtener fragmentos de texto largos desde el campo detalle de la tabla adm_parametros_sistema, con el código **CREDINVCON**. Esto permite dividir el texto en partes (considerando_parte1, considerando_parte2, etc.) para su uso en el reporte.
2. **Reemplazo de Placeholders:** La consulta realiza reemplazos de placeholders (como [[NUMERO_RESOLUCION]], [[ENTIDAD]], [[NIT]], etc.) utilizando funciones REPLACE(). Estos placeholders se sustituyen con valores específicos del proyecto (p.numero_resolucion_exencion, e.nombre_entidad, e.nit, etc.), que son valores provenientes de tablas relacionadas.
3. **Evaluación y Puntaje Promedio:** La consulta obtiene el puntaje promedio de las evaluaciones del proyecto con AVG(eval.puntaje), filtrando según el PROYECTO_ID.
4. **Variables de Acta y Normatividad:** Se obtienen datos del acta y de normatividad relevante para el reporte, como numero_acta y normatividad.

Sección de Campos (Fields)

```
<field name="SIZE_CONSIDERANDO" class="java.lang.Long"/>
<field name="CONSIDERANDO_PARTE1" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE2" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE3" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE4" class="java.lang.String"/>
<field name="CONSIDERANDO_APROBADO1" class="java.lang.String"/>
<field name="CONSIDERANDO_APROBADO2" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION2" class="java.lang.String"/>
<field name="ARTICULO1" class="java.lang.String"/>
<field name="ARTICULO2" class="java.lang.String"/>
<field name="ARTICULOS" class="java.lang.String"/>
<field name="NOMBRE" class="java.lang.String"/>
<field name="CARGO" class="java.lang.String"/>
<field name="TIPO_PROYECTO" class="java.lang.String"/>
<field name="ANIO_ACTA" class="java.lang.String"/>
<field name="NUMERO_ACTA" class="java.lang.String"/>
<field name="FORMATO_VERSION" class="java.lang.String"/>
<field name="NORMATIVIDAD" class="java.lang.String"/>
```

Los campos (<field>) definen las variables que se utilizarán en el diseño visual del reporte. Algunos campos destacados son:

1. **SIZE_CONSIDERANDO:** Longitud en bytes del texto completo en el campo detalle, lo cual podría ser útil para validar el tamaño de los datos mostrados.
2. **CONSIDERANDO_PARTE1** hasta **CONSIDERANDO_PARTE4:** Partes del texto del considerando que fueron separadas en la consulta.
3. **ARTICULO1, ARTICULO2, y ARTICULOS:** Artículos específicos en el reporte, secciones de la resolución o normativa.
4. **CONCEPTO_EVALUACION:** Detalle del concepto de evaluación del proyecto.
5. **FORMATO_VERSION:** Versión del formato, obtenida de adm_valores_definiciones.

Sección de Variables

```
<variable name="TOTAL_PAGINAS" class="java.lang.Integer">
  <variableExpression><![CDATA[{$V{PAGE_NUMBER}}]></variableExpression>
</variable>
```

Finalmente, la variable **TOTAL_PAGINAS** usa **{\$V{PAGE_NUMBER}}**, una variable interna de JasperReports, para mostrar el total de páginas del reporte.

Sección de Encabezado Pagina (PageHeader)

```
<pageHeader>
  <band height="58" splitType="Stretch">
    <textField>
      <reportElement x="150" y="0" width="182" height="31"
isRemoveLineWhenBlank="true" uuid="c21c3e5f-8264-455f-b2b0-5b786c836f7a">
        <printWhenExpression><![CDATA[{$V{PAGE_NUMBER}}
1]]></printWhenExpression>
      </reportElement>
      <textElement textAlignment="Center" verticalAlignment="Bottom">
        <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA["RESOLUCIÓN No. DE " +
{$P{VIGENCIA}}]></textFieldExpression>
    </textField>
    <image scaleImage="RealSize">
      <reportElement x="0" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="d1f062a8-2ce1-495e-8c0a-93ec77648de0"/>
      <imageExpression><![CDATA[{$P{IMG_SECUNDARIA_BASE64}} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_SECUNDARIA_BASE64}.getBytes()))
:
new
ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))]></imageExpression>
    </image>
    <textField isStretchWithOverflow="true">
      <reportElement positionType="Float" x="0" y="33" width="482"
height="25" isRemoveLineWhenBlank="true" uuid="2f5a455a-0e11-4946-b121-65a2c865f78a">
```



```

1]]></printWhenExpression>
                                <reportElement>
                                <textElement textAlignment="Center" verticalAlignment="Top"
markup="styled">
                                <font fontName="Arial" size="9" isBold="false"
isItalic="true" isStrikeThrough="false" isPdfEmbedded="false"/>
                                </textElement>
                                <textFieldExpression><![CDATA["Por la cual se califica un
proyecto como de "+ $F{TIPO_PROYECTO} +" y se otorga un Beneficio Tributario.\"
"]]></textFieldExpression>
                                </textField>
                                <image scaleImage="RealSize">
                                <reportElement x="332" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="bcc21b67-29cc-4068-ac94-35a64a343fcf"/>
                                <imageExpression><![CDATA[$P{IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))
: null]]></imageExpression>
                                </image>
                                </band>
                                </pageHeader>

```

La sección pageHeader está configurada para mostrar en las páginas donde **\$V{PAGE_NUMBER}** > 1. Explicamos el código anterior:

1. Condiciones de Impresión: La etiqueta

<printWhenExpression><![CDATA[\$V{PAGE_NUMBER} > 1]]></printWhenExpression>

en los elementos textField asegura que estos campos solo se muestren a partir de la segunda página. Esto es útil si quieres que el encabezado sea diferente en la primera página. Si se desea que algo se muestre en todas las páginas, se puede eliminar esta condición.

2. Imágenes Condicionales:

- La primera imagen se muestra si **\$P{IMG_SECUNDARIA_BASE64}** no es null; de lo contrario, utiliza **\$P{IMG_BASE64}**.
- La segunda imagen en la misma banda tiene una expresión condicional similar, pero su imageExpression solo muestra **\$P{IMG_BASE64}** si **\$P{IMG_SECUNDARIA_BASE64}** no es null.

3. Centrado de Texto:

- Los elementos **<textField>** utilizan **textAlignment="Center"**, lo cual está bien para centrar el texto en el espacio asignado. Para un resultado más preciso, verifica que el ancho y posición del campo sean consistentes con el diseño del encabezado.

4. Control de Tamaño y Desbordamiento:

- El segundo textField tiene el atributo **isStretchWithOverflow="true"**, lo cual permite que el campo se ajuste si el texto es más extenso que el espacio asignado. Esto es útil si el contenido de **{TIPO_PROYECTO}** puede variar en longitud, pero si encuentras problemas de alineación, podrías ajustar el ancho o el tamaño de fuente.

Sección de Detalles (Details)

La sección details en un informe de JasperReports es donde se generan las filas de datos del reporte. Cada vez que se itera sobre un conjunto de datos, como una lista o un conjunto de registros de una base de datos, se crea una nueva instancia de la sección details para cada registro. Aquí es donde se colocan los elementos visuales (como campos de texto, gráficos, tablas, etc.) que se llenarán con los datos correspondientes a cada registro.

1. Primer textField (Normatividad)

```
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
  <reportElement positionType="Float" x="0" y="95" width="482" height="15" uuid="8302d748-
dd87-4a32-afa9-bc7a5765e7f9"/>
  <textElement textAlignment="Center" verticalAlignment="Top" markup="html">
    <font fontName="Arial" size="11" isPdfEmbedded="false"/>
  </textElement>
  <textFieldExpression><![CDATA[{NORMATIVIDAD}.trim()]]></textFieldExpression>
</textField>
```

Este campo de texto muestra el contenido del campo **NORMATIVIDAD** y lo centra horizontalmente. La propiedad **isStretchWithOverflow="true"** permite que el campo crezca si el texto sobrepasa el tamaño especificado. Además, **isBlankWhenNull="true"** asegura que el campo aparezca vacío en lugar de mostrar null si el valor es nulo.

1. Segundo textField (Título de la Entidad)

```
<textField isStretchWithOverflow="true">
  <reportElement positionType="Float" x="0" y="59" width="482" height="30" uuid="36699d96-
812a-4905-b885-412e67d871d2"/>
  <textElement textAlignment="Center" verticalAlignment="Top" markup="styled">
    <font fontName="Arial" size="11" isBold="true" isPdfEmbedded="false"/>
  </textElement>
  <textFieldExpression><![CDATA["LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E INNOVACIÓN DEL
"+ {nombre_entidad_SIGP_Completo}.toUpperCase().trim()]]></textFieldExpression>
</textField>
```

Este campo muestra el texto en mayúsculas "LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E INNOVACIÓN DEL" seguido del nombre completo de la entidad nombre_entidad_SIGP_Completo. Se centra horizontalmente y utiliza una fuente en negrita.

2. Tercer textField (Descripción del Proyecto)

```
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
  <reportElement positionType="Float" x="0" y="32" width="482" height="20"
isRemoveLineWhenBlank="true" isPrintWhenDetailOverflows="true" uuid="385d34eb-1b0e-49d4-
a998-de01abb5302c"/>
  <textElement textAlignment="Center" verticalAlignment="Middle" markup="styled">
    <font fontName="Arial" size="10" isBold="false" isItalic="true"
isPdfEmbedded="false"/>
  </textElement>
  <textFieldExpression><![CDATA[" \Por la cual se califica un proyecto como de "+
${TIPO_PROYECTO} +" y se otorga el acceso a un beneficio tributario\"
"]]></textFieldExpression>
</textField>
```

Este campo muestra el tipo de proyecto en el formato "Por la cual se califica un proyecto como de [TIPO_PROYECTO] y se otorga el acceso a un beneficio tributario". Está centrado horizontalmente, en cursiva, y se ajusta si el texto es demasiado largo.

3. Cuarto textField (Número de Resolución y Vigencia)

```
<textField>
  <reportElement x="150" y="0" width="182" height="26" uuid="6f09484c-87df-4ad6-a727-
d6a3f443e1da"/>
  <textElement textAlignment="Center" verticalAlignment="Middle">
    <font fontName="Arial" size="11" isBold="true" isPdfEmbedded="false"/>
  </textElement>
  <textFieldExpression><![CDATA["RESOLUCIÓN No. DE " +
${VIGENCIA}]]></textFieldExpression>
</textField>
```

Este campo muestra el número de resolución y el año de vigencia, en el formato "**RESOLUCIÓN No. [número] DE [año]**". El texto está centrado horizontalmente y en negrita.

4. staticText (Título Considerando)

```
<staticText>
  <reportElement positionType="Float" x="0" y="5" width="482" height="15" uuid="64724d7c-
9256-41d5-b9d8-0df747de64e5"/>
  <textElement textAlignment="Center" verticalAlignment="Top">
    <font fontName="Arial" size="11" isBold="true" isPdfEmbedded="false"/>
  </textElement>
  <text><![CDATA[CONSIDERANDO:]]></text>
</staticText>
```

Este campo muestra el título "**CONSIDERANDO:**" en el documento, centrado y en negrita.

5. textField (Contenido de Considerando)

```
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
  <reportElement positionType="Float" stretchType="RelativeToBandHeight" x="0" y="22"
width="482" height="15" isPrintWhenDetailOverflows="true" uuid="adefeac8-c14e-4fc5-93d8-
c9c5f8bff339"/>
  <textElement textAlignment="Justified" verticalAlignment="Top" markup="html">
    <font fontName="Arial" size="11" isPdfEmbedded="false"/>
  </textElement>
  <textFieldExpression><![CDATA[ ${F{SIZE_CONSIDERANDO}} > 12000
?
${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_PARTE2}}).concat(${F{CONSIDERANDO_PARTE3}}).conc
at(${F{CONSIDERANDO_PARTE4}}).concat(${F{CONSIDERANDO_APROBADO1}})
: ${F{SIZE_CONSIDERANDO}} > 8000
?
${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_PARTE2}}).concat(${F{CONSIDERANDO_PARTE3}}).conc
at(${F{CONSIDERANDO_APROBADO1}})
: ${F{SIZE_CONSIDERANDO}} > 4000
? ${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_PARTE2}}).concat(${F{CONSIDERANDO_APROBADO1}})
: ${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_APROBADO1}})]></textFieldExpression>
</textField>
```

Este campo muestra el contenido del considerando en diferentes secciones (**CONSIDERANDO_PARTE1**, **CONSIDERANDO_PARTE2**, etc.) dependiendo de su tamaño (**SIZE_CONSIDERANDO**). El texto está justificado.

6. textField (Concepto de Evaluación)

```
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
  <reportElement positionType="Float" stretchType="RelativeToBandHeight" x="20" y="0"
width="442" height="15" isPrintWhenDetailOverflows="true" uuid="b541f2e0-8ac2-4564-97ab-
b498b2f63517"/>
  <textElement textAlignment="Justified" verticalAlignment="Top" markup="styled">
    <font fontName="Arial" size="11" isItalic="true" isUnderline="false"
isPdfEmbedded="false"/>
  </textElement>
  <textFieldExpression><![CDATA[ ${F{CONCEPTO_EVALUACION}} != null
?
(F{CONCEPTO_EVALUACION2} != null
?
"".concat(${F{CONCEPTO_EVALUACION}}).trim().concat(${F{CONCEPTO_EVALUACION2}}).trim().concat("
\").trim() : "".concat(${F{CONCEPTO_EVALUACION}}).trim().concat("").trim()
: "Sin registro de concepto de evaluación"]></textFieldExpression>
</textField>
```

Este campo muestra el **CONCEPTO_EVALUACION** y, si existe, **CONCEPTO_EVALUACION2**, o el mensaje "**Sin registro de concepto de evaluación**" si ambos son nulos.

7. subreport (Subreportes Artículos 1 y 2)

```
<subreport>
  <reportElement positionType="Float" x="0" y="1" width="482" height="20" uuid="4449cd6c-
8cd0-4984-8134-d5958981a31c"/>
  <subreportParameter name="VIGENCIA">

    <subreportParameterExpression><![CDATA[{P{VIGENCIA}}]></subreportParameterExpression>
  </subreportParameter>
  <subreportExpression><![CDATA[{P{SUBREPORT_DIR}
"entidadesVigenciasRecomendadoCredFiscalArt1.jasper"}]]></subreportExpression>
</subreport>

<subreport>
  <reportElement positionType="Float" x="0" y="1" width="482" height="20" uuid="21c1c26d-
c10c-4bd5-81f2-32cd99516064"/>
  <subreportParameter name="VIGENCIA">

    <subreportParameterExpression><![CDATA[{P{VIGENCIA}}]></subreportParameterExpression>
  </subreportParameter>
  <subreportExpression><![CDATA[{P{SUBREPORT_DIR}
"entidadesVigenciasRecomendadoCredFiscalArt2.jasper"}]]></subreportExpression>
</subreport>
```

Este bloque incluye dos subreportes para mostrar detalles adicionales. Ambos subreportes reciben parámetros como **VIGENCIA**, **SUBREPORT_DIR**, y **proyecto_id**, usando la conexión **REPORT_CONNECTION**.

Sección de Pie de Pagina (PageFooter)

```
<pageFooter>
  <band height="42" splitType="Stretch">
    <textField evaluationTime="Page">
      <reportElement x="203" y="22" width="50" height="20" forecolor="#999999"
uuid="3bfacc06-77c4-47f5-9f26-883213d073fc"/>
      <textElement>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA["Página " + ${PAGE_NUMBER} + " de
"]]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Report">
      <reportElement x="253" y="22" width="18" height="20" forecolor="#999999"
uuid="42ba6cac-8dc3-4ca5-a6af-ee42180180e8"/>
      <textElement>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA[${PAGE_NUMBER}]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Page">
      <reportElement x="0" y="0" width="112" height="30" forecolor="#999999"
uuid="5195747f-e33d-4169-8802-6a38bf19d755"/>
      <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textElement>
    </textField>
  </band>
</pageFooter>
```

```
<textFieldExpression><![CDATA[$F{FORMATO_VERSION}]]></textFieldExpression>
</textField>
</band>
</pageFooter>
```

En esta sección de **pageFooter**, hay varios elementos importantes:

1. Número de Página:

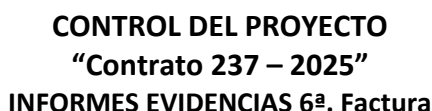
- El primer **textField** muestra el texto "Página " seguido del número de la página actual. Utiliza la expresión **`$V{PAGE_NUMBER}`** para obtener el número de la página actual.
- El segundo **textField** muestra el total de páginas utilizando también **`$V{PAGE_NUMBER}`**, lo cual puede ser útil si deseas mostrar "Página X de Y", aunque en este caso parece que no hay un total definido.

2. Formato de Versión:

- El tercer **textField** muestra el valor del campo **FORMATO_VERSION**, que puede ser un indicador de la versión del informe. Este campo se muestra en un formato de texto que se alinea verticalmente al centro.

Sección de Resumen (Summary)

```
<summary>
  <band height="161" splitType="Immediate">
    <staticText>
      <reportElement positionType="Float" x="0" y="0" width="482" height="15"
      uuid="8a85e609-d92c-4b1b-81bc-83b4a3347de9"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="Arial" size="11" isBold="true"
        isPdfEmbedded="false"/>
      </textElement>
      <text><![CDATA[NOTIFIQUESE Y CÚMPLASE,]]></text>
    </staticText>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
      <reportElement positionType="Float" x="0" y="73" width="482" height="15"
      uuid="157aa74f-f0d7-4ff9-b644-6fd620ed7cd2"/>
      <textElement textAlignment="Center" verticalAlignment="Middle"
      markup="styled">
        <font fontName="Arial" size="11" isBold="true"
        isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA[$F{CARGO}]]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
      <reportElement positionType="Float" x="0" y="58" width="482" height="15"
      uuid="b3c1a114-e661-4788-9a25-75c3dec6ee02"/>
```



```
<reportElement positionType="Float" x="0" y="151" width="37" height="10"
forecolor="#000000" uuid="bc68ce52-a367-4e48-b03f-45d08751fe10"/>
<textElement verticalAlignment="Middle">
    <font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
</textElement>
<text><![CDATA[Proyectó:]]></text>
</staticText>
<textField isBlankWhenNull="true">
    <reportElement positionType="Float" x="32" y="131" width="450"
height="10" uuid="cd14c6b5-85b8-4420-bafe-cbbf98a2e6a5"/>
    <textElement>
        <font fontName="Arial" size="7"/>
    </textElement>
    <textFieldExpression><![CDATA[ $P{func_reviso}]]></textFieldExpression>
</textField>
</band>
</summary>
```

La sección **summary** en un informe de JasperReports se utiliza para mostrar información de cierre o un resumen del contenido del informe, generalmente después de que se han presentado todos los detalles y datos en las secciones anteriores. Esta sección es útil para presentar conclusiones, notas finales o cualquier información importante que se quiera resaltar al final del documento.

En la sección de **summary**, hay varios elementos clave:

1. Texto Estático:

- El primer **staticText** muestra el mensaje "**NOTIFÍQUESE Y CÚMPLASE**," centrado y en negrita, lo que indica que es un aviso importante para el lector. Está ubicado en la parte superior de la sección y utiliza una fuente Arial de tamaño 11.

2. Nombre y Cargo:

- Se presentan campos dinámicos (**textField**) que muestran el nombre (**\$F{NOMBRE}**) y el cargo (**\$F{CARGO}**) de la persona que revisó el informe. Ambos están centrados y en negrita, asegurando que destaquen en la página.

3. Revisiones y Proyectos:

- Se incluyen etiquetas de texto estático y campos de texto que indican quién revisó y proyectó el informe, utilizando los parámetros **\$P{func_reviso}**, **\$P{func_reviso2}**, y **\$P{func_proyecto}**. Estos campos se presentan en una fuente más pequeña (tamaño 7), lo que indica que son detalles adicionales que

son relevantes pero que no necesitan el mismo nivel de atención que el encabezado.

4. Formato:

- Todos los textos estáticos y campos de texto utilizan la fuente Arial, y se asegura que la alineación sea consistente. Las etiquetas de texto tienen un color negro, mientras que los campos están en un tamaño más pequeño, lo que crea un contraste adecuado.

Resolución NEGADO

Sección de Parámetros

```
<parameter name="PROYECTO_ID" class="java.lang.String"/>
<parameter name="IMG" class="java.lang.String">
  <parameterDescription><![CDATA[]]></parameterDescription>
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="SUBREPORT_DIR" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_BASE64" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP_Completo" class="java.lang.String"/>
<parameter name="IMG_SECUNDARIA_BASE64" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_vto_bno" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso2" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_proyecto" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="VIGENCIA" class="java.lang.Integer">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
```

En esta sección se definen varios parámetros (<parameter>) que permiten la configuración dinámica del reporte. Algunos parámetros incluyen:

1. **PROYECTO_ID**: Identificador del proyecto, utilizado en la consulta para filtrar información específica.
2. **IMG, IMG_BASE64, IMG_SECUNDARIA_BASE64**: Campos para imágenes que pueden ser usadas en el reporte (base64 es un formato de codificación de imágenes).
3. **nombre_entidad_SIGP y nombre_entidad_SIGP_Completo**: Nombres de la entidad, que podrían usarse en encabezados o pies de página.
4. **VIGENCIA**: Año o periodo de vigencia del proyecto.
5. **func_vto_bno, func_reviso, func_reviso2**: Nombres de funcionarios que participaron en la revisión del proyecto.
6. **SUBREPORT_DIR**: Directorio donde se guardan los subreportes, permitiendo una estructura modular para el reporte principal.

Cada parámetro es de tipo **java.lang.String** o **java.lang.Integer** y se puede referenciar en el cuerpo del reporte.

Sección de Consulta (Query)

```
<queryString>
<![CDATA[SELECT (SELECT LENGTHB(detalle) AS length_in_bytes
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON') size_considerando
, NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
1),
'') AS considerando_parte1,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
4001),
'') AS considerando_parte2,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
8001),
'') AS considerando_parte3,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CREDINVCON'),
4000,
12001),
'') AS considerando_parte4,
```

```
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle

FROM adm_parametros_sistema

WHERE codigo_parametro =

'CREDINVCON1'),

'[[NUMERO_RESOLUCION]]',

p.numero_resolucion_exencion),

'[[ANIO_RESOLUCION]]',

p.ano_resolucion_exencion),

'[[TIPO_ENTIDAD]]',

lower(tent.descripcion_valor_definicion)),

'[[ENTIDAD]]',

e.nombre_entidad),

'[[NIT]]',

e.nit || '-' ||

e.digito_verificacion),

'[[TITULO]]',

upper(p.titulo)),
```

```
'[[CODIGO_PROYECTO]]',

p.codigo_proyecto),

                                                                    '[[TIPO_PROYECTO]]',

vdtip.descripcion_valor_definicion),

                                                                    '[[CONCEPTO_EVALUACION]]',
                                                                    p.concepto_evaluacion),

                                                                    '[[DIA_REUNION_ACTA]]',
                                                                    to_char(acta.fecha_acta,
                                                                    'DD')),

                                                                    '[[MES_REUNION_ACTA]]',
                                                                    to_char(acta.fecha_acta,
                                                                    'Month',
                                                                    'nls_date_language=spanish')),

                                                                    '[[ANO_REUNION_ACTA]]',
                                                                    to_char(acta.fecha_acta,
                                                                    'YYYY')),

'[[NUMERO_ACTA]]',

acta.numero),

                                                                    '[[ANO_ACTA]]',
                                                                    acta.anio),
                                                                    '[[PROGRAMA]]',
                                                                    lower(d.nombre_dependencia)),

                                                                    4000,
                                                                    1) AS considerando_negado1,

dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle

FROM adm_parametros_sistema

WHERE codigo_parametro =

                                                                    'CREDINVCON2')),
```

```
'[[NUMERO_RESOLUCION]]',  
  
p.numero_resolucion_exencion),  
  
'[[ANIO_RESOLUCION]]',  
  
p.ano_resolucion_exencion),  
  
'[[TIPO_ENTIDAD]]',  
  
lower(tent.descripcion_valor_definicion)),  
  
                                '[[ENTIDAD]]',  
  
                                e.nombre_entidad),  
  
                                '[[NIT]]',  
  
                                e.nit || '-' ||  
  
                                e.digito_verificacion),  
  
                                '[[TITULO]]',  
  
                                upper(p.titulo)),  
  
                                '[[CODIGO_PROYECTO]]',  
  
                                p.codigo_proyecto),  
  
'[[TIPO_PROYECTO]]',  
  
vdtip.descripcion_valor_definicion),  
  
'[[CONCEPTO_EVALUACION]]',
```

```

p.concepto_evaluacion),

                                '[[DIA_REUNION_ACTA]]',
                                to_char(acta.fecha_acta,
                                        'DD')),

                                '[[MES_REUNION_ACTA]]',
                                to_char(acta.fecha_acta,
                                        'Month',
                                        'nls_date_language=spanish')),

                                '[[ANO_REUNION_ACTA]]',
                                to_char(acta.fecha_acta,
                                        'YYYY')),

                                '[[NUMERO_ACTA]]',
                                acta.numero),

                                '[[ANO_ACTA]]',
                                acta.anio),
                                '[[PROGRAMA]]',

lower(d.nombre_dependencia)),

                                '[[PUNTAJE_PROMEDIO_EVALUACION]]',
                                (SELECT nvl(AVG(eval.puntaje),
                                        0)
                                FROM pry_evaluaciones eval
                                WHERE          eval.proyecto_id      =

p.proyecto_id)),

                                4000,
                                1) AS considerando_negado2,
                                nvl(dbms_lob.substr(p.concepto_evaluacion,
                                        4000,
                                        1),
                                '') concepto_evaluacion,
                                nvl(dbms_lob.substr(p.concepto_evaluacion,
                                        8000,
                                        4001),
                                '') concepto_evaluacion2,
                                dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle
                                        FROM adm_parametros_sistema
                                        WHERE codigo_parametro =
                                                'CREDINVARTN'),
                                '[[TITULO]]',
                                upper(p.titulo)),

```

```

        '[[CODIGO_PROYECTO]]',
        p.codigo_proyecto),
        '[[TIPO_PROYECTO]]',
        vdtip.descripcion_valor_definicion),
        '[[ENTIDAD]]',
        e.nombre_entidad),
        '[[NIT]]',
        e.nit || '-' ||
        e.digito_verificacion),
        '[[TIPO_PROYECTO]]',
        vdtip.descripcion_valor_definicion),
        4000,
        1) AS articulos,
        dbms_lob.substr((SELECT detalle
                        FROM adm_parametros_sistema
                        WHERE codigo_parametro = 'NOM_PERS'),
        4000,
        1) AS nombre,
        dbms_lob.substr((SELECT detalle
                        FROM adm_parametros_sistema
                        WHERE codigo_parametro = 'CARGO_PERS'),
        4000,
        1) AS cargo,
        vdtip.descripcion_valor_definicion tipo_proyecto,
        to_char(acta.anio,
        '9999') anio_acta,
        acta.numero numero_acta,
        (SELECT v.descripcion_valor_definicion
         FROM adm_valores_definiciones v
         WHERE v.tipo_definicion = 'CODIF_EST'
         AND v.valor_definicion = '080') formato_version,
        dbms_lob.substr((SELECT detalle
                        FROM adm_parametros_sistema
                        WHERE codigo_parametro = 'CREDINVNORN'),
        4000,
        1) AS normatividad
FROM sgp_proyectos      p,
   ent_entidades        e,
   sb_dependencias       d,
   adm_valores_definiciones tent,
   adm_valores_definiciones vdtip,
   sgp_actas_decision    acta
WHERE e.entidad_id      (+) = p.entidad_n_id
AND d.dependencia_id    (+) = p.dependencia_id
AND tent.valor_definicion_id (+) = e.tipo_entidad_id
AND vdtip.valor_definicion_id (+) = p.tipo_proyecto_id
AND acta.acta_id        (+) = p.acta_consejo_id
AND p.proyecto_id       = ${PROYECTO_ID}]]>
</queryString>

```

La consulta (**<queryString>**) está escrita en SQL y contiene varias subconsultas y funciones de Oracle para extraer y procesar datos específicos. Esta sección de la consulta SQL en el reporte está

diseñada para obtener información detallada sobre un proyecto específico desde varias tablas de la base de datos, utilizando el identificador del proyecto (**\$P{PROYECTO_ID}**). A continuación, se desglosan los principales elementos de esta consulta:

1. Selección de Secciones del Campo detalle:

- Se utiliza **dbms_lob.substr()** para dividir el campo detalle de la tabla **adm_parametros_sistema** en fragmentos de hasta 4000 caracteres, lo cual es útil para manejar texto largo en un campo LOB. Se seleccionan valores específicos de la columna detalle según el valor del parámetro **codigo_parametro**, que representa diferentes secciones de la información del proyecto (**CREDINVCON**, **CREDINVCON1**, **CREDINVCON2**, etc.).

2. Procesamiento de Texto y Reemplazo de Variables:

- En los campos considerando_negado1, considerando_negado2, y articulos, se reemplazan varios placeholders (**[[NUMERO_RESOLUCION]]**, **[[TITULO]]**, **[[ENTIDAD]]**, etc.) dentro del texto con valores específicos relacionados al proyecto o la entidad. Esto se logra mediante el uso de **REPLACE()**, sustituyendo cada marcador de posición con datos actuales como el número de resolución, título del proyecto, nombre de la entidad, entre otros.

3. Campos Adicionales de Información del Proyecto:

- Se extraen datos adicionales del proyecto y sus atributos, como:
 - **concepto_evaluacion:** Información relacionada con la evaluación del proyecto.
 - **nombre y cargo:** Nombre y cargo de una persona relevante, obtenidos de **adm_parametros_sistema**.
 - **tipo_proyecto:** Tipo de proyecto, obtenido de **adm_valores_definiciones**.
 - **anio_acta y numero_acta:** Año y número del acta de decisión asociado con el proyecto.
 - **formato_version:** Formato de versión de estado, que se obtiene a través de una subconsulta sobre **adm_valores_definiciones**.

- **normatividad:** Información normativa relacionada con el proyecto.

4. Condiciones de Unión y Filtro:

- Se utilizan varias uniones externas (LEFT JOIN) para vincular las tablas **sgp_proyectos**, **ent_entidades**, **sb_dependencias**, **adm_valores_definiciones**, y **sgp_actas_decision**, con la intención de obtener información detallada sobre el proyecto, la entidad a la que pertenece, la dependencia involucrada, el tipo de proyecto y el acta de decisión.
- La consulta se filtra específicamente para el proyecto identificado por **\$P{PROYECTO_ID}**, de modo que solo devuelve los detalles de ese proyecto en particular.

Sección de Campos (Fields)

```
<field name="SIZE_CONSIDERANDO" class="java.lang.Long"/>
<field name="CONSIDERANDO_PARTE1" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE2" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE3" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE4" class="java.lang.String"/>
<field name="CONSIDERANDO_NEGADO1" class="java.lang.String"/>
<field name="CONSIDERANDO_NEGADO2" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION2" class="java.lang.String"/>
<field name="ARTICULOS" class="java.lang.String"/>
<field name="NOMBRE" class="java.lang.String"/>
<field name="CARGO" class="java.lang.String"/>
<field name="TIPO_PROYECTO" class="java.lang.String"/>
<field name="ANIO_ACTA" class="java.lang.String"/>
<field name="NUMERO_ACTA" class="java.lang.String"/>
<field name="FORMATO_VERSION" class="java.lang.String"/>
<field name="NORMATIVIDAD" class="java.lang.String"/>
```

Esta sección define los campos que se utilizarán en el reporte y especifica el tipo de datos de cada campo. Estos campos son mapeos directos a los resultados de la consulta SQL presentada anteriormente. A continuación, se describe la funcionalidad de cada campo:

1. **SIZE_CONSIDERANDO** (java.lang.Long): Almacena el tamaño en bytes del texto completo asociado al código CREDINVCON en la tabla adm_parametros_sistema. Esto indica la longitud del texto de referencia.
2. **CONSIDERANDO_PARTE1, CONSIDERANDO_PARTE2, CONSIDERANDO_PARTE3, CONSIDERANDO_PARTE4** (java.lang.String): Cada uno de estos campos contiene un fragmento de hasta 4000 caracteres de la sección detalle de CREDINVCON. Esto permite

manejar texto largo al dividirlo en partes para que pueda ser procesado y mostrado en el reporte.

3. **CONSIDERANDO_NEGADO1 y CONSIDERANDO_NEGADO2** (java.lang.String): Campos que contienen partes del texto del campo detalle de CREDINVCON1 y CREDINVCON2, después de realizar reemplazos de marcadores de posición (placeholders) con datos específicos del proyecto y la entidad.
4. **CONCEPTO_EVALUACION y CONCEPTO_EVALUACION2** (java.lang.String): Contienen la descripción del concepto de evaluación asociado al proyecto. En algunos casos, el texto puede dividirse en varias partes si excede los 4000 caracteres.
5. **ARTICULOS** (java.lang.String): Contiene el texto del campo detalle de CREDINVARTN, con reemplazos de marcadores por información relevante del proyecto, como título, código, tipo de proyecto, nombre de la entidad y otros.
6. **NOMBRE y CARGO** (java.lang.String): Estos campos almacenan el nombre y el cargo de una persona relevante, obtenidos del sistema de parámetros de adm_parametros_sistema.
7. **TIPO_PROYECTO** (java.lang.String): Describe el tipo de proyecto según la definición asociada al proyecto actual.
8. **ANIO_ACTA y NUMERO_ACTA** (java.lang.String): Almacenan el año y el número del acta de decisión correspondiente al proyecto, lo cual es clave para la documentación y referencia del proyecto.
9. **FORMATO_VERSION** (java.lang.String): Define el formato de versión del estado, obtenido de los valores definidos en el sistema y aplicable al proyecto.
10. **NORMATIVIDAD** (java.lang.String): Contiene detalles normativos asociados al proyecto, obtenidos del parámetro CREDINVNORN.

Estos campos facilitan la manipulación y visualización de los datos del proyecto dentro del reporte, cada uno en su tipo adecuado (String para texto y Long para valores numéricos).

Sección de Variables

```
<variable name="TOTAL_PAGINAS" class="java.lang.Integer">  
  <variableExpression><![CDATA[${PAGE_NUMBER}]]></variableExpression>  
</variable>
```

Finalmente, la variable **TOTAL_PAGINAS** usa **$\$V\{PAGE_NUMBER\}$** , una variable interna de JasperReports, para mostrar el total de páginas del reporte.

Sección de Encabezado Pagina (PageHeader)

```
<pageHeader>
  <band height="58" splitType="Stretch">
    <textField>
      <reportElement x="150" y="0" width="182" height="31"
isRemoveLineWhenBlank="true" uuid="f8614e4d-50cc-4298-9419-963db51db2b1">
        <printWhenExpression><![CDATA[ $\$V\{PAGE\_NUMBER\}$ 
1]]></printWhenExpression>
      </reportElement>
      <textField textAlignment="Center" verticalAlignment="Bottom">
        <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
      </textField>
      <textFieldExpression><![CDATA["RESOLUCIÓN No. DE " +
 $\$P\{VIGENCIA\}$ ]]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true">
      <reportElement positionType="Float" x="0" y="33" width="482" height="25"
isRemoveLineWhenBlank="true" uuid="c67a5398-6533-4f34-883c-e62e88eddc65">
        <printWhenExpression><![CDATA[ $\$V\{PAGE\_NUMBER\}$ 
1]]></printWhenExpression>
      </reportElement>
      <textField textAlignment="Center" verticalAlignment="Top"
markup="styled">
        <font fontName="Arial" size="9" isBold="false" isItalic="true"
isStrikeThrough="false" isPdfEmbedded="false"/>
      </textField>
      <textFieldExpression><![CDATA[" \Por la cual se niega la calificación a
un proyecto como de " +  $\$F\{TIPO\_PROYECTO\}$  + " \ " ]]]></textFieldExpression>
    </textField>
    <image scaleImage="RealSize">
      <reportElement x="0" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="6bb89b95-dc70-47d1-b93c-6ef3618b115a"/>
      <imageExpression><![CDATA[ $\$P\{IMG\_SECUNDARIA\_BASE64\}$  != null
? new ByteArrayInputStream(Base64.decodeBase64( $\$P\{IMG\_SECUNDARIA\_BASE64\}$ ).getBytes()))
:
new ByteArrayInputStream(Base64.decodeBase64( $\$P\{IMG\_BASE64\}$ ).getBytes()))]]></imageExpression>
    </image>
    <image scaleImage="RealSize">
      <reportElement x="332" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="5f7fdf40-934e-4e18-9b4a-a5d01a58b116"/>
      <imageExpression><![CDATA[ $\$P\{IMG\_SECUNDARIA\_BASE64\}$  != null
? new ByteArrayInputStream(Base64.decodeBase64( $\$P\{IMG\_BASE64\}$ ).getBytes()))
: null]]></imageExpression>
    </image>
  </band>
</pageHeader>
```

La sección pageHeader está configurada para mostrar en las páginas donde **$\$V\{PAGE_NUMBER\}$** > 1. Explicamos el código anterior:

1. **Condiciones de Impresión:** La etiqueta

`<printWhenExpression><![CDATA[$V{PAGE_NUMBER} > 1]]></printWhenExpression>`

en los elementos `textField` asegura que estos campos solo se muestren a partir de la segunda página. Esto es útil si quieres que el encabezado sea diferente en la primera página. Si se desea que algo se muestre en todas las páginas, se puede eliminar esta condición.

2. **Imágenes Condicionales:**

- La primera imagen se muestra si `$P{IMG_SECUNDARIA_BASE64}` no es null; de lo contrario, utiliza `$P{IMG_BASE64}`.
- La segunda imagen en la misma banda tiene una expresión condicional similar, pero su `imageExpression` solo muestra `$P{IMG_BASE64}` si `$P{IMG_SECUNDARIA_BASE64}` no es null.

3. **Centrado de Texto:**

- Los elementos `<textField>` utilizan `textAlignment="Center"`, lo cual está bien para centrar el texto en el espacio asignado. Para un resultado más preciso, verifica que el ancho y posición del campo sean consistentes con el diseño del encabezado.

4. **Control de Tamaño y Desbordamiento:**

- El segundo `textField` tiene el atributo `isStretchWithOverflow="true"`, lo cual permite que el campo se ajuste si el texto es más extenso que el espacio asignado. Esto es útil si el contenido de `$F{TIPO_PROYECTO}` puede variar en longitud, pero si encuentras problemas de alineación, podrías ajustar el ancho o el tamaño de fuente.

Sección de Detalles (Details)

La sección `details` en un informe de JasperReports es donde se generan las filas de datos del reporte. Cada vez que se itera sobre un conjunto de datos, como una lista o un conjunto de registros de una base de datos, se crea una nueva instancia de la sección `details` para cada registro. Aquí es donde se colocan los elementos visuales (como campos de texto, gráficos, tablas, etc.) que se llenarán con los datos correspondientes a cada registro.

```
<detail>
  <band height="114" splitType="Stretch">
```

```

        <textField isStretchWithOverflow="true" isBlankWhenNull="true">
            <reportElement positionType="Float" x="0" y="32" width="482" height="20"
isRemoveLineWhenBlank="true" isPrintWhenDetailOverflows="true" uuid="4e48b21b-b018-492b-
a9ee-9232a9b8ce97"/>
            <textElement textAlignment="Center" verticalAlignment="Middle"
markup="styled">
                <font fontName="Arial" size="10" isBold="false" isItalic="true"
isPdfEmbedded="false"/>
            </textElement>
            <textFieldExpression><![CDATA[" \ "Por la cual se niega la calificación a
un proyecto como de "+ ${TIPO_PROYECTO} + " \ " ]]></textFieldExpression>
            </textField>
            <textField isStretchWithOverflow="true">
                <reportElement positionType="Float" x="0" y="59" width="482" height="30"
uuid="03a93dcf-f70b-44a0-b3e9-ba35d04c1dc2"/>
                <textElement textAlignment="Center" verticalAlignment="Top"
markup="styled">
                    <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
                </textElement>
                <textFieldExpression><![CDATA["LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E
INNOVACIÓN DEL "+
${nombre_entidad_SIGP_Completo}.toUpperCase().trim()]]></textFieldExpression>
            </textField>
            <textField>
                <reportElement x="150" y="0" width="182" height="26" uuid="2eb37195-
acdb-4b8d-aa2b-2b7e4ebd1bbb"/>
                <textElement textAlignment="Center" verticalAlignment="Middle">
                    <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
                </textElement>
                <textFieldExpression><![CDATA["RESOLUCIÓN No. DE " +
${VIGENCIA}]]></textFieldExpression>
            </textField>
            <textField isStretchWithOverflow="true" isBlankWhenNull="true">
                <reportElement positionType="Float" x="0" y="95" width="482" height="15"
uuid="220c49e8-9c00-400e-b720-aa986525cdf6"/>
                <textElement textAlignment="Center" verticalAlignment="Top"
markup="html">
                    <font fontName="Arial" size="11" isPdfEmbedded="false"/>
                </textElement>

                <textFieldExpression><![CDATA[${NORMATIVIDAD}.trim()]]></textFieldExpression>
            </textField>
        </band>
        <band height="37">
            <staticText>
                <reportElement positionType="Float" x="0" y="5" width="482" height="15"
uuid="e5546bfd-0044-46a9-985b-3a425d094e15"/>
                <textElement textAlignment="Center" verticalAlignment="Top">
                    <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
                </textElement>
                <text><![CDATA[CONSIDERANDO: ]]></text>
            </staticText>
            <textField isStretchWithOverflow="true" isBlankWhenNull="true">
                <reportElement positionType="Float" stretchType="RelativeToBandHeight"
x="0" y="22" width="482" height="15" isPrintWhenDetailOverflows="true" uuid="d1330410-662f-
41db-9996-9223bc068732"/>
                <textElement textAlignment="Justified" verticalAlignment="Top"
markup="html">

```

```

        <font fontName="Arial" size="11" isPdfEmbedded="false"/>
    </textElement>
    <textFieldExpression><![CDATA[{$F{SIZE_CONSIDERANDO} > 12000
?
{$F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_PARTE2}).concat($F{CONSIDERANDO_PARTE3}).conc
at($F{CONSIDERANDO_PARTE4}).concat($F{CONSIDERANDO_NEGADO1})
: {$F{SIZE_CONSIDERANDO} > 8000
?
{$F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_PARTE2}).concat($F{CONSIDERANDO_PARTE3}).conc
at($F{CONSIDERANDO_NEGADO1})
: {$F{SIZE_CONSIDERANDO} > 4000
? {$F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_PARTE2}).concat($F{CONSIDERANDO_NEGADO1})
: {$F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_NEGADO1})}]]></textFieldExpression>
    </textField>
</band>
<band height="15">
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
        <reportElement positionType="Float" x="20" y="0" width="442" height="15"
uuid="08673754-f3bb-42e6-bd91-ecc3cba2c104"/>
        <textElement textAlignment="Justified" verticalAlignment="Top"
markup="styled">
            <font fontName="Arial" size="11" isItalic="true"
isUnderline="false"/>
            </textElement>
            <textFieldExpression><![CDATA[{$F{CONCEPTO_EVALUACION} != null
?
($F{CONCEPTO_EVALUACION2} != null
?
"\\".concat($F{CONCEPTO_EVALUACION}).trim().concat($F{CONCEPTO_EVALUACION2}).trim().concat("
\").trim() : "\\".concat($F{CONCEPTO_EVALUACION}).trim().concat("\").trim())
: "Sin registro de concepto de evaluación"}]]></textFieldExpression>
            </textField>
        </band>
<band height="15">
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
        <reportElement positionType="Float" x="0" y="0" width="482" height="15"
uuid="25c2a1cf-bc9c-4e11-8d0b-177aaebcacf"/>
        <textElement textAlignment="Justified" verticalAlignment="Top"
markup="html">
            <font fontName="Arial" size="11"/>
            </textElement>

    <textFieldExpression><![CDATA[{$F{CONSIDERANDO_NEGADO2}.trim()}]]></textFieldExpression>
    </textField>
</band>
<band height="84">
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
        <reportElement positionType="Float" x="0" y="20" width="482" height="15"
uuid="b565131a-1cc4-4097-9830-44885613330e"/>
        <textElement textAlignment="Justified" markup="html">
            <font fontName="Arial" size="11"/>
            </textElement>

    <textFieldExpression><![CDATA[{$F{ARTICULOS}.trim()}]]></textFieldExpression>
    </textField>
    <staticText>
        <reportElement positionType="Float" x="0" y="0" width="482" height="15"
uuid="642f3a61-ed60-4f8c-8501-49f0d35e594d"/>
        <textElement textAlignment="Center">
            <font fontName="Arial" size="11" isBold="true"/>
            </textElement>
        <text><![CDATA[RESUELVE:]]></text>
    </staticText>

```

```
<textField>
  <reportElement positionType="Float" x="0" y="51" width="482" height="15"
  uuid="61955eb0-22b7-4ff2-9a59-0b9741a20dfd"/>
  <textElement>
    <font fontName="Arial" size="11"/>
  </textElement>
  <textFieldExpression><![CDATA["Dada en Bogotá, D.C., a los,    días del
mes de    de "+ Calendar.getInstance().get(Calendar.YEAR)]]></textFieldExpression>
</textField>
</band>
</detail>
```

1. Etiqueta <detail>:

- Contiene la sección principal del reporte donde se especifican los detalles a mostrar.
- En su interior, se usan etiquetas <band>, que dividen el reporte en bandas o secciones.

2. Primer <band> (altura="114"):

- Define una banda con una altura de 114 píxeles. La propiedad splitType="Stretch" permite que la banda se estire si el contenido ocupa más espacio.
- **Primer <textField>:**
 - Este campo de texto tiene las propiedades isStretchWithOverflow="true" y isBlankWhenNull="true", lo que significa que el texto se expandirá en caso de desbordamiento y no mostrará nada si el valor es null.
 - **<reportElement>** define la posición y dimensiones del campo:
 - positionType="Float": permite que el campo flote y se ajuste según el contenido.
 - x="0", y="32", width="482", height="20": establece la posición (x,y) y el tamaño del campo.
 - isRemoveLineWhenBlank="true" elimina la línea si el texto está vacío.

- `isPrintWhenDetailOverflows="true"` permite que el campo se imprima cuando se desborda.
 - **<textElement>**: define el estilo del texto:
 - `textAlignment="Center"`, `verticalAlignment="Middle"`: alinea el texto al centro tanto horizontal como verticalmente.
 - El tipo de fuente es Arial, tamaño 10, sin negrita y en cursiva (`isItalic="true"`).
 - **<textFieldExpression>**: contiene la expresión de texto para este campo:
 - Usa el campo TIPO_PROYECTO y lo inserta en la frase "Por la cual se niega la calificación a un proyecto como de ...".
 - **Segundo <textField>**:
 - Similar al anterior, pero:
 - Alineado en la posición y y="59" y altura de 30.
 - Usa `textAlignment="Center"`, `verticalAlignment="Top"`, con Arial tamaño 11 y negrita.
 - **<textFieldExpression>**: expresión que concatena la cadena "LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E INNOVACIÓN DEL" con `nombre_entidad_SIGP_Completo` en mayúsculas.
3. **Otros Campos de <textField> y <staticText> en <band>**:
- **<textField> para "RESOLUCIÓN No."**:
 - Campo en la posición x="150", y="0" con dimensiones width="182", height="26".
 - La expresión incluye "RESOLUCIÓN No. ... DE ..." y añade VIGENCIA.
 - **<textField> de "NORMATIVIDAD"**:
 - Campo que utiliza el valor de NORMATIVIDAD y lo coloca centrado en una altura de 15 píxeles.

4. Segundo <band> (altura="37"):

- Muestra el título "CONSIDERANDO:" en un campo <staticText>, centrado y en negrita.
- **Campo de texto justificado para los párrafos "CONSIDERANDO":**
 - Utiliza una lógica condicional para decidir qué partes del considerando mostrar, dependiendo del valor de SIZE_CONSIDERANDO. Usa operadores concat() para combinar secciones como CONSIDERANDO_PARTE1, CONSIDERANDO_NEGADO1, etc.

5. Tercer <band> y siguientes:

- **Campo para "CONCEPTO_EVALUACION":**
 - Justificado, con una expresión que verifica si CONCEPTO_EVALUACION y CONCEPTO_EVALUACION2 tienen valor y los concatena, o muestra "Sin registro de concepto de evaluación".
- **Otros campos** incluyen textos adicionales como CONSIDERANDO_NEGADO2 y ARTICULOS, cada uno con sus estilos específicos de alineación y fuente.
- Finalmente, un <textField> con una expresión que muestra una fecha basada en el año actual obtenido por Calendar.

Sección de Pie de Pagina (PageFooter)

```
<pageFooter>
  <band height="42" splitType="Stretch">
    <textField evaluationTime="Page">
      <reportElement x="203" y="22" width="50" height="20" forecolor="#999999"
        uuid="3bfacc06-77c4-47f5-9f26-883213d073fc"/>
      <textElement>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA["Página " + ${PAGE_NUMBER} + " de
    "]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Report">
      <reportElement x="253" y="22" width="18" height="20" forecolor="#999999"
        uuid="42ba6cac-8dc3-4ca5-a6af-ee42180180e8"/>
      <textElement>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA[${PAGE_NUMBER}]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Page">
```

```
<reportElement x="0" y="0" width="112" height="30" forecolor="#999999"
uuid="5195747f-e33d-4169-8802-6a38bf19d755"/>
<textElement verticalAlignment="Middle">
  <font fontName="Arial" size="8" isPdfEmbedded="false"/>
</textElement>

<textFieldExpression><![CDATA[ $F{FORMATO_VERSION} ]]></textFieldExpression>
</textField>
</band>
</pageFooter>
```

En esta sección de **pageFooter**, hay varios elementos importantes:

1. Número de Página:

- El primer **textField** muestra el texto "Página " seguido del número de la página actual. Utiliza la expresión **\$V{PAGE_NUMBER}** para obtener el número de la página actual.
- El segundo **textField** muestra el total de páginas utilizando también **\$V{PAGE_NUMBER}**, lo cual puede ser útil si deseas mostrar "Página X de Y", aunque en este caso parece que no hay un total definido.

2. Formato de Versión:

- El tercer **textField** muestra el valor del campo **FORMATO_VERSION**, que puede ser un indicador de la versión del informe. Este campo se muestra en un formato de texto que se alinea verticalmente al centro.

Sección de Resumen (Summary)

```
<summary>
  <band height="161" splitType="Immediate">
    <staticText>
      <reportElement positionType="Float" x="0" y="0" width="482" height="15"
uuid="8a85e609-d92c-4b1b-81bc-83b4a3347de9"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
      </textElement>
      <text><![CDATA[NOTIFÍQUESE Y CÚMPLASE,]]></text>
    </staticText>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
      <reportElement positionType="Float" x="0" y="73" width="482" height="15"
uuid="157aa74f-f0d7-4ff9-b644-6fd620ed7cd2"/>
      <textElement textAlignment="Center" verticalAlignment="Middle"
markup="styled">
        <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
      </textElement>
```

```

        <textFieldExpression><![CDATA[${CARGO}]]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
        <reportElement positionType="Float" x="0" y="58" width="482" height="15"
        uuid="b3c1a114-e661-4788-9a25-75c3dec6ee02"/>
        <textFieldExpression><![CDATA[${NOMBRE}]]></textFieldExpression>
    </textField>
    <staticText>
        <reportElement positionType="Float" x="0" y="141" width="32" height="10"
        forecolor="#000000" uuid="b071c509-e7cc-4edf-bfe4-c88a3f330116"/>
        <textField verticalAlignment="Middle">
            <font fontName="Arial" size="7" isBold="false"
            isPdfEmbedded="false"/>
        </textField>
        <text><![CDATA[Revisó:]]></text>
    </staticText>
    <textField isBlankWhenNull="true">
        <reportElement positionType="Float" x="32" y="121" width="450"
        height="10" uuid="1d042795-8528-4913-9b26-534fc957ca7c"/>
        <textField>
            <font fontName="Arial" size="7"/>
        </textField>
        <textFieldExpression><![CDATA[${func_vto_bno}]]></textFieldExpression>
    </textField>
    <staticText>
        <reportElement positionType="Float" x="0" y="121" width="32" height="10"
        forecolor="#000000" uuid="58af6100-17e6-4d48-84d6-ea40f6ae2a2a"/>
        <textField verticalAlignment="Middle">
            <font fontName="Arial" size="7" isBold="false"
            isPdfEmbedded="false"/>
        </textField>
        <text><![CDATA[Vto. Bo.:]]></text>
    </staticText>
    <textField isBlankWhenNull="true">
        <reportElement positionType="Float" x="37" y="151" width="445"
        height="10" uuid="bf058a34-7f9e-4349-a9bf-e4dee44f0853"/>
        <textField>
            <font fontName="Arial" size="7"/>
        </textField>

    <textFieldExpression><![CDATA[${func_proyecto}]]></textFieldExpression>
    </textField>
    <staticText>
        <reportElement positionType="Float" x="0" y="131" width="32" height="10"
        forecolor="#000000" uuid="bac943b2-4df1-4a7e-936e-dc1449755916"/>
        <textField verticalAlignment="Middle">
            <font fontName="Arial" size="7" isBold="false"
            isPdfEmbedded="false"/>
        </textField>
        <text><![CDATA[Revisó:]]></text>
    </staticText>
    <textField isBlankWhenNull="true">
        <reportElement positionType="Float" x="32" y="141" width="450"
        height="10" uuid="34c87439-9a45-45c6-b073-d197aabdace6"/>
        <textField>
            <font fontName="Arial" size="7"/>

```

```

        </textElement>
        <textFieldExpression><![CDATA[$P{func_reviso2}]]></textFieldExpression>
    </textField>
    <staticText>
        <reportElement positionType="Float" x="0" y="151" width="37" height="10"
        forecolor="#000000" uuid="bc68ce52-a367-4e48-b03f-45d08751fe10"/>
        <textElement verticalAlignment="Middle">
            <font fontName="Arial" size="7" isBold="false"
            isPdfEmbedded="false"/>
        </textElement>
        <text><![CDATA[Proyectó:]]></text>
    </staticText>
    <textField isBlankWhenNull="true">
        <reportElement positionType="Float" x="32" y="131" width="450"
        height="10" uuid="cd14c6b5-85b8-4420-bafe-cbbf98a2e6a5"/>
        <textElement>
            <font fontName="Arial" size="7"/>
        </textElement>
        <textFieldExpression><![CDATA[$P{func_reviso}]]></textFieldExpression>
    </textField>
</band>
</summary>

```

La sección **summary** en un informe de JasperReports se utiliza para mostrar información de cierre o un resumen del contenido del informe, generalmente después de que se han presentado todos los detalles y datos en las secciones anteriores. Esta sección es útil para presentar conclusiones, notas finales o cualquier información importante que se quiera resaltar al final del documento.

En la sección de **summary**, hay varios elementos clave:

1. **Texto Estático:**

- El primer **staticText** muestra el mensaje "**NOTIFÍQUESE Y CÚMPLASE,**" centrado y en negrita, lo que indica que es un aviso importante para el lector. Está ubicado en la parte superior de la sección y utiliza una fuente Arial de tamaño 11.

2. **Nombre y Cargo:**

- Se presentan campos dinámicos (**textField**) que muestran el nombre (**\$F{NOMBRE}**) y el cargo (**\$F{CARGO}**) de la persona que revisó el informe. Ambos están centrados y en negrita, asegurando que destaquen en la página.

3. **Revisiones y Proyectos:**

- Se incluyen etiquetas de texto estático y campos de texto que indican quién revisó y proyectó el informe, utilizando los parámetros **\$P{func_reviso}**, **\$P{func_reviso2}**, y **\$P{func_proyecto}**. Estos campos se presentan en una

fuerza más pequeña (tamaño 7), lo que indica que son detalles adicionales que son relevantes pero que no necesitan el mismo nivel de atención que el encabezado.

4. Formato:

- Todos los textos estáticos y campos de texto utilizan la fuente Arial, y se asegura que la alineación sea consistente. Las etiquetas de texto tienen un color negro, mientras que los campos están en un tamaño más pequeño, lo que crea un contraste adecuado.

Deducción y Descuento por Inversiones en CTel

Las **resoluciones de Deducción y Descuento por Inversiones en Ciencia, Tecnología e Innovación (CTel)** son documentos que autorizan a una empresa a beneficiarse de ciertos incentivos tributarios por haber realizado inversiones en proyectos de CTel. Estos incentivos pueden presentarse en dos formas: deducciones y descuentos sobre el impuesto de renta.

Estos incentivos son dos formas de reducir la carga tributaria de una empresa en función de las inversiones que ha realizado en proyectos de CTel:

1. **Deducción por Inversiones en CTel:** Permite a la empresa restar del ingreso gravable una parte o el total del monto invertido en proyectos de CTel, antes de calcular el impuesto de renta. Esto disminuye el ingreso sujeto a impuestos, reduciendo el total a pagar. Por ejemplo, si una empresa invierte en un proyecto de innovación y cumple con los requisitos, puede deducir esa inversión de su base imponible, logrando un menor impuesto de renta.
2. **Descuento por Inversiones en CTel:** Este incentivo permite restar un porcentaje de la inversión realizada directamente del valor del impuesto de renta a pagar. A diferencia de la deducción, que afecta el ingreso gravable, el descuento se aplica sobre el impuesto total, restándolo directamente. Esto puede ser muy beneficioso, ya que representa un ahorro directo en el monto a pagar.

Las **resoluciones** de deducción y descuento son documentos que:

1. **Certifican la aprobación del incentivo:** Confirman que la empresa cumple con los requisitos establecidos por la ley y que la inversión en CTel es válida para obtener la deducción o el descuento.

2. **Rechazan o ajustan el incentivo:** Si la inversión no cumple con los requisitos o si existen inconsistencias en la solicitud, la resolución puede rechazar la solicitud o solicitar información adicional para reconsiderar la deducción o el descuento.

Resolucion APROBADO

Seccion de Parametros

```
<parameter name="PROYECTO_ID" class="java.lang.String"/>
<parameter name="IMG" class="java.lang.String">
  <parameterDescription><![CDATA[]]></parameterDescription>
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="SUBREPORT_DIR" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_BASE64" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP_Completo" class="java.lang.String"/>
<parameter name="VIGENCIA" class="java.lang.Integer">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_SECUNDARIA_BASE64" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_vto_bno" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso2" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_proyecto" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
```

En esta sección se definen varios parámetros (<parameter>) que permiten la configuración dinámica del reporte. Algunos parámetros incluyen:

1. **PROYECTO_ID:** Define un parámetro de tipo String que representa el identificador del proyecto. Se usa para filtrar o identificar la resolución específica del proyecto en el reporte.
2. **IMG:** Es un parámetro de tipo String pensado para recibir la ruta o URL de una imagen. Esta imagen puede ser el logo principal o algún elemento visual en el encabezado o pie del reporte.

3. **SUBREPORT_DIR**: Este parámetro indica la ruta del subreporte. Si el reporte principal depende de subreportes, este parámetro le dice a iReport dónde encontrarlos.
4. **IMG_BASE64**: Otro parámetro de tipo String, que contiene una imagen en formato base64. Esto permite que la imagen se incluya directamente en el reporte sin necesidad de una ruta externa.
5. **nombre_entidad_SIGP**: Este parámetro recibe el nombre de la entidad, posiblemente en forma abreviada. Sirve para personalizar el reporte con el nombre de la entidad que está emitiendo la resolución.
6. **nombre_entidad_SIGP_Completo**: Similar al anterior, pero aquí se proporciona el nombre completo de la entidad. Esto puede ser útil si el reporte incluye detalles formales o legales.
7. **VIGENCIA**: Es un parámetro de tipo Integer que define la vigencia del reporte, generalmente el año en curso o el período de validez de la resolución.
8. **IMG_SECUNDARIA_BASE64**: Similar a **IMG_BASE64**, este parámetro contiene una segunda imagen en formato base64, probablemente una imagen secundaria o un logo adicional.
9. **func_vto_bno**: Especifica el nombre o identificación de la persona que aprobó el documento o emitió la autorización (funcionario visto bueno).
10. **func_reviso** y **func_reviso2**: Representan a los funcionarios que revisaron el reporte. Estos parámetros pueden ser utilizados para incluir los nombres o firmas en el reporte.
11. **func_proyecto**: Indica el nombre o identificador del funcionario a cargo del proyecto.

Cada uno de estos parámetros se configura en iReport para recibir valores externos, permitiendo que el reporte se ajuste de acuerdo con la información específica del proyecto o resolución que se esté generando. Esta flexibilidad permite personalizar cada reporte sin modificar el diseño.

Sección de Consulta (Query)

```
<queryString>
  <![CDATA[SELECT (SELECT LENGTHB(detalle) AS length_in_bytes
    FROM adm_parametros_sistema
    WHERE codigo_parametro = 'DEDUINVCON') size_considerando
  , NVL(dbms_lob.substr((SELECT detalle
    FROM adm_parametros_sistema
    WHERE codigo_parametro = 'DEDUINVCON'),
```

```

4000,
1),
'' AS considerando_parte1,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON'),
4000,
4001),
'' AS considerando_parte2,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON'),
4000,
8001),
'' AS considerando_parte3,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON'),
4000,
12001),
'' AS considerando_parte4,

dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle

FROM adm_parametros_sistema

WHERE codigo_parametro =

'DEDUINVCOA1'),

'[[NUMERO_RESOLUCION]]',

p.numero_resolucion_exencion),

'[[ANIO_RESOLUCION]]',

p.ano_resolucion_exencion),

'[[TIPO_ENTIDAD]]',

lower(tent.descripcion_valor_definicion)),

'[[ENTIDAD]]',

e.nombre_entidad),

'[[NIT]]',

e.nit || '-' ||

e.digito_verificacion),

'[[TITULO]]',

upper(p.titulo)),

'[[CODIGO_PROYECTO]]',

p.codigo_proyecto),

```



```
'[[TIPO_PROYECTO]]',
vdtip.descripcion_valor_definicion),
'[[CONCEPTO_EVALUACION]]',
p.concepto_evaluacion),
'[[DIA_REUNION_ACTA]]',
to_char(acta.fecha_acta,
                                'DD')),
                                '[[MES_REUNION_ACTA]]',
to_char(acta.fecha_acta,
                                'Month',
                                'nls_date_language=spanish')),
                                '[[ANO_REUNION_ACTA]]',
to_char(acta.fecha_acta,
                                'YYYY')),
                                '[[NUMERO_ACTA]]',
acta.numero),
                                '[[ANO_ACTA]]',
acta.anio),
                                '[[PROGRAMA]]',
lower(d.nombre_dependencia)),
                                4000,
                                1) AS considerando_aprobado1,

dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle

FROM adm_parametros_sistema

WHERE codigo_parametro =

'DEDUINVOA2'),
'[[NUMERO_RESOLUCION]]',
p.numero_resolucion_exencion),
'[[ANIO_RESOLUCION]]',
p.ano_resolucion_exencion),
'[[TIPO_ENTIDAD]]',
lower(tent.descripcion_valor_definicion)),
'[[ENTIDAD]]',
e.nombre_entidad),
'[[NIT]]',
e.nit || '-' ||
e.digito_verificacion),
```

```

'[[TITULO]]',
upper(p.titulo)),
'[[CODIGO_PROYECTO]]',
p.codigo_proyecto),
'[[TIPO_PROYECTO]]',
vdtip.descripcion_valor_definicion),
'[[CONCEPTO_EVALUACION]]',
p.concepto_evaluacion),
'[[DIA_REUNION_ACTA]]',
to_char(acta.fecha_acta,
'DD')),
'[[MES_REUNION_ACTA]]',
to_char(acta.fecha_acta,
'Month',
'nls_date_language=spanish')),
'[[ANO_REUNION_ACTA]]',
to_char(acta.fecha_acta,
'YYYY')),
'[[NUMERO_ACTA]]',
acta.numero),
'[[ANO_ACTA]]',
acta.anio),
'[[PROGRAMA]]',
lower(d.nombre_dependencia)),
'[[PUNTAJE_PROMEDIO_EVALUACION]]',
(SELECT nvl(AVG(eval.puntaje),
0)
FROM pry_evaluaciones eval
WHERE eval.proyecto_id = p.proyecto_id)),
4000,
1) AS considerando_aprobado2,
nvl(dbms_lob.substr(p.concepto_evaluacion,
4000,
1),
'') concepto_evaluacion,
nvl(dbms_lob.substr(p.concepto_evaluacion,
8000,
4001),
'') concepto_evaluacion2,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT
detalle
FROM
adm_parametros_sistema
WHERE
codigo_parametro =
'DEDUINVART')),

```

```

upper(p.titulo)),
'[[CODIGO_PROYECTO]]',
p.codigo_proyecto),
'[[TIPO_PROYECTO]]',
vdtip.descripcion_valor_definicion),
'[[ENTIDAD]]',
e.nombre_entidad),
'[[NIT]]',
e.nit || '-' ||
e.digito_verificacion),
'[[ANIO_ACTA]]',
to_char(p.fecha_acta_decision,
'yyyy'),
'[[ENTIDAD_IDENTIFICADA_NIT]]',
f_ent_proyecto_rubros_aprob(p.proyecto_id,
${VIGENCIA})),
4000,
1) AS articulos,
dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'NOM_PERS'),
4000,
1) AS nombre,
dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'CARGO_PERS'),
4000,
1) AS cargo,
vdtip.descripcion_valor_definicion tipo_proyecto,
to_char(acta.anio,
'9999') anio_acta,
acta.numero numero_acta,
(SELECT v.descripcion_valor_definicion
FROM adm_valores_definiciones v
WHERE v.tipo_definicion = 'CODIF_EST'
AND v.valor_definicion = '050') formato_version,
dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVNORA'),
4000,
1) AS normatividad
FROM sgp_proyectos p,
ent_entidades e,
sb_dependencias d,
adm_valores_definiciones tent,
adm_valores_definiciones vdtip,
sgp_actas_decision acta
WHERE e.entidad_id (+) = p.entidad_n_id
AND d.dependencia_id (+) = p.dependencia_id
AND tent.valor_definicion_id (+) = e.tipo_entidad_id
AND vdtip.valor_definicion_id (+) = p.tipo_proyecto_id
AND acta.acta_id (+) = p.acta_consejo_id
AND p.proyecto_id = ${PROYECTO_ID}]]>
</queryString>

```

Este bloque de código es una consulta SQL que obtiene datos de múltiples tablas relacionadas con proyectos, tales como detalles de la entidad, el proyecto y las resoluciones asociadas. Utiliza la función **dbms_lob.substr** para manipular datos de tipo **LOB** (Large Object), dividiendo o extrayendo partes de un texto de longitud considerable. A continuación, se explican las partes del código:

1. Extracción de Texto en Partes

- **considerando_parte1, considerando_parte2**, etc.: Usan **dbms_lob.substr** para dividir el contenido de un campo LOB llamado detalle (de la tabla **adm_parametros_sistema**). Esto se realiza en bloques de 4000 caracteres debido a las limitaciones de longitud de los campos LOB en consultas.
- Ejemplo de SQL:

```
NVL(dbms_lob.substr((SELECT detalle FROM adm_parametros_sistema WHERE codigo_parametro = 'DEDUINVCON'), 4000, 1), '') AS considerando_parte1
```

2. Reemplazos Dinámicos

- Campos como **considerando_aprobado1** y **considerando_aprobado2** usan múltiples funciones **REPLACE** para sustituir marcadores en el texto (como **[[NUMERO_RESOLUCION]]** o **[[TIPO_ENTIDAD]]**) con datos específicos del proyecto y su contexto, como el número de resolución, el año, el tipo de entidad, etc.
- Estos marcadores se reemplazan con valores de las tablas relacionadas (por ejemplo, **p.numero_resolucion_exencion**, **e.nombre_entidad**).

3. Parámetros y Formatos de Fecha

- Se usan funciones como **to_char** para formatear fechas (**acta.fecha_acta**, **p.fecha_acta_decision**) y adaptarlas a un formato de presentación en el reporte. Esto incluye separar el día, el mes y el año para incluirlos en la resolución.

4. Obtención de Puntajes Promedio

- **PUNTAJE_PROMEDIO_EVALUACION**: Calcula el promedio de evaluaciones para el proyecto con **AVG(eval.puntaje)**. Esto permite evaluar el desempeño o la evaluación general de cada proyecto.

5. Sección de Normatividad

- La columna normatividad obtiene contenido específico desde **adm_parametros_sistema**, configurado para el parámetro **DEDUINVNORA**. Este campo probablemente contiene referencias normativas o reglamentarias aplicables al proyecto.

6. Filtros y Relación entre Tablas

- La consulta se une a varias tablas usando condiciones WHERE con las siguientes relaciones:
 - **sgp_proyectos** (p): Tabla principal donde se encuentra el proyecto.
 - **ent_entidades** (e): Relaciona la entidad con el proyecto.
 - **sb_dependencias** (d): Asocia la dependencia con el proyecto.
 - **adm_valores_definiciones** (tent y vdtip): Relaciona valores definidos como el tipo de entidad y el tipo de proyecto.
 - **sgp_actas_decision** (acta): Registra decisiones y detalles de la aprobación del proyecto.

Sección de Campos (Fields)

```
<field name="SIZE_CONSIDERANDO" class="java.lang.Long"/>
<field name="CONSIDERANDO_PARTE1" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE2" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE3" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE4" class="java.lang.String"/>
<field name="CONSIDERANDO_APROBADO1" class="java.lang.String"/>
<field name="CONSIDERANDO_APROBADO2" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION2" class="java.lang.String"/>
<field name="ARTICULOS" class="java.lang.String"/>
<field name="NOMBRE" class="java.lang.String"/>
<field name="CARGO" class="java.lang.String"/>
<field name="TIPO_PROYECTO" class="java.lang.String"/>
<field name="ANIO_ACTA" class="java.lang.String"/>
<field name="NUMERO_ACTA" class="java.lang.String"/>
<field name="FORMATO_VERSION" class="java.lang.String"/>
<field name="NORMATIVIDAD" class="java.lang.String"/>
```

Los campos (<field>) definen las variables que se utilizarán en el diseño visual del reporte. Algunos campos destacados son:

1. **SIZE_CONSIDERANDO** (java.lang.Long): Almacena la longitud en bytes del texto en el campo detalle de adm_parametros_sistema para el parámetro **DEDUINVCON**. Este dato ayuda a verificar el tamaño del contenido extraído.
2. **CONSIDERANDO_PARTE1** a **CONSIDERANDO_PARTE4** (java.lang.String): Fragmentos del texto detalle dividido en partes de 4000 caracteres. Se utiliza para el contenido extenso del "considerando", que necesita ser seccionado debido a limitaciones de longitud en el reporte.
3. **CONSIDERANDO_APROBADO1** y **CONSIDERANDO_APROBADO2** (java.lang.String): Textos con marcadores reemplazados por valores específicos, como números de resolución, nombres de entidad, entre otros. Cada uno contiene un fragmento de texto aprobado que aparece en el reporte.
4. **CONCEPTO_EVALUACION** y **CONCEPTO_EVALUACION2** (java.lang.String): Campos de texto con el concepto de evaluación del proyecto, divididos en dos partes para adaptarse a los límites de caracteres permitidos.
5. **ARTICULOS** (java.lang.String): Contiene el texto de artículos o normativa específica aplicable al proyecto. Los datos son extraídos y formateados de manera similar al "considerando".
6. **NOMBRE** (java.lang.String): Nombre de la persona responsable o del contacto asociado al proyecto, recuperado del parámetro **NOM_PERS**.
7. **CARGO** (java.lang.String): Cargo de la persona responsable o del contacto, recuperado del parámetro **CARGO_PERS**.
8. **TIPO_PROYECTO** (java.lang.String): Tipo de proyecto, como se describe en adm_valores_definiciones. Este campo identifica la clasificación del proyecto.
9. **ANIO_ACTA** (java.lang.String): Año en el que se realizó el acta de decisión asociada al proyecto.
10. **NUMERO_ACTA** (java.lang.String): Número del acta de decisión que registra la evaluación del proyecto.
11. **FORMATO_VERSION** (java.lang.String): Versión o formato de la evaluación, obtenido de adm_valores_definiciones. Define el estándar de formato usado en el reporte.

12. **NORMATIVIDAD** (java.lang.String): Contenido de la normativa aplicable (obtenido del parámetro DEDUINVNORA), que puede incluir disposiciones legales o reglamentarias relevantes para el proyecto.

Sección de Variables

```
<variable name="TOTAL_PAGINAS" class="java.lang.Integer">
  <variableExpression><![CDATA[{$V{PAGE_NUMBER}}]></variableExpression>
</variable>
```

Finalmente, la variable **TOTAL_PAGINAS** usa **{\$V{PAGE_NUMBER}}**, una variable interna de JasperReports, para mostrar el total de páginas del reporte.

Sección de Encabezado Pagina (PageHeader)

```
<pageHeader>
  <band height="58" splitType="Stretch">
    <textField>
      <reportElement x="150" y="0" width="182" height="31"
isRemoveLineWhenBlank="true" uuid="c21c3e5f-8264-455f-b2b0-5b786c836f7a">
        <printWhenExpression><![CDATA[{$V{PAGE_NUMBER}}
1]]></printWhenExpression>
      </reportElement>
      <textElement textAlignment="Center" verticalAlignment="Bottom">
        <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA["RESOLUCIÓN No. DE " +
{$P{VIGENCIA}}]></textFieldExpression>
    </textField>
    <image scaleImage="RealSize">
      <reportElement x="0" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="d1f062a8-2ce1-495e-8c0a-93ec77648de0"/>
      <imageExpression><![CDATA[{$P{IMG_SECUNDARIA_BASE64}} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_SECUNDARIA_BASE64}.getBytes()))
:
new
ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))]]></imageExpression>
    </image>
    <textField isStretchWithOverflow="true">
      <reportElement positionType="Float" x="0" y="33" width="482" height="25"
isRemoveLineWhenBlank="true" uuid="2f5a455a-0e11-4946-b121-65a2c865f78a">
        <printWhenExpression><![CDATA[{$V{PAGE_NUMBER}}
1]]></printWhenExpression>
      </reportElement>
      <textElement textAlignment="Center" verticalAlignment="Top"
markup="styled">
        <font fontName="Arial" size="9" isBold="false" isItalic="true"
isStrikeThrough="false" isPdfEmbedded="false"/>
      </textElement>
```

```
<textFieldExpression><![CDATA["\\"Por la cual se califica un proyecto como
de "+ $F{TIPO_PROYECTO} +" y se otorga un Beneficio Tributario.\" "]]></textFieldExpression>
</textField>
<image scaleImage="RealSize">
  <reportElement x="332" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="72694b95-6457-42c7-a0f8-162f0b29f151"/>
  <imageExpression><![CDATA[$P{IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))
: null]]></imageExpression>
</image>
</band>
</pageHeader>
```

La sección pageHeader está configurada para mostrar en las páginas donde **\$V{PAGE_NUMBER}** > 1. Explicamos el código anterior:

1. **Condiciones de Impresión:** La etiqueta

<printWhenExpression><![CDATA[\$V{PAGE_NUMBER} > 1]]></printWhenExpression> en los elementos textField asegura que estos campos solo se muestren a partir de la segunda página. Esto es útil si quieres que el encabezado sea diferente en la primera página. Si se desea que algo se muestre en todas las páginas, se puede eliminar esta condición.

2. **Imágenes Condicionales:**

- La primera imagen se muestra si **\$P{IMG_SECUNDARIA_BASE64}** no es null; de lo contrario, utiliza **\$P{IMG_BASE64}**.
- La segunda imagen en la misma banda tiene una expresión condicional similar, pero su imageExpression solo muestra **\$P{IMG_BASE64}** si **\$P{IMG_SECUNDARIA_BASE64}** no es null.

3. **Centrado de Texto:**

- Los elementos **<textField>** utilizan **textAlignment="Center"**, lo cual está bien para centrar el texto en el espacio asignado. Para un resultado más preciso, verifica que el ancho y posición del campo sean consistentes con el diseño del encabezado.

4. **Control de Tamaño y Desbordamiento:**

- El segundo textField tiene el atributo **isStretchWithOverflow="true"**, lo cual permite que el campo se ajuste si el texto es más extenso que el espacio asignado. Esto es útil si el contenido de **\$F{TIPO_PROYECTO}** puede variar en

longitud, pero si encuentras problemas de alineación, podrías ajustar el ancho o el tamaño de fuente.

Sección de Detalles (Details)

```
<detail>
  <band height="114" splitType="Stretch">
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
      <reportElement positionType="Float" x="0" y="95" width="482" height="15"
      uuid="8302d748-dd87-4a32-afa9-bc7a5765e7f9"/>
      <textElement textAlignment="Center" verticalAlignment="Top"
      markup="html">
        <font fontName="Arial" size="11" isPdfEmbedded="false"/>
      </textElement>

      <textFieldExpression><![CDATA[${F{NORMATIVIDAD}.trim()}]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true">
      <reportElement positionType="Float" x="0" y="59" width="482" height="30"
      uuid="36699d96-812a-4905-b885-412e67d871d2"/>
      <textElement textAlignment="Center" verticalAlignment="Top"
      markup="styled">
        <font fontName="Arial" size="11" isBold="true"
        isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA["LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E
      INNOVACIÓN DEL $P{nombre_entidad_SIGP_Completo}.toUpperCase().trim()}]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
      <reportElement positionType="Float" x="0" y="32" width="482" height="20"
      isRemoveLineWhenBlank="true" isPrintWhenDetailOverflows="true" uuid="385d34eb-1b0e-49d4-
      a998-de01abb5302c"/>
      <textElement textAlignment="Center" verticalAlignment="Middle"
      markup="styled">
        <font fontName="Arial" size="10" isBold="false" isItalic="true"
        isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA[" \"Por la cual se califica un proyecto
      como de "+ ${TIPO_PROYECTO} +" y se otorga el acceso a un beneficio tributario\"
      "]]></textFieldExpression>
    </textField>
    <textField>
      <reportElement x="150" y="0" width="182" height="26" uuid="6f09484c-
      87df-4ad6-a727-d6a3f443e1da"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="Arial" size="11" isBold="true"
        isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA["RESOLUCIÓN No. DE " +
      $P{VIGENCIA}]]></textFieldExpression>
    </textField>
  </band>
  <band height="37">
    <staticText>
      <reportElement positionType="Float" x="0" y="5" width="482" height="15"
      uuid="64724d7c-9256-41d5-b9d8-0df747de64e5"/>
      <textElement textAlignment="Center" verticalAlignment="Top">
```

```

<font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
</textElement>
<text><![CDATA[CONSIDERANDO:]]></text>
</staticText>
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
<reportElement positionType="Float" stretchType="RelativeToBandHeight"
x="0" y="22" width="482" height="15" isPrintWhenDetailOverflows="true" uuid="adeaefac8-c14e-
4fc5-93d8-c9c5f8bfff339"/>
<textElement textAlignment="Justified" verticalAlignment="Top"
markup="html">
<font fontName="Arial" size="11" isPdfEmbedded="false"/>
</textElement>
<textFieldExpression><![CDATA[ ${F{SIZE_CONSIDERANDO}} > 12000
?
${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_PARTE2}}).concat(${F{CONSIDERANDO_PARTE3}}).conc
at(${F{CONSIDERANDO_PARTE4}}).concat(${F{CONSIDERANDO_APROBADO1}})
: ${F{SIZE_CONSIDERANDO}} > 8000
?
${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_PARTE2}}).concat(${F{CONSIDERANDO_PARTE3}}).conc
at(${F{CONSIDERANDO_APROBADO1}})
: ${F{SIZE_CONSIDERANDO}} > 4000
? ${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_PARTE2}}).concat(${F{CONSIDERANDO_APROBADO1}})
: ${F{CONSIDERANDO_PARTE1}}.concat(${F{CONSIDERANDO_APROBADO1}})]></textFieldExpression>
</textField>
</band>
<band height="15">
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
<reportElement positionType="Float" stretchType="RelativeToBandHeight"
x="20" y="0" width="442" height="15" isPrintWhenDetailOverflows="true" uuid="b541f2e0-8ac2-
4564-97ab-b498b2f63517"/>
<textElement textAlignment="Justified" verticalAlignment="Top"
markup="styled">
<font fontName="Arial" size="11" isItalic="true"
isUnderline="false" isPdfEmbedded="false"/>
</textElement>
<textFieldExpression><![CDATA[ ${F{CONCEPTO_EVALUACION}} != null
?
(${F{CONCEPTO_EVALUACION2}} != null
?
"\\".concat(${F{CONCEPTO_EVALUACION}}).trim().concat(${F{CONCEPTO_EVALUACION2}}).trim().concat("
\").trim() : "\\".concat(${F{CONCEPTO_EVALUACION}}).trim().concat("\").trim()
: "Sin registro de concepto de evaluación"]></textFieldExpression>
</textField>
</band>
<band height="15">
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
<reportElement positionType="Float" stretchType="RelativeToBandHeight"
x="0" y="0" width="482" height="15" isPrintWhenDetailOverflows="true" uuid="d07c35bc-e3a5-
4cd2-a692-7a6201cc6079"/>
<textElement textAlignment="Justified" verticalAlignment="Top"
markup="html">
<font fontName="Arial" size="11" isPdfEmbedded="false"/>
</textElement>
<textFieldExpression><![CDATA[ ${F{CONSIDERANDO_APROBADO2}}.trim()]]></textFieldExpression>
>
</textField>
</band>
<band height="15">
<staticText>
<reportElement positionType="Float" x="0" y="0" width="482" height="15"
uuid="d93be984-3888-4ac8-bb6e-10323ffe01d9"/>

```

```

        <textElement textAlignment="Center" verticalAlignment="Top">
            <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
        </textElement>
        <text><![CDATA[RESUELVE:]]></text>
        </staticText>
    </band>
    <band height="47">
        <subreport>
            <reportElement positionType="Float" x="0" y="1" width="482" height="20"
uuid="4449cd6c-8cd0-4984-8134-d5958981a31c"/>
            <subreportParameter name="VIGENCIA">

                <subreportParameterExpression><![CDATA[${P{VIGENCIA}}]></subreportParameterExpression>
                </subreportParameter>
                <subreportParameter name="SUBREPORT_DIR">

                <subreportParameterExpression><![CDATA[${P{SUBREPORT_DIR}}]></subreportParameterExpressi
on>
                </subreportParameter>
                <subreportParameter name="proyecto_id">

                <subreportParameterExpression><![CDATA[${P{PROYECTO_ID}}]></subreportParameterExpression
>
                </subreportParameter>

                <connectionExpression><![CDATA[${P{REPORT_CONNECTION}}]></connectionExpression>
                <subreportExpression><![CDATA[${P{SUBREPORT_DIR}
"entidadesVigenciasRecomendadoDeducccionArt1.jasper"}]></subreportExpression>
                </subreport>
                <subreport>
                    <reportElement positionType="Float" x="0" y="27" width="482" height="20"
uuid="1e87e814-c3df-4fd7-b0e0-2afb004e996a"/>
                    <subreportParameter name="VIGENCIA">

                        <subreportParameterExpression><![CDATA[${P{VIGENCIA}}]></subreportParameterExpression>
                        </subreportParameter>
                        <subreportParameter name="SUBREPORT_DIR">

                        <subreportParameterExpression><![CDATA[${P{SUBREPORT_DIR}}]></subreportParameterExpressi
on>
                        </subreportParameter>
                        <subreportParameter name="proyecto_id">

                        <subreportParameterExpression><![CDATA[${P{PROYECTO_ID}}]></subreportParameterExpression
>
                        </subreportParameter>

                        <connectionExpression><![CDATA[${P{REPORT_CONNECTION}}]></connectionExpression>
                        <subreportExpression><![CDATA[${P{SUBREPORT_DIR}
"entidadesVigenciasRecomendadoDeducccionArt2.jasper"}]></subreportExpression>
                        </subreport>
                    </band>
                    <band height="48">
                        <textField>
                            <reportElement positionType="Float" x="0" y="29" width="482" height="15"
isPrintWhenDetailOverflows="true" uuid="42012e2e-3999-44af-816d-cd2d5833aad7"/>
                            <textElement verticalAlignment="Top">
                                <font fontName="Arial" size="11" isPdfEmbedded="false"/>
                            </textElement>

```

```
<textFieldExpression><![CDATA["Dada en Bogotá, D.C., a los,      días del
mes de      de "+ Calendar.getInstance().get(Calendar.YEAR)]]></textFieldExpression>
</textField>
<textField isStretchWithOverflow="true" isBlankWhenNull="true">
  <reportElement positionType="Float" x="0" y="0" width="482" height="15"
isPrintWhenDetailOverflows="true" uuid="b970a8cf-1d52-4388-a395-39a0afa465cf"/>
  <textField      textAlignment="Justified"      verticalAlignment="Top"
markup="html">
    <font fontName="Arial" size="11" isPdfEmbedded="false"/>
    </textField>

    <textFieldExpression><![CDATA[${F{ARTICULOS}.trim()}]]></textFieldExpression>
    </textField>
  </band>
</detail>
```

La sección details en un informe de JasperReports es donde se generan las filas de datos del reporte. Cada vez que se itera sobre un conjunto de datos, como una lista o un conjunto de registros de una base de datos, se crea una nueva instancia de la sección details para cada registro. Aquí es donde se colocan los elementos visuales (como campos de texto, gráficos, tablas, etc.) que se llenarán con los datos correspondientes a cada registro.

Esta sección de detalles contiene múltiples elementos textField, staticText, y subreport. Aquí te explico las funciones principales de cada uno:

1. Campos de Texto (textField):

- Los campos textField extraen y muestran datos según sus expresiones. Algunos ejemplos incluyen:
 - **`$F{NORMATIVIDAD}.trim()`**: Muestra el campo NORMATIVIDAD centrado.
 - **"LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E INNOVACIÓN DEL " + `$P{nombre_entidad_SIGP_Completo}`**: Inserta un texto concatenado con el parámetro **nombre_entidad_SIGP_Completo**, en mayúsculas y centrado.
 - Otro campo define el texto de resolución: **"RESOLUCIÓN No. DE " + `$P{VIGENCIA}`**.

2. Condicional de Concatenación de Textos (SIZE_CONSIDERANDO):

- La expresión condicional del campo **`$F{SIZE_CONSIDERANDO}`** evalúa el tamaño del texto para concatenar diferentes partes (**CONSIDERANDO_PARTE1**,

CONSIDERANDO_PARTE2, etc.) según el tamaño. Esto permite manejar texto variable en función de su longitud.

3. Subreportes (subreport):

- Se incluyen dos subreportes para agregar datos de deducción de entidades y vigencias. Ambos reciben parámetros (**VIGENCIA**, **SUBREPORT_DIR**, **PROYECTO_ID**) y comparten la conexión **REPORT_CONNECTION**.

4. Texto Estático (staticText):

- Ejemplos de texto estático incluyen los títulos "**CONSIDERANDO:**" y "**RESUELVE:**", que son encabezados en el documento.

5. Campos de Fecha:

- El campo de texto para la fecha muestra el año actual utilizando **Calendar.getInstance().get(Calendar.YEAR)**.

6. Manejo de Desbordamiento de Texto (isStretchWithOverflow):

- Los atributos **isStretchWithOverflow** y **isBlankWhenNull** permiten que el texto se extienda o se oculte cuando es nulo, ajustando la presentación.

Sección de Pie de Pagina (PageFooter)

```
<pageFooter>
  <band height="42" splitType="Stretch">
    <textField evaluationTime="Page">
      <reportElement x="203" y="22" width="50" height="20" forecolor="#999999"
        uuid="3bfacc06-77c4-47f5-9f26-883213d073fc"/>
      <textField>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textField>
      <textFieldExpression><![CDATA["Página " + $V{PAGE_NUMBER} + " de
        "]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Report">
      <reportElement x="253" y="22" width="18" height="20" forecolor="#999999"
        uuid="42ba6cac-8dc3-4ca5-a6af-ee42180180e8"/>
      <textField>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textField>
      <textFieldExpression><![CDATA[$V{PAGE_NUMBER}]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Page">
      <reportElement x="0" y="0" width="112" height="30" forecolor="#999999"
        uuid="5195747f-e33d-4169-8802-6a38bf19d755"/>
      <textField>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textField>
      <textFieldExpression><![CDATA["Contrato No. 237 / 2025"]]></textFieldExpression>
    </textField>
  </band>
</pageFooter>
```

```
<font fontName="Arial" size="8" isPdfEmbedded="false"/>
</textElement>

<textFieldExpression><![CDATA[ $F{FORMATO_VERSION} ]]></textFieldExpression>
</textField>

</band>
</pageFooter>
```

En esta sección de **pageFooter**, hay varios elementos importantes:

1. Número de Página:

- El primer **textField** muestra el texto "Página " seguido del número de la página actual. Utiliza la expresión **\$V{PAGE_NUMBER}** para obtener el número de la página actual.
- El segundo **textField** muestra el total de páginas utilizando también **\$V{PAGE_NUMBER}**, lo cual puede ser útil si deseas mostrar "Página X de Y", aunque en este caso parece que no hay un total definido.

2. Formato de Versión:

- El tercer **textField** muestra el valor del campo **FORMATO_VERSION**, que puede ser un indicador de la versión del informe. Este campo se muestra en un formato de texto que se alinea verticalmente al centro.

Sección de Resumen (Summary)

```
<summary>
  <band height="161" splitType="Immediate">
    <staticText>
      <reportElement positionType="Float" x="0" y="0" width="482" height="15"
      uuid="8a85e609-d92c-4b1b-81bc-83b4a3347de9"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="Arial" size="11" isBold="true"
        isPdfEmbedded="false"/>
        </textElement>
        <text><![CDATA[NOTIFÍQUESE Y CÚMPLASE,]]></text>
      </staticText>
      <textField isStretchWithOverflow="true" isBlankWhenNull="true">
        <reportElement positionType="Float" x="0" y="73" width="482" height="15"
        uuid="157aa74f-f0d7-4ff9-b644-6fd620ed7cd2"/>
        <textElement textAlignment="Center" verticalAlignment="Middle"
        markup="styled">
          <font fontName="Arial" size="11" isBold="true"
          isPdfEmbedded="false"/>
          </textElement>
          <textFieldExpression><![CDATA[ $F{CARGO} ]]></textFieldExpression>
        </textField>
        <textField isStretchWithOverflow="true" isBlankWhenNull="true">
```

```

<reportElement positionType="Float" x="0" y="58" width="482" height="15"
uuid="b3c1a114-e661-4788-9a25-75c3dec6ee02"/>
<textElement textAlignment="Center" verticalAlignment="Middle"
markup="styled">
<font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
</textElement>
<textFieldExpression><![CDATA[{F{NOMBRE}}]></textFieldExpression>
</textField>
<textField isBlankWhenNull="true">
<reportElement positionType="Float" x="32" y="120" width="450"
height="10" uuid="da925050-29d6-4e52-b150-7d11fa547503"/>
<textElement>
<font fontName="Arial" size="7"/>
</textElement>
<textFieldExpression><![CDATA[{P{func_vto_bno}}]></textFieldExpression>
</textField>
<staticText>
<reportElement positionType="Float" x="0" y="150" width="37" height="10"
forecolor="#000000" uuid="5f8584e1-686b-42b5-a0f6-4871aa8f6dc9"/>
<textElement verticalAlignment="Middle">
<font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
</textElement>
<text><![CDATA[Proyectó:]]></text>
</staticText>
<textField isBlankWhenNull="true">
<reportElement positionType="Float" x="32" y="130" width="450"
height="10" uuid="cc30cbc7-f693-4fab-8165-74b0652ff5d5"/>
<textElement>
<font fontName="Arial" size="7"/>
</textElement>
<textFieldExpression><![CDATA[{P{func_reviso}}]></textFieldExpression>
</textField>
<staticText>
<reportElement positionType="Float" x="0" y="130" width="32" height="10"
forecolor="#000000" uuid="7b00e9ce-f1ed-460b-8bfd-bd1610b9d1ea"/>
<textElement verticalAlignment="Middle">
<font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
</textElement>
<text><![CDATA[Revisó:]]></text>
</staticText>
<staticText>
<reportElement positionType="Float" x="0" y="120" width="32" height="10"
forecolor="#000000" uuid="3f592d26-6184-411d-9eeb-14765ed7d2d8"/>
<textElement verticalAlignment="Middle">
<font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
</textElement>
<text><![CDATA[Vto. Bo.:]]></text>
</staticText>
<textField isBlankWhenNull="true">
<reportElement positionType="Float" x="37" y="150" width="445"
height="10" uuid="e1071fd0-8f25-4d85-9491-0924af8b4284"/>
<textElement>
<font fontName="Arial" size="7"/>
</textElement>
<textFieldExpression><![CDATA[{P{func_proyecto}}]></textFieldExpression>
</textField>

```

```
<textField isBlankWhenNull="true">
  <reportElement positionType="Float" x="32" y="140" width="450"
height="10" uuid="1a2cf88a-c43d-4e72-8248-e3e8b3357bb"/>
  <textElement>
    <font fontName="Arial" size="7"/>
  </textElement>
  <textFieldExpression><![CDATA[{$P{func_reviso2}}]></textFieldExpression>
</textField>
<staticText>
  <reportElement positionType="Float" x="0" y="140" width="32" height="10"
forecolor="#000000" uuid="a9faff02-3d5d-4a0f-a42a-e188f2d46983"/>
  <textElement verticalAlignment="Middle">
    <font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
  </textElement>
  <text><![CDATA[Revisó:]]></text>
</staticText>
</band>
</summary>
```

La sección **summary** en un informe de JasperReports se utiliza para mostrar información de cierre o un resumen del contenido del informe, generalmente después de que se han presentado todos los detalles y datos en las secciones anteriores. Esta sección es útil para presentar conclusiones, notas finales o cualquier información importante que se quiera resaltar al final del documento.

En la sección de **summary**, hay varios elementos clave:

1. Texto Estático:

- El primer **staticText** muestra el mensaje "**NOTIFÍQUESE Y CÚMPLASE,**" centrado y en negrita, lo que indica que es un aviso importante para el lector. Está ubicado en la parte superior de la sección y utiliza una fuente Arial de tamaño 11.

2. Nombre y Cargo:

- Se presentan campos dinámicos (**textField**) que muestran el nombre (**{\$F{NOMBRE}}**) y el cargo (**{\$F{CARGO}}**) de la persona que revisó el informe. Ambos están centrados y en negrita, asegurando que destaquen en la página.

3. Revisiones y Proyectos:

- Se incluyen etiquetas de texto estático y campos de texto que indican quién revisó y proyectó el informe, utilizando los parámetros **{\$P{func_reviso}}**, **{\$P{func_reviso2}}**, y **{\$P{func_proyecto}}**. Estos campos se presentan en una fuente más pequeña (tamaño 7), lo que indica que son detalles adicionales que

son relevantes pero que no necesitan el mismo nivel de atención que el encabezado.

4. Formato:

- Todos los textos estáticos y campos de texto utilizan la fuente Arial, y se asegura que la alineación sea consistente. Las etiquetas de texto tienen un color negro, mientras que los campos están en un tamaño más pequeño, lo que crea un contraste adecuado.

Resolucion NEGADO

Seccion de Parametros

```
<parameter name="PROYECTO_ID" class="java.lang.String"/>
<parameter name="IMG" class="java.lang.String">
  <parameterDescription><![CDATA[]]></parameterDescription>
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="SUBREPORT_DIR" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_BASE64" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP" class="java.lang.String"/>
<parameter name="nombre_entidad_SIGP_Completo" class="java.lang.String"/>
<parameter name="IMG_SECUNDARIA_BASE64" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_vto_bno" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_reviso2" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="func_proyecto" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="VIGENCIA" class="java.lang.Integer">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
```

En esta sección se definen varios parámetros (<parameter>) que permiten la configuración dinámica del reporte. Algunos parámetros incluyen:

1. **PROYECTO_ID**: Define un parámetro de tipo String que representa el identificador del proyecto. Se usa para filtrar o identificar la resolución específica del proyecto en el reporte.
2. **IMG**: Es un parámetro de tipo String pensado para recibir la ruta o URL de una imagen. Esta imagen puede ser el logo principal o algún elemento visual en el encabezado o pie del reporte.
3. **SUBREPORT_DIR**: Este parámetro indica la ruta del subreporte. Si el reporte principal depende de subreportes, este parámetro le dice a iReport dónde encontrarlos.
4. **IMG_BASE64**: Otro parámetro de tipo String, que contiene una imagen en formato base64. Esto permite que la imagen se incluya directamente en el reporte sin necesidad de una ruta externa.
5. **nombre_entidad_SIGP**: Este parámetro recibe el nombre de la entidad, posiblemente en forma abreviada. Sirve para personalizar el reporte con el nombre de la entidad que está emitiendo la resolución.
6. **nombre_entidad_SIGP_Completo**: Similar al anterior, pero aquí se proporciona el nombre completo de la entidad. Esto puede ser útil si el reporte incluye detalles formales o legales.
7. **VIGENCIA**: Es un parámetro de tipo Integer que define la vigencia del reporte, generalmente el año en curso o el período de validez de la resolución.
8. **IMG_SECUNDARIA_BASE64**: Similar a **IMG_BASE64**, este parámetro contiene una segunda imagen en formato base64, probablemente una imagen secundaria o un logo adicional.
9. **func_vto_bno**: Especifica el nombre o identificación de la persona que aprobó el documento o emitió la autorización (funcionario visto bueno).
10. **func_reviso** y **func_reviso2**: Representan a los funcionarios que revisaron el reporte. Estos parámetros pueden ser utilizados para incluir los nombres o firmas en el reporte.
11. **func_proyecto**: Indica el nombre o identificador del funcionario a cargo del proyecto.

Sección de Consulta (Query)

<queryString>

```

<![CDATA[SELECT (SELECT LENGTHB(detalle) AS length_in_bytes
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON') size_considerando
, NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON'),
4000,
1),
'' ) AS considerando_parte1,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON'),
4000,
4001),
'' ) AS considerando_parte2,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON'),
4000,
8001),
'' ) AS considerando_parte3,
NVL(dbms_lob.substr((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVCON'),
4000,
12001),
'' ) AS considerando_parte4,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle

FROM adm_parametros_sistema

WHERE codigo_parametro =

'DEDUINVCON1'),

'[[NUMERO_RESOLUCION]]',

p.numero_resolucion_exencion),

'[[ANIO_RESOLUCION]]',

p.ano_resolucion_exencion),

'[[TIPO_ENTIDAD]]',

```

```

lower(tent.descripcion_valor_definicion)),

                                '[[ENTIDAD]]',

                                e.nombre_entidad),

                                '[[NIT]]',

                                e.nit || '-' ||

                                e.digito_verificacion),

                                '[[TITULO]]',

                                upper(p.titulo)),

                                '[[CODIGO_PROYECTO]]',

                                p.codigo_proyecto),

                                '[[TIPO_PROYECTO]]',

                                vdtip.descripcion_valor_definicion),

                                '[[CONCEPTO_EVALUACION]]',

                                p.concepto_evaluacion),

                                '[[DIA_REUNION_ACTA]]',

                                to_char(acta.fecha_acta,

                                'DD')),

                                '[[MES_REUNION_ACTA]]',

                                to_char(acta.fecha_acta,

                                'Month',

                                'nls_date_language=spanish')),

                                '[[ANO_REUNION_ACTA]]',

                                to_char(acta.fecha_acta,

                                'YYYY')),

```

```
'[[NUMERO_ACTA]]',
acta.numero),
                                '[[ANO_ACTA]]',
                                acta.anio),
                                '[[PROGRAMA]]',
                                lower(d.nombre_dependencia)),
                                4000,
                                1) AS considerando_negado1,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPL
ACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle
FROM adm_parametros_sistema
WHERE codigo_parametro =
'DEDUINVCON2'),
'[[NUMERO_RESOLUCION]]',
p.numero_resolucion_exencion),
'[[ANIO_RESOLUCION]]',
p.ano_resolucion_exencion),
'[[TIPO_ENTIDAD]]',
lower(tent.descripcion_valor_definicion)),
                                '[[ENTIDAD]]',
                                e.nombre_entidad),
                                '[[NIT]]',
                                e.nit || '-' ||
```

```

e.digito_verificacion),

'[[TITULO]]',

upper(p.titulo)),

'[[CODIGO_PROYECTO]]',

p.codigo_proyecto),

'[[TIPO_PROYECTO]]',

vdtip.descripcion_valor_definicion),

'[[CONCEPTO_EVALUACION]]',

p.concepto_evaluacion),

'[[DIA_REUNION_ACTA]]',

to_char(acta.fecha_acta,

'DD')),

'[[MES_REUNION_ACTA]]',

to_char(acta.fecha_acta,

'Month',

'nls_date_language=spanish')),

'[[ANO_REUNION_ACTA]]',

to_char(acta.fecha_acta,

'YYYY')),

'[[NUMERO_ACTA]]',

acta.numero),

'[[ANO_ACTA]]',

acta.anio),

'[[PROGRAMA]]',

lower(d.nombre_dependencia)),

'[[PUNTAJE_PROMEDIO_EVALUACION]]',

(SELECT nvl(AVG(eval.puntaje),

0)

FROM pry_evaluaciones eval

```

```

                                WHERE          eval.proyecto_id          =
p.proyecto_id)),
                                4000,
                                1) AS considerando_negado2,
nvl(dbms_lob.substr(p.concepto_evaluacion,
                                4000,
                                1),
    '') concepto_evaluacion,
nvl(dbms_lob.substr(p.concepto_evaluacion,
                                8000,
                                4001),
    '') concepto_evaluacion2,
dbms_lob.substr(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((SELECT detalle
                                FROM adm_parametros_sistema
                                WHERE codigo_parametro =
                                'DEDUINVARTN'),
                                '[[TITULO]]',
                                upper(p.titulo)),
                                '[[CODIGO_PROYECTO]]',
                                p.codigo_proyecto),
                                '[[TIPO_PROYECTO]]',
                                vdtip.descripcion_valor_definicion),
'[[ENTIDAD]]',
e.nombre_entidad),
                                '[[NIT]]',
                                e.nit || '-' ||
e.digito_verificacion),
                                '[[TIPO_PROYECTO]]',
                                vdtip.descripcion_valor_definicion),
                                4000,
                                1) AS articulos,
dbms_lob.substr((SELECT detalle
                                FROM adm_parametros_sistema
                                WHERE codigo_parametro = 'NOM_PERS'),
                                4000,
                                1) AS nombre,
dbms_lob.substr((SELECT detalle
                                FROM adm_parametros_sistema
                                WHERE codigo_parametro = 'CARGO_PERS'),
                                4000,
                                1) AS cargo,
vdtip.descripcion_valor_definicion tipo_proyecto,
to_char(acta.anio,
    '9999') anio_acta,
acta.numero numero_acta,
(SELECT v.descripcion_valor_definicion
FROM adm_valores_definiciones v
WHERE v.tipo_definicion = 'CODIF_EST'
AND v.valor_definicion = '060') formato_version,
dbms_lob.substr((SELECT detalle

```

```

FROM adm_parametros_sistema
WHERE codigo_parametro = 'DEDUINVNORN'),
4000,
1) AS normatividad
FROM sgp_proyectos      p,
ent_entidades          e,
sb_dependencias        d,
adm_valores_definiciones tent,
adm_valores_definiciones vdtip,
sgp_actas_decision      acta
WHERE e.entidad_id      (+) = p.entidad_n_id
AND d.dependencia_id    (+) = p.dependencia_id
AND tent.valor_definicion_id (+) = e.tipo_entidad_id
AND vdtip.valor_definicion_id (+) = p.tipo_proyecto_id
AND acta.acta_id        (+) = p.acta_consejo_id
AND p.proyecto_id       = $P{PROYECTO_ID}]>
</queryString>

```

La consulta (**<queryString>**) está escrita en SQL y contiene varias subconsultas y funciones de Oracle para extraer y procesar datos específicos. Esta sección de la consulta SQL en el reporte está diseñada para obtener información detallada sobre un proyecto específico desde varias tablas de la base de datos, utilizando el identificador del proyecto (**\$P{PROYECTO_ID}**). A continuación, se desglosan los principales elementos de esta consulta:

1. **Uso de dbms_lob.substr:** La función **dbms_lob.substr** se usa repetidamente para dividir textos largos en trozos de 4000 bytes, debido a que Oracle limita la longitud de texto en ciertas operaciones. Esto permite extraer partes de los textos contenidos en los campos de tipo **LOB**.
2. **Parámetros estáticos:** Se extraen valores de la tabla **adm_parametros_sistema**, donde parece que se almacenan textos predefinidos, como en el caso de los códigos **'DEDUINVCON'**, **'DEDUINVCON1'**, etc. Estos valores se ajustan con información del proyecto mediante **REPLACE** para personalizar los textos según cada proyecto.
3. **Datos del proyecto:** La consulta extrae varios campos de la tabla **sgp_proyectos (p)**, como **numero_resolucion_exencion**, **ano_resolucion_exencion**, **titulo**, **codigo_proyecto**, y otros. También utiliza las tablas relacionadas (**ent_entidades**, **sb_dependencias**, etc.) para completar los datos.
4. **Variables de reemplazo:** La consulta emplea **REPLACE** para insertar valores específicos en los textos largos, tales como el número de resolución, la fecha, y otros detalles específicos del proyecto (**p.numero_resolucion_exencion**, **e.nombre_entidad**, etc.).

5. **Fragmentación de información:** El texto considerando está dividido en múltiples partes (**considerando_parte1**, **considerando_parte2**, etc.) para evitar problemas de tamaño en Oracle.
6. **Subconsultas:** Hay varias subconsultas, como la que obtiene el promedio del puntaje de evaluaciones o el valor de ciertas definiciones (**v.descripcion_valor_definicion**), lo cual le agrega profundidad a la consulta, pero también aumenta su complejidad.

Sección de Campos (Fields)

```
<field name="SIZE_CONSIDERANDO" class="java.lang.Long"/>
<field name="CONSIDERANDO_PARTE1" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE2" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE3" class="java.lang.String"/>
<field name="CONSIDERANDO_PARTE4" class="java.lang.String"/>
<field name="CONSIDERANDO_NEGADO1" class="java.lang.String"/>
<field name="CONSIDERANDO_NEGADO2" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION" class="java.lang.String"/>
<field name="CONCEPTO_EVALUACION2" class="java.lang.String"/>
<field name="ARTICULOS" class="java.lang.String"/>
<field name="NOMBRE" class="java.lang.String"/>
<field name="CARGO" class="java.lang.String"/>
<field name="TIPO_PROYECTO" class="java.lang.String"/>
<field name="ANIO_ACTA" class="java.lang.String"/>
<field name="NUMERO_ACTA" class="java.lang.String"/>
<field name="FORMATO_VERSION" class="java.lang.String"/>
<field name="NORMATIVIDAD" class="java.lang.String"/>
```

Los campos (<field>) definen las variables que se utilizarán en el diseño visual del reporte. Algunos campos destacados son:

1. **SIZE_CONSIDERANDO** (java.lang.Long): Representa el tamaño en bytes del texto almacenado en el parámetro '**DEDUINVCON**', probablemente para información de longitud.
2. **CONSIDERANDO_PARTE1** a **CONSIDERANDO_PARTE4** (java.lang.String): Fragmentos del texto considerando, divididos para respetar las limitaciones de longitud de texto en Oracle.
3. **CONSIDERANDO_NEGADO1** y **CONSIDERANDO_NEGADO2** (java.lang.String): Textos personalizados de considerando cuando la evaluación del proyecto es negativa. Se obtienen al reemplazar etiquetas en los textos definidos por **DEDUINVCON1** y **DEDUINVCON2**.

4. **CONCEPTO_EVALUACION** y **CONCEPTO_EVALUACION2** (java.lang.String): Partes del texto de evaluación del concepto, también segmentadas para cumplir con las restricciones de longitud.
5. **ARTICULOS** (java.lang.String): Texto extraído y modificado del parámetro DEDUINVARTN, que probablemente contiene referencias legales o normativas aplicables al proyecto.
6. **NOMBRE** (java.lang.String): Nombre de la persona o entidad responsable, obtenido de **NOM_PERS**.
7. **CARGO** (java.lang.String): Cargo de la persona o entidad, obtenido del parámetro **CARGO_PERS**.
8. **TIPO_PROYECTO** (java.lang.String): Descripción del tipo de proyecto (por ejemplo, investigación, desarrollo, etc.).
9. **ANIO_ACTA** (java.lang.String): Año del acta de consejo donde se tomó la decisión sobre el proyecto.
10. **NUMERO_ACTA** (java.lang.String): Número del acta asociada a la decisión del proyecto.
11. **FORMATO_VERSION** (java.lang.String): Representa el formato o versión del informe, obtenido de una definición de código **CODIF_EST**.
12. **NORMATIVIDAD** (java.lang.String): Texto referente a la normativa aplicable, obtenido del parámetro **DEDUINVNORN**.

Estos campos son esenciales para estructurar y presentar la información obtenida en la consulta, asegurando que cada dato específico del proyecto se integre correctamente en el reporte.

Sección de Variables

```
<variable name="TOTAL_PAGINAS" class="java.lang.Integer">  
  <variableExpression><![CDATA[${PAGE_NUMBER}]]></variableExpression>  
</variable>
```

Finalmente, la variable **TOTAL_PAGINAS** usa **`\${PAGE\_NUMBER}`**, una variable interna de JasperReports, para mostrar el total de páginas del reporte.

Sección de Encabezado Pagina (PageHeader)

```
<pageHeader>
  <band height="58" splitType="Stretch">
    <textField>
      <reportElement x="150" y="0" width="182" height="31"
isRemoveLineWhenBlank="true" uuid="f8614e4d-50cc-4298-9419-963db51db2b1">
        <printWhenExpression><![CDATA[ $V{PAGE_NUMBER}
1]]></printWhenExpression>
      </reportElement>
      <textElement textAlignment="Center" verticalAlignment="Bottom">
        <font fontName="Arial" size="11" isBold="true"
isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA[ "RESOLUCIÓN No. DE " +
$P{VIGENCIA}]]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true">
      <reportElement positionType="Float" x="0" y="33" width="482" height="25"
isRemoveLineWhenBlank="true" uuid="c67a5398-6533-4f34-883c-e62e88eddc65">
        <printWhenExpression><![CDATA[ $V{PAGE_NUMBER}
1]]></printWhenExpression>
      </reportElement>
      <textElement textAlignment="Center" verticalAlignment="Top"
markup="styled">
        <font fontName="Arial" size="9" isBold="false" isItalic="true"
isStrikeThrough="false" isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA[ " \\"Por la cual se niega la calificación a
un proyecto como de "+ $F{TIPO_PROYECTO} +" \\" "]]></textFieldExpression>
    </textField>
    <image scaleImage="RealSize">
      <reportElement x="0" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="6bb89b95-dc70-47d1-b93c-6ef3618b115a"/>
      <imageExpression><![CDATA[ $P{IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_SECUNDARIA_BASE64}.getBytes()))
:
new ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))]]></imageExpression>
    </image>
    <image scaleImage="RealSize">
      <reportElement x="332" y="0" width="150" height="31"
isRemoveLineWhenBlank="true" uuid="894c6ae9-3927-4e2c-a5fc-98e1f8f15072"/>
      <imageExpression><![CDATA[ $P{IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))
: null]]></imageExpression>
    </image>
  </band>
</pageHeader>
```

La sección pageHeader está configurada para mostrar en las páginas donde \$V{PAGE_NUMBER}

> 1. Explicamos el código anterior:

1. **Condiciones de Impresión:** La etiqueta

```
<printWhenExpression><![CDATA[ $V{PAGE_NUMBER} > 1]]></printWhenExpression>
```

en los elementos textField asegura que estos campos solo se muestren a partir de la segunda página. Esto es útil si quieres que el encabezado sea diferente en la primera

página. Si se desea que algo se muestre en todas las páginas, se puede eliminar esta condición.

2. Imágenes Condicionales:

- La primera imagen se muestra si **\$P{IMG_SECUNDARIA_BASE64}** no es null; de lo contrario, utiliza **\$P{IMG_BASE64}**.
- La segunda imagen en la misma banda tiene una expresión condicional similar, pero su **imageExpression** solo muestra **\$P{IMG_BASE64}** si **\$P{IMG_SECUNDARIA_BASE64}** no es null.

3. Centrado de Texto:

- Los elementos **<textField>** utilizan **textAlignment="Center"**, lo cual está bien para centrar el texto en el espacio asignado. Para un resultado más preciso, verifica que el ancho y posición del campo sean consistentes con el diseño del encabezado.

4. Control de Tamaño y Desbordamiento:

- El segundo **textField** tiene el atributo **isStretchWithOverflow="true"**, lo cual permite que el campo se ajuste si el texto es más extenso que el espacio asignado. Esto es útil si el contenido de **\$F{TIPO_PROYECTO}** puede variar en longitud, pero si encuentras problemas de alineación, podrías ajustar el ancho o el tamaño de fuente.

Sección de Detalles (Details)

```
<detail>
  <band height="114" splitType="Stretch">
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
      <reportElement positionType="Float" x="0" y="32" width="482" height="20"
isRemoveLineWhenBlank="true" isPrintWhenDetailOverflows="true" uuid="4e48b21b-b018-492b-
a9ee-9232a9b8ce97"/>
      <textElement textAlignment="Center" verticalAlignment="Middle"
markup="styled">
        <font fontName="Arial" size="10" isBold="false" isItalic="true"
isPdfEmbedded="false"/>
      </textElement>
      <textFieldExpression><![CDATA[" \Por la cual se niega la calificación a
un proyecto como de "+ $F{TIPO_PROYECTO} +" \"]]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true">
      <reportElement positionType="Float" x="0" y="59" width="482" height="30"
uuid="03a93dcf-f70b-44a0-b3e9-ba35d04c1dc2"/>
    </textField>
  </band>
</detail>
```

```

markup="styled">
    <textElement
        textAlignment="Center"
        verticalAlignment="Top"
        isPdfEmbedded="false">
        <font
            fontName="Arial"
            size="11"
            isBold="true"
        </font>
        <textFieldExpression><![CDATA["LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E
INNOVACIÓN
DEL
$P{nombre_entidad_SIGP_Completo}.toUpperCase().trim()]]></textFieldExpression>
    </textField>
    <reportElement
        x="150" y="0" width="182" height="26" uuid="2eb37195-
acdb-4b8d-aa2b-2b7e4ebd1bbb"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
    <font
        fontName="Arial"
        size="11"
        isBold="true"
    </font>
    <textFieldExpression><![CDATA["RESOLUCIÓN No.
DE " +
$P{VIGENCIA}]]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
    <reportElement positionType="Float" x="0" y="95" width="482" height="15"
uuid="220c49e8-9c00-400e-b720-aa986525cdf6"/>
    <textElement
        textAlignment="Center"
        verticalAlignment="Top"
    </textElement>
    <font fontName="Arial" size="11" isPdfEmbedded="false"/>
    </font>
    <textFieldExpression><![CDATA[$F{NORMATIVIDAD}.trim()]]></textFieldExpression>
    </textField>
</band>
<band height="37">
    <staticText>
    <reportElement positionType="Float" x="0" y="5" width="482" height="15"
uuid="e5546bfd-0044-46a9-985b-3a425d094e15"/>
    <textElement textAlignment="Center" verticalAlignment="Top">
    <font
        fontName="Arial"
        size="11"
        isBold="true"
    </font>
    <text><![CDATA[CONSIDERANDO:]]></text>
    </staticText>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
    <reportElement positionType="Float" stretchType="RelativeToBandHeight"
x="0" y="22" width="482" height="15" isPrintWhenDetailOverflows="true" uuid="d1330410-662f-
41db-9996-9223bc068732"/>
    <textElement
        textAlignment="Justified"
        verticalAlignment="Top"
    </textElement>
    <font fontName="Arial" size="11" isPdfEmbedded="false"/>
    </font>
    <textFieldExpression><![CDATA[$F{SIZE_CONSIDERANDO} > 12000
?
$F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_PARTE2}).concat($F{CONSIDERANDO_PARTE3}).conc
at($F{CONSIDERANDO_PARTE4}).concat($F{CONSIDERANDO_NEGADO1})
: $F{SIZE_CONSIDERANDO} > 8000
?
$F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_PARTE2}).concat($F{CONSIDERANDO_PARTE3}).conc
at($F{CONSIDERANDO_NEGADO1})
: $F{SIZE_CONSIDERANDO} > 4000
? $F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_PARTE2}).concat($F{CONSIDERANDO_NEGADO1})
: $F{CONSIDERANDO_PARTE1}.concat($F{CONSIDERANDO_NEGADO1}]]></textFieldExpression>
    </textField>
</band>

```

```

    <band height="15">
        <textField isStretchWithOverflow="true" isBlankWhenNull="true">
            <reportElement positionType="Float" x="20" y="0" width="442" height="15"
            uuid="08673754-f3bb-42e6-bd91-ecc3cba2c104"/>
            <textField textAlignment="Justified" verticalAlignment="Top"
            markup="styled">
                <font fontName="Arial" size="11" isItalic="true"
                isUnderline="false"/>
                <textFieldExpression><![CDATA[{F{CONCEPTO_EVALUACION} != null
                ? ({F{CONCEPTO_EVALUACION} != null ?
                "\".concat({F{CONCEPTO_EVALUACION}).trim().concat({F{CONCEPTO_EVALUACION2}).trim().concat("
                \").trim() : "\".concat({F{CONCEPTO_EVALUACION}).trim().concat("\").trim()
                : "Sin registro de concepto de evaluación"]}></textFieldExpression>
            </textField>
        </band>
        <band height="15">
            <textField isStretchWithOverflow="true" isBlankWhenNull="true">
                <reportElement positionType="Float" x="0" y="0" width="482" height="15"
                uuid="25c2a1cf-bc9c-4e11-8d0b-177aaeabcacf"/>
                <textField textAlignment="Justified" verticalAlignment="Top"
                markup="html">
                    <font fontName="Arial" size="11"/>
                    </textField>

                <textFieldExpression><![CDATA[{F{CONSIDERANDO_NEGADO2}.trim()}]></textFieldExpression>
            </textField>
        </band>
        <band height="84">
            <textField isStretchWithOverflow="true" isBlankWhenNull="true">
                <reportElement positionType="Float" x="0" y="20" width="482" height="15"
                uuid="b565131a-1cc4-4097-9830-44885613330e"/>
                <textField textAlignment="Justified" markup="html">
                    <font fontName="Arial" size="11"/>
                    </textField>

                <textFieldExpression><![CDATA[{F{ARTICULOS}.trim()}]></textFieldExpression>
            </textField>
            <staticText>
                <reportElement positionType="Float" x="0" y="0" width="482" height="15"
                uuid="642f3a61-ed60-4f8c-8501-49f0d35e594d"/>
                <textField textAlignment="Center">
                    <font fontName="Arial" size="11" isBold="true"/>
                    </textField>
                <text><![CDATA[RESUELVE:]]></text>
            </staticText>
            <textField>
                <reportElement positionType="Float" x="0" y="51" width="482" height="15"
                uuid="61955eb0-22b7-4ff2-9a59-0b9741a20dfd"/>
                <textField>
                    <font fontName="Arial" size="11"/>
                    </textField>
                <textFieldExpression><![CDATA["Dada en Bogotá, D.C., a los, días del
                mes de de "+ Calendar.getInstance().get(Calendar.YEAR)]]></textFieldExpression>
            </textField>
        </band>
    </detail>

```

Esta sección define el detalle de un reporte, donde se estructuran y formatean distintos elementos de texto que aparecerán en el cuerpo del documento. Aquí tienes una descripción de cada banda y sus elementos:

1. Primera Banda (Altura 114)

- Contiene varios campos de texto que presentan información introductoria y el encabezado de la resolución:
 - **Primer Campo:** Muestra el mensaje "Por la cual se niega la calificación a un proyecto como de [TIPO_PROYECTO]", centrado y en cursiva, donde [TIPO_PROYECTO] es una variable que se inserta dinámicamente.
 - **Segundo Campo:** Despliega el nombre completo de la entidad en mayúsculas (proveniente del parámetro **nombre_entidad_SIGP_Completo**), alineado al centro y en negritas.
 - **Tercer Campo:** Muestra la resolución, en el formato "RESOLUCIÓN No. DE [VIGENCIA]", donde [VIGENCIA] se inserta dinámicamente.
 - **Cuarto Campo:** Muestra el texto de normatividad asociado al proyecto (NORMATIVIDAD), alineado al centro y en HTML.

2. Segunda Banda (Altura 37)

- Comienza con el encabezado "**CONSIDERANDO:**", en negritas y centrado.
- **Campo de Texto Considerando:** Muestra el texto de considerando, dividido en partes según su tamaño (SIZE_CONSIDERANDO):
 - Si el tamaño es mayor a 12000 bytes, se concatenan CONSIDERANDO_PARTE1, CONSIDERANDO_PARTE2, CONSIDERANDO_PARTE3, CONSIDERANDO_PARTE4, y CONSIDERANDO_NEGADO1.
 - Para tamaños menores, se reducen los fragmentos concatenados de manera similar.

3. Tercera Banda (Altura 15)

- **Campo de Texto Concepto de Evaluación:** Muestra el concepto de evaluación:

- Si el campo CONCEPTO_EVALUACION no es nulo, muestra el contenido concatenado de CONCEPTO_EVALUACION y, si está presente, CONCEPTO_EVALUACION2, con comillas alrededor.
- Si ambos son nulos, despliega el mensaje "Sin registro de concepto de evaluación".

4. Cuarta Banda (Altura 15)

- **Campo de Texto Considerando Negado 2:** Muestra el texto en CONSIDERANDO_NEGADO2, justificado y en formato HTML.

5. Quinta Banda (Altura 84)

- **Campo de Artículos:** Despliega el contenido del campo **ARTICULOS**, en texto justificado y HTML.
- **Texto Estático "RESUELVE:"**: Este texto está centrado, en negritas, y marca la sección de resolución del documento.
- **Campo de Texto Fecha:** Presenta la fecha con el formato "Dada en Bogotá, D.C., a los [días] días del mes de [mes] de [año]", usando el año actual obtenido dinámicamente.

Sección de Pie de Pagina (PageFooter)

```
<pageFooter>
  <band height="42" splitType="Stretch">
    <textField evaluationTime="Page">
      <reportElement x="203" y="22" width="50" height="20" forecolor="#999999"
        uuid="3bfacc06-77c4-47f5-9f26-883213d073fc"/>
      <textField>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textField>
      <textFieldExpression><![CDATA["Página " + ${PAGE_NUMBER} + " de
        "]]></textFieldExpression>
      </textField>
    <textField evaluationTime="Report">
      <reportElement x="253" y="22" width="18" height="20" forecolor="#999999"
        uuid="42ba6cac-8dc3-4ca5-a6af-ee42180180e8"/>
      <textField>
        <font fontName="Arial" size="8" isPdfEmbedded="false"/>
      </textField>
      <textFieldExpression><![CDATA[${PAGE_NUMBER}]]></textFieldExpression>
      </textField>
    <textField evaluationTime="Page">
      <reportElement x="0" y="0" width="112" height="30" forecolor="#999999"
        uuid="5195747f-e33d-4169-8802-6a38bf19d755"/>
```



```
<textElement verticalAlignment="Middle">
  <font fontName="Arial" size="8" isPdfEmbedded="false"/>
</textElement>

<textFieldExpression><![CDATA[ $F{FORMATO_VERSION} ]]></textFieldExpression>
</textField>
</band>
</pageFooter>
```

En esta sección de **pageFooter**, hay varios elementos importantes:

1. Número de Página:

- El primer **textField** muestra el texto "Página " seguido del número de la página actual. Utiliza la expresión **\$V{PAGE_NUMBER}** para obtener el número de la página actual.
- El segundo **textField** muestra el total de páginas utilizando también **\$V{PAGE_NUMBER}**, lo cual puede ser útil si deseas mostrar "Página X de Y", aunque en este caso parece que no hay un total definido.

2. Formato de Versión:

- El tercer **textField** muestra el valor del campo **FORMATO_VERSION**, que puede ser un indicador de la versión del informe. Este campo se muestra en un formato de texto que se alinea verticalmente al centro.

Sección de Resumen (Summary)

```
<summary>
  <band height="161" splitType="Stretch">
    <staticText>
      <reportElement positionType="Float" x="0" y="0" width="482" height="15"
      uuid="8a85e609-d92c-4b1b-81bc-83b4a3347de9"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="Arial" size="11" isBold="true"/>
      </textElement>
      <text><![CDATA[NOTIFÍQUESE Y CÚMPLASE,]]></text>
    </staticText>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
      <reportElement positionType="Float" x="0" y="58" width="482" height="15"
      uuid="59677bfd-ef0f-4227-8d0c-a67ba3c1b1b8"/>
      <textElement textAlignment="Center" verticalAlignment="Middle"
      markup="styled">
        <font fontName="Arial" size="11" isBold="true"/>
      </textElement>
      <textFieldExpression><![CDATA[ $F{NOMBRE} ]]></textFieldExpression>
    </textField>
    <textField isStretchWithOverflow="true" isBlankWhenNull="true">
```

```

<reportElement positionType="Float" x="0" y="73" width="482" height="15"
uuid="f00a388b-16fb-4a4b-86c2-75df5b8a08fa"/>
<textElement textAlignment="Center" verticalAlignment="Middle"
markup="styled">
    <font fontName="Arial" size="11" isBold="true"/>
</textElement>
<textFieldExpression><![CDATA[{F{CARGO}}]></textFieldExpression>
</textField>
<textField isBlankWhenNull="true">
    <reportElement positionType="Float" x="32" y="121" width="450"
height="10" uuid="f1c48d70-2392-4366-a7cc-2d26b2518aba"/>
    <textElement>
        <font fontName="Arial" size="7"/>
    </textElement>
    <textFieldExpression><![CDATA[{P{func_vto_bno}}]></textFieldExpression>
</textField>
<staticText>
    <reportElement positionType="Float" x="0" y="151" width="37" height="10"
forecolor="#000000" uuid="2c54923d-5b62-4b33-b40c-897ac4823de2"/>
    <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
    </textElement>
    <text><![CDATA[Proyecto:]]></text>
</staticText>
<textField isBlankWhenNull="true">
    <reportElement positionType="Float" x="32" y="131" width="450"
height="10" uuid="ee32bf57-21ab-48ba-850b-c15453254657"/>
    <textElement>
        <font fontName="Arial" size="7"/>
    </textElement>
    <textFieldExpression><![CDATA[{P{func_reviso}}]></textFieldExpression>
</textField>
<staticText>
    <reportElement positionType="Float" x="0" y="131" width="32" height="10"
forecolor="#000000" uuid="a9d75d1b-7700-49e6-9e6d-39664c0a9b6f"/>
    <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
    </textElement>
    <text><![CDATA[Revisó:]]></text>
</staticText>
<staticText>
    <reportElement positionType="Float" x="0" y="121" width="32" height="10"
forecolor="#000000" uuid="023d4188-ae3d-4a4a-aa7a-45c17afa6345"/>
    <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
    </textElement>
    <text><![CDATA[Vto. Bo.:]]></text>
</staticText>
<textField isBlankWhenNull="true">
    <reportElement positionType="Float" x="37" y="151" width="445"
height="10" uuid="af12291a-5fb7-4411-bdca-e4abff4d6765"/>
    <textElement>
        <font fontName="Arial" size="7"/>
    </textElement>
    <textFieldExpression><![CDATA[{P{func_proyecto}}]></textFieldExpression>
</textField>
<textField isBlankWhenNull="true">

```

```
<reportElement positionType="Float" x="32" y="141" width="450"
height="10" uuid="c060bb93-1b91-42ac-8b7d-ea5eac312955"/>
<textElement>
  <font fontName="Arial" size="7"/>
</textElement>
<textFieldExpression><![CDATA[ $P{func_reviso2}]]></textFieldExpression>
</textField>
<staticText>
  <reportElement positionType="Float" x="0" y="141" width="32" height="10"
forecolor="#000000" uuid="1f88e47a-9eec-48a8-8591-3888313707ed"/>
  <textElement verticalAlignment="Middle">
    <font fontName="Arial" size="7" isBold="false"
isPdfEmbedded="false"/>
  </textElement>
  <text><![CDATA[Revisó:]]></text>
</staticText>
</band>
</summary>
```

La sección **summary** en un informe de JasperReports se utiliza para mostrar información de cierre o un resumen del contenido del informe, generalmente después de que se han presentado todos los detalles y datos en las secciones anteriores. Esta sección es útil para presentar conclusiones, notas finales o cualquier información importante que se quiera resaltar al final del documento.

En la sección de **summary**, hay varios elementos clave:

5. Texto Estático:

- El primer **staticText** muestra el mensaje "**NOTIFÍQUESE Y CÚMPLASE**," centrado y en negrita, lo que indica que es un aviso importante para el lector. Está ubicado en la parte superior de la sección y utiliza una fuente Arial de tamaño 11.

6. Nombre y Cargo:

- Se presentan campos dinámicos (**textField**) que muestran el nombre (**\$F{NOMBRE}**) y el cargo (**\$F{CARGO}**) de la persona que revisó el informe. Ambos están centrados y en negrita, asegurando que destaquen en la página.

7. Revisiones y Proyectos:

- Se incluyen etiquetas de texto estático y campos de texto que indican quién revisó y proyectó el informe, utilizando los parámetros **\$P{func_reviso}**, **\$P{func_reviso2}**, y **\$P{func_proyecto}**. Estos campos se presentan en una fuente más pequeña (tamaño 7), lo que indica que son detalles adicionales que

son relevantes pero que no necesitan el mismo nivel de atención que el encabezado.

8. Formato:

- Todos los textos estáticos y campos de texto utilizan la fuente Arial, y se asegura que la alineación sea consistente. Las etiquetas de texto tienen un color negro, mientras que los campos están en un tamaño más pequeño, lo que crea un contraste adecuado.

Ajustar Validación visualización de evaluación de proyectos (fecha de cierre)

El requerimiento consistió en permitir mostrar la evaluación de proyectos/programas al cambiar la vigencia siempre y cuando esta estuviera aún vigente.

El sistema SIGP valida por año de vigencia actual contra el año de creación de la evaluación, es decir, solo muestra las evaluaciones que hayan sido creadas el mismo año actual. Adicional a la anterior vigencia, se agregaron dos condiciones:

11. Validar que la fecha de cierre de evaluación de la convocatoria sea mayor a la fecha actual
12. Validar que si la evaluación está habilitada individualmente y si la fecha de cierre de la evaluación es mayor a la fecha actual.

Para este ajuste se modificó el filtro que realizaba la carga de los proyectos según el año de asignación:

1. Se modifica condicional

ANTES

```
//Filtro por AÑO de asignación
Object anio = filters.get("anio");
if (anio != null) {
    String anio2 = String.valueOf((Long) anio);
    lista.add(cb.equal(cb.function("TO_CHAR",
        proyectoEvaluacion.get("fechaCreacion"), cb.literal("YYYY"), anio2)));
}
```

DESPUES

```
/**
 * Filtro por Año de creación de la evaluación OR fecha de cierre de
 * evaluación de la convocatoria OR habilitación individual de
 * evaluación y fecha de cierre de evaluación
 */
Object anio = filters.get("anio");
if (anio != null) {
    String anio2 = String.valueOf((Long) anio);
    List<Predicate> predicatesFechaVigenciaCierre = new ArrayList<>();

    // Agregar la condición de que el año de la fecha de Creación de la evaluación sea mayor
    a la vigencia actual
    predicatesFechaVigenciaCierre.add(cb.equal(cb.function("TO_CHAR", String.class,
    proyectoEvaluacionRoot.get("fechaCreacion"), cb.literal("YYYY")), anio2));

    // Agregar la condición de fechaCierreEvaluacion IS NOT NULL y fechaCierreEvaluacion >
    SYSDATE
    predicatesFechaVigenciaCierre.add(cb.and(

        cb.isNull(convocatoriaIdJoin.get(Convocatoria_.fechaCierreEvaluacion)),

        cb.greaterThan(convocatoriaIdJoin.get(Convocatoria_.fechaCierreEvaluacion), new Date())
    ));

    // Agregar la condición de que si esta habilitada la evaluación individual y
    fechaCierreEvaluacion > SYSDATE
    predicatesFechaVigenciaCierre.add(cb.and(
        cb.equal(proyectoEvaluacionRoot.get("habilitarEvaluacion"), "S"),

        cb.greaterThan(proyectoEvaluacionRoot.get(ProyectoEvaluacion_.fechaCierreEvaluacion),
        new Date())
    ));

    // Combinar todas las condiciones usando OR
    Predicate finalPredicate = cb.or(predicatesFechaVigenciaCierre.toArray(new
    Predicate[0]));
    lista.add(finalPredicate);
}
```

Este fragmento de código implementa un filtro para buscar registros basados en el año de creación de la evaluación, la fecha de cierre de la convocatoria o la habilitación individual de la evaluación, combinando estos criterios en una lista de condiciones (predicatesFechaVigenciaCierre). A continuación, se explica cada parte:

1. **Descripción del Filtro** El comentario inicial explica el propósito del filtro: se utiliza para buscar registros de evaluación según el año de creación, la fecha de cierre de la evaluación en una convocatoria o la habilitación individual de la evaluación junto con la fecha de cierre de la evaluación.
2. **Obtención del Año**

```
Object anio = filters.get("anio");
```

Se recupera el valor del año (**anio**) del mapa de filtros (**filters**). Este valor se utiliza para comparar con el año de la fecha de creación de la evaluación.

3. Conversión del Año

```
String anio2 = String.valueOf((Long) anio);
```

El valor **anio** se convierte a una cadena (**anio2**). Esta conversión asegura que el valor esté en el formato adecuado para la comparación de cadenas en el filtro de fecha.

4. Lista de Predicados

```
List<Predicate> predicatesFechaVigenciaCierre = new ArrayList<>();
```

Se crea una lista de **Predicate** llamada **predicatesFechaVigenciaCierre**, que almacenará las condiciones (predicados) de filtrado.

5. Condición de Año de Creación

```
predicatesFechaVigenciaCierre.add(cb.equal(
    cb.function("TO_CHAR", String.class, proyectoEvaluacionRoot.get("fechaCreacion"),
    cb.literal("YYYY")),
    anio2
));
```

- Aquí, se añade un predicado que verifica que el año de la fecha de creación de la evaluación (**fechaCreacion**) sea igual al valor de **anio2**.
- Para esto, se utiliza la función **TO_CHAR** de SQL para extraer el año en formato de cadena ("YYYY") de **fechaCreacion** y compararlo con **anio2**.

6. Condición de Fecha de Cierre de Evaluación

```
predicatesFechaVigenciaCierre.add(cb.and(
    cb.isNull(convocatoriaIdJoin.get(Convocatoria_.fechaCierreEvaluacion)),
    cb.greaterThan(convocatoriaIdJoin.get(Convocatoria_.fechaCierreEvaluacion), new Date())
));
```

- Este bloque añade una condición combinada (**AND**) que verifica dos aspectos:
 1. **Fecha no nula:** `fechaCierreEvaluacion` de la convocatoria no es nula (**IS NOT NULL**).
 2. **Fecha posterior a la actual:** La fecha de cierre de la evaluación es posterior a la fecha actual (**SYSDATE** en SQL, representado aquí como **new Date()** en Java).
- Esta condición permite obtener solo las evaluaciones con fecha de cierre en el futuro.

En resumen, el filtro construido en este código permite seleccionar evaluaciones donde el año de creación coincide con un valor específico o donde la fecha de cierre de la evaluación es futura y está configurada en la convocatoria. Estos predicados se pueden utilizar posteriormente en una consulta para filtrar registros según los criterios definidos.

Implementar que por convocatoria se habilite si los funcionarios internos pueden modificar la calificación evaluación, siempre y cuando estos pertenezcan a un grupo específico

Para este requerimiento se solicita implementar la funcionalidad de habilitar a un grupo de funcionarios internos de Minciencias la potestad de modificar y evaluar las calificaciones de evaluación de una evaluación asignada a los proyectos.

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

- Se crea un campo por convocatoria que permite indicar si se habilita o no para funcionarios internos, la modificación de la evaluación de los proyectos
- Se crean dos parámetros del sistema encargados de indicar el nombre del grupo al cual van a pertenecer los funcionarios con los permisos para modificar las evaluaciones
- Se agrega logica que permita consultar los usuarios que pueden modificar las calificaciones de evaluación, siempre y cuando estos pertenezcan a un grupo específico que está parametrizado

Habilitar funcionarios por Convocatoria

- Creación de nuevo campo en la tabla SGP_CONVOCATORIAS

```
-- Add/modify columns
alter table SGP_CONVOCATORIAS add habilitar_eval_funcionario VARCHAR2(1) default 'N';
-- Add comments to the columns
comment on column SGP_CONVOCATORIAS.habilitar_eval_funcionario
is 'Indica si la convocatoria esta habilitada para que funcionarios internos puedan modificar
las evaluaciones de los proyectos';
```

- Mapear nuevo campo en la entidad de la convocatoria (Convocatoria.java)

```
@Column(name = "HABILITAR_EVAL_FUNCIONARIO")
private String habilitarEvalFuncionario;

// GETTER and SETTER
public String getHabilitarEvalFuncionario() {
    return habilitarEvalFuncionario;
}

public void setHabilitarEvalFuncionario(String habilitarEvalFuncionario) {
    this.habilitarEvalFuncionario = habilitarEvalFuncionario;
}
```

- Mapear nuevo campo en DTO de la convocatoria (ConvocatoriaDTO.java)

```
@Bind(name = "habilitarEvalFuncionario")
private String habilitarEvalFuncionario;

public String getHabilitarEvalFuncionario() {
    return habilitarEvalFuncionario;
}

private Boolean habilitarEvalFuncionarioBool;

// GETTER and SETTER

public void setHabilitarEvalFuncionario(String habilitarEvalFuncionario) {
    if (habilitarEvalFuncionario != null && habilitarEvalFuncionario.equals("S")) {
        this.habilitarEvalFuncionarioBool = true;
    }
    this.habilitarEvalFuncionario = habilitarEvalFuncionario;
}

public Boolean getHabilitarEvalFuncionarioBool() {
    return habilitarEvalFuncionarioBool;
}

public void setHabilitarEvalFuncionarioBool(Boolean habilitarEvalFuncionarioBool) {
    if (habilitarEvalFuncionarioBool) {
        this.habilitarEvalFuncionario = "S";
    }
}
```



```
    } else {  
        this.habilitarEvalFuncionario = "N";  
    }  
    this.habilitarEvalFuncionarioBool = habilitarEvalFuncionarioBool;  
}
```

Este código permite manejar el estado de habilitación de un funcionario en dos formatos (**String** y **Boolean**), facilitando la integración con otros componentes o interfaces. La lógica asegura que los valores de ambas variables (**habilitarEvalFuncionario** y **habilitarEvalFuncionarioBool**) se actualicen en función de los cambios del otro.

- Creación de grupos del sistema en donde van a ser asignados los usuarios (funcionarios internos) quienes podrán modificar las evaluaciones de los proyectos

```
-- Grupos  
insert into sb_grupos (GRUPO_ID, GRUPO, DESCRIPCION_GRUPO, ESTADO_GRUPO, CATEGORIA_GRUPO,  
    APLICACION, GRUPO_PADRE, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR,  
    FECHA_MODIFICACION)  
values (null, 'SGPS_EVAL_INT_PRY', 'Grupo de Funcionarios Internos que modifican las  
    evaluaciones calificaciones de proyectos/programas', 'H', 'U', 'SIGP_WEB', null, 'HERCULES',  
    to_date('02-02-2024', 'dd-mm-yyyy'), null, null);  
  
insert into sb_grupos (GRUPO_ID, GRUPO, DESCRIPCION_GRUPO, ESTADO_GRUPO, CATEGORIA_GRUPO,  
    APLICACION, GRUPO_PADRE, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR,  
    FECHA_MODIFICACION)  
values (null, 'SGPS_EVAL_INT_EST', 'Grupo de Funcionarios Internos que modifican las  
    evaluaciones calificaciones de estímulos tributario', 'H', 'U', 'SIGP_WEB', null, 'HERCULES',  
    to_date('02-02-2024', 'dd-mm-yyyy'), null, null);
```

- Creación de parámetros del sistema para indicar los grupos que habilitan a los funcionarios internos

```
-- Parametros del sistema  
insert into adm_parametros_sistema (CODIGO_PARAMETRO, DESCRIPCION, TIPO_PARAMETRO, FECHA,  
    VALOR, CADENA, TIPO_DEFINICION, VALOR_DEFINICION_ID, USUARIO_CREADOR, FECHA_CREACION,  
    USUARIO_MODIFICADOR, FECHA_MODIFICACION, DETALLE)  
values ('EVALINTPRY', 'Indica el grupo al que pertenecen los Funcionarios Internos que pueden  
    modificar las evaluaciones calificación de Proyectos/Programas', 'A', null, null,  
    'SGPS_EVAL_INT_PRY', null, null, 'SIGP_WEB', to_date('02-02-2024', 'dd-mm-yyyy'), null, null,  
    '<CLOB>');  
  
insert into adm_parametros_sistema (CODIGO_PARAMETRO, DESCRIPCION, TIPO_PARAMETRO, FECHA,  
    VALOR, CADENA, TIPO_DEFINICION, VALOR_DEFINICION_ID, USUARIO_CREADOR, FECHA_CREACION,  
    USUARIO_MODIFICADOR, FECHA_MODIFICACION, DETALLE)  
values ('EVALINTEST', 'Indica el grupo al que pertenecen los Funcionarios Internos que pueden  
    modificar las evaluaciones calificación de Beneficios Tributarios', 'A', null, null,  
    'SGPS_EVAL_INT_EST', null, null, 'SIGP_WEB', to_date('02-02-2024', 'dd-mm-yyyy'), null, null,  
    '<CLOB>');
```

Se crearon los parámetros:

2. **'EVALINTPRY'**: Indica el grupo al que pertenecen los funcionarios Internos que pueden modificar las evaluaciones calificación de Proyectos/Programas'
 3. **'EVALINTEST'**: Indica el grupo al que pertenecen los funcionarios Internos que pueden modificar las evaluaciones calificación de Beneficios Tributarios'
- En la pantalla de configuración de la convocatoria, se crea campo para indicar si se habilita o no que los funcionarios internos puedan modificar las evaluacion dentro de la convocatoria

```
<div class="row">
  <!-- Habilitar Evaluación a Funcionarios Minciencias -->
  <div class="col-lg-6">
    <div class="form-group">
      <p:outputLabel
value="#{msg['gestion.configuracion.convocatorias.habilitarEvalFuncionario.label']}"
                                styleClass="col-lg-6 control-label"
                                style="text-align: right"
                                for="habilitarEvalFuncionario"/>
      <div class="col-lg-5">
        <h:selectBooleanCheckbox id="habilitarEvalFuncionario"

value="#{convocatoriaSessController.selected.habilitarEvalFuncionarioBool}"

required="false"

requiredMessage="#{msg['gestion.configuracion.convocatorias.habilitarEvalFuncionario.require
d']}"

styleClass="js-switch"

                                >
      </h:selectBooleanCheckbox>
    </div>
  </div>
</div>
</div>
```

Este código en XHTML genera un elemento de interfaz en una página web que permite habilitar o deshabilitar la evaluación para funcionarios, usando un checkbox. Está diseñado con componentes de JSF (JavaServer Faces) y PrimeFaces, y utiliza un diseño basado en columnas para la presentación en pantalla.

Etiqueta para el checkbox (p:outputLabel):

- Se utiliza **p:outputLabel** para mostrar la etiqueta del checkbox.

- La propiedad value usa **`{msg['gestion.configuracion.convocatorias.habilitarEvalFuncionario.label']}`**, lo cual indica que el valor de la etiqueta se obtiene de un archivo de mensajes para soporte multilinguaje.
- **`styleClass="col-lg-6 control-label"`** aplica estilos **CSS** y ajusta el ancho de la etiqueta en la columna.
- La propiedad **`style="text-align: right"`** alinea la etiqueta a la derecha.
- **`for="habilitarEvalFuncionario"`** asocia la etiqueta con el checkbox mediante el id.

Checkbox (h:selectBooleanCheckbox):

- Este componente **`h:selectBooleanCheckbox`** genera un checkbox.
- La propiedad **`id="habilitarEvalFuncionario"`** asigna un identificador único al checkbox.
- **`value="{convocatoriaSessController.selected.habilitarEvalFuncionarioBool}"`** enlaza el valor del checkbox con la propiedad **`habilitarEvalFuncionarioBool`** del objeto **`selected`** en **`convocatoriaSessController`**. Este valor controla si el checkbox está marcado (habilitado) o no.
- **`required="false"`** indica que este campo no es obligatorio, pero si fuera requerido, el mensaje **`requiredMessage`** especifica el texto de error a mostrar.
- **`styleClass="js-switch"`** aplica estilos adicionales para personalizar el checkbox, usando la clase **`js-switch`** para darle un aspecto personalizado (como un interruptor visual).

Validaciones para habilitar modificación de evaluación a funcionarios internos

- Cambios en el controlador de la pantalla de calificación evaluación
(**`ProyectoBeneficiosEvaluacionCalificacionSessController.java`**)

Se crea campo que permite indicar si el usuario conectado está o no habilitado para calificar la evaluación

```
private boolean usuarioHabilitadoEvaluar;

public boolean isUsuarioHabilitadoEvaluar() {
    return usuarioHabilitadoEvaluar;
}

public void setUsuarioHabilitadoEvaluar(boolean usuarioHabilitadoEvaluar) {
    this.usuarioHabilitadoEvaluar = usuarioHabilitadoEvaluar;
}
```

Al momento de ingresar a la pantalla de calificación se llama al método **cargarListas()** para cargar todas las listas y atributos, a su vez este método llama a otro método (**consultarCierreEvaluacion()**) encargado de validar si la evaluación se encuentra o no cerrada aun.

```
/**
 * Metodo para consultar si la evaluación ha sido cerrada o no
 *
 * @param evaluacionProyecto
 */
public void consultarCierreEvaluacion(ProyectoEvaluacionDTO evaluacionProyecto) {

    evaluacionCerrada = Boolean.FALSE;
    if (evaluacionProyecto.getProyectoId().getCierreEvaluacion() != null &&
        evaluacionProyecto.getProyectoId().getCierreEvaluacion().equals("S")) {
        evaluacionCerrada = Boolean.TRUE;
    } else if
        (evaluacionProyecto.getProyectoId().getConvocatoriaId().getFechaCierreEvaluacion() != null) {

        SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy HH:mm");

        //fecha Cierre Evaluacion
        Calendar c1 = Calendar.getInstance();

        c1.setTime(evaluacionProyecto.getProyectoId().getConvocatoriaId().getFechaCierreEvaluacion());

        //fecha Actual
        Calendar c2 = Calendar.getInstance();

        // Si la fecha de cierre configurada en la CONVOCATORIA es menor a la fecha Actual
        if (c1.before(c2)) {
            evaluacionCerrada = Boolean.TRUE;

            //Validamos Fecha de Cierre Configurado por EVALUACION INDIVIDUAL
            if (evaluacionProyecto.getHabilitarEvaluacion() != null &&
                evaluacionProyecto.getHabilitarEvaluacion().equals("S")) {
                // Si la fecha de cierre es posterior a la fecha Actual, se abre evaluacion
                if (evaluacionProyecto.getFechaCierreEvaluacion() != null &&
                    evaluacionProyecto.getFechaCierreEvaluacion().after(new Date())) {
                    evaluacionCerrada = Boolean.FALSE;
                }
            }
        }

    }

    validarUsuarioHabilitadoEvaluar(evaluacionProyecto);
}
```

En este método se creó llama al método **validarUsuarioHabilitadoEvaluar()** para verificar si el usuario logueado está habilitado para evaluar

```
/**
 * Metodo que valida si el usuario logueado esta habilitado para evaluar
```

```

*
* @param evaluacionProyecto
*/
public void validarUsuarioHabilitadoEvaluar(ProyectoEvaluacionDTO evaluacionProyecto) {
    try {
        usuarioHabilitadoEvaluar = Boolean.FALSE;
        //Si la evaluacion no está cerrada
        if (!evaluacionCerrada) {
            if (evaluacionProyecto.getUsuarioSigp() != null
                &&
                evaluacionProyecto.getUsuarioSigp().equals(seguridadSessController.getSelected().getUsuario(
                ))) {
                usuarioHabilitadoEvaluar = Boolean.TRUE;
            } else {
                // SI la convocatoria tiene habilitada que los funcionarios
                internos puedan modificar la evaluacion
                if (evaluacionProyecto.getProyectoId().getConvocatoriaId() !=
                null
                &&
                evaluacionProyecto.getProyectoId().getConvocatoriaId().getHabilitarEvalFuncionarioBool()) {
                    /**
                     * Parametro que Indica el grupo al que pertenecen los
                     * Funcionarios Internos que pueden modificar las
                     * evaluaciones calificación de Estimulos Tributarios
                     */
                    String grupoHabilitado =
                    configuracionServiceFacade.obtenerParametroSistemaURL("EVALINTEST").getCadena();
                    if (grupoHabilitado != null &&
                    !grupoHabilitado.isEmpty()) {
                        Map<String, Object> filters = new HashMap<>();
                        filters.put("grupo", grupoHabilitado); // Filtro
                        por Grupo
                        filters.put("usuario",
                        seguridadSessController.getSelected().getUsuario()); // Filtro por usuario
                        int count =
                        administracionServiceFacade.usuarioGrupoListCount(filters);
                        if (count > 0) {
                            usuarioHabilitadoEvaluar =
                            Boolean.TRUE;
                        }
                    }
                }
            }
        }
        System.out.println("usuarioHabilitadoEvaluar: " + usuarioHabilitadoEvaluar);
    } catch (Exception ex) {
        LOG.error("Error al validar si el Usuario esta Habilitado para Evaluar ", ex);
    }
}

```

Este método, **validarUsuarioHabilitadoEvaluar**, verifica si el usuario que ha iniciado sesión tiene los permisos necesarios para realizar la evaluación de un proyecto específico. A continuación, se desglosa el funcionamiento del método:

Descripción General

- **Propósito:** Comprobar si el usuario autenticado tiene permiso para evaluar el proyecto que recibe como parámetro.
- **Parámetro:**
 - **evaluacionProyecto:** un objeto de tipo **ProyectoEvaluacionDTO** que contiene información sobre la evaluación del proyecto a validar.
- **Resultado:** Establece **usuarioHabilitadoEvaluar** como **true** o **false** dependiendo de si el usuario tiene permiso para evaluar.

Explicación Paso a Paso

1. Inicialización del Permiso:

- **usuarioHabilitadoEvaluar** se establece inicialmente en **false**, indicando que el usuario no tiene permiso para evaluar hasta que se valide lo contrario.

2. Comprobación de la Evaluación Cerrada:

- Se verifica si **evaluacionCerrada** es **false**. Esto indica que la evaluación está abierta y puede ser modificada.

3. Verificación del Usuario Asignado:

- Si el proyecto tiene un **usuarioSigg** asignado y este coincide con el usuario autenticado (**seguridadSessController.getSelected().getUsuario()**), se concede el permiso (**usuarioHabilitadoEvaluar = true**). Esto significa que el usuario autenticado es el evaluador asignado para el proyecto.

4. Permiso para Funcionarios Internos:

- Si el usuario autenticado no es el evaluador asignado, se verifica si la convocatoria del proyecto permite que funcionarios internos modifiquen la evaluación. Para ello:
 - Se comprueba si **habilitarEvalFuncionarioBool** es **true**, lo que indica que funcionarios internos pueden modificar la evaluación.

5. Validación del Grupo Autorizado:

- Se obtiene un parámetro de configuración llamado "**EVALINTTEST**" (parámetro que tiene el grupo de funcionarios habilitado para beneficios tributarios) que representa el grupo de usuarios internos autorizados.
- Si **grupoHabilitado** no es nulo o vacío, se crea un filtro que contiene:
 - **grupo**: el grupo autorizado para modificar.
 - **usuario**: el usuario autenticado.
- Se utiliza este filtro para contar cuántos registros existen en el grupo autorizado para el usuario autenticado (**administracionServiceFacade.usuarioGrupoListCount(filters)**).
- La consulta de los grupos a los que pertenece el usuario se hace a la tabla **SB_USUARIOS_GRUPOS**
- Si **count** es mayor que cero, significa que el usuario autenticado pertenece al grupo autorizado, por lo que se concede el permiso (**usuarioHabilitadoEvaluar = true**).

6. Registro de Resultado:

- Imprime el valor de **usuarioHabilitadoEvaluar** para verificar si el usuario está habilitado para evaluar.

7. Manejo de Errores:

- Si ocurre una excepción durante la ejecución, el error se registra en los logs con el mensaje "**Error al validar si el Usuario esta Habilitado para Evaluar**".

- Modificación de los botones para grabar los registros de evaluacion

ANTES

```
disabled="#{evaluacionCalificacionSessController.evaluacionCerrada}"
```

DESPUES

```
disabled="#{evaluacionCalificacionSessController.evaluacionCerrada or  
!evaluacionCalificacionSessController.usuarioHabilitadoEvaluar}"
```

La propiedad **disabled="#{evaluacionCalificacionSessController.evaluacionCerrada or !evaluacionCalificacionSessController.usuarioHabilitadoEvaluar}"** en el código controla la habilitación o deshabilitación del componente de la interfaz basado en ciertas condiciones lógicas. A continuación se detalla su funcionamiento:

a) Condición disabled:

- La propiedad **disabled** determina si el componente (**botón**) está deshabilitado o no.
- En este caso, la propiedad toma su valor de una expresión booleana que usa el operador **or**, por lo que el componente estará deshabilitado (**true**) si cualquiera de las siguientes condiciones es verdadero.

b) Condiciones en la expresión:

- **evaluacionCalificacionSessController.evaluacionCerrada**: Verifica si la evaluación está cerrada. Si **evaluacionCerrada** es **true**, el componente se desactiva para evitar que se realicen cambios en una evaluación finalizada.
- **!evaluacionCalificacionSessController.usuarioHabilitadoEvaluar**: Comprueba si el usuario no tiene permiso para evaluar. El signo **!** significa "no", por lo que si **usuarioHabilitadoEvaluar** es **false**, el componente se deshabilitará, restringiendo el acceso a usuarios no autorizados.

c) Resumen de la expresión:

- Si la evaluación está cerrada o el usuario no está habilitado para evaluar, el componente será deshabilitado.
- Esto asegura que solo los usuarios autorizados puedan interactuar con el componente y que no se realicen cambios en evaluaciones que ya han sido cerradas.

Esta lógica garantiza un control de acceso dinámico y evita modificaciones no deseadas en función del estado de la evaluación y de los permisos del usuario.

- Etiquetas de Internacionalización en español e inglés

```
#ES
gestion.configuracion.convocatorias.habilitarEvalFuncionario.label    = Habilitar Evaluación a
Funcionarios Minciencias
gestion.configuracion.convocatorias.habilitarEvalFuncionario.required  = Debe    Habilitar
Evaluación a Funcionarios Minciencias

#EN
gestion.configuracion.convocatorias.habilitarEvalFuncionario.label    = Enable Evaluation for
Minciencias Officials
gestion.configuracion.convocatorias.habilitarEvalFuncionario.required  = Must    Enable
Evaluation for Minciencias Officials
```

Parametrizar una cantidad máxima y mínima de proyectos que pueden presentarse por programa

Para este requerimiento se solicita poder parametrizar por convocatorias de programas, la cantidad mínima y máxima de proyectos que un programa puede asociar

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

- Creación de campos de mínimo y máximo de proyectos en la configuración de convocatorias. (Solo para convocatorias de Programas).
- Adición de validaciones en el formulario de proyectos que tengan en cuenta la nueva lógica de configuración de cantidad mínima y máxima de proyectos asociados al programa

Aplicativo de Gestión

Implementación de nuevos campos en la configuración de la convocatoria

- Creación de nuevo campo en la tabla SGP_CONVOCATORIAS

-- Add/modify columns

```
alter table SGP_CONVOCATORIAS add max_proyectos_programa NUMBER(15);
alter table SGP_CONVOCATORIAS add min_proyectos_programa NUMBER(15);
-- Add comments to the columns
comment on column SGP_CONVOCATORIAS.max_proyectos_programa
is 'Indica el máximo de proyectos a registrar por programa';
comment on column SGP_CONVOCATORIAS.min_proyectos_programa
is 'Indica el mínimo de proyectos a registrar por programa';
```

- Mapear nuevo campo en la entidad de la convocatoria (Convocatoria.java)

```
@Column(name = "MAX_PROYECTOS_PROGRAMA")
private Long maxProyectosPrograma;
@Column(name = "MIN_PROYECTOS_PROGRAMA")
private Long minProyectosPrograma;

// GETTER and SETTER

public Long getMaxProyectosPrograma() {
    return maxProyectosPrograma;
}

public void setMaxProyectosPrograma(Long maxProyectosPrograma) {
    this.maxProyectosPrograma = maxProyectosPrograma;
}

public Long getMinProyectosPrograma() {
    return minProyectosPrograma;
}

public void setMinProyectosPrograma(Long minProyectosPrograma) {
    this.minProyectosPrograma = minProyectosPrograma;
}
```

- Mapear nuevo campo en DTO de la convocatoria (ConvocatoriaDTO.java)

```
@Bind(name = "maxProyectosPrograma")
private Long maxProyectosPrograma;

@Bind(name = "minProyectosPrograma")
private Long minProyectosPrograma;

// GETTER and SETTER

public Long getMaxProyectosPrograma() {
    return maxProyectosPrograma;
}

public void setMaxProyectosPrograma(Long maxProyectosPrograma) {
    this.maxProyectosPrograma = maxProyectosPrograma;
}

public Long getMinProyectosPrograma() {
```

```
        return minProyectosPrograma;
    }

    public void setMinProyectosPrograma(Long minProyectosPrograma) {
        this.minProyectosPrograma = minProyectosPrograma;
    }
}
```

Este código en Java define dos atributos, **maxProyectosPrograma** y **minProyectosPrograma**, que representan límites máximo y mínimo de proyectos permitidos en un programa específico.

- En la pantalla de configuración de la convocatoria, se crean los campos para registro de mínimo y máximo proyectos

```
<!-- Mínimo de proyectos -->
<div class="col-lg-4">
    <div class="form-group">
        <p:outputLabel
value="#{msg['gestion.configuracion.convocatorias.minimoProyectos.label']}"
                                styleClass="col-lg-6 control-label"
                                style="text-align: right"
                                for="minProyectos"/>

        <div class="col-lg-5">
            <p:inputNumber id="minProyectos"

value="#{convocatoriaSessController.selected.minProyectosPrograma}"
                                required="false"

title="#{msg['gestion.configuracion.convocatorias.minimoProyectos.placeholder']}"

placeholder="#{msg['gestion.configuracion.convocatorias.minimoProyectos.placeholder']}"
                                decimalPlaces="0"
                                maxValue="9999999999999999"
                                styleClass="sbTooltip"
                                />

        </div>
    </div>
</div>
<!-- Máximo de proyectos -->
<div class="col-lg-4">
    <div class="form-group">
        <p:outputLabel
value="#{msg['gestion.configuracion.convocatorias.maximoProyectos.label']}"
                                styleClass="col-lg-6 control-label"
                                style="text-align: right"
                                for="maxProyectos"/>

        <div class="col-lg-5">
            <p:inputNumber id="maxProyectos"

value="#{convocatoriaSessController.selected.maxProyectosPrograma}"
                                required="false"

title="#{msg['gestion.configuracion.convocatorias.maximoProyectos.placeholder']}"

placeholder="#{msg['gestion.configuracion.convocatorias.maximoProyectos.placeholder']}"
                                decimalPlaces="0"
                                maxValue="9999999999999999"
                                />

    </div>
</div>
```

```

                                styleClass="sbTooltip"
                                />
        </div>
    </div>
</div>

```

Este código define en el formulario de configuración de convocatorias, permitir configurar los valores mínimo y máximo de proyectos permitidos para una convocatoria. Está estructurado en dos secciones, una para el valor mínimo y otra para el máximo. Cada sección utiliza componentes de PrimeFaces y etiquetas HTML para presentar los campos de entrada. A continuación, se describe cada parte del código:

Sección: Mínimo de proyectos

1. Etiqueta <p:outputLabel>:

- Muestra una etiqueta para el campo de entrada del valor mínimo de proyectos. El texto se obtiene de un archivo de mensajes **(#{msg['gestion.configuracion.convocatorias.minimoProyectos.label']})**, permitiendo la internacionalización y el manejo dinámico de etiquetas.
- **styleClass="col-lg-6 control-label"** aplica estilos específicos a la etiqueta para una adecuada alineación.
- **style="text-align: right"** alinea el texto a la derecha, adaptándose a una disposición visual estándar en formularios.

2. Campo <p:inputNumber>:

- Permite al usuario ingresar el número mínimo de proyectos. La propiedad **value** está enlazada a la variable **minProyectosPrograma** en el controlador **convocatoriaSessController.selected**.
- **required="false"** especifica que el campo no es obligatorio.
- **title** y **placeholder** proporcionan texto de ayuda para guiar al usuario sobre el propósito del campo, extraído del archivo de mensajes.
- **decimalPlaces="0"** indica que solo se permiten números enteros, evitando decimales.

- **maxValue="9999999999999999"** establece el valor máximo permitido, asegurando que el campo no acepte números extremadamente grandes.
- **styleClass="sbTooltip"** aplica una clase **CSS** para mostrar información adicional mediante un tooltip.

Sección: Máximo de proyectos

- **Etiqueta <p:outputLabel>:**
 - Similar a la sección anterior, muestra una etiqueta para el campo de entrada del valor máximo de proyectos, utilizando el archivo de mensajes para obtener el texto de la etiqueta.
 - Los atributos **styleClass** y **style** son los mismos que en la etiqueta anterior, alineando y aplicando el formato de visualización adecuado.
- **Campo <p:inputNumber>:**
 - Este campo es similar al del mínimo de proyectos, pero está enlazado a **maxProyectosPrograma** en el controlador.
 - Configurado con **decimalPlaces="0"** para aceptar solo valores enteros.
 - **maxValue="9999999999999999"** establece un valor máximo mayor, adecuado para el campo de máximo de proyectos.
 - **styleClass="sbTooltip"** aplica la clase **CSS** para el tooltip, manteniendo la consistencia visual.
- Etiquetas de Internacionalización en español e inglés

```
#ES
gestion.configuracion.convocatorias.cantidadPry.msg           = A continuación puede
configurar el mínimo o máximo de proyectos a registrar por programa. Además indicar si se
desea importar los datos de Programas a Proyectos.
gestion.configuracion.convocatorias.minimoProyectos.label     = Mínimo de proyectos
gestion.configuracion.convocatorias.minimoProyectos.placeholder = Ingrese el mínimo de
proyectos a registrar
gestion.configuracion.convocatorias.maximoProyectos.label     = Máximo de proyectos
gestion.configuracion.convocatorias.maximoProyectos.placeholder = Ingrese el máximo de
proyectos a registrar

#EN
```

```
gestion.configuracion.convocatorias.cantidadPry.msg = Next you can configure the
minimum or maximum number of projects to register per program. Also indicate if you want to
import the data from Programs to Projects.
gestion.configuracion.convocatorias.minimoProyectos.label = Minimum of projects
gestion.configuracion.convocatorias.minimoProyectos.placeholder = Enter the minimum number of
projects to register
gestion.configuracion.convocatorias.maximoProyectos.label = Maximum projects
gestion.configuracion.convocatorias.maximoProyectos.placeholder = Enter the maximum number of
projects to register
```

Formulario de Proyectos y Programas

Validaciones de Mínimo y Máximo registro de proyectos

- De igual manera, en este formulario se mapean los nuevos campos creados tanto en la entidad como en el DTO de la convocatoria
- Ajuste de método editar al registrar la cantidad de proyectos que va a asociar el programa.

Se agrega cuando es programa, validación que verifica la cantidad de proyectos.

```
if (proyectoDTO.getExencion().equals("R") && proyectoDTO.getNumeroProyectos() != null) {
    this.verificarCantidadProyectos(proyectoDTO);
}
```

El método **verificarCantidadProyectos(proyectoDTO)** es el siguiente:

```
public void verificarCantidadProyectos(ProyectoDTO proyectoDTO) throws SystemException {
    Long minimo = null;
    Long maximo = null;
    //Validamos por la convocatoria al que se presentó el programa
    if (proyectoDTO.getConvocatoriaId().getMinProyectosPrograma() != null) {
        minimo = proyectoDTO.getConvocatoriaId().getMinProyectosPrograma();
    }
    if (proyectoDTO.getConvocatoriaId().getMaxProyectosPrograma() != null) {
        maximo = proyectoDTO.getConvocatoriaId().getMaxProyectosPrograma();
    }
    //Validamos por la convocatoria convocatoria padre del programa
    if (proyectoDTO.getConvocatoriaId().getConvocatoriaPadreId() != null) {
        if
        (proyectoDTO.getConvocatoriaId().getConvocatoriaPadreId().getMinProyectosPrograma() != null
        && minimo == null) {
            minimo
            proyectoDTO.getConvocatoriaId().getConvocatoriaPadreId().getMinProyectosPrograma();
        }
    }
}
```

```

    }
    if
    (proyectoDTO.getConvocatoriaId().getConvocatoriaPadreId().getMaxProyectosPrograma() != null
    && maximo == null) {
        maximo
        proyectoDTO.getConvocatoriaId().getConvocatoriaPadreId().getMaxProyectosPrograma();
    }
    }

    if (minimo != null && maximo != null) {
        if (proyectoDTO.getNumeroProyectos() < minimo ||
        proyectoDTO.getNumeroProyectos() > maximo) {
            SystemException error = new
            SystemException("#{msg['formulario.proyectos.modificar.numProyectos.error.numeroPryMinMax']}");
            error.getParameters().add(minimo.toString());
            error.getParameters().add(maximo.toString());
            throw error;
        }
        } else if (minimo != null) {
            if (proyectoDTO.getNumeroProyectos() < minimo) {
                SystemException error = new
                SystemException("#{msg['formulario.proyectos.modificar.numProyectos.error.numeroPryMin']}");
                error.getParameters().add(minimo.toString());
                throw error;
            }
        } else if (maximo != null) {
            if (proyectoDTO.getNumeroProyectos() > maximo) {
                SystemException error = new
                SystemException("#{msg['formulario.proyectos.modificar.numProyectos.error.numeroPryMax']}");
                error.getParameters().add(maximo.toString());
                throw error;
            }
        }
    }
}

```

Este método tiene la función de verificar que la cantidad de proyectos de un programa esté dentro de los límites establecidos en la convocatoria. Si el número de proyectos no cumple con los valores mínimos o máximos permitidos, el método genera una excepción del tipo `SystemException`, con un mensaje de error personalizado.

Explicación de cada parte del código:

1. Declaración de Variables:

- **Long mínimo y Long máximo:** Estas variables almacenan los valores mínimo y máximo de proyectos permitidos para el programa, obtenidos de la convocatoria a la que pertenece.

2. Validación de los Valores de la Convocatoria:

- Verifica si la convocatoria asociada al proyecto (**proyectoDTO.getConvocatoriaId()**) tiene valores definidos para **minProyectosPrograma** y **maxProyectosPrograma**.
- Si existen valores, estos se asignan a las variables **minimo** y **maximo**, respectivamente.

3. Validación de los Valores en la Convocatoria Padre:

- Si la convocatoria tiene una convocatoria padre asociada (**proyectoDTO.getConvocatoriaId().getConvocatoriaPadreId()**), el método también revisa los valores **minProyectosPrograma** y **maxProyectosPrograma** de esta convocatoria padre.
- Solo asigna estos valores a **minimo** y **maximo** si no se han establecido previamente por la convocatoria hija, permitiendo que los valores de la convocatoria hija tengan prioridad.

4. Verificación de la Cantidad de Proyectos:

- Si tanto **minimo** como **maximo** tienen valores:
 - 1. Comprueba que el número de proyectos (**proyectoDTO.getNumeroProyectos()**) esté dentro del rango establecido. Si el valor está fuera de este rango, lanza una excepción **SystemException** con un mensaje de error que incluye los valores **minimo** y **maximo**.
- Si solo **minimo** tiene un valor:
 - 1. Verifica que el número de proyectos no sea menor al mínimo permitido. Si es menor, lanza una excepción con un mensaje que incluye el valor **minimo**.
- Si solo **maximo** tiene un valor:
 - 1. Comprueba que el número de proyectos no supere el máximo permitido. Si lo supera, lanza una excepción con un mensaje que incluye el valor **maximo**.

5. Generación de Mensajes de Error:

- Los mensajes de error se obtienen a través de un archivo de mensajes (`{msg['clave_del_mensaje']}`), permitiendo la personalización y traducción de los mensajes.
- Los parámetros **minimo** y **maximo** se agregan a la excepción para que el mensaje de error sea específico y contenga los límites establecidos.

El anterior método es replicado al momento de validar y enviar el programa, con la finalidad que en caso de que los valores hayan cambiado posterior al registro, se valide con la información actualizada.

Implementar Reporte Certificados Evaluadores BT

Para este requerimiento se solicita lo siguiente:

1. El equipo técnico de Beneficios Tributarios. deberá habilitar una opción de “evaluado” una vez se haya verificado que la evaluación es correcta y dar por cerrada la evaluación.
2. Permitir a los pares evaluadores acceder al SIGP y descargar certificados de las evaluaciones de Beneficios Tributarios realizadas. Para este requerimiento se deberá habilitar una opción de “evaluado” que le permita al equipo técnico del Ministerio validar si se puede generar la certificación, o, por el contrario, la evaluación no ha cumplido con las condiciones solicitadas para la evaluación, y por lo cual, no se puede expedir el certificado. En el momento que el equipo técnico habilite la opción “evaluado”, el par evaluador a través de su usuario y contraseña podrá acceder al SIGP y descargar su certificación.

La certificación deberá contener información tal como:

- Nombre de la convocatoria
- Código de la propuesta evaluada
- Título de la propuesta evaluada
- Fecha de evaluación
- Nombre del evaluador
- Documento de identidad del evaluador

(Se adjunto modelo de certificación con la información que deberá contener el certificado generado por la plataforma).

Por otra parte, si el par evaluador desea consultar todas las evaluaciones que ha realizado a la fecha, el sistema le deberá permitir generar esta certificación. No se contempla que las certificaciones estén firmadas.

Los ajustes fueron los siguientes:

Aplicativo de Gestión

Ajuste Fecha Cierre Evaluación por Convocatoria

En el módulo de evaluación de Beneficios Tributarios, el equipo técnico ajustara la Fecha Cierre Evaluación por Convocatoria

- En la pantalla principal de evaluación (**app\beneficiosTributarios\evaluacion.xhtml**), se crea panel para registrar la fecha y cierre de la evaluación de la convocatoria.

```
<!-- Ajuste fecha de cierre de evaluacion de la convocatoria -->
<p:separator/>
<p:outputPanel id="panelFechaCierreEvaluacion">
    <div class="well" style="padding: 20px">
        <ui:fragment
            rendered="#{!SeguridadSessController.listaGruposUsuario.contains('SGPS_EVALUADOR_PRY')}">
            <div class="row" style="padding-top: 20px">
                <div class="col-lg-12">
                    <div class="form-group">
                        <p:outputLabel
                            value="#{msg['gestion.configuracion.convocatorias.fechaCierreEvaluacion.label']}"
                            styleClass="col-lg-6
                                control-label"
                                style="text-align:
                                    right"
                                    for="fechaCierreEval"/>
                        <div class="col-lg-3">
                            <p:inputText id="fechaCierreEval"
                                styleClass="form-control"
                                value="#{proyectoBeneficiosEvaluadoresSessController.selected.convocatoriaId.fechaCierreEvaluacion != null
                                    ?
                                    proyectoBeneficiosEvaluadoresSessController.selected.convocatoriaId.fechaCierreEvaluacion
                                    : 'Sin registro' }"
                                disabled="true"
                                >
                                <f:convertDateTime
                                    pattern="dd/MM/yyyy HH:mm" timeZone="EST" locale="es"/>
                                </p:inputText>
                            </div>
                        <div class="col-lg-2">
```

```

<p:commandLink styleClass="btn btn-xs
btn-primary"
update="formCierreEvaluacionConvocatoria"
title="#{msg['gestion.proyectos.evaluacion.proyectos.convocatoria.fechaCierreEvaluacion.btn'
]}"
disabled="#{proyectoBeneficiosEvaluadoresSessController.selected.estadoProyectoId.descripcionValorDefinicion != 'REVISADO'}"
process="@this"
oncomplete="$('#dlgCierreEvaluacionConvocatoria').modal('show')">
<i class="fa fa-plus"></i>
</p:commandLink>
</div>
</div>
</div>
</div>
</div>
</div>
</p:outputPanel>

```

Este código crea una sección en la interfaz de usuario que muestra la fecha de cierre de evaluación de una convocatoria, permitiendo al usuario visualizar esta fecha y, en algunos casos, ajustar su valor. La sección está restringida para ciertos usuarios y muestra un botón para abrir un diálogo que permite ajustar la fecha si se cumplen condiciones específicas.

Explicación de cada componente:

1. **Separador (<p:separator/>):**
 - Agrega un separador visual para diferenciar esta sección de otras en la interfaz.
2. **Panel de Salida (<p:outputPanel id="panelfechaCierreEvaluacion">):**
 - Define un panel de salida con el ID **panelfechaCierreEvaluacion**. Todo el contenido de este panel será visible o actualizado con el ID especificado.
3. **Contenedor Estilizado (<div class="well">):**
 - Crea un contenedor visual con estilo (**well**), que agrega un fondo sombreado y espacio adicional alrededor de los elementos internos.
4. **Condición de Visualización (<ui:fragment rendered="#{!SeguridadSessController.listaGruposUsuario.contains('SGPS_EVALUADOR_PRY')}>):**

- La etiqueta **<ui:fragment>** solo renderiza el contenido interno si el usuario no pertenece al grupo **SGPS_EVALUADOR_PRY**, ocultando el contenido para los evaluadores específicos.

5. Etiqueta de Fecha de Cierre de Evaluación (<p:outputLabel>):

- Muestra el texto "Fecha de Cierre de Evaluación" como etiqueta. El valor se obtiene del archivo de mensajes para facilitar la traducción.
- Aplica un estilo de alineación y le asigna el **for="fechaCierreEval"** para asociarlo con el campo de texto correspondiente.

6. Campo de Fecha de Cierre de Evaluación (<p:inputText id="fechaCierreEval">):

- Es un campo de entrada de texto que muestra la fecha de cierre de evaluación.
- **Condición de Visualización:** Si la fecha existe (**proyectoBeneficiosEvaluadoresSessController.selected.convocatoriaId.fechaCierreEvaluacion**), se muestra; de lo contrario, muestra "Sin registro".
- **Conversión de Fecha:** Utiliza **<f:convertDateTime>** para mostrar la fecha en el formato **dd/MM/yyyy HH:mm**, con la zona horaria **EST** y en idioma español.
- **Deshabilitado:** El campo está deshabilitado (**disabled="true"**) para evitar modificaciones directas.

7. Botón para Ajuste de Fecha (<p:commandLink>):

- Define un botón pequeño de estilo **btn btn-xs btn-primary** con un ícono (**fa fa-plus**) para abrir un diálogo de ajuste de fecha.
- **Condición de Activación:** El botón se deshabilita si el estado del proyecto (**proyectoBeneficiosEvaluadoresSessController.selected.estadoProyectoId.descripcionValorDefinicion**) no es "REVISADO", limitando el acceso a este ajuste a proyectos en el estado correcto.
- **Función de Actualización:** Actualiza la sección **formCierreEvaluacionConvocatoria** al hacer clic.
- **Diálogo Modal:** Al completarse el proceso, se muestra el diálogo modal **dlgCierreEvaluacionConvocatoria** para permitir el ajuste de fecha.

- Creación Dialogo de ajuste de fecha de cierre de evaluación por convocatoria

```
<!-- Fecha Cierre Evaluacion por Convocatoria -->
<h:form id="formCierreEvaluacionConvocatoria">
  <div class="modal inmodal" id="dlgCierreEvaluacionConvocatoria" tabindex="-1"
  role="dialog" aria-hidden="true">
    <div class="modal-dialog modal-lg">
      <div class="modal-content animated fadeIn">
        <div class="modal-header">
          <button type="button" class="close" data-
dismiss="modal"><span
          aria-hidden="true">&times;</span><span
          class="sr-
only">Close</span></button>
          <i class="fa fa-calendar-check-o modal-icon"></i>
          <h4
          class="modal-
title">#{msg['gestion.proyectos.evaluacion.proyectos.convocatoria.fechaCierreEvaluacion.titl
e']}</h4>
          <p:spacer/>
          <small>
            <h:outputFormat
value="#{msg['gestion.proyectos.evaluacion.proyectos.convocatoria.fechaCierreEvaluacion.msg'
]}">
            <f:param
value="#{proyectoBeneficiosEvaluadoresSessController.selected.convocatoriaId.descripcionConv
ocatoria}" />
            </h:outputFormat>
          </small>
        </div>
        <div class="modal-body">
          <p:messages id="messagesFechaCierre" closable="true"
showDetail="true" skipDetailIfEqualsSummary="true" >
            <p:autoUpdate /><f:passThroughAttribute
name="tabindex" value="0" />
          </p:messages>
          <div class="form-horizontal">
            <div class="row">
              <div class="col-lg-12">
                <div class="form-group">
                  <p:outputLabel
value="#{msg['gestion.configuracion.convocatorias.fechaCierreEvaluacion.label']}"
                  styleClass="col-lg-6 control-label"
                  for="fechaCierreEvaluacion"/>
                  <div class="col-lg-3">
                    <p:calendar
id="fechaCierreEvaluacion"
                    locale="es"
                    value="#{proyectoBeneficiosEvaluadoresSessController.selected.convocatoriaId.fechaCierr
eEvaluacion}"
                    navigator="true"
                    mask="true"
                    pattern="dd/MM/yyyy HH:mm"
```



```

        <p:inputText id="fechaRegistroCierreEva"
                    styleClass="form-control"

value="#{proyectoBeneficiosEvaluadoresSessController.selected.convocatoriaId.fechaRegistroCierreEva}"

                    disabled="true"
                >

        <f:convertDateTime pattern="dd/MM/yyyy HH:mm" timeZone="EST" locale="es"/>
    </p:inputText>

</div>
</div>
</div>
</div>
</ui:fragment>
<div class="modal-footer">
    <button type="button" class="btn btn-
white" data-dismiss="modal">#{msg['sb.component.boton.cancelar']}</button>
    <p:commandLink styleClass="btn btn-
primary"
value="#{msg['sb.component.boton.modificar']}"
actionListener="#{proyectoBeneficiosEvaluadoresConvController.registrarFechaCierreEvaluacion
Convocatoria}"
update="#{proyectoBeneficiosEvaluadoresSessController.modulo}FormDetalle:panelFechaCierreEva
luacion,
formCierreEvaluacionConvocatoria:messagesFechaCierre,
#{proyectoBeneficiosEvaluadoresSessController.modulo}FormDetalle:#{proyectoBeneficiosEvalua
oresSessController.modulo}Table"
process="@form"

/>
</div>
</div>
</div>
</div>
</div>
</div>
</h:form>

```

Este código define un diálogo modal que permite al usuario establecer o modificar la fecha de cierre de evaluación de una convocatoria. El diálogo se abre desde el botón en la interfaz principal y está diseñado para actualizar esta fecha y mostrar detalles relacionados.

Explicación de cada componente:

1. Formulario (<h:form id="formCierreEvaluacionConvocatoria">):

- Define el formulario principal del diálogo. Incluye todos los elementos y controles necesarios para gestionar el ajuste de la fecha de cierre de evaluación.

2. Contenedor Modal (<div class="modal inmodal" id="dlgCierreEvaluacionConvocatoria">):

- Especifica el contenedor del diálogo modal. Este se muestra como una ventana emergente y es activado por el botón en la interfaz principal.

3. Encabezado del Modal (<div class="modal-header">):

- Incluye:
 - Botón de cierre (**close**) para cerrar el diálogo.
 - Ícono de calendario (**fa fa-calendar-check-o**) que indica que se trata de un ajuste de fecha.
 - Título del diálogo, que se obtiene del archivo de mensajes para facilitar la traducción.
 - Mensaje de contexto (pequeño) que muestra una descripción de la convocatoria.

4. Cuerpo del Modal (<div class="modal-body">):

- Contiene los elementos principales para ajustar la fecha:
 - **Mensajes (<p:messages>):** Muestra mensajes de error o éxito después de realizar una acción en el diálogo.
 - **Campo de Fecha de Cierre (<p:calendar>):**
 - Permite seleccionar y ajustar la fecha de cierre de evaluación.
 - **Restricciones:**
 - Solo permite fechas posteriores a la fecha de cierre inicial (**mindate**).
 - Requiere un valor (**required="true"**) y muestra un mensaje específico si el campo está vacío.

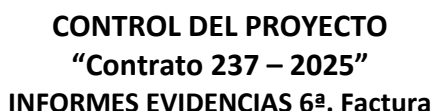
- **Conversión de Fecha:** Muestra la fecha en formato **dd/MM/yyyy HH:mm**, en español y con zona horaria **EST**.

5. **Fragmento Condicional (<ui:fragment rendered="{proyectoBeneficiosEvaluadoresSessController.selected.convocatoriaId.usuarioRegistroCierreEva != null}">):**

- Solo muestra el fragmento si el usuario ha registrado previamente una fecha de cierre.
- Incluye dos campos adicionales:
 - **Usuario que Registró la Fecha (<p:inputText>):** Campo deshabilitado que muestra el nombre del usuario que realizó el último registro de la fecha de cierre.
 - **Fecha de Registro (<p:inputText>):** Campo deshabilitado que muestra la fecha y hora de registro del último ajuste de fecha de cierre, con el mismo formato de conversión.

6. **Pie del Modal (<div class="modal-footer">):**

- Incluye dos botones:
 - **Cancelar:** Cierra el diálogo sin realizar ninguna acción.
 - **Modificar:**
 - Ejecuta el método **registrarFechaCierreEvaluacionConvocatoria** en el **proyectoBeneficiosEvaluadoresConvController** para registrar la nueva fecha de cierre.
 - Actualiza varios elementos en la interfaz, incluyendo:
 - Panel de fecha de cierre en el formulario principal.
 - Mensajes de error o confirmación en el diálogo.



Habilitar opción de “evaluado”

En el módulo de evaluación de Beneficios Tributarios, el equipo técnico deberá habilitar una opción de “evaluado” una vez se haya verificado que la evaluación es correcta y dar por cerrada la evaluación.

- En la pantalla principal de evaluacion (**app\beneficiosTributarios\evaluacion.xhtml**), en el panel de fecha y cierre de la evaluacion de la convocatoria, se crea campo que permita indicar si el proyecto ha sido o no evaluado (S/N)

```

<ui:fragment
rendered="#{!SeguridadSessController.listaGruposUsuario.contains('SGPS_EVALUADOR_PRY'))}">
    <div class="row" style="padding-top: 20px">
        <div class="col-lg-12">
            <div class="form-group">
                <p:outputLabel
value="#{msg['gestion.proyectos.evaluacion.proyectos.evaluado.label']}"
                                styleClass="col-lg-6 control-label"
                                style="text-align: right"
                                for="cerrarEva"/>
                <div class="col-lg-3">
                    <p:commandLink id="cerrarEva"
title="#{projectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO
?
msg['gestion.proyectos.evaluacion.proyectos.cerrarEvaluacionOff.title']
:
msg['gestion.proyectos.evaluacion.proyectos.cerrarEvaluacion.title']}"
                                styleClass="btn
                                #{projectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO ? 'btn-primary':
'btn-danger'} sbTooltip"
oncomplete="$('#dlgJustCierre').modal();"
                                process="@this"
                                update="formJustCierre"
disabled="#{projectoBeneficiosEvaluadoresSessController.selected.estadoProyectoId.descripcionValorDefinicion != 'REVISADO'}"
                                >
                    <i
                                class="fa
                                #{projectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO ? 'fa-check':
'fa-close'}">
                        <div class="col-lg-6">
                            <p:commandLink
                                value="#{projectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO
?
msg['listaSINO.S'] : msg['listaSINO.N']}"
                                </i>
                                </p:commandLink>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </ui:fragment>

```

Este código define un botón que permite indicar que un proyecto ya ha sido evaluado, lo cual cierra el proceso de evaluación. La interfaz de usuario permite a ciertos usuarios (excluyendo al grupo "SGPS_EVALUADOR_PRY") marcar un proyecto como "Evaluado", lo que modifica su estado en el sistema y refleja visualmente su nuevo estado.

Explicación de cada componente:

1. **Fragmento Condicional (<ui:fragment rendered="#{!SeguridadSessController.listaGruposUsuario.contains('SGPS_EVALUADOR_PRY')}">):**
 - Solo muestra el botón si el usuario no pertenece al grupo **SGPS_EVALUADOR_PRY**, lo que permite restringir esta funcionalidad a ciertos roles.
2. **Estructura de Presentación (<div class="row">):**
 - Este div crea una fila para organizar el botón y el texto en la interfaz con un espaciado superior de **20px**.
3. **Etiqueta del Botón (<p:outputLabel>):**
 - Muestra una etiqueta de texto para el botón, obteniendo el valor desde el archivo de mensajes (**msg**) con la clave **gestion.proyectos.evaluacion.proyectos.evaluado.label**.
 - La etiqueta se alinea a la derecha y se aplica a la clase de estilo **col-lg-6 control-label**.
4. **Botón de Comando (<p:commandLink>):**
 - Es el botón principal que permite cerrar la evaluación del proyecto:
 - **ID:** **cerrarEva** para identificar este elemento.
 - **Título (title):** Cambia dinámicamente dependiendo de si el proyecto ya fue cerrado o no:
 - Si el proyecto fue cerrado (**cierreEvaluacionSINO** es **true**), el título es **gestion.proyectos.evaluacion.proyectos.cerrarEvaluacionOff.title**.

- Si el proyecto aún no ha sido cerrado (**cierreEvaluacionSINO** es **false**), el título es **gestion.proyectos.evaluacion.proyectos.cerrarEvaluacion.title**.
- **Estilo (styleClass)**: Cambia el estilo del botón según el estado de evaluación:
 - Si **cierreEvaluacionSINO** es **true**, el estilo es **btn-primary** (indica que está cerrado).
 - Si es **false**, el estilo es **btn-danger** (indica que no ha sido cerrado).
- **Ícono (<i class="fa...">)**:
 - Muestra un icono dinámico que también cambia según el estado de evaluación:
 - Si **cierreEvaluacionSINO** es **true**, se muestra **fa-check** (marca de verificación).
 - Si es **false**, se muestra **fa-close** (ícono de cierre).
- **Texto de Estado**
(#{proyectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO ? msg['listaSINO.S'] : msg['listaSINO.N']}):
 - El texto del botón muestra **S** o **N** dependiendo del estado (**true** o **false**) de **cierreEvaluacionSINO**.

5. Eventos y Actualizaciones:

- **Evento oncomplete**: Cuando se hace clic, abre el diálogo modal **dlgJustCierre**.
- **process="@this"**: Procesa solo este botón para minimizar la carga de procesamiento.
- **update="formJustCierre"**: Actualiza el formulario **formJustCierre** después de hacer clic en el botón, lo que permite reflejar los cambios en la interfaz.
- **disabled**: Si el estado del proyecto (**estadoProyectoId.descripcionValorDefinicion**) no es "REVISADO", el botón

estará deshabilitado, lo que impide cerrarlo hasta que el proyecto esté en estado revisado.

- Dialogo para cerrar evaluacion del proyecto

```
<!-- Dialogo CERRAR EVALUACION -->
<h:form id="formJustCierre">
  <div class="modal inmodal" id="dlgJustCierre" tabindex="-1" role="dialog" aria-
hidden="true">
    <div class="modal-dialog modal-lg">
      <div class="modal-content animated fadeIn">
        <div class="modal-header">
          <button type="button" class="close" data-
dismiss="modal"><span
aria-hidden="true">&times;</span><span class="sr-
only">Close</span></button>
          <i class="fa fa-check-square-o modal-icon"></i>
          <h4 class="modal-
title">#{proyectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO
?
msg['gestion.proyectos.evaluacion.proyectos.cerrarEvaluacionOff.dlg.title']}
:
msg['gestion.proyectos.evaluacion.proyectos.cerrarEvaluacion.dlg.title']}</h4>

          <medium>#{proyectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO
?
msg['gestion.proyectos.evaluacion.proyectos.cerrarEvaluacionOff.dlg.mensaje']}
:
msg['gestion.proyectos.evaluacion.proyectos.cerrarEvaluacion.dlg.mensaje']}</medium>
        </div>
        <div class="modal-body">
          <div class="form-horizontal">
            <div class="row">
              <div class="col-lg-12">
                <div class="form-group">
                  <p:outputLabel
value="#{msg['gestion.configuracion.registroEntidades.observacion.label']}"
styleClass="col-lg-3 control-label"
for="justificacionCierre"/>
                </div>
                <div class="col-lg-9">
                  <h:inputTextarea
value="#{proyectoBeneficiosEvaluadoresConvController.observacionJustificacion}"
id="justificacionCierre"
styleClass="form-control"
required="true"
requiredMessage="#{msg['gestion.configuracion.registroEntidades.observacion.required']}">
                </div>
              </div>
            </div>
          </div>
          <div class="modal-footer">
            <button type="button" class="btn btn-white"
data-dismiss="modal">#{msg['sb.component.boton.cancelar']}</button>
            <p:commandLink styleClass="btn btn-primary">
```

```
value="#{msg['gestion.proyectos.evaluacion.proyectos.cerrarEvaluacion.dlg.btn']}"
actionListener="#{proyectoBeneficiosEvaluadoresConvController.agregarObservacion('cierre')}"
oncomplete="$('#dlgJustCierre').modal('hide')"
update="#{proyectoBeneficiosEvaluadoresSessController.modulo}FormDetalle,
#{proyectoBeneficiosEvaluadoresSessController.modulo}FormDetalle:#{proyectoBeneficiosEvaluadoresSessController.modulo}Table"
</p:commandLink>
</div>
</div>
</div>
</div>
</h:form>
```

Este código define un diálogo modal para confirmar el cierre de la evaluación de un proyecto. Cuando el usuario decide cerrar una evaluación, se abre este diálogo para permitirle justificar el cierre proporcionando una observación.

Explicación de cada componente:

1. Formulario del Diálogo (<h:form id="formJustCierre">):

- Todo el contenido del diálogo está encapsulado en un formulario JSF, permitiendo la gestión y envío de datos.

2. Diálogo Modal (<div class="modal inmodal" id="dlgJustCierre">):

- Define una ventana modal (**id="dlgJustCierre"**) que se presenta con una animación de **fadeIn**.
- Esta ventana es **modal**, es decir, requiere interacción antes de volver a la interfaz principal.

3. Encabezado del Diálogo (<div class="modal-header">):

- **Botón de Cierre** (<button type="button" class="close" data-dismiss="modal">): Permite cerrar el diálogo sin confirmar la acción.
- **Icono y Título** (<i class="fa fa-check-square-o modal-icon">, <h4 class="modal-title">):

- Se muestra un ícono **fa-check-square-o** para indicar el propósito de confirmación.
 - El título cambia según el estado de **cierreEvaluacionSINO**:
 - Si el proyecto está cerrado (**true**), muestra el título **cerrarEvaluacionOff.dlg.title**.
 - Si no está cerrado (**false**), muestra **cerrarEvaluacion.dlg.title**.
 - **Mensaje de Descripción (<medium>):**
 - Muestra un mensaje descriptivo en el diálogo. Este mensaje cambia dinámicamente según el estado de **cierreEvaluacionSINO**.
4. **Cuerpo del Diálogo (<div class="modal-body">):**
- **Campo de Observación (<h:inputTextarea>):**
 - Se solicita una justificación de cierre mediante un campo de texto (**textarea**).
 - Etiqueta: Muestra la etiqueta **observacion.label** obtenida del archivo de mensajes.
 - Campo de Entrada:
 - Utiliza **proyectoBeneficiosEvaluadoresConvController.observacionJustificacion** para almacenar la observación ingresada.
 - Requiere que el usuario complete este campo antes de cerrar la evaluación (con **required="true"** y **requiredMessage**).
5. **Pie de Página del Diálogo (<div class="modal-footer">):**
- **Botón de Cancelar:**
 - Botón con clase **btn btn-white** que permite cerrar el diálogo sin realizar ninguna acción adicional.
 - **Botón de Confirmación (<p:commandLink>):**

- Permite agregar la observación y cerrar el proyecto:
 - **Valor (value):** Muestra el texto del botón desde el mensaje cerrarEvaluacion.dlg.btn.
 - **Acción (actionListener):** Llama al método **agregarObservacion('cierre')** de **proyectoBeneficiosEvaluadoresConvController** para agregar la observación y completar la acción de cierre.
 - **Evento oncomplete:** Oculta el diálogo **(\$('#dlgJustCierre').modal('hide'))** una vez que se ejecuta la acción.
 - **Actualización (update):**
 - Actualiza el formulario de detalles y la tabla de proyectos con los IDs dinámicos especificados **(#{proyectoBeneficiosEvaluadoresSessController.modulo}FormDetalle y #{proyectoBeneficiosEvaluadoresSessController.modulo}Table)**, asegurando que se reflejen los cambios en la interfaz.
- Script de actualización campo de **cierre_evaluacion** de proyectos de beneficios tributarios.

```
UPDATE sgp_proyectos p
  SET p.cierre_evaluacion = 'S'
 WHERE p.estado_proyecto_id IN (SELECT vd.valor_definicion_id
                                FROM adm_valores_definiciones vd
                                WHERE vd.tipo_definicion = 'ESTADO'
                                AND vd.descripcion_valor_definicion IN
('EVALUADO', 'NEGADO', 'APROBADO', 'SEGUIMIENTO'))
  AND p.exencion = 'E'
  AND p.cierre_evaluacion IS NULL
```


El script actualiza el campo **cierre_evaluacion** de ciertos proyectos en la tabla **sgp_proyectos**, estableciéndolo en 'S' (probablemente indicando que la evaluación está cerrada) para proyectos que cumplen con condiciones específicas.

Explicación del Código

1. Tabla de Actualización:

- La instrucción **UPDATE** afecta la tabla **sgp_proyectos**, en particular el campo **cierre_evaluacion**.

2. Valor de Actualización:

- El campo **cierre_evaluacion** se actualiza a 'S' para los proyectos seleccionados.

3. Condiciones del WHERE:

- Las condiciones de filtro limitan los registros que se actualizarán:
 - **Estado del Proyecto:**
 - **p.estado_proyecto_id** debe coincidir con uno de los valores de **valor_definicion_id** obtenidos mediante una subconsulta en la tabla **adm_valores_definiciones**.
 - La subconsulta selecciona los valores **valor_definicion_id** donde el tipo de definición (**tipo_definicion**) es 'ESTADO' y la descripción del valor de definición (**descripcion_valor_definicion**) corresponde a uno de los siguientes estados: 'EVALUADO', 'NEGADO', 'APROBADO', o 'SEGUIMIENTO'.
 - **Tipo de Exención:**
 - **p.exencion** debe ser igual a 'E'.
 - **Campo de Evaluación Cerrado:**
 - **p.cierre_evaluacion** debe ser **NULL**, asegurando que solo se actualicen registros donde **cierre_evaluacion** aún no tiene un valor asignado.

- Se adicionan validaciones en la pestaña de asignación de evaluadores que permita deshabilitar las funcionalidades de la pantalla al dar por evaluada la evaluación, ya sean campos, botones, etc.

```
disabled="#{proyectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO
or
proyectoBeneficiosEvaluadoresSessController.selected.estadoProyectoId.descripcionValorDefini
cion != 'REVISADO'}"
```

Este código utiliza una expresión condicional para deshabilitar un componente cuando se cumplen ciertas condiciones en el módulo de evaluación.

Explicación del código:

1. **disabled="..."**: Este atributo indica que el componente estará deshabilitado si la condición es true.
2. **Condiciones**:
 - **proyectoBeneficiosEvaluadoresSessController.selected.cierreEvaluacionSINO**: Si esta variable es **true**, el proyecto ya fue cerrado para evaluación, y el botón o función estará deshabilitada.
 - **proyectoBeneficiosEvaluadoresSessController.selected.estadoProyectoId.descripcionValorDefinicion != 'REVISADO'**: La función también se deshabilita si el estado actual del proyecto es distinto de "REVISADO". Esto asegura que solo los proyectos en estado "REVISADO" puedan acceder a ciertas funcionalidades.

En resumen, el componente estará deshabilitado cuando el proyecto ya haya sido cerrado o su estado no sea "REVISADO". Esto ayuda a controlar el acceso y garantiza que ciertas funciones solo estén disponibles en condiciones específicas.

- Mensajes de alerta el proyecto ha cerrado su evaluación, independientemente si las fechas de evaluación están abiertas.

```
<ui:fragment rendered="#{evaluacionSessController.selected.proyectoId.cierreEvaluacion !=
null
and
evaluacionSessController.selected.proyectoId.cierreEvaluacion eq 'S'}">
  <div class="row">
    <div class="col-lg-12">
      <div class="alert alert-danger alert-dismissible">
        <b> <i class="fa fa-info-circle"></i>
```

```

        #{msg['gestion.beneficiosTributarios.evaluadores.calificacion.cierreEvaluacionPry.close
        .msg']}
        </b>
    </div>
</div>
</div>
</ui:fragment>

```

Este código muestra un mensaje de alerta cuando la evaluación de un proyecto ha sido cerrada. La alerta se presenta en un formato visual llamativo (color rojo y símbolo de información), indicando que ya no es posible realizar cambios en la evaluación.

Explicación detallada del código:

1. Condición de Renderizado:

- **rendered="#{evaluacionSessController.selected.proyectold.cierreEvaluacion != null and evaluacionSessController.selected.proyectold.cierreEvaluacion eq 'S'}"**:
 - La alerta solo se renderiza cuando **cierreEvaluacion** no es **null** y tiene el valor **'S'** (cerrada).
 - Esto garantiza que el mensaje de alerta se muestre solo cuando la evaluación del proyecto ha sido efectivamente cerrada.

2. Estructura del Mensaje:

- **Clase CSS alert alert-danger alert-dismissible**:
 - Estas clases aplican estilos que resaltan el mensaje en rojo y permiten al usuario cerrarlo manualmente.
- **Icono y Texto del Mensaje**:
 - **<i class="fa fa-info-circle"></i>**: Muestra un ícono informativo para llamar la atención del usuario.
 - **Texto**:
 #{msg['gestion.beneficiosTributarios.evaluadores.calificacion.cierreEvaluacionPry.close.msg']}

- El mensaje se obtiene de un archivo de recursos (**msg**) y proporciona una advertencia específica sobre la imposibilidad de modificar la evaluación.

Descargar certificados de las evaluaciones a pares evaluadores

Permitir a los pares evaluadores acceder al SIGP (módulo de evaluar de Beneficios tributarios) y descargar certificados de las evaluaciones de Beneficios Tributarios realizadas.

Se crea boton en la pantalla de evaluar de proyectos de Beneficios Tributarios (**app/beneficiosTributarios/proyectosEvaluador.xhtml**), que abra ventana de dialogo y permita al evaluador indicar la convocatoria y el tipo de certificado que desea generar

- Botón certificado evaluador

```
<!-- Boton certificado evaluador -->
<div class="row">
  <div class="col-lg-12">
    <p:commandButton
value="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn']}"
                                styleClass="btn btn-success pull-right
sbTooltip"
                                style="padding: 3px"

onclick="$('#dlgCertificadoEvaluador').modal()"

update="#{evaluacionSessController.modulo}formReporteEvaluador"

title="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn.titl
e']}"
                                icon="fa fa-file-pdf-o"

actionListener="#{proyectoBeneficiosEvaluadoresConvController.cargarListaConvocatorias()}"
                                >
    </p:commandButton>
  </div>
</div>
```

Este botón, denominado "Certificado Evaluador", fue desarrollado para abrir un diálogo en la pantalla de evaluación de proyectos de Beneficios Tributarios, permitiendo al evaluador seleccionar la convocatoria y el tipo de certificado a generar. Aquí está la explicación del código:

Explicación del Código del Botón "Certificado Evaluador":

1. Clase CSS:

- **styleClass="btn btn-success pull-right sbTooltip":**

- **btn** y **btn-success**: Estilo del botón en color verde, destacándolo como acción positiva.
- **pull-right**: Alinea el botón hacia la derecha dentro de su contenedor.
- **sbTooltip**: Clase utilizada para mostrar el mensaje de ayuda al pasar el cursor por encima.

2. Propiedades del Botón:

- **value="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn']}'**:
 - Establece el texto del botón utilizando un valor del archivo de mensajes (**msg**), facilitando su internacionalización.
- **title="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn.title']}'**:
 - Muestra un texto adicional como título del botón al pasar el ratón, también utilizando el archivo de mensajes.

3. Icono:

- **icon="fa fa-file-pdf-o"**:
 - Añade un ícono en formato PDF al botón (usando **FontAwesome**), indicando visualmente que la acción genera un archivo PDF.

4. Eventos y Funcionalidades:

- **onclick="\$('#dlgCertificadoEvaluador').modal()"**:
 - Llama al método **modal()** de jQuery para abrir el diálogo con el ID **dlgCertificadoEvaluador**.
 - Este diálogo contendrá la interfaz que permite seleccionar la convocatoria y el tipo de certificado.
- **update="#{evaluacionSessController.modulo}formReporteEvaluador"**:

- Actualiza el formulario correspondiente (**formReporteEvaluador**) antes de abrir el diálogo para asegurarse de que la lista de convocatorias esté actualizada.
- **actionListener="#{proyectoBeneficiosEvaluadoresConvController.cargarListaConvocatorias()}"**:
 - Llama a **cargarListaConvocatorias()**, que carga las opciones de convocatorias disponibles, asegurando que el evaluador pueda seleccionarlas en el diálogo.
- Diálogo de Generación de Certificado para el Evaluador

```
<!--DIALOGO DE CERTIFICADO DEL EVALUADOR -->
<div class="modal inmodal" id="dlgCertificadoEvaluador" tabindex="-1" role="dialog" aria-
hidden="true">
  <div class="modal-dialog modal-lg">
    <div class="modal-content animated bounceInRight">
      <h:form id="#{evaluacionSessController.modulo}formReporteEvaluador">
        <div class="modal-header">
          <button type="button" class="close" data-
dismiss="modal"><span aria-hidden="true"></span><span class="sr-only">Close</span></button>
          <i class="fa fa-pdf-o modal-icon"></i>
          <h4 tabindex="0" class="modal-
title">#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.header']}
</h4>
          <medium class="font-
bold">#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.mensaje']}
</medium>
        </div>
        <div class="modal-body">
          <div class="form-horizontal">
            <div class="col-lg-10 col-lg-offset-1">
              <div class="alert alert-info">
                <p class="medium"><i class="fa
fa-info-circle"></i>
                #{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.alert']}
                </p>
              </div>
            </div>
            <div class="row">
              <div class="col-lg-12">
                <p:messages
id="msgCertificados"
closable="true"/>
              </div>
            </div>
            <div class="row">
              <div class="col-lg-12">
                <div class="form-group">
```

```

value="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.convocat
oria']}"
    for="convocatoriaEva-EBT"
    style="text-align: right"
    styleClass="col-lg-2 control-label"
    />
    <div class="col-lg-10">
    <p:selectOneMenu id="convocatoriaEva-EBT"
        value="#{evaluacionSessController.convocatoriaEvaluador}"
        required="true"
        requiredMessage="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluado
r.convocatoria.req']}"
        appendTo="@this"
        panelStyle="position: initial;"
        converter="convocatoriaConverter"
        style="width:100%"
        filter="true"
        filterMatchMode="contains"
    >
    <f:selectItem itemLabel="#{msg['sb.selectOneMenu.registro']}" />
    <f:selectItems value="#{evaluacionSessController.convocatoriaEvaluadorList}"
        var="convocatoria"
        itemLabel="#{convocatoria.descripcionConvocatoria}"
        itemValue="#{convocatoria}"
    />
    </p:selectOneMenu>
    </div>
    </div>
    </div>
    <div class="row">
        <div class="col-lg-12">
            <div class="form-group">
                <p:outputLabel
value="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoCert
ificado']}"

```

```

        for="tipoReporte"
        style="text-align: right"
        styleClass="col-lg-2 control-label"
    />

<div class="col-lg-4">

    <p:selectOneMenu id="tipoReporte"

        value="#{evaluacionSessController.tipoReporte}"

        required="true"

requiredMessage="#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluado
r.tipoCertificado.re']}"

        appendTo="@this"
        panelStyle="position: initial;"
        style="width:100%;"
    >

    <f:selectItems value="#{configApplController.listaTipoCertificado}"/>
</p:selectOneMenu>

</div>
</div>
</div>
<p:spacer />
<div class="row" style="text-align: center">
    <div class="col-lg-12">
        <p:commandLink
id="generarReporte"
actionListener="#{proyectoBeneficiosEvaluadoresConvController.generarReporteCertificadoEvalu
ador()}"
process="@this, convocatoriaEva-EBT, tipoReporte"
ajax="false"
update="@form"
styleClass="btn btn btn-primary"
>

        #{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.generarCer
tificado']}
    </p:commandLink>
    </div>
</div>
</div>
</div>
<div class="modal-footer">

```



```
<button type="button" class="btn btn-white" data-  
dismiss="modal">#{msg[ 'sb.component.boton.cancelar' ]}</button>  
</div>  
</h:form>  
</div>  
</div>  
</div>
```

Este diálogo se presenta al evaluador en la pantalla de evaluación de proyectos de Beneficios Tributarios. Permite seleccionar la convocatoria y el tipo de certificado que se desea generar.

Estructura General del Diálogo

1. Etiqueta div con clases modal:

- **modal inmodal**: Define el contenedor del diálogo como modal, con estilo de animación inmodal.
- **id="dlgCertificadoEvaluador"**: Identificador del modal que permite abrirlo desde otros elementos como botones.
- **modal-dialog modal-lg**: Tamaño del diálogo (modal grande).
- **modal-content animated bounceInRight**: Estilo de animación para la entrada del diálogo.

2. Formulario h:form:

- El formulario (**#{evaluacionSessController.modulo}formReporteEvaluador**) es necesario para encapsular los elementos interactivos y permite la actualización parcial de componentes.

Encabezado del Diálogo

- **Título e ícono:**
 - **modal-title**: Texto dinámico definido mediante el archivo de mensajes (msg) para facilitar la internacionalización.
 - **<i class="fa fa-file-pdf-o modal-icon"></i>**: Ícono de PDF que indica el tipo de acción (generación de documento).

Cuerpo del Diálogo

1. Mensaje Informativo:

- Una alerta con un ícono de información proporciona detalles adicionales al evaluador sobre la funcionalidad del diálogo.

2. Mensajes:

- **<p:messages id="msgCertificados" closable="true"/>**: Panel de mensajes que muestra errores o confirmaciones.

3. Campos de Selección:

- **p:selectOneMenu para Convocatoria:**
 - **value="#{evaluacionSessController.convocatoriaEvaluador}"**: Enlaza el campo con la variable de sesión convocatoriaEvaluador.
 - **<f:selectItems value="#{evaluacionSessController.convocatoriaEvaluadorList}"**: Carga la lista de convocatorias disponibles.
 - **converter="convocatoriaConverter"**: Utiliza un convertidor personalizado para gestionar correctamente el objeto convocatoria.
- **p:selectOneMenu para Tipo de Certificado:**
 - **value="#{evaluacionSessController.tipoReporte}"**: Variable de sesión que almacena el tipo de reporte seleccionado.
 - **<f:selectItems value="#{configApplController.listaTipoCertificado}"**: Lista de tipos de certificado configurada previamente.

4. Botón Generar Certificado:

- **<p:commandLink id="generarReporte"**:
 - **actionListener="#{proyectoBeneficiosEvaluadoresConvController.generarReporteCertificadoEvaluador()}"**: Llama al método para generar el certificado.
 - **process="@this, convocatoriaEva-EBT, tipoReporte"**: Procesa el botón y los campos de selección.

- **ajax="false"**: Desactiva AJAX para permitir que el archivo se genere y descargue sin interferencias.

Pie de Diálogo

- **Botón de Cancelación:**
 - **<button type="button" class="btn btn-white" data-dismiss="modal">**: Cierra el diálogo al hacer clic en "Cancelar".
- Metodo **cargarListaConvocatorias** en el controlador de conversión **proyectoBeneficiosEvaluadoresConvController.java**

```
public void cargarListaConvocatorias() {  
    List<ConvocatoriaDTO> listaConvocatorias = new ArrayList<>();  
    try {  
        listaConvocatorias  
beneficiosServiceFacade.obtenerConvocatoriasBTEvaluador(seguridadSessController.getSelected( =  
) .getUsuario());  
        evaluacionSessController.setConvocatoriaEvaluadorList(listaConvocatorias);  
    } catch (Exception ex) {  
        LOG.error("Error al obtener el listado de convocatorias del Evaluador ", ex);  
    }  
}
```

Este método se invoca al hacer clic en el botón de generación de certificados. Al ejecutarse, carga dinámicamente las convocatorias en las que ha participado el evaluador y las muestra en el diálogo para que pueda seleccionar la convocatoria específica.

Este método permite al evaluador tener una lista actualizada de sus convocatorias, mejorando la usabilidad y asegurando que solo se muestren convocatorias relevantes para la generación de certificados.

Detalles Técnicos:

1. **Inicialización de la Lista:** Se crea una lista vacía listaConvocatorias de tipo **ConvocatoriaDTO** para almacenar las convocatorias del evaluador.
2. **Obtención de las Convocatorias:**

- Llama al servicio **beneficiosServiceFacade.obtenerConvocatoriasBTEvaluador(usuario)**, pasando el usuario obtenido de **seguridadSessController.getSelected().getUsuario()**.
- **obtenerConvocatoriasBTEvaluador** devuelve una lista de objetos **ConvocatoriaDTO** que representan las convocatorias asociadas al evaluador.

3. Asignación a la Sesión:

- La lista de convocatorias obtenida se almacena en **evaluacionSessController** usando **setConvocatoriaEvaluadorList(listaConvocatorias)**.
- Esto permite que la lista esté disponible en el diálogo de generación de certificados en la interfaz de usuario.

4. Gestión de Errores:

- Si ocurre un error durante la ejecución, se captura en un bloque **catch**.
 - Se registra el error en los logs con un mensaje descriptivo para facilitar la identificación de problemas.
- Crear método de consulta de convocatorias en EJB facade, EJB service y store manager que permita consultar las convocatorias asociadas a un evaluador

BeneficiosServiceFacade.java

```
//----- Método para obtener las convocatorias en las cuáles ha participado el
//----- evaluador -----
List<ConvocatoriaDTO> obtenerConvocatoriasBTEvaluador(String usuario) throws Exception;
```

BeneficiosServiceFacadeBean.java

```
@Override
public List<ConvocatoriaDTO> obtenerConvocatoriasBTEvaluador(String usuario) throws Exception
{
    ObjectConverter<ConvocatoriaDTO, Convocatoria> converter =
    objectConverterFactory.checkOut();
    List<ConvocatoriaDTO> convocatoriaEvaluadorList =
    converter.toTransferList(beneficiosService.obtenerConvocatoriasBTEvaluador(usuario),
    ConvocatoriaDTO.class);
    objectConverterFactory.checkIn(converter);
    return convocatoriaEvaluadorList;
}
```

BeneficiosService.java

```
List<Convocatoria> obtenerConvocatoriasBTEvaluador(String usuario) throws SystemException;
```

BeneficiosServiceBean.java

```
@Override  
public List<Convocatoria> obtenerConvocatoriasBTEvaluador(String usuario) throws  
SystemException {  
    return  
    this.getStoreManager(BeneficiosStoreManager.class).obtenerConvocatoriasBTEvaluador(usuario);  
}
```

BeneficiosStoreManager.java

```
List<Convocatoria> obtenerConvocatoriasBTEvaluador(String usuario);
```

BeneficiosStoreManagerImpl.java

```
@Override  
public List<Convocatoria> obtenerConvocatoriasBTEvaluador(String usuario) {  
    Query query = em.createQuery("SELECT DISTINCT c "  
        + " FROM Convocatoria c "  
        + "      , Proyecto p "  
        + "      , ProyectoEvaluacion eva "  
        + " WHERE p.proyectoId = eva.proyectoId.proyectoId "  
        + "      AND c.convocatoriaId = p.convocatoriaId.convocatoriaId "  
        + "      AND p.exencion IN ('E') "  
        + "      AND eva.usuarioSigp = :usuario "  
        + " ORDER BY c.descripcionConvocatoria ASC ");  
    query.setParameter("usuario", usuario);  
  
    return query.getResultList();  
}
```

Método obtenerConvocatoriasBTEvaluador

Este método es llamado por el controlador **proyectoBeneficiosEvaluadoresConvController** al cargar las convocatorias para el evaluador en el diálogo de generación de certificados. Proporciona una lista filtrada de convocatorias en las que el evaluador ha participado, permitiéndole seleccionar la convocatoria relevante para el certificado.

Este método realiza una consulta en la base de datos para obtener una lista de convocatorias asociadas a un evaluador específico, basándose en su participación en proyectos de evaluación con exención de beneficios tributarios.

Detalles Técnicos:

1. Consulta JPA:

- Utiliza **createQuery** para construir una consulta JPQL que selecciona convocatorias (**Convocatoria**) donde el evaluador (identificado por **usuario**) ha participado en proyectos asociados a esa convocatoria.
- La consulta realiza un **JOIN** implícito entre las entidades **Convocatoria**, **Proyecto**, y **ProyectoEvaluacion**.

2. Condiciones de Filtro:

- **p.proyectold = eva.proyectold.proyectold**: Filtra proyectos que coinciden con los registros de evaluación del evaluador.
- **c.convocatoriald = p.convocatoriald.convocatoriald**: Establece la relación entre el proyecto y la convocatoria.
- **p.exencion IN ('E')**: Solo selecciona proyectos con exención de beneficios tributarios ('E').
- **eva.usuarioSigp = :usuario**: Filtra registros de evaluación realizados por el usuario especificado en el parámetro.

3. Ordenamiento:

- Ordena los resultados por **c.descripcionConvocatoria** en orden ascendente para una visualización organizada en la interfaz.

4. Parámetro de Usuario:

- **query.setParameter("usuario", usuario)**; establece el valor del parámetro usuario, permitiendo la búsqueda específica para el evaluador actual.

5. Resultado:

- La consulta devuelve una lista de objetos **Convocatoria** sin duplicados, utilizando **SELECT DISTINCT** para evitar convocatorias repetidas en el caso de múltiples evaluaciones en el mismo proyecto.
- Método **generarReporteCertificadoEvaluador**, que permite generar el certificado de evaluación del evaluador en formato PDF.

Este método es invocado cuando el evaluador solicita la generación de su certificado desde el diálogo de certificación. Genera un archivo PDF que sirve como certificado de evaluación para el evaluador. Dependiendo del tipo de reporte (general o específico), ajusta el contenido del certificado y lo descarga con un nombre de archivo personalizado.

```
public void generarReporteCertificadoEvaluador() {
    FacesContext context = FacesContext.getCurrentInstance();
    HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();

    String usuario = seguridadSessController.getSelected().getUsuario();
    String nombreReporte = "";
    Long convocatoriaId =
evaluacionSessController.getConvocatoriaEvaluador().getConvocatoriaId();
    String tipoReporte = evaluacionSessController.getTipoReporte();

    try {
        if (tipoReporte != null) {
            beneficiosServiceFacade.verificarProyectosBTEvaluados(usuario,
convocatoriaId);
            ReporteUtil reporteUtil = new ReporteUtil();
            HashMap hashMap = new HashMap();

            reporteUtil.setNomCarpeta("CertificadoEvaluadorProyectoBT");

            if (tipoReporte.equals("G")) { // General
                nombreReporte = "certificacionGeneral";
            } else { // Especifico
                nombreReporte = "certificacionEspecifico";
            }

            //Se setea la imagen del logo parametrizada, guardada en la base de datos
            String imagenLogoBase64 =
beneficiosServiceFacade.obtenerParametroSistema("LOGIN_LOGO").getDetalle();
            String[] parts = imagenLogoBase64.split(",");
            //String part1 = parts[0]; // data:image/jpeg;base64
            String imagen = parts[1]; // IMAGEN
            hashMap.put("IMG_BASE64", imagen);

            //Se setea la imagen secundaria del logo parametrizada, guardada en la
base de datos
            String imagenLogoSecundarioBase64 =
beneficiosServiceFacade.obtenerParametroSistema("LOGIN_LOGO2").getDetalle();
            String imagenSecundaria = null;
            if (imagenLogoSecundarioBase64 != null) {
                String[] parts2 = imagenLogoSecundarioBase64.split(",");
                imagenSecundaria = parts2[1]; // IMAGEN
            }
        }
    }
}
```

```

    }
    hashMap.put("IMG_SECUNDARIA_BASE64", imagenSecundaria);

    //Se setea la convocatoria
    hashMap.put("CONVOCATORIA_ID", convocatoriaId);
    //Se setea el usuario conectado
    hashMap.put("USUARIO_SIGP", usuario);

    byte reporte[] = reporteUtil.generarReporte(hashMap, nombreReporte,
"PDF", context);
    response.setContentType("application/pdf");
    if (tipoReporte.equals("G")) {
        response.setHeader("Content-Disposition", "attachment;
filename=\"CertificadoGeneral\" + convocatoriaId.toString() + ".pdf\"");
    } else {
        response.setHeader("Content-Disposition", "attachment;
filename=\"CertificadoEspecifico\" + convocatoriaId.toString() + ".pdf\"");
    }
    response.setContentLength(reporte.length);
    try (ServletOutputStream ouputStream = response.getOutputStream()) {
        ouputStream.write(reporte, 0, reporte.length);
        ouputStream.flush();
        ouputStream.close();
    }
}
evaluacionSessController.setConvocatoriaEvaluador(new ConvocatoriaDTO());
evaluacionSessController.setTipoReporte("");
evaluacionSessController.init();
//Cerramos el Dialog

PrimeFaces.current().executeScript("$('#dlgCertificadoEvaluador').modal('hide')");
} catch (Exception ex) {
    evaluacionSessController.setConvocatoriaEvaluador(new ConvocatoriaDTO());
    evaluacionSessController.setTipoReporte("");
    evaluacionSessController.init();
    PrimeFaces.current().executeScript("toastr.error('"
this.resolveExpression(ex.getMessage() + "')");
// LOG.error("Error al generar el reporte de certificado del evaluador ", ex);
}
}
}

```

Detalles Técnicos:

1. Inicialización:

- Obtiene el contexto **FacesContext** y la respuesta **HttpServletResponse** para manipular la descarga del archivo.
- Recupera el usuario actual, el **convocatoriaId** de la convocatoria seleccionada y el **tipoReporte** especificado (general o específico).

2. Condición de Tipo de Reporte:

- Verifica que **tipoReporte** no sea **null** antes de proceder con la generación del certificado.
- Según el valor de **tipoReporte**, asigna **nombreReporte** como **"certificacionGeneral"** o **"certificacionEspecifica"**.

3. Parámetros de Imagen:

- Obtiene las imágenes base64 desde la base de datos mediante **beneficiosServiceFacade.obtenerParametroSistema("LOGIN_LOGO")** y **beneficiosServiceFacade.obtenerParametroSistema("LOGIN_LOGO2")**.
- Extrae la imagen en **base64** y la agrega a **hashMap** como parámetros **IMG_BASE64** e **IMG_SECUNDARIA_BASE64**.

4. Generación del Reporte:

- Llama a **reporteUtil.generarReporte**, pasando **hashMap**, **nombreReporte**, **"PDF"**, y el contexto **context**.
- Establece el **Content-Disposition** en la respuesta **HTTP** para definir el nombre del archivo descargable (general o específico según el tipo de reporte).

5. Descarga del Archivo PDF:

- Escribe el contenido del reporte en el **ServletOutputStream** y asegura que el flujo se cierre correctamente después de la escritura.

6. Manejo de Errores y Reseteo:

- Si ocurre un error, muestra un mensaje con **toastr.error**.
- Restablece los valores de **convocatoriaEvaluador** y **tipoReporte** en **evaluacionSessController** y cierra el diálogo **modal** de manera automática.
- Método **verificarProyectosBTEvaluados**, para verificar que los proyectos del evaluador hayan finalizado su etapa de evaluación

El método **verificarProyectosBTEvaluados** es una verificación previa para la generación del certificado de un evaluador. Su propósito es comprobar que todos los proyectos de la convocatoria específica hayan finalizado la etapa de evaluación. Si algún proyecto aún no ha finalizado su etapa de evaluación, lanza una excepción **SystemException** para prevenir la generación del certificado.

Método de consulta en EJB facade, EJB service y storeManager.

BeneficiosServiceFacade.java

```
//----- Método para verificar que los proyectos del evaluador hayan finalizado su etapa de evaluación -----  
void verificarProyectosBTEvaluados(String usuario, Long convocatoriaId) throws Exception;
```

BeneficiosServiceFacadeBean.java

```
@Override  
public void verificarProyectosBTEvaluados(String usuario, Long convocatoriaId) throws  
Exception {  
    beneficiosService.verificarProyectosBTEvaluados(usuario, convocatoriaId);  
}
```

BeneficiosService.java

```
void verificarProyectosBTEvaluados(String usuario, Long convocatoriaId) throws  
SystemException;
```

BeneficiosServiceBean.java

```
@Override  
public void verificarProyectosBTEvaluados(String usuario, Long convocatoriaId) throws  
SystemException {  
    Boolean pryNoEvaluados =  
this.getStoreManager(BeneficiosStoreManager.class).verificarProyectosBTEvaluados(usuario,  
convocatoriaId);  
    if (pryNoEvaluados) {  
        //Si hay proyectos que no han finalizado su etapa de evaluacion  
        throw new  
SystemException("#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluado  
r.msgError']}");  
    }  
}
```

BeneficiosStoreManager.java

```
boolean verificarProyectosBTEvaluados(String usuario, Long convocatoriaId) throws  
SystemException;
```

BeneficiosStoreManagerImpl.java

```
@Override
public boolean verificarProyectosBTEvaluados(String usuario, Long convocatoriaId) throws
SystemException {
    Query query = em.createQuery("SELECT p "
        + " FROM Convocatoria c "
        + "      , Proyecto p "
        + "      , ProyectoEvaluacion eva "
        + " WHERE p.proyectoId = eva.proyectoId.proyectoId "
        + "       AND c.convocatoriaId = p.convocatoriaId.convocatoriaId "
        + "       AND p.exencion IN ('E') "
        + "       AND (p.cierreEvaluacion IS NULL OR p.cierreEvaluacion = 'N') "
        + "       AND eva.usuarioSigp = :usuario "
        + "       AND p.convocatoriaId.convocatoriaId = :convocatoriaId");
    query.setParameter("usuario", usuario);
    query.setParameter("convocatoriaId", convocatoriaId);
    return query.getResultList().size() > 0;
}
```

Este método en el StoreManager (**BeneficiosStoreManagerImpl**) consulta la base de datos para verificar el estado de los proyectos asociados a una convocatoria y evaluador en particular.

Detalles Técnicos:

a) Consulta de Verificación de Proyectos:

- Se ejecuta una consulta que busca proyectos asociados al evaluador (**usuario**) y a la convocatoria (**convocatoriaId**) donde:
 - El campo **exencion** del proyecto tiene el valor 'E'.
 - La evaluación del proyecto (**cierreEvaluacion**) no ha finalizado, ya sea porque está en **NULL** o tiene el valor 'N'.

b) Resultado de la Consulta:

- Si la consulta retorna proyectos, esto indica que hay al menos uno que aún no ha concluido su etapa de evaluación.
- En ese caso, el método devuelve **true**, lo que provocará que se lance una **SystemException** con un mensaje de error predefinido, bloqueando la generación del certificado.

c) Manejo de Excepción:

- Si se detecta que quedan proyectos sin evaluar, se lanza una **SystemException**, utilizando un mensaje de error configurado en el archivo de mensajes de la aplicación
(#{msg['gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.msgError']}).

Implementar reporte de "Certificado Evaluador Especifico".

En Jaspersoft fue construido un reporte llamado **certificación específica del evaluador** (**certificacionEspecificajrxml**), diseñado para generar un certificado formal para un evaluador. A continuación, se explican las secciones claves del código:

Encabezado del XML

```
<property name="ireport.zoom" value="1.0980750000000004"/>
<property name="ireport.x" value="10"/>
<property name="ireport.y" value="0"/>
<property name="net.sf.jasperreports.export.pdf.force.linebreak.policy" value="true"/>
<import value="org.apache.commons.codec.binary.*"/>
```

- **Declaración del XML:** Define la versión y codificación, y define las dependencias necesarias.
- **Atributos principales:** Establecen dimensiones de página, márgenes y propiedades adicionales.

Propiedades y Parámetros

```
<parameter name="SUBREPORT_DIR" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_BASE64" class="java.lang.String"/>
<parameter name="IMG_SECUNDARIA_BASE64" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="USUARIO_SIGP" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="CONVOCATORIA_ID" class="java.lang.Long">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
```

- **Parámetros de entrada:**

- **SUBREPORT_DIR:** Directorio de subreporte, que almacena la ubicación de otros archivos de reporte relacionados.
- **IMG_BASE64 y IMG_SECUNDARIA_BASE64:** Almacenan las imágenes codificadas en **Base64** para incluir logotipos en el reporte.
- **USUARIO_SIGP:** Identifica al usuario del sistema de gestión de proyectos (**SIGP**).
- **CONVOCATORIA_ID:** Filtra la convocatoria específica a la que corresponde el certificado.

Consulta SQL (queryString)

```
<queryString>
<![CDATA[SELECT DISTINCT(SGPCONVOCATORIAS.DESCRIPCION_CONVOCATORIA) AS DC,
(CASE
    WHEN PRYEVALUACIONES.tipo_evaluador = 'I' THEN
        UPPER(PRYEVALUACIONES.funcionario)
    ELSE
        (CASE
            WHEN PRYEVALUACIONES.evaluador_scienti_id IS NOT NULL THEN
                (UPPER(CONCAT(CONCAT(SGPEVALUADORESSCIENTI.nombre_evaluador, '
'),
                CONCAT(CONCAT(SGPEVALUADORESSCIENTI.primer_apellido,
                SGPEVALUADORESSCIENTI.segundo_apellido))))
            ELSE
                (UPPER(CONCAT(CONCAT(SGPPERSONAS.nombres, '
'),
                CONCAT(CONCAT(SGPPERSONAS.primer_apellido, ' '), SGPPERSONAS.segundo_apellido))))
        END)
    END) AS evaluador,
(CASE
    WHEN PRYEVALUACIONES.TIPO_EVALUADOR = 'I' THEN
        (SELECT pr.numero_identificacion
        FROM SGP_PERSONAS pr
        WHERE pr.usuario_sigp = ${USUARIO_SIGP})
    ELSE
        (CASE
            WHEN PRYEVALUACIONES.evaluador_scienti_id IS NOT NULL THEN
                SGPEVALUADORESSCIENTI.NRO_DOCUMENTO_IDENT
            ELSE
                SGPPERSONAS.Numero_Identificacion
        END)
    END) DOCUMENTO,
(SELECT v.descripcion_valor_definicion
FROM adm_valores_definiciones v
WHERE v.tipo_definicion = 'CODIF_EST'
AND v.valor_definicion = '090') FORMATO_VERSION
FROM PRY_EVALUACIONES PRYEVALUACIONES,
SGP_PERSONAS SGPPERSONAS,
SGP_PROYECTOS SGPPROYECTOS,
SGP_EVALUADORES_SCIENTI SGPEVALUADORESSCIENTI,
SGP_CONVOCATORIAS SGPCONVOCATORIAS
WHERE PRYEVALUACIONES.PERSONA_ID = SGPPERSONAS.PERSONA_ID(+)
AND (PRYEVALUACIONES.EVALUACION_PROYECTO_ID IN
(SELECT EVALUACION_PROYECTO_ID FROM VW_FILTRO_EVALUACION))
AND SGPPROYECTOS.PROYECTO_ID = PRYEVALUACIONES.PROYECTO_ID
```

```
AND SGPEVALUADORESSCIENTI.EVALUADOR_SCIENTI_ID(+) =
    PRYEVALUACIONES.EVALUADOR_SCIENTI_ID
AND SGPPROYECTOS.CONVOCATORIA_ID = SGPCONVOCATORIAS.CONVOCATORIA_ID
AND SGPCONVOCATORIAS.CONVOCATORIA_ID = ${CONVOCATORIA_ID}
AND PRYEVALUACIONES.USUARIO_SIGP = ${USUARIO_SIGP}]]>
</queryString>
```

- La consulta **SQL** obtiene información sobre el evaluador y la convocatoria, como:
 - Nombre del evaluador:** Genera el nombre completo del evaluador según su tipo (**tipo_evaluador**), utilizando el sistema **SCIENTI** o el registro personal (**SGPPERSONAS**).
 - Documento de identidad:** Recupera el número de identificación del evaluador.
 - Versión del formato:** Define el formato del documento, basado en los valores de definición de **ADM_VALORES_DEFINICIONES**.
- La consulta filtra los registros en función de la **CONVOCATORIA_ID** y **USUARIO_SIGP**.

Campos y Variables

```
<field name="DC" class="java.lang.String"/>
<field name="EVALUADOR" class="java.lang.String"/>
<field name="DOCUMENTO" class="java.lang.String"/>
<field name="FORMATO_VERSION" class="java.lang.String"/>
<variable name="TOTAL_PAGINAS" class="java.lang.Integer">
    <variableExpression><![CDATA[${PAGE_NUMBER}]]></variableExpression>
</variable>
```

- Define campos para cada dato recuperado en la consulta (**DC**, **EVALUADOR**, **DOCUMENTO**, **FORMATO_VERSION**).
- Variable TOTAL_PAGINAS:** Contador de páginas del reporte.

Encabezado de Página (pageHeader)

```
<pageHeader>
    <band height="75">
        <image scaleImage="RetainShape">
            <reportElement x="0" y="0" width="150" height="60"
isRemoveLineWhenBlank="true" uid="f02b44eb-a85b-4e1c-996e-69f4f8ffd3ef"/>
            <imageExpression><![CDATA[${IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64(${IMG_SECUNDARIA_BASE64}.getBytes()))
:
ByteArrayInputStream(Base64.decodeBase64(${IMG_BASE64}.getBytes()))]]></imageExpression>
        </image>
```

```
<image scaleImage="RetainShape">
  <reportElement x="362" y="0" width="150" height="60"
isRemoveLineWhenBlank="true" uuid="2ba43384-bed0-4714-8d9b-8e79fe5c9933"/>
  <imageExpression><![CDATA[ $P{IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))
: null]]></imageExpression>
</image>
</band>
</pageHeader>
```

- **Imágenes del encabezado:** Se añaden logotipos en formato **Base64**, colocados a la izquierda y derecha del encabezado.

Detalle (detail)

```
<detail>
  <band height="161">
    <staticText>
      <reportElement x="0" y="0" width="512" height="60" uuid="15b544cb-dcb8-
48fd-9e67-570c9de6da92"/>
      <textElement textAlignment="Center">
        <font fontName="Arial" size="14" isBold="true"/>
      </textElement>
      <text><![CDATA[LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E INNOVACIÓN DEL
MINISTERIO DE CIENCIA, TECNOLOGÍA E INNOVACIÓN]]></text>
    </staticText>
    <staticText>
      <reportElement positionType="Float" x="0" y="92" width="512" height="17"
uuid="d6fbbb0d-7ff9-4eeb-a71a-782a6ed7a8c5"/>
      <textElement textAlignment="Center">
        <font fontName="Arial" size="14" isBold="true"/>
      </textElement>
      <text><![CDATA[CERTIFICA]]></text>
    </staticText>
  </band>
  <band height="128">
    <textField>
      <reportElement x="0" y="0" width="512" height="72" uuid="c90a6457-23b6-
4c72-9e70-7fc0b520cd66"/>
      <textElement textAlignment="Justified" verticalAlignment="Top"
markup="none">
        <font fontName="Arial" size="11"/>
      </textElement>
      <textFieldExpression><![CDATA["Que el(la) evaluador(a) " + ${EVALUADOR}
+ " identificado(a) con cédula de ciudadanía No. " + ${DOCUMENTO} + " participó como
evaluador(a) experto(a) reconocido(a) por el Ministerio de Ciencia, Tecnología e Innovación
en las siguientes convocatorias/Invitaciones, con base en los registros disponibles en la
plataforma del Sistema de Información de Gestión de Proyectos
(SIGP):"]]]></textFieldExpression>
    </textField>
    <subreport>
      <reportElement positionType="Float" x="0" y="72" width="512" height="34"
uuid="5fc221ab-28ef-41c0-b4cd-0162eb21fc96"/>
      <subreportParameter name="USUARIO_SIGP">
        <subreportParameterExpression><![CDATA[$P{USUARIO_SIGP}]]></subreportParameterExpressio
n>
```

```

</subreportParameter>
<subreportParameter name="CONVOCATORIA_ID">

  <subreportParameterExpression><![CDATA[${P{CONVOCATORIA_ID}}]></subreportParameterExpression>
  </subreportParameter>

  <connectionExpression><![CDATA[${P{REPORT_CONNECTION}}]></connectionExpression>
  <subreportExpression><![CDATA[${P{SUBREPORT_DIR}}
"proyectosEvaluadosEvaluador.jasper"]]></subreportExpression>
  </subreport>
</band>
<band height="50">
  <textField>
    <reportElement x="0" y="0" width="512" height="50" uuid="019fe1b1-790d-4218-a981-8967d0ea1662"/>
    <textElement>
      <font fontName="Arial" size="11"/>
    </textElement>
    <textFieldExpression><![CDATA["Certificado generado a los " +
Calendar.getInstance().get(Calendar.DAY_OF_MONTH) + " días del mes de " + new
java.text.SimpleDateFormat("MMMM").format(new java.util.Date()).toLowerCase() + " de " +
Calendar.getInstance().get(Calendar.YEAR) + " por el Ministerio de Ciencia, Tecnología e
Innovación."]]></textFieldExpression>
  </textField>
</band>
</detail>

```

- **Texto estático:** Presenta un mensaje introductorio con el título del Ministerio y el propósito del certificado.
- **Campo textField:** Muestra el nombre del evaluador, su documento de identidad, y un mensaje certificando su rol como evaluador en las convocatorias especificadas.
- **Subreporte:** Incluye otro reporte **proyectosEvaluadosEvaluador.jasper**, que lista los proyectos evaluados por este evaluador específico. Los parámetros **USUARIO_SIGP** y **CONVOCATORIA_ID** se pasan al subreporte para filtrar la información.

Pie de Página (pageFooter)

```

<pageFooter>
  <band height="30" splitType="Stretch">
    <textField evaluationTime="Page" isBlankWhenNull="true">
      <reportElement x="0" y="0" width="112" height="30" forecolor="#999999"
uuid="e16776a0-4644-4fd0-8c5b-27e3a1095ebf"/>
      <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="8"/>
      </textElement>

      <textFieldExpression><![CDATA[${F{FORMATO_VERSION}}]></textFieldExpression>
    </textField>
    <textField evaluationTime="Page">

```



```
<reportElement x="235" y="10" width="45" height="20" forecolor="#999999"
uuid="4b05a348-07eb-42e8-b5a9-3e9c5c5912df"/>
<textElement verticalAlignment="Middle">
<font fontName="Arial" size="8"/>
</textElement>
<textFieldExpression><![CDATA[ "Página " + ${PAGE_NUMBER} + " de
"]]></textFieldExpression>
</textField>
<textField evaluationTime="Report">
<reportElement x="280" y="10" width="18" height="20" forecolor="#999999"
uuid="6201de6c-9c6f-4f4b-b175-fa81b4fd97c1"/>
<textElement verticalAlignment="Middle">
<font fontName="Arial" size="8"/>
</textElement>
<textFieldExpression><![CDATA[${PAGE_NUMBER}]]></textFieldExpression>
</textField>
</band>
</pageFooter>
```

- Incluye información sobre la versión del formato y la numeración de páginas.
 - **FORMATO_VERSION:** Versión del formato en texto gris.
 - **Número de página:** Muestra "Página X de Y".

Expresiones de Fecha

- **Certificado de emisión:** Genera dinámicamente la fecha actual en el certificado, indicando día, mes y año.

SUBREPORT proyectosEvaluadosEvaluador.jrxml

Este subreporte está diseñado para mostrar información de proyectos evaluados en una convocatoria específica. A continuación, se explican los elementos principales de este subreporte en iReport:

1. **Parámetros:**
 - **USUARIO_SIGP (String):** Usuario responsable de la evaluación.
 - **CONVOCATORIA_ID (Long):** ID de la convocatoria específica.
2. **Consulta SQL:** Esta consulta extrae datos de las tablas **pry_evaluaciones**, **sgp_proyectos**, y **sgp_convocatorias**:
 - **descripcion_convocatoria:** Describe la convocatoria.
 - **codigo_proyecto:** Código único del proyecto.

- **titulo:** Título del proyecto.
- **fecha_recepcion_evaluacion:** Fecha en que se recibió la evaluación.

La consulta filtra proyectos por el usuario y la convocatoria específica, y sólo incluye proyectos con exención (**p.exencion = 'E'**), ordenando los resultados por **convocatoria_id** y **proyecto_id**.

3. Campos de datos:

- **DESCRIPCION_CONVOCATORIA, CODIGO_PROYECTO, TITULO, y FECHA_RECEPCION_EVALUACION** son los campos extraídos de la consulta y utilizados en el diseño del reporte.

4. Diseño de detalles:

- Cada proyecto muestra:
 - Código (**CODIGO_PROYECTO**)
 - Descripción de la convocatoria (**DESCRIPCION_CONVOCATORIA**)
 - Título del proyecto (**TITULO**)
 - Fecha de evaluación (**FECHA_RECEPCION_EVALUACION**), que está formateada en español para mostrar "dd 'días del mes de' MMMM 'de' YYYY".

5. Estilos visuales:

- Los textos como "Código proyecto", "Título del proyecto", y "Fecha de evaluación" tienen estilos personalizados (fuente Arial, tamaño 11) y están resaltados con un fondo azul para diferenciar las secciones.

Implementar reporte de "Certificado Evaluador General".

Este archivo de reporte en **JasperReports** (**certificacionGeneral.jrxml**) está diseñado para generar un certificado general de evaluadores. A continuación, se explican los elementos clave del código:

Parámetros y Configuración

```
<property name="ireport.zoom" value="0.9982500000000003"/>
```

```
<property name="ireport.x" value="0"/>
<property name="ireport.y" value="0"/>

<import value="org.apache.commons.codec.binary.*"/>
<parameter name="CONVOCATORIA_ID" class="java.lang.Long">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="USUARIO_SIGP" class="java.lang.String">
  <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
<parameter name="IMG_BASE64" class="java.lang.String"/>
<parameter name="IMG_SECUNDARIA_BASE64" class="java.lang.String"/>
```

- **Parámetros:**
 - **CONVOCATORIA_ID:** Identificador de la convocatoria.
 - **USUARIO_SIGP:** Identificador del usuario del evaluador en el sistema.
 - **IMG_BASE64 y IMG_SECUNDARIA_BASE64:** Imágenes en formato **Base64** para insertar en el encabezado del reporte.
- **Propiedades:** Se especifican propiedades para configurar el zoom y posición inicial en la herramienta **iReport**.

Consulta de Datos (<queryString>)

```
<queryString>
  <![CDATA[SELECT      DISTINCT(SGPCONVOCATORIAS.DESCRIPCION_CONVOCATORIA)      AS
DESCRIPCION_CONVOCATORIA,
      (CASE
        WHEN PRYEVALUACIONES.tipo_evaluador = 'I' THEN
          UPPER(PRYEVALUACIONES.funcionario)
        ELSE
          (CASE
            WHEN PRYEVALUACIONES.evaluador_scienti_id IS NOT NULL THEN
              (UPPER(CONCAT(CONCAT(SGPEVALUADORESSCIENTI.nombre_evaluador, '
'),
              CONCAT(CONCAT(SGPEVALUADORESSCIENTI.primer_apellido,
              ' '),
              SGPEVALUADORESSCIENTI.segundo_apellido))))
            ELSE
              (UPPER(CONCAT(CONCAT(SGPPERSONAS.nombres,
              ' '),
              CONCAT(CONCAT(SGPPERSONAS.primer_apellido, ' '), SGPPERSONAS.segundo_apellido))))
          )
        END) AS NOMBRE,
      (CASE
        WHEN PRYEVALUACIONES.TIPO_EVALUADOR = 'I' THEN
          (SELECT pr.numero_identificacion
            FROM SGP_PERSONAS pr
            WHERE pr.usuario_sigp = ${USUARIO_SIGP})
        ELSE
          (CASE
            WHEN PRYEVALUACIONES.evaluador_scienti_id IS NOT NULL THEN
              SGPEVALUADORESSCIENTI.NRO_DOCUMENTO_IDENT
```

```

ELSE
    SGPPERSONAS.Numero_Identificacion
END)
END) DOCUMENTO,
(SELECT COUNT(DISTINCT(p.evaluacion_proyecto_id))
FROM PRY_EVALUACIONES p, SGP_PROYECTOS pr
WHERE p.usuario_sigp      = ${P{USUARIO_SIGP}}
AND p.proyecto_id      (+)= pr.proyecto_id
AND pr.convocatoria_id  = ${P{CONVOCATORIA_ID}}) AS NRO_EVALUACIONES,
(SELECT v.descripcion_valor_definicion
FROM adm_valores_definiciones v
WHERE v.tipo_definicion  = 'CODIF_EST'
AND v.valor_definicion  = '100') FORMATO_VERSION
FROM PRY_EVALUACIONES      PRYEVALUACIONES,
SGP_PERSONAS              SGPPERSONAS,
SGP_PROYECTOS              SGPPROYECTOS,
SGP_EVALUADORES_SCIENTI   SGPEVALUADORESSCIENTI,
SGP_CONVOCATORIAS         SGPCONVOCATORIAS
WHERE PRYEVALUACIONES.PERSONA_ID = SGPPERSONAS.PERSONA_ID(+)
AND (PRYEVALUACIONES.EVALUACION_PROYECTO_ID IN
(SELECT EVALUACION_PROYECTO_ID FROM VW_FILTRO_EVALUACION))
AND SGPPROYECTOS.PROYECTO_ID = PRYEVALUACIONES.PROYECTO_ID
AND SGPEVALUADORESSCIENTI.EVALUADOR_SCIENTI_ID(+) =
PRYEVALUACIONES.EVALUADOR_SCIENTI_ID
AND SGPPROYECTOS.CONVOCATORIA_ID = SGPCONVOCATORIAS.CONVOCATORIA_ID
AND SGPCONVOCATORIAS.CONVOCATORIA_ID = ${P{CONVOCATORIA_ID}}
AND PRYEVALUACIONES.USUARIO_SIGP = ${P{USUARIO_SIGP}}]]>
</queryString>

```

La consulta SQL extrae información relevante del evaluador, incluyendo:

- **Descripción de la convocatoria** (DESCRIPCION_CONVOCATORIA).
- **Nombre completo** (NOMBRE) y **documento de identificación** (DOCUMENTO) del evaluador.
- **Número de evaluaciones** realizadas por el evaluador en la convocatoria (NRO_EVALUACIONES).
- **Formato de versión** (FORMATO_VERSION), que muestra una descripción estática de la definición de estado.

La consulta utiliza un caso condicional (**CASE**) para determinar el tipo de evaluador y sus datos asociados, accediendo a distintas tablas y campos según el tipo de evaluador.

Definición de Campos (<field>)

```

<field name="DESCRIPCION_CONVOCATORIA" class="java.lang.String"/>
<field name="NOMBRE" class="java.lang.String"/>

```

```
<field name="DOCUMENTO" class="java.lang.String"/>
<field name="NRO_EVALUACIONES" class="java.math.BigDecimal"/>
<field name="FORMATO_VERSION" class="java.lang.String"/>
```

Define los campos que se utilizarán en el diseño del reporte, tales como:

- DESCRIPCION_CONVOCATORIA, NOMBRE, DOCUMENTO, NRO_EVALUACIONES y FORMATO_VERSION.

Encabezado de Página (<pageHeader>)

```
<pageHeader>
  <band height="75" splitType="Stretch">
    <image scaleImage="RetainShape">
      <reportElement x="0" y="0" width="150" height="60"
isRemoveLineWhenBlank="true" uuid="03b1e6f9-5ab6-4cad-816a-bae06f934f06"/>
      <imageExpression><![CDATA[{$P{IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_SECUNDARIA_BASE64}.getBytes()))
: new ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))}]></imageExpression>
    </image>
    <image scaleImage="RetainShape">
      <reportElement x="362" y="0" width="150" height="60"
isRemoveLineWhenBlank="true" uuid="867d3fc9-c849-482c-a969-19716f3d19b6"/>
      <imageExpression><![CDATA[{$P{IMG_SECUNDARIA_BASE64} != null
? new ByteArrayInputStream(Base64.decodeBase64($P{IMG_BASE64}.getBytes()))
: null}]></imageExpression>
    </image>
  </band>
</pageHeader>
```

Incluye dos imágenes en la parte superior del reporte:

- La primera imagen se basa en el parámetro **IMG_SECUNDARIA_BASE64** o **IMG_BASE64**, y se ubica en la esquina izquierda.
- La segunda imagen se basa en **IMG_BASE64** y se muestra en la esquina derecha.

Detalle del Contenido (<detail>)

```
<detail>
  <band height="161" splitType="Stretch">
    <staticText>
      <reportElement x="0" y="14" width="512" height="52" uuid="e3cb5889-c45a-4f95-8df8-130de71ad5a1"/>
      <textElement textAlignment="Center">
        <font fontName="Arial" size="14" isBold="true"/>
```

```

</textElement>
<text><![CDATA[LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E INNOVACIÓN DEL
MINISTERIO DE CIENCIA, TECNOLOGÍA E INNOVACIÓN]]></text>
</staticText>
<staticText>
<reportElement positionType="Float" x="0" y="92" width="512" height="17"
uuid="cdd59f4a-06aa-4c68-98dc-444050a0cb54"/>
<textElement textAlignment="Center">
<font fontName="Arial" size="14" isBold="true"/>
</textElement>
<text><![CDATA[CERTIFICA]]></text>
</staticText>
</band>
<band height="233">
<textField>
<reportElement x="0" y="0" width="512" height="196" uuid="e55028c3-d4af-
46c0-9274-60637ca07359"/>
<textElement textAlignment="Justified" verticalAlignment="Top"
markup="none">
<font fontName="Arial" size="11"/>
</textElement>
<textFieldExpression><![CDATA["Que el(la) evaluador(a) " + ${NOMBRE} +
" identificado(a) con cédula de ciudadanía No. " + ${DOCUMENTO} + " participó como
evaluador(a) experto(a) reconocido(a) por el Ministerio de Ciencia, Tecnología e Innovación
en la convocatoria/Invitación \" + ${DESCRIPCION_CONVOCATORIA} + "\".\n\n" +

"Que el(la) evaluador(a) " + ${NOMBRE} + " evaluó " + ${NRO_EVALUACIONES} + " propuestas
presentadas a esta convocatoria, con base en los registros disponibles en la plataforma del
Sistema de Información de Gestión de Proyectos (SIGP).\n\n\n" +

"Certificado generado a los " + Calendar.getInstance().get(Calendar.DAY_OF_MONTH) + " días
del mes de " + new java.text.SimpleDateFormat("MMMM").format(new
java.util.Date()).toLowerCase() + " de " + Calendar.getInstance().get(Calendar.YEAR) + " por
el Ministerio de Ciencia, Tecnología e Innovación."]]></textFieldExpression>
</textField>
</band>
</detail>

```

El cuerpo del reporte contiene:

- **Título de la entidad certificadora** en texto centrado.
- **Certificado de Participación:** Un campo de texto dinámico que genera el certificado con detalles como:
 - Nombre y documento del evaluador.
 - Convocatoria en la que participó.
 - Número de evaluaciones realizadas.
 - Fecha en que se emite el certificado, generado automáticamente.

El texto es generado dinámicamente con expresiones de Java para obtener la fecha actual y formatearla.

Pie de Página (<pageFooter>)

```
<pageFooter>
  <band height="30" splitType="Stretch">
    <textField evaluationTime="Page">
      <reportElement x="239" y="10" width="45" height="20" forecolor="#999999"
      uuid="37afe06b-76cc-4759-b42c-bbe4b7e18eb5"/>
      <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="8"/>
      </textElement>
      <textFieldExpression><![CDATA["Página " + $V{PAGE_NUMBER} + " de
      "]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Report">
      <reportElement x="284" y="10" width="18" height="20" forecolor="#999999"
      uuid="15312717-b274-45c0-a784-df04a7476bdf"/>
      <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="8"/>
      </textElement>
      <textFieldExpression><![CDATA[$V{PAGE_NUMBER}]]></textFieldExpression>
    </textField>
    <textField evaluationTime="Page" isBlankWhenNull="true">
      <reportElement x="0" y="0" width="112" height="30" forecolor="#999999"
      uuid="efc2f226-21f9-4fa7-ab33-375a745eefdc"/>
      <textElement verticalAlignment="Middle">
        <font fontName="Arial" size="8"/>
      </textElement>

      <textFieldExpression><![CDATA[$F{FORMATO_VERSION}]]></textFieldExpression>
    </textField>
  </band>
</pageFooter>
```

En el pie de página se muestran:

- **Número de página:** "Página X de Y".
- **Formato de versión:** Muestra el campo **FORMATO_VERSION** en texto gris claro.

Expresiones Java

Utiliza expresiones para generar contenido dinámico, como:

- **Texto del certificado:** Generado en la sección de detalle utilizando expresiones **CDATA**.
- **Fecha del certificado:** Se obtiene mediante **Calendar.getInstance()** y **SimpleDateFormat** para mostrar el día y mes actuales.

Funcionalidad de Generación y Descargar de Certificados de Evaluación por Par Evaluador Scienti desde la pantalla de Configuración - Personal - Evaluadores Scienti

Se crea en el módulo **Configuración – personas** en la pantalla de evaluadores scienti (**app/configuracion/evaluadoresScienti/evaluadoresScienti.xhtml**), la opción de generar un certificado de evaluador, el cual abre ventana de dialogo y permita al evaluador indicar el tipo de convocatoria (proyectos/programas o beneficios tributarios), la convocatoria y el tipo de certificado (especifico o general) que desea generar

- Botón certificado evaluador

```
<p:commandLink id="generarCertificados"
                process="@this"

title="#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.btn']
}"

update="#{evaluadoresScientiProViewController.modulo}formReporteEvaluador"

actionListener="#{evaluadoresScientiConvController.inicializarEvaluador(evalScienti)}"
                oncomplete="$('#dlgCertificadoEvaluador').modal()"
                styleClass="btn btn-xs btn-primary">
    <i class="fa fa-file-pdf-o"></i>
</p:commandLink>
```

El componente **p:commandLink** para generar el certificado de evaluador funciona de la siguiente manera:

1. Configuración del Botón:

- El botón está estilizado como un botón primario pequeño con un ícono (**fa fa-file-pdf-o**) para indicar la generación de un **PDF**.
- El atributo **title** establece un texto de ayuda con una clave de mensaje.

2. Inicialización del Diálogo:

- **process="@this"** limita el procesamiento del formulario solo al botón, reduciendo la carga.
- **update="#{evaluadoresScientiProViewController.modulo}formReporteEvaluador"** especifica qué componente actualizar después de completar la acción.

- **actionListener="#{evaluadoresScientiConvController.inicializarEvaluador(evalScienti)}"** inicializa los datos del evaluador llamando a un método en el controlador (**inicializarEvaluador**), pasando **evalScienti** como argumento.
- **oncomplete="\$('#dlgCertificadoEvaluador').modal()"** abre el cuadro de diálogo (**dlgCertificadoEvaluador**) para permitir que los usuarios seleccionen los parámetros para la generación del certificado.

3. Propósito del Diálogo:

- Una vez que se abre el diálogo, los usuarios pueden elegir opciones como el "tipo de convocatoria" (**proyectos/programas o beneficios tributarios**), la "convocatoria" específica y el tipo de certificado (específico o general).
- Diálogo de Generación de Certificado para el Evaluador

Este diálogo permite al evaluador personalizar el certificado de acuerdo con el tipo de convocatoria, la convocatoria específica, y el tipo de certificado, asegurando que el documento se adapte a las necesidades de cada contexto.

```
<!--DIALOGO DE CERTIFICADO DEL EVALUADOR -->
<div class="modal inmodal" id="dlgCertificadoEvaluador" tabindex="-1" role="dialog" aria-
hidden="true">
    <div class="modal-dialog modal-lg">
        <div class="modal-content animated bounceInRight">
            <h:form
id="#{evaluadoresScientiProViewController.modulo}formReporteEvaluador">
                <div class="modal-header">
                    <button type="button" class="close" data-
dismiss="modal"><span aria-hidden="true"></span><span class="sr-only">Close</span></button>
                    <i class="fa fa-file-pdf-o modal-icon"></i>
                    <h4 tabindex="0" class="modal-
title">#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.heade
r']}</h4>
                    <medium class="font-
bold">#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.mensaj
e']}</medium>
                </div>
                <div class="modal-body">
                    <div class="form-horizontal">
                        <div class="col-lg-10 col-lg-offset-1">
                            <div class="alert alert-info">
                                <p class="medium"><i class="fa
fa-info-circle"></i>
                                #{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.alert']}
                            </div>
                        </div>
                    </div>
                </div>
            </h:form>
        </div>
    </div>
</div>
```

```

                                <div class="col-lg-12">
                                    <p:messages
id="msgCertificados"
                                closable="true"/>
                                </div>
                                </div>
                                <div class="row">
                                    <div class="col-lg-12">
                                        <div class="form-group">
                                            <p:outputLabel
value="#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.tipoC
onvocatoria']}"
                                for="tipoConvocatoria"
                                style="text-align: right"
                                styleClass="col-lg-2 control-label"
                                />
                                <div class="col-lg-10">
                                    <p:selectOneMenu id="tipoConvocatoria"
                                value="#{evaluadoresScientiSessController.tipoConvocatoria}"
                                required="true"
                                requiredMessage="#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvalu
ador.tipoConvocatoria.req']}"
                                appendTo="@this"
                                panelStyle="position: initial;"
                                style="width:100%"
                                filter="false"
                                filterMatchMode="contains"
                                >
                                    <f:selectItems value="#{configAppController.listaTipoConvocatorias}"/>
                                <p:ajax
update="convocatoriaEva-ES,
#{evaluadoresScientiProViewController.modulo}formReporteEvaluador:panelGenerarReporte"
                                process="@this,tipoConvocatoria"
                                event="change"
                                listener="#{evaluadoresScientiConvController.cargarListaConvocatoriasEvaluadores(evalu
adoresScientiSessController.tipoConvocatoria)}"
                                />
                                </p:selectOneMenu>

```

```

</div>
</div>
</div>
<div class="row">
  <div class="col-lg-12">
    <div class="form-group">
      <p:outputLabel
value="#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.convo
catoria']}"
for="convocatoriaEva-ES"
style="text-align: right"
styleClass="col-lg-2 control-label"
/>
<div class="col-lg-10">
  <p:selectOneMenu id="convocatoriaEva-ES"
value="#{evaluadoresScientiSessController.convocatoriaEvaluador}"
required="true"
requiredMessage="#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvalu
ador.convocatoria.req']}"
appendTo="@this"
panelStyle="position: initial;"
converter="convocatoriaConverter"
style="width:100%"
filter="true"
filterMatchMode="contains"
>
  <f:selectItem itemLabel="#{msg['sb.selectOneMenu.registro']}" />
  <f:selectItems value="#{evaluadoresScientiSessController.convocatoriaEvaluadorList}"
var="convocatoria"
itemLabel="#{convocatoria.descripcionConvocatoria}"
itemValue="#{convocatoria}"
/>
</p:selectOneMenu>
</div>
</div>
</div>
</div>

```

```

<div class="row">
    <div class="col-lg-12">
        <div class="form-group">
            <p:outputLabel
value="#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.tipoC
ertificado']}"
for="tipoReporte"
style="text-align: right"
styleClass="col-lg-2 control-label"
/>
<div class="col-lg-4">
<p:selectOneMenu id="tipoReporte"
value="#{evaluadoresScientiSessController.tipoReporte}"
required="true"
requiredMessage="#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvalu
ador.tipoCertificado.re']}"
appendTo="@this"
panelStyle="position: initial;"
style="width:100%;"
>
<f:selectItems value="#{configApplController.listaTipoCertificado}"/>
</p:selectOneMenu>
</div>
</div>
</div>
<p:spacer />
<p:outputPanel id="panelGenerarReporte">
    <div class="row" style="text-align:
center">
        <div class="col-lg-12">
            <p:commandLink
id="generarReporte"
actionListener="#{evaluadoresScientiConvController.generarReporteCertificadoEvaluadores('SCI
ENTI')}"
process="@this,tipoConvocatoria,convocatoriaEva-ES, tipoReporte"
ajax="false"
update="@form"
styleClass="btn btn btn-primary"

```

```

        disabled="#{evaluadoresScientiSessController.tipoConvocatoria == null}"
    >
        #{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.generar
Certificado']}
                                </p:commandLink>
                                </div>
                                </div>
                                </p:outputPanel>
                                </div>
                                </div>
                                <div class="modal-footer">
                                    <button type="button" class="btn btn-white" data-
dismiss="modal">#{msg['sb.component.boton.cancelar']}</button>
                                </div>
                                </h:form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

```

A continuación, se detallan los componentes y su función en el formulario.

a) Estructura y Diseño del Diálogo:

- **Etiqueta <div class="modal inmodal" id="dlgCertificadoEvaluador">**: Este div crea un cuadro de diálogo modal con animación, que se abre al seleccionar la opción de "Generar Certificado" en la pantalla de evaluadores.
- **<div class="modal-dialog modal-lg">**: Configura el tamaño del diálogo para que ocupe un espacio amplio en pantalla (**modal-lg**).
- **Icono y Título**: La cabecera incluye un icono PDF y un título de diálogo, configurado con el mensaje
#{msg['gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.header']}.

b) Cuerpo del Formulario (<h:form

id="#{evaluadoresScientiProViewController.modulo}formReporteEvaluador">):

- **Mensaje Informativo**: Dentro del **<div class="alert alert-info">**, se muestra un mensaje de ayuda que informa al usuario sobre los pasos para generar el certificado.

- **Componente de Mensajes:** El componente `<p:messages id="msgCertificados" closable="true"/>` permite mostrar mensajes de éxito o error al usuario, y estos mensajes se pueden cerrar manualmente.
- c) **Selección de Parámetros:**
 - **Tipo de Convocatoria:**
 - Campo: `<p:selectOneMenu id="tipoConvocatoria">`
 - Permite al evaluador seleccionar el tipo de convocatoria (como proyectos/programas o beneficios tributarios).
 - Incluye una lista de opciones generada a partir del controlador `configApplController.listaTipoConvocatorias`.
 - **Evento Ajax:** Al cambiar el valor, se actualiza la lista de convocatorias mediante `evaluadoresScientiConvController.cargarListaConvocatoriasEvaluadores`, que ajusta el contenido según el tipo seleccionado.
 - **Convocatoria:**
 - Campo: `<p:selectOneMenu id="convocatoriaEva-ES">`
 - Muestra una lista de convocatorias específicas relacionadas al tipo previamente seleccionado. Utiliza un converter para manejar la conversión de datos y mostrar etiquetas descriptivas.
 - **Tipo de Certificado:**
 - Campo: `<p:selectOneMenu id="tipoReporte">`
 - Permite elegir el tipo de certificado que se generará, con opciones como "específico" o "general". La lista se obtiene de `configApplController.listaTipoCertificado`.
- d) **Generación del Certificado (`<p:outputPanel id="panelGenerarReporte">`):**
 - El botón "Generar Certificado" (`<p:commandLink id="generarReporte">`) llama al método

evaluadoresScientiConvController.generarReporteCertificadoEvaluadores, que procesa y genera el certificado PDF.

○ **Propiedades del Botón:**

- **process="@this, tipoConvocatoria, convocatoriaEva-ES, tipoReporte"** asegura que solo los campos relevantes sean procesados.
- **ajax="false"** realiza una redirección sin **AJAX** para manejar el archivo descargado.
- **disabled** evita que el botón esté activo si el **tipoConvocatoria** no está seleccionado.

e) **Pie de Diálogo:**

- **Botón Cancelar:** El botón de cancelar cierra el diálogo sin realizar cambios.

- **Metodo cargarListaConvocatoriasEvaluadores**

Este método **cargarListaConvocatoriasEvaluadores** es responsable de cargar la lista de convocatorias asociadas al evaluador seleccionado en función del tipo de convocatoria elegido en el diálogo.

```
/**
 * Metodo para cargar el listado de convocatorias del evaluador desde la
 * pantalla de evaluadores SIGP
 *
 * @param tipoConvocatoria
 */
public void cargarListaConvocatoriasEvaluadores(String tipoConvocatoria) {
    List<ConvocatoriaDTO> listaConvocatorias = new ArrayList<>();
    try {
        if (evaluadoresScientiSessController.getSelected().getUsuarioSigp() != null
            &&
            !evaluadoresScientiSessController.getSelected().getUsuarioSigp().isEmpty()) {
            String usuario =
            evaluadoresScientiSessController.getSelected().getUsuarioSigp();
            if (tipoConvocatoria.equals("E")) { // Convocatorias de Beneficios
                listaConvocatorias =
                beneficiosServicefacade.obtenerConvocatoriasBTEvaluador(usuario);
            } else { // Convocatorias de Proyectos y Programas
                listaConvocatorias =
                proyectoServicefacade.obtenerConvocatoriasEvaluador(usuario);
            }
        }
    }
}
```

```
    }  
    evaluadoresScientiSessController.setConvocatoriaEvaluadorList(listaConvocatorias);  
  } catch (Exception ex) {  
    LOG.error("Error al obtener el listado de convocatorias del Evaluador ", ex);  
  }  
}
```

Aquí está la explicación paso a paso:

Descripción del Método

1. Parámetro de Entrada:

- **tipoConvocatoria**: Define el tipo de convocatoria que se seleccionó en el diálogo (por ejemplo, "E" para Beneficios Tributarios o cualquier otro valor para Proyectos y Programas).

2. Inicialización de Variables:

- Se crea una lista vacía **listaConvocatorias** de tipo **ConvocatoriaDTO** para almacenar las convocatorias recuperadas según el tipo de convocatoria.

3. Condición de Verificación:

- Comprueba que el evaluador actual tenga un usuario asociado (**usuarioSigg**). Si el usuario es válido y no está vacío, se procede a buscar las convocatorias.
- El usuario del evaluador se obtiene de **evaluadoresScientiSessController.getSelected().getUsuarioSigg()**.

4. Lógica de Carga de Convocatorias:

- **Tipo de Convocatoria "E" (Beneficios Tributarios):**
 - Si el **tipoConvocatoria** es igual a "E", se obtienen las convocatorias de beneficios tributarios usando **beneficiosServicefacade.obtenerConvocatoriasBTEvaluador(usuario)**.
- **Otros Tipos de Convocatorias (Proyectos y Programas):**

- Si el **tipoConvocatoria** no es "E", se obtienen las convocatorias relacionadas con proyectos y programas mediante **proyectoServicefacade.obtenerConvocatoriasEvaluador(usuario)**.

5. Asignación de Convocatorias:

- Una vez obtenida la lista de convocatorias, se asigna al controlador de sesión **evaluadoresScientiSessController** usando **setConvocatoriaEvaluadorList(listaConvocatorias)**.

6. Manejo de Excepciones:

- En caso de que ocurra una excepción, se registra un error en el log indicando que hubo un problema al obtener el listado de convocatorias del evaluador.

- Metodo **generarReporteCertificadoEvaluadores**.

Este método, **generarReporteCertificadoEvaluadores**, es responsable de generar un certificado en PDF para los evaluadores, con la opción de generar un certificado general o específico. La generación se basa en el tipo de convocatoria y otros parámetros seleccionados en la interfaz. Aquí está la explicación paso a paso del proceso:

```
/**
 * Metodo para cargar el listado de convocatorias del evaluador desde la
 * pantalla de evaluadores SIGP
 *
 * @param tipoConvocatoria
 */
public void cargarListaConvocatoriasEvaluadores(String tipoConvocatoria) {
    List<ConvocatoriaDTO> listaConvocatorias = new ArrayList<>();
    try {
        if (evaluadoresScientiSessController.getSelected().getUsuarioSigp() != null
            &&
            !evaluadoresScientiSessController.getSelected().getUsuarioSigp().isEmpty()) {
            String usuario =
            evaluadoresScientiSessController.getSelected().getUsuarioSigp();
            if (tipoConvocatoria.equals("E")) { // Convocatorias de Beneficios
            Tributarios
                listaConvocatorias =
            beneficiosServicefacade.obtenerConvocatoriasBTEvaluador(usuario);
            } else { // Convocatorias de Proyectos y Programas
                listaConvocatorias =
            proyectoServicefacade.obtenerConvocatoriasEvaluador(usuario);
            }
        }
    }
}
```

```
evaluadoresScientiSessController.setConvocatoriaEvaluadorList(listaConvocatorias);  
} catch (Exception ex) {  
    LOG.error("Error al obtener el listado de convocatorias del Evaluador ", ex);  
}  
}
```

Descripción del Método

1. Parámetros de Entrada:

- **option**: Define el origen del usuario (por ejemplo, "SIGP" o "SCIENTI").

2. Contexto de la Respuesta HTTP:

- Obtiene el contexto de Faces (**FacesContext**) y la respuesta HTTP (**HttpServletResponse**) para configurar y enviar el archivo PDF al cliente.

3. Asignación del Usuario:

- Según el valor de **option**, asigna el usuario correspondiente desde el controlador de sesión **evaluadoresScientiSessController**.

4. Obtención de Datos:

- Obtiene la ID de convocatoria (**convocatoriald**), el tipo de convocatoria (**tipoConvocatoria**), y el tipo de reporte (**tipoReporte**) desde el controlador de sesión.

5. Configuración de Parámetros para el Reporte:

- Crea una instancia de **ReporteUtil** y un **HashMap** para almacenar parámetros necesarios para la generación del reporte.

6. Verificación de Tipo de Convocatoria:

- **"E" para Beneficios Tributarios**: Llama a **verificarProyectosBTEvaluados** para confirmar los proyectos evaluados en esta categoría y establece la carpeta del reporte a "CertificadoEvaluadorProyectoBT".
- **Proyectos y Programas**: Llama a **verificarProyectosEvaluados** y establece la carpeta a "CertificadoEvaluadorProyecto".

7. Determinación del Nombre del Reporte:

- Asigna nombreReporte según el tipo de reporte: "**certificacionGeneral**" para un reporte general, y "**certificacionEspecificica**" para un reporte específico.

8. Configuración de Logos en el Reporte:

- Obtiene las imágenes de los logos desde la base de datos, convertidos a formato base64, y los inserta en el hashMap como **IMG_BASE64** y **IMG_SECUNDARIA_BASE64**.

9. Agrega Otros Parámetros al hashMap:

- Asigna la convocatoria (**CONVOCATORIA_ID**) y el usuario (**USUARIO_SIGP**) en el **hashMap**.

10. Generación y Envío del PDF:

- Utiliza **reporteUtil.generarReporte** para crear el PDF y lo configura como archivo de descarga en la respuesta **HTTP**.
- Establece el nombre del archivo adjunto como "CertificadoGeneral" o "CertificadoEspecifico" seguido de la ID de convocatoria.
- Escribe el contenido del archivo en el **ServletOutputStream** y finaliza la respuesta.

11. Limpieza y Actualización de la Interfaz:

- Restablece los valores de **ConvocatoriaDTO** y **tipoReporte** en el controlador de sesión.
- Recarga las listas necesarias con **cargarListas()** y cierra el diálogo en la interfaz.

12. Manejo de Excepciones:

- En caso de error, se muestra un mensaje de error mediante toastr y se registra el error en el log. También se restablecen los datos de sesión y se cierra el diálogo.

Este método asegura que el certificado del evaluador se genere y se descargue en función de los parámetros seleccionados, mostrando un flujo eficiente y detallado desde la generación hasta la entrega del archivo PDF.

- Creacion de parámetros para formatos de los reportes de certificados

```
-- formatos de reportes

insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
values (null, 'CODIF_EST', '090', 'Código: A204PR01M01
Versión: 01
Fecha: 2020-02-07', 'S', 'CARACTER', null, null, 'FORMATO CERTIFICADO ESPECIFICO EVALUADOR
CONVOCATORIA BENEFICIOS TRIBUTARIOS', 'N', null, 'SIGP_WEB', to_date('23-02-2024', 'dd-mm-
yyyy'), null, null, null, null, null, null, null, null);

insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
values (null, 'CODIF_EST', '100', 'Código: A204PR01M01
Versión: 01
Fecha: 2020-02-07', 'S', 'CARACTER', null, null, 'FORMATO CERTIFICADO GENERAL EVALUADOR
CONVOCATORIA BENEFICIOS TRIBUTARIOS', 'N', null, 'SIGP_WEB', to_date('23-02-2024', 'dd-mm-
yyyy'), null, null, null, null, null, null, null, null);
```

- Se crearon reportes similares a los certificados de beneficios tributarios para proyectos/programas, con la diferencia que el valor **exencion** en la consulta de estos incluye proyectos y programas (**exencion IN ('P','R')**)
- Internacionalización en Español e Ingles

```
#ES
sb.mensajes.procesando.label          = Procesando su solicitud...
sb.mensajes.procesando.msg            = Por favor, espere. Esto puede tomar algunos segundos.

gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.tipoConvocatoria
= Tipo de Convocatoria
gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.tipoConvocatoria.re
q  = Debe seleccionar el tipo de convocatoria

gestion.beneficiosTributarios.evaluadores.calificacion.cierreEvaluacionPry.close.msg      =
Se le informa que la Evaluación del Proyecto ha sido cerrada y finalizada por el Equipo Técnico
del Ministerio de Ciencia, Tecnología e Innovación.

gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn              =
Certificado Evaluador
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn.title        =
Generar certificado de evaluación por convocatoria
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.header          =
Certificado Evaluador
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.mensaje          =
Generar certificado de evaluación de proyectos
```

```

gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.alert =
Solo podrá descargar los certificados de una convocatoria, siempre y cuando los proyectos que
usted evaluó hayan finalizado su etapa de evaluación.
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoConvocatoria
= Tipo de Convocatoria
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoConvocatoria.req
= Debe seleccionar el tipo de convocatoria
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.convocatoria =
Convocatoria
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.convocatoria.req
= Debe seleccionar una convocatoria
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoCertificado
= Tipo de Certificado
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoCertificado.re
= Debe seleccionar un tipo de certificado
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.generarCertificado
= Generar Certificado
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.msgError =
No puede generar el certificado de evaluación ya que alguno de los proyectos que usted evaluó
en la convocatoria seleccionada, no ha finalizado su evaluación

#EN
sb.mensajes.procesando.label = Processing your request...
sb.mensajes.procesando.msg = Please wait. This may take a few seconds.

gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.tipoConvocatoria
= Type of Call
gestion.proyectosProgramas.evaluadores.calificacion.certificadoEvaluador.tipoConvocatoria.re
q = You must select the type of call

gestion.beneficiosTributarios.evaluadores.calificacion.cierreEvaluacionPry.close.msg =
You are informed that the Project Evaluation has been closed and finalized by the Technical
Team of the Ministry of Science, Technology and Innovation.

gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn =
Evaluador Certificate
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.btn.title =
Generate evaluation certificate by call
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.header =
Evaluador Certificate
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.mensaje =
Generate project evaluation certificate
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.alert =
You will only be able to download the certificates of a call, as long as the projects that
you evaluated have completed their evaluation stage.
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoConvocatoria
= Type of Call
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoConvocatoria.req
= You must select the type of call
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.convocatoria =
Call
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.convocatoria.req
= You must select a call
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoCertificado
= Type of Certificate
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.tipoCertificado.re
= You must select a type of certificate
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.generarCertificado
= Generate Certificate

```

```
gestion.beneficiosTributarios.evaluadorProyectos.certificadoEvaluador.msgError =  
You cannot generate the evaluation certificate since one of the projects that you evaluated  
in the selected call has not completed its evaluation
```

Configurar Gran Áreas Por Convocatoria y Disciplina OCDE proyectos y beneficios tributarios

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

1. Configurar Gran Áreas Por Convocatoria y Disciplina OCDE
2. Registrar Código BPIN y Número de Contrato en proyectos de BT

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

- Creación de nueva pantalla de configuración de grandes áreas por convocatoria
- Se agrega campo que permite identificar si se registra código BPIN
- Se agrega campo que permite mostrar el código BPIN
- Se agrega campo que permite identificar si se registra número de contrato
- Se agrega campo que permite mostrar el número de contrato
- Se agrega campo que permite mostrar la disciplina OCDE a partir del área OCDE seleccionada

Aplicativo de Gestión

Creación pantalla de Configuración de Gran Áreas por Convocatoria

- Creación nuevo menú de configuración de Áreas por Convocatoria

```
--funciones  
insert into sb_funciones (FUNCION_ID, APLICACION, FUNCION, NOMBRE_FUNCION, TIPO_FUNCION, MENU,  
OPCION, URL, URL_AYUDA, FUNCION_PADRE, ICONO, USUARIO_CREADOR, FECHA_CREACION,  
USUARIO_MODIFICADOR, FECHA_MODIFICACION)  
values (null, 'SIGP_WEB', 'GEST1330', 'Gran Áreas por Convocatorias', 'F', null, null,  
'/app/configuracion/convocatorias/granAreasConvocatorias.xhtml', null, 'GEST1050', null,  
'SIGP_WEB', to_date('23-05-2024', 'dd-mm-yyyy'), null, null);  
  
--derechos  
insert into sb_derechos_funciones (DERECHO_FUNCION_ID, APLICACION, FUNCION, GRUPO, ESTADO,  
USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION)  
values (null, 'SIGP_WEB', 'GEST1330', 'SGPS_ADMINISTRADOR', 'H', 'SIGP_WEB', to_date('23-05-  
2024', 'dd-mm-yyyy'), null, null);
```

- Crear nueva tabla de configuración de grandes áreas por convocatoria (**SGP_CONVOCATORIAS_GRAN_AREAS**) en la base de datos sigp

```
-- Create table
create table SGP_CONVOCATORIAS_GRAN_AREAS
(
  convocatoria_gran_area_id NUMBER(15) not null,
  convocatoria_id           NUMBER(15) not null,
  gran_area_id             NUMBER(15) not null,
  usuario_creador          VARCHAR2(30) not null,
  fecha_creacion           DATE not null,
  usuario_modificador      VARCHAR2(30),
  fecha_modificacion       DATE
)
tablespace DATOS
pctfree 10
initrans 1
maxtrans 255
storage
(
  initial 64K
  next 1M
  minextents 1
  maxextents unlimited
);
-- Add comments to the table
comment on table SGP_CONVOCATORIAS_GRAN_AREAS
  is 'Almacena la información de la configuración de gran areas por convocatoria';
-- Add comments to the columns
comment on column SGP_CONVOCATORIAS_GRAN_AREAS.convocatoria_gran_area_id
  is 'Identificador llave primaria';
comment on column SGP_CONVOCATORIAS_GRAN_AREAS.convocatoria_id
  is 'Identificador de la convocatoria';
comment on column SGP_CONVOCATORIAS_GRAN_AREAS.gran_area_id
  is 'Identificador de la gran área';
comment on column SGP_CONVOCATORIAS_GRAN_AREAS.usuario_creador
  is 'Usuario creador del registro';
comment on column SGP_CONVOCATORIAS_GRAN_AREAS.fecha_creacion
  is 'Fecha de creacion';
comment on column SGP_CONVOCATORIAS_GRAN_AREAS.usuario_modificador
  is 'Usuario modificador del registro';
comment on column SGP_CONVOCATORIAS_GRAN_AREAS.fecha_modificacion
  is 'Fecha última de modificacion del registro';
-- Create/Recreate indexes
create index SGPCONVGARE_ADMVALDEF_FK_I on SGP_CONVOCATORIAS_GRAN_AREAS (GRAN_AREA_ID)
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
  initial 64K
  next 1M
  minextents 1
  maxextents unlimited
);
create index SGPCONVGARE_SGPCONV_FK_I on SGP_CONVOCATORIAS_GRAN_AREAS (CONVOCATORIA_ID)
tablespace INDICES
pctfree 10
```



- Adición del menú **"Gran Áreas por Convocatoria"** en la plantilla del menú del aplicativo de gestión

```
<!-- Gran Areas -->
<p:menuitem value="#{msg['gestion.configuracion.convocatoria.granAreas.button']}"
            icon="fa fa-cubes"
outcome="/app/configuracion/convocatorias/granAreasConvocatorias.xhtml"
            rendered="#{convocatoriaSessController.selected.convEstimulos eq 'N'
                        and
(convocatoriaSessController.selected.convocatoriaPatente == null
                        or
convocatoriaSessController.selected.convocatoriaPatente eq 'N')
                        and
SeguridadSessController.mapaMenu.containsKey('GEST1330')}"
            />
```

El botón "Gran Áreas" permite acceder a la configuración de las grandes áreas para una convocatoria en particular. Este botón solo se muestra bajo ciertas condiciones, lo cual ayuda a controlar el acceso según el tipo de convocatoria y los permisos de usuario.

Este botón, representado por el componente `<p:menuitem>`, está diseñado para redirigir al usuario a la página **granAreasConvocatorias.xhtml** cuando se selecciona. A continuación, se explica el código:

Explicación de los Atributos

1. Etiqueta del Botón:

- **value="#{msg['gestion.configuracion.convocatoria.granAreas.button']}"**
- Este atributo **value** asigna el texto que se muestra en el botón. El texto se obtiene de un archivo de mensajes (por ejemplo, un archivo de propiedades) mediante la clave **gestion.configuracion.convocatoria.granAreas.button**, permitiendo soporte para múltiples idiomas.

2. Ícono:

- **icon="fa fa-cubes"**
- Asigna un icono al botón utilizando la clase **fa fa-cubes** de **FontAwesome**. Esto agrega un ícono de cubos al botón, visualmente representativo de las "gran áreas".

3. Redirección a una Página:

- **outcome="/app/configuracion/convocatorias/granAreasConvocatorias.xhtml"**
- Este atributo **outcome** define la página de destino a la que se redirige el usuario al hacer clic en el botón, en este caso, a **granAreasConvocatorias.xhtml**, que contiene la configuración de las grandes áreas de la convocatoria.

4. Condición para Renderizar el Botón:

- **rendered="#{convocatoriaSessController.selected.convEstimulos eq 'N' and (convocatoriaSessController.selected.convocatoriaPatente == null or convocatoriaSessController.selected.convocatoriaPatente eq 'N') and SeguridadSessController.mapaMenu.containsKey('GEST1330')}"**
- Este atributo **rendered** establece las condiciones bajo las cuales el botón será visible para el usuario. La condición evalúa tres criterios:
 - **convEstimulos:** El valor de **convocatoriaSessController.selected.convEstimulos** debe ser 'N', indicando que la convocatoria no está configurada para estímulos.
 - **convocatoriaPatente:** La convocatoria no debe estar asociada a una patente. Esto se verifica comprobando que **convocatoriaSessController.selected.convocatoriaPatente** sea null o tenga el valor 'N'.
 - **Permiso de Usuario:** El usuario debe tener acceso al menú identificado por la clave 'GEST1330' en el mapaMenu del controlador de sesión de seguridad (**SeguridadSessController**). Esto permite que solo los usuarios autorizados puedan ver el botón.
- Creación Controlador de Vista **ConvocatoriaGranAreaViewController.java**, encargado de mostrar la información (listado de registros de Grandes Áreas) de convocatorias.

```
@Named(value = "convocatoriaGranAreaViewController")
@ViewScoped
public class ConvocatoriaGranAreaViewController extends
GenericSessionController<ConvocatoriaGranAreaDTO> implements Serializable {

    @EJB
    private ConvocatoriaServiceFacade convocatoriaServiceFacade;
    @Inject
    private ColcienciasSessController colcienciasSessController;
    @Inject
    private ConfigApplController configApplController;
```

```

private LazyDataModel<ConvocatoriaGranAreaDTO> model;
private ConvocatoriaDTO convocatoriaSelected;
private Map<String, ValorDefinicionDTO> listaGranAreas;

private static final org.apache.log4j.Logger LOG =
org.apache.log4j.Logger.getLogger(ConvocatoriaGranAreaViewController.class);

public ConvocatoriaGranAreaViewController() {
    super(ConvocatoriaGranAreaDTO.class);
    convocatoriaSelected = new ConvocatoriaDTO();
}

@PostConstruct
public void cargarListas() {

    Map<String, Object> mapReferencias = (Map<String, Object>)
colcienciasSessController.getUtilObject();
    convocatoriaSelected = (ConvocatoriaDTO) mapReferencias.get("convocatoriaSelected");

    listaGranAreas = configApplController.getMapValorDefinicion().get("GRANAROCDE");

    this.model = new LazyDataModel<ConvocatoriaGranAreaDTO>() {

        @Override
        public List<ConvocatoriaGranAreaDTO> load(int first, int pageSize, String
sortField, SortOrder sortOrder, Map<String, FilterMeta> filterBy) {
            Map<String, Object> filters = new TreeMap<>();
            filterBy.forEach((k, v) -> {
                if (k.equals("globalFilter")) {
                    if (v.getFilterValue() != null &&
!v.getFilterValue().toString().isEmpty()) {
                        filters.put(k, v.getFilterValue());
                    }
                }
            });

            try {
                //Filtro por Convocatoria ID
                filters.put("convocatoriaId", convocatoriaSelected.getConvocatoriaId());

                int count =
convocatoriaServiceFacade.convocatoriaGranAreasListCount(filters);
                if (count > 0) {
                    list = convocatoriaServiceFacade.convocatoriaGranAreasList(first,
pageSize, sortField, sortOrder.equals(SortOrder.ASCENDING) ? "A" : "D", filters);
                }
                model.setRowCount(count);
            } catch (Exception ex) {
                LOG.error("Error al cargar el listado de gran areas por convocatoria ",
ex);
            }
            return list;
        }

        @Override
        public Object getRowKey(ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO) {
            return convocatoriaGranAreaDTO.getConvocatoriaGranAreaId();
        }

        @Override

```

```

        public ConvocatoriaGranAreaDTO getRowData(String rowKey) {

            try {
                //Se consulta en la base de datos para mantener independiente el objeto
                //que trae en la lista principal y no lo actualice por referencia
                ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO =
                convocatoriaServiceFacade.queryConvocatoriaGranArea(Long.parseLong(rowKey));
                return convocatoriaGranAreaDTO;
            } catch (Exception ex) {
                LOG.error("no se pudo cargar el gran area ( id => " + rowKey + ")", ex);
            }
            return null;
        }
    };
}

@Override
public LazyDataModel<ConvocatoriaGranAreaDTO> getModel() {
    return model;
}

@Override
public void setModel(LazyDataModel<ConvocatoriaGranAreaDTO> model) {
    this.model = model;
}

public ConfigApplController getConfigApplController() {
    return configApplController;
}

public void setConfigApplController(ConfigApplController configApplController) {
    this.configApplController = configApplController;
}

public ConvocatoriaDTO getConvocatoriaSelected() {
    return convocatoriaSelected;
}

public void setConvocatoriaSelected(ConvocatoriaDTO convocatoriaSelected) {
    this.convocatoriaSelected = convocatoriaSelected;
}

public Map<String, ValorDefinicionDTO> getListaGranAreas() {
    return listaGranAreas;
}

public void setListaGranAreas(Map<String, ValorDefinicionDTO> listaGranAreas) {
    this.listaGranAreas = listaGranAreas;
}
}

```

Clase ConvocatoriaGranAreaViewController

Esta clase controla la vista de la pantalla de "Gran Áreas" dentro de una convocatoria en la aplicación. Está diseñada para gestionar la lógica de negocio relacionada con las grandes áreas que

corresponden a una convocatoria específica, incluyendo la carga de datos de forma paginada y la recuperación de listas de grandes áreas disponibles.

Anotaciones y Alcance

- **@Named:** Define el nombre de referencia de este controlador para que pueda ser utilizado en vistas **JSF**.
- **@ViewScoped:** Define el alcance de la vista, permitiendo que los datos y el estado del controlador persistan mientras el usuario esté en la misma página.

Atributos Principales

1. **model:** Es un objeto **LazyDataModel<ConvocatoriaGranAreaDTO>**, que implementa la carga de datos de manera perezosa (**lazy loading**). Esto optimiza el rendimiento, ya que permite cargar solo los registros necesarios por página en la vista.
2. **convocatoriaSelected:** Almacena la convocatoria seleccionada en la sesión, permitiendo que los datos se carguen basados en esta convocatoria específica.
3. **listaGranAreas:** Un mapa que contiene las áreas temáticas (grandes áreas) disponibles para selección en la convocatoria, cargado desde **configApplController**.

Método cargarListas()

Este método es invocado automáticamente después de la construcción de la instancia (anotado con **@PostConstruct**). Realiza las siguientes acciones:

- **Carga la convocatoria seleccionada** a partir del controlador de sesión **colcienciasSessController**.
- **Carga las grandes áreas** desde **configApplController** para poblar el mapa **listaGranAreas**.
- **Inicializa el modelo perezoso (LazyDataModel)** con una función **load()** que se encarga de realizar la consulta paginada.

Modelo Perezoso (LazyDataModel)

El modelo perezoso es esencial para manejar grandes volúmenes de datos de manera eficiente. Dentro del **load()** del **LazyDataModel**:

- **Filtros:** Se agrega un filtro específico convocatoriald basado en la convocatoria seleccionada (**convocatoriaSelected**).
- **Carga de datos:** Usa **convocatoriaServiceFacade** para obtener la lista de **ConvocatoriaGranAreaDTO** según los parámetros de paginación (**first**, **pageSize**, etc.).
- **Manejo de errores:** Utiliza LOG.error para capturar cualquier excepción durante la carga de datos.

Métodos Sobrescritos

1. **getRowKey():** Devuelve la clave de fila para un objeto **ConvocatoriaGranAreaDTO**, que se utiliza para identificar cada registro en el modelo.
2. **getRowData():** Recupera datos de una fila específica a partir de su clave (**rowKey**), permitiendo cargar el registro desde la base de datos si es necesario.

Métodos Getter y Setter

Los métodos getter y setter proporcionan acceso a los atributos model, configApplController, convocatoriaSelected y listaGranAreas para ser utilizados en la vista.

Dependencias Externas

- **ConvocatoriaServiceFacade:** Componente EJB que provee acceso a la capa de servicios para realizar las consultas necesarias.
- **ColcienciasSessController y ConfigApplController:** Controladores de sesión inyectados, usados para obtener datos contextuales de la aplicación.
- Creación Controlador de Conversación **ConvocatoriaGranAreaConvController.java**, el cual permite la comunicación e interacción de la pantalla con el usuario y funcionalidades de **CRUD** (Create, Read, Update y Delete).

```
@Named(value = "convocatoriaGranAreaConvController")
@ConversationScoped
public class ConvocatoriaGranAreaConvController extends
GenericConversationController<ConvocatoriaGranAreaDTO> implements Serializable {

    @EJB
    private ConvocatoriaServiceFacade convocatoriaServiceFacade;
    @Inject
```

```
private ConvocatoriaGranAreaViewController convocatoriaGranAreaViewController;
@Inject
private SeguridadSessController seguridadSessController;
@Inject
private ColcienciasSessController colcienciasSessController;

private final String programa = "GEST1330";

private static final org.apache.log4j.Logger LOG =
org.apache.log4j.Logger.getLogger(ConvocatoriaGranAreaConvController.class);

public ConvocatoriaGranAreaConvController() {
    this.selected = new ConvocatoriaGranAreaDTO();
}

@Override
public void guardar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO =
convocatoriaGranAreaViewController.getSelected();

convocatoriaGranAreaDTO.setUsuarioCreador(seguridadSessController.getSelected().getUsuario());

convocatoriaGranAreaDTO.setFechaCreacion(new Date());

convocatoriaGranAreaDTO.setConvocatoriaId(convocatoriaGranAreaViewController.getConvocatoria
Selected());
//Auditoria
AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
appSessionVO.setPrograma(programa);
convocatoriaGranAreaDTO.setAppSessionVO(appSessionVO);
convocatoriaServiceFacade.guardarConvocatoriaGranArea(convocatoriaGranAreaDTO);

convocatoriaGranAreaViewController.setSelected(new ConvocatoriaGranAreaDTO());
convocatoriaGranAreaViewController.cargarListas();
//se debilita el boton de editar / eliminar ya que no habra nada seleccionado
despues de la operación
this.onRowUnselect(null);
this.obtenerPestana = 0;
} catch (SystemException ex) {
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
    LOG.error("Error al guardar la gran area por convocatoria ", ex);
} catch (Exception ex) {
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
    LOG.error("Error al guardar la gran area por convocatoria ", ex);
} finally {
    FacesContext.getCurrentInstance().addMessage(null, message);
}
}

@Override
public void eliminar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
}
```

```

        try {
            ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO =
convocatoriaGranAreaViewController.getSelected();
            //Auditoria
            AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
            appSessionVO.setProgram(programa);
            convocatoriaGranAreaDTO.setAppSessionVO(appSessionVO);
            convocatoriaServiceFacade.eliminarConvocatoriaGranArea(convocatoriaGranAreaDTO);

            convocatoriaGranAreaViewController.setSelected(new ConvocatoriaGranAreaDTO());
            convocatoriaGranAreaViewController.cargarListas();
            this.onRowUnselect(null);
        } catch (Exception ex) {
            message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
            LOG.error("Error al eliminar la gran area de la Convocatoria ", ex);
        } finally {
            FacesContext.getCurrentInstance().addMessage(null, message);
        }
    }

    @Override
    public void editar(ActionEvent ae) {
        FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
            resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
            resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));

        try {
            ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO =
convocatoriaGranAreaViewController.getSelected();

            convocatoriaGranAreaDTO.setUsuarioModificador(seguridadSessController.getSelected().getUsuar
io());

            convocatoriaGranAreaDTO.setFechaModificacion(new Date());

            convocatoriaGranAreaDTO.setConvocatoriaId(convocatoriaGranAreaViewController.getConvocatoria
Selected());
            //Auditoria
            AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
            appSessionVO.setProgram(programa);
            convocatoriaGranAreaDTO.setAppSessionVO(appSessionVO);
            convocatoriaServiceFacade.editarConvocatoriaGranArea(convocatoriaGranAreaDTO);

            convocatoriaGranAreaViewController.setSelected(new ConvocatoriaGranAreaDTO());
            convocatoriaGranAreaViewController.cargarListas();
            //se deshabilita el boton de editar / eliminar ya que no habra nada seleccionado
despues de la operación
            this.onRowUnselect(null);
            this.obtenerPestana = 0;
        } catch (SystemException ex) {
            message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
            LOG.error("Error al editar la gran area por convocatoria ", ex);
        } catch (Exception ex) {
            message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
            LOG.error("Error al editar la gran area de la Convocatoria ", ex);
        } finally {
            FacesContext.getCurrentInstance().addMessage(null, message);
        }
    }
}

```



```
@Override
public void buscar(ActionEvent ae) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
}

@Override
public void mostrar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    //Obtengo la tabla+
    String modulo = convocatoriaGranAreaViewController.getModulo();
    DataTable dataTable = (DataTable)
FacesContext.getCurrentInstance().getViewRoot().findComponent(modulo + "Form:" + modulo +
"TabList:" + modulo + "Table");
    //limpio los filtros
    dataTable.setFilterBy(null);
    //Se actualiza el formulario que contiene la tabla
    PrimeFaces.current().ajax().update(modulo + "Form");
    //Se muestra el growl con el mensaje
    FacesContext.getCurrentInstance().addMessage(null, message);

    convocatoriaGranAreaViewController.cargarListas();
    this.onRowUnselect(null);
}

@Override
public void goNuevo(ActionEvent ae) {
    convocatoriaGranAreaViewController.setSelected(new ConvocatoriaGranAreaDTO());
    convocatoriaGranAreaViewController.modosGuardar();
    this.obtenerPestana = 1;
}

@Override
public void goEditar(ActionEvent ae) {
    convocatoriaGranAreaViewController.modosEditar();
    this.obtenerPestana = 1;
}

@Override
public void goBuscar(int i) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
}

@Override
public void goEliminar() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
}

@Override
public void onRowSelect(SelectEvent event) {
    super.onRowSelect(event);
    convocatoriaGranAreaViewController.setSelected(this.selected);
    convocatoriaGranAreaViewController.setActivaEditar(true);
    convocatoriaGranAreaViewController.modosEditar();
}
```

```
@Override
public void onRowUnselect(UnselectEvent event) {
    super.onRowUnselect(event);
    convocatoriaGranAreaViewController.setSelected(new ConvocatoriaGranAreaDTO());
    convocatoriaGranAreaViewController.setActivaEditar(false);
    convocatoriaGranAreaViewController.modaGuardar();
    selected = null;
}

public void volverConvocatorias() {
    Map<String, Object> mapReferencias = new HashMap<>();
    colcienciasSessController.setUtilObject(mapReferencias);

    this.redirectPage("app/configuracion/convocatorias/cortesConvocatorias.xhtml");
}
}
```

Este controlador **ConvocatoriaGranAreaConvController** permite gestionar la creación, edición y eliminación de "gran áreas" asociadas a una convocatoria, aplicando transacciones que incluyen la auditoría y la interacción con el controlador de vista. Aquí hay un desglose de las partes clave del controlador y su funcionalidad:

1. Atributos y Dependencias

- **convocatoriaServiceFacade:** Gestiona las operaciones de acceso a la base de datos para las "gran áreas" asociadas a convocatorias.
- **convocatoriaGranAreaViewController:** Controlador de vista para interactuar con el modelo y la interfaz.
- **seguridadSessController** y **colcienciasSessController:** Manejan aspectos de sesión y seguridad, incluyendo el usuario actual y la auditoría de operaciones.
- **programa:** Un identificador constante para el programa "**GEST1330**", probablemente usado para auditoría o segmentación.

2. Métodos de Gestión de CRUD

- **guardar(ActionEvent ae)**

Este método guarda una nueva "gran área" de la convocatoria, y realiza las siguientes operaciones:

- **Asigna datos de auditoría:** Incluye el usuario creador y la fecha de creación.
- **Auditoría extendida:** Usa el objeto **AppSessionVO** para definir detalles del programa.

- Invoca el servicio de guardado con `convocatoriaServiceFacade.guardarConvocatoriaGranArea`.
- Limpia el formulario: Reinicia el elemento seleccionado en `convocatoriaGranAreaViewController` y deshabilita el botón de edición.
- **eliminar(ActionEvent ae)**

Permite eliminar una "gran área" seleccionada de la convocatoria:

- **Asigna datos de auditoría** al elemento antes de llamar al servicio de eliminación.
- **Actualiza la vista:** Reinicia la selección y actualiza la lista cargada en la vista.
- **editar(ActionEvent ae)**

Edita la información de una "gran área" existente:

- **Actualización de datos de auditoría:** Incluye usuario modificador y fecha de modificación.
- **Guarda los cambios** a través de `convocatoriaServiceFacade.editarConvocatoriaGranArea`.
- **Actualiza la vista** limpiando el formulario y deshabilitando el botón de edición después de guardar.

3. Métodos de Navegación y Actualización de Interfaz

- **mostrar(ActionEvent ae)**

Este método restaura el estado de la tabla:

- **Accede y limpia la tabla de datos** en la interfaz.
- **Actualiza la tabla** y la lista cargada en `convocatoriaGranAreaViewController`.
- **goNuevo(ActionEvent ae) y goEditar(ActionEvent ae)**

Ambos métodos preparan la vista para crear o editar una "gran área":

- **Inicialización:** **goNuevo** limpia la selección, mientras que **goEditar** habilita el modo de edición.
- **Configuración de la pestaña activa** (`obtenerPestana`) para controlar la interfaz de usuario.

4. Métodos de Selección y Deselección de Filas

- **onRowSelect(SelectEvent event) y onRowUnselect(UnselectEvent event)**

Estos métodos manejan la selección y deselección de elementos en la tabla:

- **Seleccionar:** Asigna el elemento seleccionado a **convocatoriaGranAreaViewController** y habilita el modo de edición.
- **Deseleccionar:** Limpia la selección y restablece el formulario en modo de guardado.

5. Método de Redirección

- **volverConvocatorias()**

Este método redirige a otra pantalla de configuración de convocatorias:

- **Redirecciona** a la página **cortesConvocatorias.xhtml**.

6. Manejo de Excepciones y Mensajes

En cada operación **CRUD**, se maneja cualquier error con un **FacesMessage** para notificar a la interfaz. Esto asegura que los usuarios reciban retroalimentación adecuada, ya sea en éxito o en caso de error.

- Se definen métodos de crud en Facade EJB para funcionalidad de pantalla

```
//  
//  
//GRAN AREAS POR CONVOCATORIA  
//  
/**  
 * Consulta el listado de GranAreas por convocatoria registrados en la base  
 * de datos  
 *  
 * @param first indica desde donde empieza la consulta  
 * @param pageSize indica el final de la consulta (first + pageSize)  
 * @param sortField indica el campo por el que se realiza el ordenamiento  
 * @param sortOrder indica el orden (ascendente o descendente)  
 * @param filters contiene los filtros al momento de solicitar el listado de  
 * GranAreas por convocatoria  
 * @return lista de GranAreas por convocatoria encontrados en el rango  
 * enviado y basado en los filtros enviados  
 * @throws Exception  
 */  
List<ConvocatoriaGranAreaDTO> convocatoriaGranAreasList(int first, int pageSize, String  
sortField, String sortOrder, Map<String, Object> filters) throws Exception;
```

```
/**
 * Cuenta la cantidad de GranAreas por convocatoria actualmente registrados
 * en la base de datos
 *
 * @param filters
 * @return
 * @throws Exception
 */
int convocatoriaGranAreasListCount(Map<String, Object> filters) throws Exception;

/**
 * Guarda el registro de GranArea por convocatoria en la base de datos
 *
 * @param convocatoriaGranAreaDTO
 * @throws Exception
 */
void guardarConvocatoriaGranArea(ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO) throws
Exception;

/**
 * Edita el registro de GranArea por convocatoria en la base de datos
 *
 * @param convocatoriaGranAreaDTO
 * @throws Exception
 */
void editarConvocatoriaGranArea(ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO) throws
Exception;

/**
 * Elimina el registro de GranArea por convocatoria en la base de datos
 *
 * @param convocatoriaGranAreaDTO
 * @throws Exception
 */
void eliminarConvocatoriaGranArea(ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO) throws
Exception;

/**
 * Buscar el registro de una GranArea por convocatoria y lo retorna basado
 * en el id (llave primaria)
 *
 * @param convocatoriaGranAreaId
 * @return la GranArea por convocatoria que coincide con el id enviado
 * @throws Exception
 */
ConvocatoriaGranAreaDTO queryConvocatoriaGranArea(Long convocatoriaGranAreaId) throws
Exception;
```

- Se implementan métodos crud en facade EJB para funcionalidad de pantalla

```
//----- Configuración Gran Areas por Convocatoria -----
-----//
@Override
public List<ConvocatoriaGranAreaDTO> convocatoriaGranAreasList(int first, int pageSize,
String sortField, String sortOrder, Map<String, Object> filters) throws Exception {
```

```

        ObjectConverter<ConvocatoriaGranAreaDTO,      ConvocatoriaGranArea> converter =
objectConverterFactory.checkOut();
        List<ConvocatoriaGranAreaDTO> listConvocatoriaGranAreaDTOs =
converter.toTransferList(convocatoriaGranAreaService.convocatoriaGranAreasList(first,
pageSize, sortField, sortOrder, filters), ConvocatoriaGranAreaDTO.class);
        objectConverterFactory.checkIn(converter);
        return listConvocatoriaGranAreaDTOs;
    }

    @Override
    public int convocatoriaGranAreasListCount(Map<String, Object> filters) throws Exception {
        return convocatoriaGranAreaService.convocatoriaGranAreasListCount(filters);
    }

    @Override
    public void guardarConvocatoriaGranArea(ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO)
throws Exception {
        ObjectConverter<ConvocatoriaGranAreaDTO,      ConvocatoriaGranArea> converter =
objectConverterFactory.checkOut();
        convocatoriaGranAreaService.guardarConvocatoriaGranArea(converter.toEntity(convocatoria
GranAreaDTO, ConvocatoriaGranArea.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public void editarConvocatoriaGranArea(ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO)
throws Exception {
        ObjectConverter<ConvocatoriaGranAreaDTO,      ConvocatoriaGranArea> converter =
objectConverterFactory.checkOut();
        convocatoriaGranAreaService.editarConvocatoriaGranArea(converter.toEntity(convocatoriaG
ranAreaDTO, ConvocatoriaGranArea.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public void eliminarConvocatoriaGranArea(ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO)
throws Exception {
        ObjectConverter<ConvocatoriaGranAreaDTO,      ConvocatoriaGranArea> converter =
objectConverterFactory.checkOut();
        convocatoriaGranAreaService.eliminarConvocatoriaGranArea(converter.toEntity(convocatoria
aGranAreaDTO, ConvocatoriaGranArea.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public ConvocatoriaGranAreaDTO queryConvocatoriaGranArea(Long convocatoriaGranAreaId) throws
Exception {
        ObjectConverter<ConvocatoriaGranAreaDTO,      ConvocatoriaGranArea> converter =
objectConverterFactory.checkOut();
        ConvocatoriaGranAreaDTO convocatoriaGranAreaDTO =
converter.toTransfer(convocatoriaGranAreaService.queryConvocatoriaGranArea(convocatoriaGranA
reaId), ConvocatoriaGranAreaDTO.class);
        objectConverterFactory.checkIn(converter);
        return convocatoriaGranAreaDTO;
    }
}

```

- Se crea interfaz **ConvocatoriaGranAreaService.java** que define métodos de crud en Service EJB para funcionalidad de pantalla

```
@Local
public interface ConvocatoriaGranAreaService {

    List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters) throws SystemException;

    int convocatoriaGranAreasListCount(Map<String, Object> filters) throws SystemException;

    void guardarConvocatoriaGranArea(ConvocatoriaGranArea convocatoriaGranArea) throws
SystemException;

    void editarConvocatoriaGranArea(ConvocatoriaGranArea convocatoriaGranArea) throws
SystemException;

    void eliminarConvocatoriaGranArea(ConvocatoriaGranArea convocatoriaGranArea) throws
SystemException;

    ConvocatoriaGranArea queryConvocatoriaGranArea(Long convocatoriaGranAreaId) throws
SystemException;
}
```

- Se crea **ConvocatoriaGranAreaServiceBean.java** donde implementan métodos crud en Service EJB para funcionalidad de pantalla

```
@Stateless
public class ConvocatoriaGranAreaServiceBean extends StoreManagerImpl implements
ConvocatoriaGranAreaService {

    //----- Convocatoria Gran Area -----//
    @Override
    public List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize,
String sortField, String sortOrder, Map<String, Object> filters) throws SystemException {
        return
this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaGranAreasList(first,
pageSize, sortField, sortOrder, filters);
    }

    @Override
    public int convocatoriaGranAreasListCount(Map<String, Object> filters) throws
SystemException {
        return
this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaGranAreasListCount(filters)
;
    }

    @Override
    public void guardarConvocatoriaGranArea(ConvocatoriaGranArea convocatoriaGranArea) throws
SystemException {
        this.crear(convocatoriaGranArea, ConvocatoriaGranArea.class);
    }
}
```

```
@Override
public void editarConvocatoriaGranArea(ConvocatoriaGranArea convocatoriaGranArea) throws
SystemException {
    this.modificar(convocatoriaGranArea, ConvocatoriaGranArea.class);
}

@Override
public void eliminarConvocatoriaGranArea(ConvocatoriaGranArea convocatoriaGranArea)
throws SystemException {
    this.eliminar(convocatoriaGranArea, ConvocatoriaGranArea.class);
}

@Override
public ConvocatoriaGranArea queryConvocatoriaGranArea(Long convocatoriaGranAreaId) throws
SystemException {
    return this.buscar(convocatoriaGranAreaId, ConvocatoriaGranArea.class);
}
}
```

- Se crean métodos en **ConvocatoriaStoreManager.java** donde se definen métodos de consulta criteria para funcionalidad de pantalla

```
List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters);

int convocatoriaGranAreasListCount(Map<String, Object> filters);
```

- Se crean métodos en **ConvocatoriaStoreManagerImpl.java** donde implementan métodos de consulta en criteria para funcionalidad de pantalla

```
//----- GRAN AREAS POR CONVOCATORIA -----//
@Override
public List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters) {
    cb = em.getCriteriaBuilder();
    queryConvocatoriaGranArea = cb.createQuery(ConvocatoriaGranArea.class);
    convocatoriaGranAreaRoot = queryConvocatoriaGranArea.from(ConvocatoriaGranArea.class);
    queryConvocatoriaGranArea.select(convocatoriaGranAreaRoot);
    //Where
    queryConvocatoriaGranArea.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new
Predicate[]{})));
    //Order
    if (sortOrder.equals("A")) {
        queryConvocatoriaGranArea.orderBy(cb.asc(getConvocatoriaGranAreaPath(sortField)));
    } else {
        queryConvocatoriaGranArea.orderBy(cb.desc(getConvocatoriaGranAreaPath(sortField)));
    }
    //pagination
    TypedQuery<ConvocatoriaGranArea> tq = em.createQuery(queryConvocatoriaGranArea);
```



```

        if (pageSize >= 0) {
            tq.setMaxResults(pageSize);
        }
        if (first >= 0) {
            tq.setFirstResult(first);
        }
        return tq.getResultList();
    }

    @Override
    public int convocatoriaGranAreasListCount(Map<String, Object> filters) {
        CriteriaQuery<Long> qCount;
        cb = em.getCriteriaBuilder();
        qCount = cb.createQuery(Long.class);
        convocatoriaGranAreaRoot = qCount.from(ConvocatoriaGranArea.class);
        qCount.select(cb.count(convocatoriaGranAreaRoot));
        //Where
        qCount.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new Predicate[]{})));
        Long res = em.createQuery(qCount).getSingleResult();
        return res.intValue();
    }

    private List<Predicate> filtrosConvocatoriaGranArea(Map<String, Object> filters) {
        List<Predicate> lista = new ArrayList<>();
        List<String> camposFiltro = new ArrayList<>();
        Path pathFilter;
        camposFiltro.add("granAreaId.descripcionValorDefinicion");
        camposFiltro.add("convocatoriaId.descripcionConvocatoria");

        //Filtro por convocatoria ID
        Object convocatoriaId = filters.get("convocatoriaId");
        if (convocatoriaId != null) {

            lista.add(cb.equal(getConvocatoriaGranAreaPath("convocatoriaId.convocatoriaId"), (Long)
convocatoriaId));
        }

        //Filtro Global
        globalFilter(filters, camposFiltro, "getConvocatoriaGranAreaPath", lista);

        return lista;
    }

    private Path<?> getConvocatoriaGranAreaPath(String field) {
        //Joins
        Join<ConvocatoriaGranArea, Convocatoria> convocatoriaId
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.convocatoriaId, JoinType.LEFT);
        Join<ConvocatoriaGranArea, ValorDefinicion> granAreaId
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.granAreaId, JoinType.LEFT);

        Path<?> path = null;
        if (null == field) {
            path
convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.convocatoriaGranAreaId);
        } else {
            switch (field) {
                case "convocatoriaGranAreaId":
                    path
convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.convocatoriaGranAreaId);
                    break;
                case "usuarioCreador":

```

```

        path = convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.usuarioCreador);
        break;
        case "fechaCreacion":
        path = convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.fechaCreacion);
        break;
        case "convocatoriaId.convocatoriaId":
        path = convocatoriaId.get(Convocatoria_.convocatoriaId);
        break;
        case "convocatoriaId.descripcionConvocatoria":
        path = convocatoriaId.get(Convocatoria_.descripcionConvocatoria);
        break;
        case "granAreaId.valorDefinicionId":
        path = granAreaId.get(ValorDefinicion_.valorDefinicionId);
        break;
        case "granAreaId.valorDefinicion":
        path = granAreaId.get(ValorDefinicion_.valorDefinicion);
        break;
        case "granAreaId.descripcionValorDefinicion":
        path = granAreaId.get(ValorDefinicion_.descripcionValorDefinicion);
        break;
        default:
        break;
    }
    return path;
}

```

A continuación, se explica cada método relacionado con Criterias API que se usó para el módulo **ConvocatoriaGranArea**. Se revisa en detalle las implementaciones de **convocatoriaGranAreasList**, **convocatoriaGranAreasListCount**, **filtrosConvocatoriaGranArea**, y **getConvocatoriaGranAreaPath**.

1. Método convocatoriaGranAreasList

Este método obtiene una lista de entidades **ConvocatoriaGranArea** con filtros, ordenamiento, y paginación. Veamos cada paso:

- **Inicialización de CriteriaBuilder y CriteriaQuery:**

```

cb = em.getCriteriaBuilder();
queryConvocatoriaGranArea = cb.createQuery(ConvocatoriaGranArea.class);

```

CriteriaBuilder (cb) se utiliza para construir la consulta, proporcionando operaciones de criterios. **CriteriaQuery (queryConvocatoriaGranArea)** es la consulta específica para **ConvocatoriaGranArea**.

- **Definición de la raíz de la consulta:**

```
convocatoriaGranAreaRoot = queryConvocatoriaGranArea.from(ConvocatoriaGranArea.class);
queryConvocatoriaGranArea.select(convocatoriaGranAreaRoot);
```

convocatoriaGranAreaRoot define el punto de inicio para los atributos de la entidad **ConvocatoriaGranArea**, similar a una tabla en SQL.

- **Aplicación de filtros:**

```
queryConvocatoriaGranArea.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new
Predicate[]{})));
```

Aquí se llaman los predicados de **filtrosConvocatoriaGranArea** y se aplican con **.where**. Los filtros son dinámicos, adaptándose a las condiciones proporcionadas en **filters**.

- **Ordenamiento:**

```
if (sortOrder.equals("A")) {
    queryConvocatoriaGranArea.orderBy(cb.asc(getConvocatoriaGranAreaPath(sortField)));
} else {
    queryConvocatoriaGranArea.orderBy(cb.desc(getConvocatoriaGranAreaPath(sortField)));
}
```

Dependiendo del valor de **sortOrder**, la consulta se ordena de forma ascendente (**asc**) o descendente (**desc**). El método **getConvocatoriaGranAreaPath** se usa para obtener la ruta específica del campo de ordenamiento, adaptándose al campo seleccionado.

- **Paginación:**

```
TypedQuery<ConvocatoriaGranArea> tq = em.createQuery(queryConvocatoriaGranArea);
if (pageSize >= 0) {
    tq.setMaxResults(pageSize);
}
if (first >= 0) {
    tq.setFirstResult(first);
}
```

pageSize y **first** controlan la cantidad de registros devueltos y el índice inicial. Esto es útil para manejar grandes cantidades de datos al cargar solo una parte de los registros en cada llamada.

2. Método **convocatoriaGranAreasListCount**

Este método calcula el número total de registros que cumplen con los filtros, útil para la paginación en el front-end:

- **Configuración de **CriteriaQuery** para contar:**

```
qCount = cb.createQuery(Long.class);  
convocatoriaGranAreaRoot = qCount.from(ConvocatoriaGranArea.class);  
qCount.select(cb.count(convocatoriaGranAreaRoot));
```

Aquí, **CriteriaQuery<Long>** se configura para realizar una operación de conteo (**count**). **cb.count** cuenta el número de elementos en la raíz **convocatoriaGranAreaRoot**.

- **Aplicación de filtros:**

```
qCount.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new Predicate[]{})));
```

Se aplican los mismos filtros que en **convocatoriaGranAreasList**, lo cual asegura que el conteo coincida con los registros obtenidos en la lista.

3. Método **filtrosConvocatoriaGranArea**

Este método construye la lista de **Predicates** (condiciones) basados en los filtros proporcionados:

- **Lista de campos de filtro:**

```
List<String> camposFiltro = new ArrayList<>();  
camposFiltro.add("granAreaId.descripcionValorDefinicion");  
camposFiltro.add("convocatoriaId.descripcionConvocatoria");
```

camposFiltro contiene los nombres de los campos sobre los que se aplicará un filtro global.

- **Filtro específico para convocatoriaId:**

```
Object convocatoriaId = filters.get("convocatoriaId");  
if (convocatoriaId != null) {
```

```
lista.add(cb.equal(getConvocatoriaGranAreaPath("convocatoriaId.convocatoriaId"), (Long)
convocatoriaId));
}
```

Si el filtro **convocatoriaId** está presente, se agrega una condición equal para buscar coincidencias exactas en **convocatoriaId**.

- **Filtro global:**

```
globalFilter(filters, camposFiltro, "getConvocatoriaGranAreaPath", lista);
```

globalFilter aplica un filtro más amplio, permitiendo buscar coincidencias en varios campos al mismo tiempo (como una búsqueda de texto completo).

4. Método **getConvocatoriaGranAreaPath**

Este método es crucial para construir las rutas de los atributos en Criterios API para diferentes campos. Esto permite crear una lógica flexible para orden y filtrado en función del campo solicitado.

- **Join con otras entidades:**

```
Join<ConvocatoriaGranArea, Convocatoria> convocatoriaId =
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.convocatoriaId, JoinType.LEFT);
Join<ConvocatoriaGranArea, ValorDefinicion> granAreaId =
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.granAreaId, JoinType.LEFT);
```

Aquí se establecen joins con las entidades **Convocatoria** y **ValorDefinicion** para acceder a sus campos. El **JoinType.LEFT** asegura que se incluyan registros de **ConvocatoriaGranArea** que puedan no tener coincidencias en estas tablas relacionadas.

- **Rutas de campos específicos:**

```
switch (field) {
    case "convocatoriaGranAreaId":
        path = convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.convocatoriaGranAreaId);
        break;
    // Otros casos...
}
```

Para cada campo solicitado (**field**), el método devuelve la ruta adecuada. Esto permite que los filtros y el ordenamiento sean dinámicos y adaptables a los campos solicitados.

- Internacionalización en español e inglés

```
#ES
gestion.configuracion.convocatoria.granAreas.button           = Gran Áreas
gestion.configuracion.convocatoria.granAreas.header           = Gran Áreas de la
Convocatoria
gestion.configuracion.convocatoria.granAreas.granArea.header   = Gran Área
gestion.configuracion.convocatoria.granAreas.usuarioCreador.header = Usuario Creador
gestion.configuracion.convocatoria.granAreas.fechaCreacion.header = Fecha Creación

gestion.configuracion.convocatoria.granAreas.granArea.label     = Gran Área
gestion.configuracion.convocatoria.granAreas.granArea.required   = Debe seleccionar la gran
área
com.sb.persistence.constraint.ORA-00001.SGPCONVGARE_UK           = Ya se ha configurado el
gran área seleccionada para esta convocatoria
com.sb.persistence.constraint.ORA-02292.SGPCONVGARE_ADMVALDEF_FK = La gran área no puede
ser eliminada porque se encuentra asociada a una convocatoria

#EN
gestion.configuracion.convocatoria.granAreas.button           = Large Areas
gestion.configuracion.convocatoria.granAreas.header           = Large Areas of the Call
gestion.configuracion.convocatoria.granAreas.granArea.header   = Large Area
gestion.configuracion.convocatoria.granAreas.usuarioCreador.header = User Creator
gestion.configuracion.convocatoria.granAreas.fechaCreacion.header = Creation Date

gestion.configuracion.convocatoria.granAreas.granArea.label     = Large Area
gestion.configuracion.convocatoria.granAreas.granArea.required   = You must select the large
area
com.sb.persistence.constraint.ORA-00001.SGPCONVGARE_UK           = The large area selected
for this call has already been configured
com.sb.persistence.constraint.ORA-02292.SGPCONVGARE_ADMVALDEF_FK = The large area cannot be
deleted because it is associated with a call
```

- Se crea tabla de visualización de la información en la pantalla de Gran áreas

```
<!--PESTAÑA TABLE -->
<p:tab title="#{msg['gestion.configuracion.convocatoria.granAreas.header']}">
  <!-- TABLA -->
  <div class="row">
    <div class="col-lg-12" style="overflow-x: scroll;">
      <p:dataTable id="#{convocatoriaGranAreaViewController.modulo}Table"
        var = "convGranArea"

value="#{convocatoriaGranAreaViewController.model}"

rowKey="#{convGranArea.convocatoriaGranAreaId}"

selection="#{convocatoriaGranAreaConvController.selected}"
        selectionMode="single"
        paginator="true"
        rows="10"
        lazy="true"
```

```

reflow="true"

sortBy="#{convGranArea.convocatoriaGranAreaId}"
sortOrder="ASCENDING"
currentPageReportTemplate="Total de registros:
{totalRecords}"
paginatorTemplate="{CurrentPageReport}
{FirstPageLink} {PreviousPageLink} {PageLinks} {NextPageLink} {LastPageLink}
{RowsPerPageDropdown}"
paginatorPosition="bottom"

emptyMessage="#{msg['sb.datatable.registro.empty']}"

widgetVar="items#{convocatoriaGranAreaViewController.modulo}"
<f:facet name="header">
    <ui:decorate
template="../../../../template/_tableHeader.xhtml">
    <ui:define name="comandos">
        <ui:param name="sessionController"
value="#{convocatoriaGranAreaViewController}" />
        <ui:param name="conversationController"
value="#{convocatoriaGranAreaConvController}" />
        <ui:param name="btnBusqueda" value="N"
/>
        <ui:param name="buscarVisible"
value="S" />
    </ui:define>
    </ui:decorate>
</f:facet>
<p:ajax event="rowSelect"

listener="#{convocatoriaGranAreaConvController.onRowSelect}"

update="#{convocatoriaGranAreaViewController.modulo}Table"
process="@this"/>
<p:ajax event="rowUnselect"

listener="#{convocatoriaGranAreaConvController.onRowUnselect}"

update="#{convocatoriaGranAreaViewController.modulo}Table"
process="@this"/>
<p:column
headerText="#{msg['gestion.configuracion.convocatoria.granAreas.granArea.header']}"
filterBy="#{convGranArea.granAreaId.descripcionValorDefinicion}"
filterMatchMode="contains"
filterStyle="display:none;"
>
    <h:outputText
value="#{convGranArea.granAreaId.descripcionValorDefinicion}"/>
</p:column>
<p:column
headerText="#{msg['gestion.configuracion.convocatoria.granAreas.fechaCreacion.header']}"
filterBy="#{convGranArea.fechaCreacion}"
filterMatchMode="contains"
filterStyle="display:none;"
width="20%"
style="text-align: center"
>
    <h:outputText value="#{convGranArea.fechaCreacion}">

```

```

timeZone="EST"/>
                                <f:convertDateTime pattern="dd/MM/yyyy hh:mm a"
                                </h:outputText>
                                </p:column>
                                </p:dataTable>
                                </div>
                                </div>
                                </p:tab>

```

En la tabla de registros de **ConvocatoriaGranArea**, los datos se muestran con el componente **<p:dataTable>** de **PrimeFaces**, que incluye características clave para manejar grandes conjuntos de datos de manera eficiente. A continuación, se muestra un desglose de los atributos y funcionalidades clave utilizados en la tabla:

1. Carga Diferida (Lazy Loading) (lazy="true"):

- Al establecer **lazy** en **true**, los datos se cargan bajo demanda, una página a la vez. Esto evita cargar todos los registros de una sola vez, mejorando el rendimiento cuando hay una gran cantidad de datos.

2. Paginación (paginator="true" y rows="10"):

- La paginación está habilitada, permitiendo que los usuarios vean 10 registros por página de manera predeterminada. El paginador está posicionado en la parte inferior, y la plantilla de paginador (**paginatorTemplate**) define los controles y el diseño para navegar entre páginas.

3. Ordenación y Filtro:

- La tabla se ordena inicialmente por **convocatoriaGranAreald** en orden ascendente.
- Cada **<p:column>** tiene **filterBy** y **filterMatchMode**, lo que permite que los datos de la columna se filtren por criterios específicos (como **"contains"** para coincidencias parciales). Actualmente, los filtros están ocultos (**filterStyle="display:none;"**), pero pueden mostrarse para que los usuarios los usen si es necesario.

4. Selección de Filas (selectionMode="single"):

- Está habilitada la selección de una sola fila, con eventos **rowSelect** y **rowUnselect**. Estos eventos disparan acciones específicas en el controlador (**onRowSelect** y **onRowUnselect**), lo que permite comportamientos interactivos cuando los usuarios seleccionan o deselectan filas.

5. Formato de Fecha:

- La columna **fechaCreacion** muestra las fechas con un patrón específico (**dd/MM/yyyy hh:mm a**) y usa una zona horaria de "EST", mejorando la legibilidad y consistencia para los usuarios que visualizan los tiempos de creación.
- Se crea formulario de captura de datos

```
<!--PESTAÑA GUARDAR/EDITAR -->
<p:tab title="#{msg['sb.component.tabview.guardarEditar']}">
  <p:panel id="formularioEditarCrear">
    <div class="ibox-content">
      <div class="form-horizontal">
        <div class="row">
          <div class="col-lg-12">
            <div class="form-group">
              <p:outputLabel
value="#{msg['gestion.configuracion.convocatoria.granAreas.granArea.label']}"
styleClass="col-lg-2 control-label"
for="vadef-
#{msg['sb.convert.selectOne.tipoGranAreas.ID']}"
style="text-
align: right"/>
              <div class="col-lg-9">
                <p:selectOneMenu id="vadef-
#{msg['sb.convert.selectOne.tipoGranAreas.ID']}"
value="#{convocatoriaGranAreaViewController.selected.granAreaId}"
required="true"
requiredMessage="#{msg['gestion.configuracion.convocatoria.granAreas.granArea.required']}"
converter="valorDefinicionConverter"
style="width:100%;"
autoWidth="false"
panelStyle="width:180px"
effect="fade"
filter="true"
filterMatchMode="contains"
>
                <f:selectItem
itemLabel="#{msg['sb.selectOneMenu.registro']}" />
                <f:selectItems
value="#{convocatoriaGranAreaViewController.listaGranAreas}"
```

```

var="granArea"

itemLabel="#{granArea.descripcionValorDefinicion}"

itemValue="#{granArea}"

/>

</p:selectOneMenu>
</div>
</div>
</div>
</div>
<br/>
<div class="hr-line-dashed"></div>
<ui:decorate template="../../template/_formButtons.xhtml">
  <ui:define name="comandos">
    <ui:param name="sessionController"
value="#{convocatoriaGranAreaViewController}" />
    <ui:param name="conversationController"
value="#{convocatoriaGranAreaConvController}" />
  </ui:define>
</ui:decorate>
</div>
</div>
</p:panel>
</p:tab>

```

Este es el formulario de edición y creación de registros en la pantalla de ConvocatoriaGranArea. Te explico cada elemento y su funcionalidad en el formulario:

1. Estructura General:

- El formulario está contenido dentro de un **<p:tab>**, con el título **Guardar/Editar**, para organizar las pestañas de la interfaz.
- El panel principal (**<p:panel id="formularioEditarCrear">**) encapsula el contenido del formulario y le da un aspecto visual consistente con la interfaz de **PrimeFaces**.

2. Campo granArea con <p:selectOneMenu>:

- Se usa para seleccionar el **granArea** en el registro de **ConvocatoriaGranArea**.
- **Atributo id:** Incluye **vadef-#{msg['sb.convert.selectOne.tipoGranAreas.ID']}**, un identificador dinámico para el menú de selección, probablemente utilizado para evitar conflictos en la ID generada.
- **value:** Se enlaza a **#{convocatoriaGranAreaViewController.selected.granAreaId}**, que guarda el valor seleccionado en la propiedad **granAreaId** de la entidad seleccionada.

- **converter:** Se usa **valorDefinicionConverter** para convertir el objeto seleccionado entre el modelo de Java y la vista. Este conversor se asegura de que los elementos mostrados sean correctamente manejados en el backend.
- **required y requiredMessage:** Este campo es obligatorio y se muestra un mensaje (**requiredMessage**) si el usuario intenta enviar el formulario sin seleccionarlo.
- **Atributos de Estilo:** **style="width:100%;"** establece que el campo ocupe todo el ancho disponible, mientras que **panelStyle="width:180px"** define el ancho del panel de opciones desplegables.
- **Filtrado:** **filter="true"** y **filterMatchMode="contains"** permiten que los usuarios busquen valores en el menú usando coincidencias parciales.

3. Opciones de Selección (<f:selectItem> y <f:selectItems>):

- **<f:selectItem>:** Incluye una opción inicial para indicarle al usuario que seleccione un valor.
- **<f:selectItems>:** Genera las opciones dinámicamente a partir de **#{{convocatoriaGranAreaViewController.listaGranAreas}}**, donde cada elemento de **listaGranAreas** representa un objeto **granArea**.
 - **itemLabel:** Es el texto visible en el menú desplegable, que aquí corresponde a **descripcionValorDefinicion** del objeto **granArea**.
 - **itemValue:** Es el valor seleccionado cuando el usuario elige esta opción; se asigna directamente el objeto **granArea**.

4. Sección de Botones (<ui:decorate>):

- La plantilla **_formButtons.xhtml** se carga para proporcionar botones de envío, guardado, y cancelación.
- Mediante **ui:param**, se pasan referencias de los controladores (**sessionController** y **conversationController**), lo que permite que los botones ejecuten acciones específicas definidas en estos controladores.

Este formulario está configurado para que el usuario seleccione un **granArea**, con validaciones y filtrado para mejorar la experiencia de usuario. Los botones adicionales permiten guardar o cancelar las acciones en esta pantalla de edición o creación de registros de **ConvocatoriaGranArea**.

Visualización del menu de gran areas ocde - areas ocde y disciplinas del proyecto (Proyectos/Programas)

- Se adiciona en la pestaña de Info Básica la sección Gran Areas – Areas OCDE y Disciplinas del proyecto, esta sección es visible en Proyectos/Programas – Proyectos – Detalles

```
<!-- PRY - Gran Área OCDE y Área OCDE o PRY - Gran Área OCDE, Área OCDE y Disciplina OCDE -->
<ui:fragment rendered="#{proyectoSessController.existeMenu('FOR1150') or
proyectoSessController.existeMenu('FOR1155')}">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-parent="#accordion"
href="#collapseOCDE">#{msg['gestion.proyectosProgramas.proyectos.OCDE']}</a>
      </h4>
    </div>
    <div id="collapseOCDE" class="panel-collapse collapse">
      <div class="panel-body">
        <div class="row">
          <div class="col-lg-12">
            <h:panelGroup layout="block" styleClass="well">
              <div class="row">
                <div class="col-lg-12">
                  <dl
horizontal">
                    <dt>#{msg['gestion.proyectosProgramas.proyectos.granAreaOCDE']}</dt>
                    <dd
class="text-
justify">#{proyectoSessController.selected.granAreaOcdeId.descripcionValorDefinicion}</dd>
                    <br/>
                    <dt>#{msg['gestion.proyectosProgramas.proyectos.areaOCDE']}</dt>
                    <dd
class="text-
justify">#{proyectoSessController.selected.areaOcdeId.descripcionValorDefinicion}</dd>
                    <ui:fragment
rendered="#{proyectoSessController.existeMenu('FOR1155')}">
                    <br/>
                    <dt>#{msg['gestion.proyectosProgramas.proyectos.disciplinaOCDE']}</dt>
                    <dd
class="text-
justify">#{proyectoSessController.selected.disciplinaOcdeId.descripcionValorDefinicion}</dd>
                    </ui:fragment>
                  </dl>
                </div>
              </div>
            </h:panelGroup>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
</ui:fragment>
```

Esta sección está diseñada para mostrar detalles específicos de un proyecto relacionado con el área OCDE. Vamos a desglosar los elementos clave:

Estructura General

- La sección está envuelta en un **<ui:fragment>** que solo se renderiza si al menos uno de los menús **FOR1150** o **FOR1155** existe, como lo indica el atributo **rendered** de la etiqueta. Esto permite controlar la visibilidad de esta sección de acuerdo con ciertas condiciones.

Panel de Desplegable (Accordion)

- **Panel Principal:** Utiliza un panel de Bootstrap (**<div class="panel panel-default">**) para organizar visualmente el contenido. Dentro del panel, hay un encabezado con un título que dice **"#{msg['gestion.proyectosProgramas.proyectos.OCDE']}"**, que se traduce con el mensaje correspondiente desde el archivo de recursos.
- **Colapsable:** La sección dentro del panel tiene un comportamiento de colapsable (**<div id="collapseOCDE" class="panel-collapse collapse">**), lo que significa que el contenido puede ser expandido o contraído al hacer clic en el título. Esto permite que la información no ocupe mucho espacio cuando no es necesaria.
 - **ID collapseOCDE:** Este ID se utiliza para asociar la sección de contenido con el enlace del encabezado. Es decir, al hacer clic en el enlace del encabezado, se expandirá o colapsará el contenido correspondiente.

Contenido de la Sección (Detalles del Proyecto)

- Dentro del cuerpo del panel, se está utilizando una estructura de presentación en bloque (**h:panelGroup**) para mostrar los detalles del proyecto. Los datos se muestran en un formato de lista horizontal mediante el componente **<dl class="dl-horizontal">**.
 - **Gran Área OCDE:** Se muestra el valor de **granAreaOcdeld.descripcionValorDefinicion**, lo cual es un texto descriptivo que hace referencia a la "Gran Área OCDE" del proyecto seleccionado.
 - **Área OCDE:** Similarmente, se muestra el valor de **areaOcdeld.descripcionValorDefinicion**, que hace referencia a "Área OCDE".

- **Disciplina OCDE:** Esta parte solo se muestra si el menú **FOR1155** existe, lo que se logra mediante el uso de un `<ui:fragment>` con el atributo `rendered="#{proyectoSessController.existeMenu('FOR1155')}"`. Si la condición es verdadera, se muestra también el valor de `disciplinaOcdeld.descripcionValorDefinicion`.

Detalles del Formato

- **<dl> (Lista de Definición):**
 - **<dt>:** Define el término o encabezado de cada par **clave-valor**, como "Gran Área OCDE", "Área OCDE", y "Disciplina OCDE".
 - **<dd>:** Define la descripción o valor que se muestra al usuario, como `#{proyectoSessController.selected.granAreaOcdeld.descripcionValorDefinicion}`, que es el valor de la "Gran Área OCDE".
- **class="text-justify":** La clase **CSS** se usa para justificar el texto en cada **<dd>**, lo que asegura que el contenido se vea alineado correctamente.

Condicionalidad de Disciplina OCDE

- **<ui:fragment rendered="#{proyectoSessController.existeMenu('FOR1155')}":** Este fragmento asegura que el campo "Disciplina OCDE" solo se muestre si el menú con el código **FOR1155** está habilitado o existe. Si no es así, esta parte se omite del HTML generado.

Visualización del menu de gran areas ocde - areas ocde y disciplinas del proyecto (Beneficios Tributarios) y codigo bpin y numero de contrato

- Se adiciona en la pestaña de Info Básica la sección para consultar la información de los campos de gran areas ocde - areas ocde y disciplinas del proyecto. Esta sección es visible en Beneficios Tributarios – Proyectos – Detalles

```
<ui:fragment                                rendered="#{proyectoBeneficiosAllSessController.renderCampo('410',
'CAMPOSPRY')}">
  <dt>#{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.granAreaOcde']}:
</dt>
  <dd>#{proyectoBeneficiosAllSessController.selected.granAreaOcdeId.descripcionValorDefinicion}
</dd>
</ui:fragment>
```

```
<ui:fragment                                rendered="#{proyectoBeneficiosAllSessController.renderCampo('420',
'CAMPOSPRY')}">
    <dt>#{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.areaOcde']}</dt>
    <dd>#{proyectoBeneficiosAllSessController.selected.areaOcdeId.descripcionValorDefinicion}</dd>
</ui:fragment>
<ui:fragment                                rendered="#{proyectoBeneficiosAllSessController.renderCampo('470',
'CAMPOSPRY')}">
    <dt>#{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.disciplinaOcde']}</dt>
    <dd>#{proyectoBeneficiosAllSessController.selected.disciplinaOcdeId.descripcionValorDefinicion}</dd>
</ui:fragment>
```

Este código está diseñado para mostrar ciertos campos de un proyecto relacionado con beneficios tributarios, específicamente campos como "Gran Área OCDE", "Área OCDE" y "Disciplina OCDE". Se está utilizando **ui:fragment** y expresiones condicionales **rendered** para controlar la visibilidad de cada campo. Vamos a desglosar cada parte del código:

Estructura de cada <ui:fragment>

Cada uno de los **<ui:fragment>** controla la visualización de un campo específico del proyecto. La sintaxis se parece mucho al código anterior, con las siguientes diferencias:

1. Condicionalidad con rendered:

- Cada campo se renderiza (muestra en la interfaz de usuario) solo si la condición especificada en **rendered** es verdadera.
- En este caso, se está utilizando el método **proyectoBeneficiosAllSessController.renderCampo()** para verificar si se debe renderizar el campo en función de ciertos parámetros: el primer parámetro parece ser un código (por ejemplo, '410', '420', '470') y el segundo parámetro es la categoría **'CAMPOSPRY'**.
- Por ejemplo, el primer campo se muestra si **proyectoBeneficiosAllSessController.renderCampo('410', 'CAMPOSPRY')** retorna **true**.

2. Campos a Mostrar:

- **Gran Área OCDE:** El primer fragmento renderiza la "Gran Área OCDE", mostrando su descripción con el valor de

proyectoBeneficiosAllSessController.selected.granAreaOcdeld.descripcionValorDefinicion.

- **Área OCDE:** El segundo fragmento muestra la "Área OCDE", obteniendo su valor a través de **proyectoBeneficiosAllSessController.selected.areaOcdeld.descripcionValorDefinicion.**
- **Disciplina OCDE:** El tercer fragmento muestra la "Disciplina OCDE", con el valor de **proyectoBeneficiosAllSessController.selected.disciplinaOcdeld.descripcionValorDefinicion.**

3. Estructura de Visualización:

- **<dt>:** Define la etiqueta o título de cada campo, que se traduce desde el archivo de mensajes con **{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.granAreaOcdeld']}. Esto permite una traducción dinámica dependiendo del idioma o configuración del sistema.**
 - **<dd>:** Muestra el valor correspondiente de cada campo, es decir, la descripción del valor de "Gran Área OCDE", "Área OCDE", y "Disciplina OCDE" para el proyecto seleccionado.
- Se adiciona en la pestaña de Info Básica la sección para consultar la información de los campos código BPIN y numero de contrato del proyecto, esta sección es visible en Beneficios Tributarios – Proyectos – Detalles

```
<div class="row">
  <div class="col-lg-6">
    <dl class="dl-horizontal">

      <dt>#{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.registraCodigoBPIN']}</dt> <dd>#{proyectoBeneficiosAllSessController.selected.registraCodigoBPIN eq 'S' ? 'Sí' : 'No'}</dd>

      <dt>#{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.registraNumeroContrato']}</dt> <dd>#{proyectoBeneficiosAllSessController.selected.registraNumeroContrato eq 'S' ? 'Sí' : 'No'}</dd>

    </dl>
  </div>
  <div class="col-lg-6" id="cluster_info">
```



```
<dl class="dl-horizontal" >
  <ui:fragment
rendered="#{proyectoBeneficiosAllSessController.selected.registraCodigoBPIN eq 'S'}">

    <dt>#{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.codigoBPIN']}:</
dt> <dd>#{proyectoBeneficiosAllSessController.selected.codigoBPIN}</dd>
    </ui:fragment>

    <ui:fragment
rendered="#{proyectoBeneficiosAllSessController.selected.registraNumeroContrato eq 'S'}">

    <dt>#{msg['gestion.beneficiosTributarios.proyectosRegistrados.exencion.numeroContrato']
}:</dt> <dd>#{proyectoBeneficiosAllSessController.selected.numeroContrato}</dd>
    </ui:fragment>
  </dl>
</div>
</div>
```

Este código se encarga de mostrar información detallada de un proyecto relacionado con beneficios tributarios, específicamente la información sobre el registro de un código BPIN (Número de Proyecto de Inversión) y un número de contrato.

Desglose del código:

1. Estructura General (Filas y Columnas):

- **<div class="row">**: Esta etiqueta crea una fila que contiene dos columnas (**col-lg-6**), cada una ocupando la mitad del ancho de la pantalla en una vista grande (como un dispositivo de escritorio).
- **Primera columna (<div class="col-lg-6">**: Contendrá los títulos y los valores que indican si ciertos campos (como el código BPIN y el número de contrato) están registrados o no.
- **Segunda columna (<div class="col-lg-6" id="cluster_info">**: Contendrá los detalles adicionales, como el código BPIN y el número de contrato, pero solo se mostrarán si se cumplen ciertas condiciones.

2. Primera columna - Información de registro:

- **<dl class="dl-horizontal">**: Esta es una lista de definiciones horizontal, donde cada término (definido por **<dt>**) tiene su descripción (definida por **<dd>**).
- Los dos campos que se muestran en esta columna son:

- **Registra Código BPIN:** Se muestra "Sí" o "No" dependiendo de si `proyectoBeneficiosAllSessController.selected.registraCodigoBPIN` es igual a 'S' o no.
- **Registra Número de Contrato:** Se muestra "Sí" o "No" dependiendo de si `proyectoBeneficiosAllSessController.selected.registraNumeroContrato` es igual a 'S' o no.

3. Segunda columna - Detalles adicionales (ocultos hasta cumplir condición):

- **<ui:fragment rendered="...">**: Los fragmentos dentro de esta columna se renderizan solo si la condición **rendered** es verdadera. Las condiciones que determinan la visibilidad de los campos son:
 - **Código BPIN:** Solo se muestra si `proyectoBeneficiosAllSessController.selected.registraCodigoBPIN eq 'S'`. Si es cierto, se muestra el valor de `proyectoBeneficiosAllSessController.selected.codigoBPIN`.
 - **Número de Contrato:** Solo se muestra si `proyectoBeneficiosAllSessController.selected.registraNumeroContrato eq 'S'`. Si es cierto, se muestra el valor de `proyectoBeneficiosAllSessController.selected.numeroContrato`.

4. Mensajes traducibles:

- Los textos dentro de `{msg['...']}` se resuelven usando el archivo de mensajes (`.properties` en el servidor). Esto permite que las etiquetas y los valores se adapten a diferentes idiomas según la configuración del sistema.

Este bloque de código es un diseño de interfaz de usuario que muestra de manera condicional si un proyecto registrado en el sistema tiene asociado un código BPIN o un número de contrato. Los valores "Sí" o "No" son el primer indicador de registro de estos datos, y si el campo está registrado, el sistema mostrará detalles adicionales de ese campo (como el código BPIN o el número de contrato) en la segunda columna. Esto permite que solo se muestren los campos relevantes dependiendo de los datos disponibles.

- Internacionalización en Español e Inglés

```
##### ES
#Configuracion - Gran Areas por Convocatoria
sb.menu.gestion.configuracion.convocatorias.granAreas           = Gran Áreas
gestion.configuracion.convocatoria.granAreas.button            = Gran Áreas
gestion.configuracion.convocatoria.granAreas.header             = Gran Áreas de la
Convocatoria
gestion.configuracion.convocatoria.granAreas.granArea.header    = Gran Área
gestion.configuracion.convocatoria.granAreas.usuarioCreador.header = Usuario Creador
gestion.configuracion.convocatoria.granAreas.fechaCreacion.header = Fecha Creación

gestion.configuracion.convocatoria.granAreas.granArea.label      = Gran Área
gestion.configuracion.convocatoria.granAreas.granArea.required    = Debe seleccionar la gran
área
com.sb.persistence.constraint.ORA-00001.SGPCONVGARE_UK           = Ya se ha configurado el
gran área seleccionada para esta convocatoria
com.sb.persistence.constraint.ORA-02292.SGPCONVGARE_ADMVALDEF_FK = La gran área no puede
ser eliminada porque se encuentra asociada a una convocatoria

gestion.beneficiosTributarios.proyectosRegistrados.exencion.registraCodigoBPIN = Registra
Código BPIN
gestion.beneficiosTributarios.proyectosRegistrados.exencion.codigoBPIN          = Código
BPIN
gestion.beneficiosTributarios.proyectosRegistrados.exencion.registraNumeroContrato =
Registra Número Contrato
gestion.beneficiosTributarios.proyectosRegistrados.exencion.numeroContrato        = Número
Contrato
gestion.beneficiosTributarios.proyectosRegistrados.exencion.disciplinaOcde         =
Disciplina OCDE

##### EN
#Configuracion - Gran Areas por Convocatoria
sb.menu.gestion.configuracion.convocatorias.granAreas           = Large Areas
gestion.configuracion.convocatoria.granAreas.button            = Large Areas
gestion.configuracion.convocatoria.granAreas.header             = Large Areas of the Call
gestion.configuracion.convocatoria.granAreas.granArea.header    = Large Area
gestion.configuracion.convocatoria.granAreas.usuarioCreador.header = User Creator
gestion.configuracion.convocatoria.granAreas.fechaCreacion.header = Creation Date

gestion.configuracion.convocatoria.granAreas.granArea.label      = Large Area
gestion.configuracion.convocatoria.granAreas.granArea.required    = You must select the large
area
com.sb.persistence.constraint.ORA-00001.SGPCONVGARE_UK           = The large area selected
for this call has already been configured
com.sb.persistence.constraint.ORA-02292.SGPCONVGARE_ADMVALDEF_FK = The large area cannot be
deleted because it is associated with a call

gestion.beneficiosTributarios.proyectosRegistrados.exencion.registraCodigoBPIN = Register
BPIN Code
gestion.beneficiosTributarios.proyectosRegistrados.exencion.codigoBPIN          = BPIN Code
gestion.beneficiosTributarios.proyectosRegistrados.exencion.registraNumeroContrato =
Register Contract Number
gestion.beneficiosTributarios.proyectosRegistrados.exencion.numeroContrato        = Contract
Number
gestion.beneficiosTributarios.proyectosRegistrados.exencion.disciplinaOcde         =
Discipline OCDE
```

Formulario de Proyectos y Programas

Menú Gran Área OCDE, Área OCDE y Disciplina OCDE por Proyecto

- Creación nuevo menú de Gran Área OCDE, Área OCDE y Disciplina OCDE por proyecto, asignación de funciones y derechos para la nueva pantalla

```
insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
values (null, 'FDMENU', '21155', 'PRY - Gran Área OCDE, Área OCDE y Disciplina OCDE', 'S',
null, null, null, 'FOR1155', null, null, 'SIGP_WEB', to_date('23-05-2024', 'dd-mm-yyyy'),
null, null, null, null, null, null, null, null);

--funciones
insert into sb_funciones (FUNCION_ID, APLICACION, FUNCION, NOMBRE_FUNCION, TIPO_FUNCION, MENU,
OPCION, URL, URL_AYUDA, FUNCION_PADRE, ICONO, USUARIO_CREADOR, FECHA_CREACION,
USUARIO_MODIFICADOR, FECHA_MODIFICACION)
values (null, 'SIGP_WEB', 'FOR1155', 'Gran Área OCDE, Área OCDE y Disciplina OCDE', 'F', null,
null, '/app/proyecto/modificarProyecto.xhtml#granAreaOcd', null, 'FOR1010', null,
'SIGP_WEB', to_date('23-05-2024', 'dd-mm-yyyy'), null, null);

--derechos
insert into sb_derechos_funciones (DERECHO_FUNCION_ID, APLICACION, FUNCION, GRUPO, ESTADO,
USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION)
values (null, 'SIGP_WEB', 'FOR1155', 'SGPS_ADMINISTRADOR', 'H', 'SIGP_WEB', to_date('23-05-
2024', 'dd-mm-yyyy'), null, null);
```

- Crear nuevos campos de **DISCIPLINA_ID** en la tabla de **SGP_PROYECTOS** en la base de datos Sigp

```
-- Add/modify columns
alter table SGP_PROYECTOS add disciplina_ocde_id NUMBER(15);
-- Add comments to the columns
comment on column SGP_PROYECTOS.disciplina_ocde_id
is 'Indica la Disciplina para la Organización para la Cooperación y el Desarrollo Económico
(OCDE) a la cual apunta el proyecto';
-- Create/Recreate indexes
create index SGPPRY_ADMVDEF_DISC_OCDE_FK_I on SGP_PROYECTOS (disciplina_ocde_id)
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
initial 64K
next 1M
minextents 1
maxextents unlimited
);
-- Create/Recreate primary, unique and foreign key constraints
alter table SGP_PROYECTOS
```

```
add constraint SGPPRY_ADMVDEF_DISC_OCDE_FK foreign key (DISCIPLINA_OCDE_ID)
references adm_valores_definiciones (VALOR_DEFINICION_ID);
```

- Se mapean nuevos campos en entidad **Proyecto.java** de la nueva tabla

```
@JoinColumn(name = "DISCIPLINA_OCDE_ID", referencedColumnName = "VALOR_DEFINICION_ID")
@ManyToOne
private ValorDefinicion disciplinaOcdeId;

public ValorDefinicion getDisciplinaOcdeId() {
    return disciplinaOcdeId;
}

public void setDisciplinaOcdeId(ValorDefinicion disciplinaOcdeId) {
    this.disciplinaOcdeId = disciplinaOcdeId;
}
```

- Se mapea nuevos campos en **ProyectoDTO.java** de la nueva tabla

```
@Bind(name = "disciplinaOcdeId", returnType = ValorDefinicionDTO.class, type =
ValorDefinicion.class)
private ValorDefinicionDTO disciplinaOcdeId;

public ValorDefinicionDTO getDisciplinaOcdeId() {
    return disciplinaOcdeId;
}

public void setDisciplinaOcdeId(ValorDefinicionDTO disciplinaOcdeId) {
    this.disciplinaOcdeId = disciplinaOcdeId;
}
```

- Se agrega en pantalla de información general del proyecto el menu

```
<ui:fragment                                rendered="#{appSessController.renderPage('FOR1150')}"                                or
appSessController.renderPage('FOR1155')}>">
    <div class="row" id="granAreaOcde">
        <div class="col-lg-12">
            <div class="form-group">
                <p:outputLabel
value="#{msg['formulario.proyectos.modificar.granAreaOcde.label']}"
                                styleClass="col-lg-3 control-label"
                                for="areaOcde-
#{msg['sb.convert.selectOne.granAreaOCDE.ID']}">
                                />
                <div class="col-lg-9">
                    <p:selectOneMenu                                id="areaOcde-
#{msg['sb.convert.selectOne.granAreaOCDE.ID']}">
value="#{appSessController.proyectoSelected.granAreaOcdeId}"
```

```

disabled="#{!proyectoConvController.renderRegistroProyecto()}"
converter="valorDefinicionConverter"
required="true"
requiredMessage="#{msg['formulario.proyectos.modificar.granAreaOcde.required']}"
style="width:100%"
autoWidth="false"
>
<f:selectItem
itemLabel="#{msg['sb.selectOneMenu.registro']}" />
<f:selectItems
value="#{proyectoSessController.listaGranAreasOcde}"
var="granAreaOcde"

itemDescription="#{granAreaOcde.descripcionValorDefinicion}"
itemLabel="#{granAreaOcde.descripcionValorDefinicion}"
itemValue="#{granAreaOcde}"

/>
<p:ajax event="change"

update="panelAreasOcde,panelDisciplinasOCDE"
process="@this"

listener="#{proyectoSessController.cargarListaAreasOCDE()}"

oncomplete="$('##{proyectoSessController.modulo}Form\\\\\\\\areaOcde-
#{msg['sb.convert.selectOne.granAreaOCDE.ID']}').focus();"
/>
</p:selectOneMenu>
</div>
</div>
</div>
</div>
<div class="row">
<h:panelGroup id="panelAreasOcde" layout="block" >
<div class="col-lg-12">
<div class="form-group">
<p:outputLabel
value="#{msg['formulario.proyectos.modificar.areaOcde.label']}"
styleClass="col-lg-3
control-label"
for="areaOcde-A0"
/>
<div class="col-lg-9">
<p:selectOneMenu id="areaOcde-A0"

value="#{appSessController.proyectoSelected.areaOcdeId}"
disabled="#{!proyectoConvController.renderRegistroProyecto()}"
converter="valorDefinicionConverter1"
required="true"
requiredMessage="#{msg['formulario.proyectos.modificar.areaOcde.required']}"
style="width:100%"

```

```

autoWidth="false"
                                >
itemLabel="#{msg['sb.selectOneMenu.registro']}" />
                                <f:selectItems
value="#{proyectoSessController.listaAreasOcdc}"
var="areaOcdc"
itemDescription="#{areaOcdc.descripcionValorDefinicion}"
itemLabel="#{areaOcdc.descripcionValorDefinicion}"
itemValue="#{areaOcdc}"
                                />
                                <p:ajax event="change"
                                listener="#{proyectoSessController.cargarListaDisciplinasOCDE()}"
                                update="panelDisciplinasOCDE"
                                process="@this"
                                oncomplete="$('##{proyectoSessController.modulo}Form\\\\\\\\areaOcdc-AO').focus();"
                                />
                                </p:selectOneMenu>
                                </div>
                                </div>
                                </div>
                                </h:panelGroup>
                                </div>
                                <ui:fragment rendered="#{appSessController.renderPage('FOR1155')}">
                                <h:panelGroup id="panelDisciplinasOCDE" layout="block" >
                                <div class="row">
                                <div class="col-lg-12">
                                <div class="form-group">
                                <p:outputLabel
value="#{msg['formulario.proyectos.modificar.disciplinaOcdc.label']}"
                                styleClass="col-lg-3
control-label"
                                for="disciplinaOcdc-
discOCDE"/>
                                <div class="col-lg-9">
                                <p:selectOneMenu id="disciplinaOcdc-
discOCDE"
value="#{appSessController.proyectoSelected.disciplinaOcdcId}"
required="true"
requiredMessage="#{msg['formulario.proyectos.modificar.disciplinaOcdc.required']}"
disabled="#{!proyectoConvController.renderRegistroProyecto()}"
converter="valorDefinicionConverter1"
style="width:100%;"
autoWidth="false"
panelStyle="width:180px"

```

```

effect="fade"
filter="true"
filterMatchMode="contains"

itemLabel="#{msg['sb.selectOneMenu.registro']}" />
value="#{proyectoSessController.listaDisciplinasOcde}"
var="disciplinaOcde"
itemDescription="#{disciplinaOcde.descripcionValorDefinicion}"
itemLabel="#{disciplinaOcde.descripcionValorDefinicion}"
itemValue="#{disciplinaOcde}"
/>
</p:selectOneMenu>
</div>
</div>
</div>
</h:panelGroup>
</ui:fragment>
</ui:fragment>

```

El código presenta un formulario en el que se permite seleccionar un Gran Área OCDE, un Área OCDE y una Disciplina OCDE. La visibilidad y disponibilidad de estos campos están condicionadas por los permisos de página y las interacciones previas. A través de los menús desplegables, los usuarios pueden elegir estos valores, y en función de la selección, se cargan dinámicamente nuevas opciones en los campos relacionados (como las disciplinas, dependiendo del área seleccionada). Además, los campos están validados para asegurar que el usuario haga una selección válida y obligatoria.

A continuación, se desglosan los elementos del menú, sus condiciones y la funcionalidad implementada:

Desglose del código:

1. **Fragmento de UI Visible Bajo Condición de Renderización (rendered):** El formulario y sus campos solo se renderizan si el usuario tiene permisos para ver las páginas 'FOR1150' o 'FOR1155'. Esto se maneja con las condiciones `#{appSessController.renderPage('FOR1150')}` or `appSessController.renderPage('FOR1155')}`.
 - Si alguna de estas condiciones es verdadera, el bloque completo de contenido se renderiza.

2. Selección del Gran Área OCDE:

- **<p:selectOneMenu>**: Este es un menú desplegable (**dropdown**) donde el usuario selecciona el "Gran Área OCDE".
- **Condición de habilitación (disabled)**: El campo solo es editable si **proyectoConvController.renderRegistroProyecto()** es verdadero, lo que indica si el proyecto está en un estado donde se puede registrar.
- **f:selectItems**: La lista de opciones se obtiene de **proyectoSessController.listaGranAreasOcde**, con cada elemento siendo representado por **granAreaOcde.descripcionValorDefinicion**.
- **p:ajax**: Se ejecuta cuando el valor cambia, lo que actualiza el panel de las áreas OCDE y las disciplinas OCDE asociadas. Además, se ejecuta un método en el controlador para cargar las áreas y disciplinas correspondientes.

3. Selección de Área OCDE:

- Similar al componente anterior, este campo permite seleccionar un área específica dentro del Gran Área OCDE seleccionada previamente.
- La lista de opciones se obtiene de **proyectoSessController.listaAreasOcde**.
- **p:ajax**: Se ejecuta cuando el valor cambia, actualizando el panel de disciplinas OCDE y ejecutando un método para cargar la lista de disciplinas correspondientes.

4. Selección de Disciplina OCDE:

- Este componente solo se renderiza si la condición **{appSessController.renderPage('FOR115')}** es verdadera, lo que significa que la página 'FOR115' tiene que ser visible para que este campo sea accesible.
- **p:selectOneMenu**: Este campo permite seleccionar una disciplina dentro del área OCDE seleccionada.
- La lista de disciplinas se obtiene de **proyectoSessController.listaDisciplinasOcde**, y se filtra por coincidencia parcial gracias al atributo **filter="true"** y **filterMatchMode="contains"**.
- **p:ajax**: Actualiza los paneles relacionados y ejecuta un método de controlador al cambiar el valor.

5. Comportamiento de los Componentes:

- **required="true" y requiredMessage:** Cada campo tiene la validación de que se debe seleccionar una opción. Si el campo no se selecciona, se muestra un mensaje de error indicando que la selección es obligatoria.
- **Condición de habilitación (disabled):** Los campos están deshabilitados si el proyecto no está en un estado donde se pueda modificar. Esta habilitación/deshabilitación depende de la función `proyectoConvController.renderRegistroProyecto()`.

- **Metodo cargarConvocatoriaGranAreas**

```
/**
 * Metodo para cargar las gran areas configuradas por convocatoria
 */
public void cargarConvocatoriaGranAreas() {
    try {
        Map<String, Object> filters = new HashMap<>();
        //Filtro por Convocatoria ID
        filters.put("convocatoriaId",
appSessController.getProyectoSelected().getConvocatoriaId());
        int count = convocatoriaServiceFacade.convocatoriaGranAreasListCount(filters);
        if (count > 0) {
            List<ConvocatoriaGranAreaDTO> listaGranAreas =
convocatoriaServiceFacade.convocatoriaGranAreasList(-1,
"granAreaId.descripcionValorDefinicion", "A", filters);
            listaGranAreasOcdc = new ArrayList<>();

            listaGranAreas.forEach((convocatoriaGranAreaDTO) -> {

                listaGranAreasOcdc.add(convocatoriaGranAreaDTO.getGranAreaId());
            });
        }
    } catch (Exception ex) {
        LOG.error("Error al cargar el listado de gran areas configurados por convocatoria", ex);
    }
}
```

El método **cargarConvocatoriaGranAreas** tiene como propósito cargar las grandes áreas configuradas por convocatoria, utilizando un filtro por **convocatoriaId**.

Desglose del Método:

1. Inicialización de los Filtros:

```
Map<String, Object> filters = new HashMap<>();
```

Se crea un mapa (**filters**) para almacenar los filtros que se aplicarán en la consulta.

2. Filtro por Convocatoria ID:

```
filters.put("convocatoriaId",  
appSessController.getProyectoSelected().getConvocatoriaId().getConvocatoriaId());
```

Se agrega un filtro al mapa con la clave **convocatoriaId**. El valor se obtiene de **appSessController.getProyectoSelected().getConvocatoriaId().getConvocatoriaId()**, lo que indica que se filtrarán los resultados por el ID de la convocatoria seleccionada en el proyecto actual.

3. Obtención del Conteo de Resultados:

```
int count = convocatoriaServiceFacade.convocatoriaGranAreasListCount(filters);
```

Se realiza una consulta para obtener el número total de grandes áreas configuradas para la convocatoria, aplicando el filtro de **convocatoriaId**. Este valor se almacena en **count**.

4. Condicional para Verificar la Existencia de Resultados:

```
if (count > 0) {
```

Si se encuentran resultados (es decir, si **count > 0**), se procede a cargar la lista de grandes áreas configuradas.

5. Obtención de la Lista de Grandes Áreas:

```
List<ConvocatoriaGranAreaDTO> listaGranAreas =  
convocatoriaServiceFacade.convocatoriaGranAreasList(-1,  
"granAreaId.descripcionValorDefinicion", "A", filters);
```

Se llama al servicio **convocatoriaServiceFacade.convocatoriaGranAreasList** para obtener la lista de grandes áreas configuradas para la convocatoria. Se pasan los siguientes parámetros:

- -1, -1: Los valores indican que no se aplican límites de paginación (es decir, se obtienen todos los resultados).
- "granAreaId.descripcionValorDefinicion": El nombre del campo por el cual se ordenan los resultados.
- "A": La dirección del orden (presumiblemente ascendente).
- filters: El mapa de filtros, que incluye el **convocatoriaId**.

6. Creación de la Lista de Grandes Áreas (OCDE):

```
listaGranAreasOcde = new ArrayList<>();
listaGranAreas.forEach((convocatoriaGranAreaDTO) -> {
    listaGranAreasOcde.add(convocatoriaGranAreaDTO.getGranAreaId());
});
```

Se crea una nueva lista llamada **listaGranAreasOcde** que almacenará las grandes áreas OCDE correspondientes. Luego, se recorre la lista **listaGranAreas** obtenida previamente, y se extrae el campo **granAreaId** de cada objeto **ConvocatoriaGranAreaDTO** y se agrega a **listaGranAreasOcde**.

7. Manejo de Excepciones:

```
} catch (Exception ex) {
    LOG.error("Error al cargar el listado de gran areas configurados por convocatoria ", ex);
}
```

Si ocurre alguna excepción durante el proceso, se captura y se registra un mensaje de error.

- Se definen métodos de consulta de grandes áreas configuradas por convocatoria en Facade EJB

```
//
//
//GRAN AREAS POR CONVOCATORIA
//
/**
 * Consulta el listado de GranAreas por convocatoria registrados en la base
 * de datos
 *
 * @param first indica desde donde empieza la consulta
 * @param pageSize indica el final de la consulta (first + pageSize)
 * @param sortField indica el campo por el que se realiza el ordenamiento
 * @param sortOrder indica el orden (ascendente o descendente)
 * @param filters contiene los filtros al momento de solicitar el listado de
 * GranAreas por convocatoria
 * @return lista de GranAreas por convocatoria encontrados en el rango
 * enviado y basado en los filtros enviados
 * @throws Exception
 */
List<ConvocatoriaGranAreaDTO> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters) throws Exception;

/**
 * Cuenta la cantidad de GranAreas por convocatoria actualmente registrados
 * en la base de datos
```

```
*
* @param filters
* @return
* @throws Exception
*/
int convocatoriaGranAreasListCount(Map<String, Object> filters) throws Exception;
```

- Se implementan métodos de consulta de grandes áreas configuradas por convocatoria facade EJB para funcionalidad de pantalla

```
//----- Configuración Gran Áreas por Convocatoria -----
-----//
@Override
public List<ConvocatoriaGranAreaDTO> convocatoriaGranAreasList(int first, int pageSize,
String sortField, String sortOrder, Map<String, Object> filters) throws Exception {
    ObjectConverter<ConvocatoriaGranAreaDTO, ConvocatoriaGranArea> converter =
    objectConverterFactory.checkOut();
    List<ConvocatoriaGranAreaDTO> listConvocatoriaGranAreaDTOS =
    converter.toTransferList(convocatoriaService.convocatoriaGranAreasList(first, pageSize,
sortField, sortOrder, filters), ConvocatoriaGranAreaDTO.class);
    objectConverterFactory.checkIn(converter);
    return listConvocatoriaGranAreaDTOS;
}

@Override
public int convocatoriaGranAreasListCount(Map<String, Object> filters) throws Exception {
    return convocatoriaService.convocatoriaGranAreasListCount(filters);
}
```

- Se agrega en interfaz **ConvocatoriaService.java** métodos de consulta de grandes áreas configuradas por convocatoria en Service EJB.

```
List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters) throws SystemException;

int convocatoriaGranAreasListCount(Map<String, Object> filters) throws SystemException;
```

- Se agregan en **ConvocatoriaServiceBean.java** donde implementan métodos métodos de consulta de grandes áreas configuradas por convocatoria en Service EJB.

```
//----- Convocatoria Gran Área -----//
@Override
public List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters) throws SystemException {
    return
    this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaGranAreasList(first,
pageSize, sortField, sortOrder, filters);
}

@Override
```

```
public int convocatoriaGranAreasListCount(Map<String, Object> filters) throws SystemException
{
    return
    this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaGranAreasListCount(filters)
    ;
}
```

- Se crean métodos en **ConvocatoriaStoreManager.java** donde se definen métodos de consulta criteria para funcionalidad de pantalla

```
List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters);

int convocatoriaGranAreasListCount(Map<String, Object> filters);
```

- Se crean métodos en **ConvocatoriaStoreManagerImpl.java** donde implementan métodos de consulta en criteria para funcionalidad de pantalla

```
//----- GRAN AREAS POR CONVOCATORIA -----//
@Override
public List<ConvocatoriaGranArea> convocatoriaGranAreasList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters) {
    cb = em.getCriteriaBuilder();
    queryConvocatoriaGranArea = cb.createQuery(ConvocatoriaGranArea.class);
    convocatoriaGranAreaRoot = queryConvocatoriaGranArea.from(ConvocatoriaGranArea.class);
    queryConvocatoriaGranArea.select(convocatoriaGranAreaRoot);
    //Where
    queryConvocatoriaGranArea.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new
Predicate[]{})));
    //Order
    if (sortOrder.equals("A")) {
        queryConvocatoriaGranArea.orderBy(cb.asc(getConvocatoriaGranAreaPath(sortField)));
    } else {
        queryConvocatoriaGranArea.orderBy(cb.desc(getConvocatoriaGranAreaPath(sortField)));
    }
    //pagination
    TypedQuery<ConvocatoriaGranArea> tq = em.createQuery(queryConvocatoriaGranArea);
    if (pageSize >= 0) {
        tq.setMaxResults(pageSize);
    }
    if (first >= 0) {
        tq.setFirstResult(first);
    }
    return tq.getResultList();
}

@Override
public int convocatoriaGranAreasListCount(Map<String, Object> filters) {
    CriteriaQuery<Long> qCount;
    cb = em.getCriteriaBuilder();
    qCount = cb.createQuery(Long.class);
}
```

```

convocatoriaGranAreaRoot = qCount.from(ConvocatoriaGranArea.class);
qCount.select(cb.count(convocatoriaGranAreaRoot));
//Where
qCount.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new Predicate[]{})));
Long res = em.createQuery(qCount).getSingleResult();
return res.intValue();
}

private List<Predicate> filtrosConvocatoriaGranArea(Map<String, Object> filters) {
    List<Predicate> lista = new ArrayList<>();
    List<String> camposFiltro = new ArrayList<>();
    Path pathFilter;
    camposFiltro.add("granAreaId.descripcionValorDefinicion");
    camposFiltro.add("convocatoriaId.descripcionConvocatoria");

    //Filtro por convocatoria ID
    Object convocatoriaId = filters.get("convocatoriaId");
    if (convocatoriaId != null) {

        lista.add(cb.equal(getConvocatoriaGranAreaPath("convocatoriaId.convocatoriaId"), (Long)
convocatoriaId));
    }

    //Filtro Global
    globalFilter(filters, camposFiltro, "getConvocatoriaGranAreaPath", lista);

    return lista;
}

private Path<?> getConvocatoriaGranAreaPath(String field) {
    //Joins
    Join<ConvocatoriaGranArea, Convocatoria> convocatoriaId =
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.convocatoriaId, JoinType.LEFT);
    Join<ConvocatoriaGranArea, ValorDefinicion> granAreaId =
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.granAreaId, JoinType.LEFT);

    Path<?> path = null;
    if (null == field) {
        path =
convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.convocatoriaGranAreaId);
    } else {
        switch (field) {
            case "convocatoriaGranAreaId":
                path =
convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.convocatoriaGranAreaId);
                break;
            case "usuarioCreador":
                path =
convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.usuarioCreador);
                break;
            case "fechaCreacion":
                path =
convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.fechaCreacion);
                break;
            case "convocatoriaId.convocatoriaId":
                path = convocatoriaId.get(Convocatoria_.convocatoriaId);
                break;
            case "convocatoriaId.descripcionConvocatoria":
                path =
convocatoriaId.get(Convocatoria_.descripcionConvocatoria);
                break;

```

```

        case "granAreaId.valorDefinicionId":
            path = granAreaId.get(ValorDefinicion_.valorDefinicionId);
            break;
        case "granAreaId.valorDefinicion":
            path = granAreaId.get(ValorDefinicion_.valorDefinicion);
            break;
        case "granAreaId.descripcionValorDefinicion":
            path
granAreaId.get(ValorDefinicion_.descripcionValorDefinicion);
            break;
        default:
            break;
    }
    }
    return path;
}

```

A continuación, se explica cada método relacionado con Criterios API que se usó para el módulo **ConvocatoriaGranArea**. Se revisa en detalle las implementaciones de **convocatoriaGranAreasList**, **convocatoriaGranAreasListCount**, **filtrosConvocatoriaGranArea**, y **getConvocatoriaGranAreaPath**.

1. Método **convocatoriaGranAreasList**

Este método obtiene una lista de entidades **ConvocatoriaGranArea** con filtros, ordenamiento, y paginación. Veamos cada paso:

- **Inicialización de **CriteriaBuilder** y **CriteriaQuery**:**

```

cb = em.getCriteriaBuilder();
queryConvocatoriaGranArea = cb.createQuery(ConvocatoriaGranArea.class);

```

CriteriaBuilder (cb) se utiliza para construir la consulta, proporcionando operaciones de criterios. **CriteriaQuery (queryConvocatoriaGranArea)** es la consulta específica para **ConvocatoriaGranArea**.

- **Definición de la raíz de la consulta:**

```

convocatoriaGranAreaRoot = queryConvocatoriaGranArea.from(ConvocatoriaGranArea.class);
queryConvocatoriaGranArea.select(convocatoriaGranAreaRoot);

```

convocatoriaGranAreaRoot define el punto de inicio para los atributos de la entidad **ConvocatoriaGranArea**, similar a una tabla en SQL.

- **Aplicación de filtros:**


```
queryConvocatoriaGranArea.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new  
Predicate[]{})));
```

Aquí se llaman los predicados de **filtrosConvocatoriaGranArea** y se aplican con **.where**. Los filtros son dinámicos, adaptándose a las condiciones proporcionadas en **filters**.

- **Ordenamiento:**

```
if (sortOrder.equals("A")) {  
    queryConvocatoriaGranArea.orderBy(cb.asc(getConvocatoriaGranAreaPath(sortField)));  
} else {  
    queryConvocatoriaGranArea.orderBy(cb.desc(getConvocatoriaGranAreaPath(sortField)));  
}
```

Dependiendo del valor de **sortOrder**, la consulta se ordena de forma ascendente (**asc**) o descendente (**desc**). El método **getConvocatoriaGranAreaPath** se usa para obtener la ruta específica del campo de ordenamiento, adaptándose al campo seleccionado.

- **Paginación:**

```
TypedQuery<ConvocatoriaGranArea> tq = em.createQuery(queryConvocatoriaGranArea);  
if (pageSize >= 0) {  
    tq.setMaxResults(pageSize);  
}  
if (first >= 0) {  
    tq.setFirstResult(first);  
}
```

pageSize y **first** controlan la cantidad de registros devueltos y el índice inicial. Esto es útil para manejar grandes cantidades de datos al cargar solo una parte de los registros en cada llamada.

2. Método convocatoriaGranAreasListCount

Este método calcula el número total de registros que cumplen con los filtros, útil para la paginación en el front-end:

- **Configuración de CriteriaQuery para contar:**

```
qCount = cb.createQuery(Long.class);  
convocatoriaGranAreaRoot = qCount.from(ConvocatoriaGranArea.class);  
qCount.select(cb.count(convocatoriaGranAreaRoot));
```

Aquí, **CriteriaQuery<Long>** se configura para realizar una operación de conteo (**count**). **cb.count** cuenta el número de elementos en la raíz **convocatoriaGranAreaRoot**.

- **Aplicación de filtros:**

```
qCount.where(cb.and(filtrosConvocatoriaGranArea(filters).toArray(new Predicate[]{})));
```

Se aplican los mismos filtros que en **convocatoriaGranAreasList**, lo cual asegura que el conteo coincida con los registros obtenidos en la lista.

3. Método filtrosConvocatoriaGranArea

Este método construye la lista de **Predicates** (condiciones) basados en los filtros proporcionados:

- **Lista de campos de filtro:**

```
List<String> camposFiltro = new ArrayList<>();  
camposFiltro.add("granAreaId.descripcionValorDefinicion");  
camposFiltro.add("convocatoriaId.descripcionConvocatoria");
```

camposFiltro contiene los nombres de los campos sobre los que se aplicará un filtro global.

- **Filtro específico para convocatoriaId:**

```
Object convocatoriaId = filters.get("convocatoriaId");  
if (convocatoriaId != null) {  
    lista.add(cb.equal(getConvocatoriaGranAreaPath("convocatoriaId.convocatoriaId"), (Long)  
convocatoriaId));  
}
```

Si el filtro **convocatoriaId** está presente, se agrega una condición equal para buscar coincidencias exactas en **convocatoriaId**.

- **Filtro global:**

```
globalFilter(filters, camposFiltro, "getConvocatoriaGranAreaPath", lista);
```

globalFilter aplica un filtro más amplio, permitiendo buscar coincidencias en varios campos al mismo tiempo (como una búsqueda de texto completo).

4. Método getConvocatoriaGranAreaPath

Este método es crucial para construir las rutas de los atributos en Criteria API para diferentes campos. Esto permite crear una lógica flexible para orden y filtrado en función del campo solicitado.

- **Join con otras entidades:**

```
Join<ConvocatoriaGranArea, Convocatoria> convocatoriaId =
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.convocatoriaId, JoinType.LEFT);
Join<ConvocatoriaGranArea, ValorDefinicion> granAreaId =
convocatoriaGranAreaRoot.join(ConvocatoriaGranArea_.granAreaId, JoinType.LEFT);
```

Aquí se establecen joins con las entidades **Convocatoria** y **ValorDefinicion** para acceder a sus campos. El **JoinType.LEFT** asegura que se incluyan registros de **ConvocatoriaGranArea** que puedan no tener coincidencias en estas tablas relacionadas.

- **Rutas de campos específicos:**

```
switch (field) {
    case "convocatoriaGranAreaId":
        path = convocatoriaGranAreaRoot.get(ConvocatoriaGranArea_.convocatoriaGranAreaId);
        break;
    // Otros casos...
}
```

Para cada campo solicitado (**field**), el método devuelve la ruta adecuada. Esto permite que los filtros y el ordenamiento sean dinámicos y adaptables a los campos solicitados.

- **Método cargarListaDisciplinasOCDE**

```
/**
 * Metodo para cargar el listado de disciplinas OCDE de conocimiento
 */
public void cargarListaDisciplinasOCDE() {
    try {
        listaDisciplinasOcde = new ArrayList<>();
        if (selected.getAreaOcdeId() != null) {
            if (selected.getAreaOcdeId().getTipoValidacion() != null) {
                listaDisciplinasOcde =
                proyectoServiceFacade.obtenerValoresDefinicion(selected.getAreaOcdeId().getTipoValidacion().
                getTipoDefinicion());
            }
        }
    } catch (Exception ex) {
        LOG.error("Error al consultar la lista de disciplinas OCDE", ex);
    }
}
```

```
}  
}
```

El método **cargarListaDisciplinasOCDE** tiene como objetivo cargar el listado de disciplinas OCDE de conocimiento basadas en el área OCDE seleccionada.

Desglose del Método:

1. Inicialización de la Lista de Disciplinas:

```
listaDisciplinasOcde = new ArrayList<>();
```

Se inicializa una lista vacía `listaDisciplinasOcde` donde se almacenarán las disciplinas OCDE obtenidas.

2. Verificación de si el Área OCDE Está Seleccionada:

```
if (selected.getAreaOcdeId() != null) {
```

Se verifica si el objeto `selected` (presumiblemente una entidad o DTO que representa el proyecto o la convocatoria seleccionada) tiene un área OCDE (`getAreaOcdeId()`) seleccionada. Si es null, el proceso se detendrá.

3. Verificación del Tipo de Validación del Área OCDE:

```
if (selected.getAreaOcdeId().getTipoValidacion() != null) {
```

Se comprueba si el `TipoValidacion` del área OCDE seleccionada no es null. Si no existe un `TipoValidacion`, el proceso se detendrá.

4. Obtención de las Disciplinas OCDE:

```
listaDisciplinasOcde  
proyectoServiceFacade.obtenerValoresDefinicion(selected.getAreaOcdeId().getTipoValidacion(),  
getTipoDefinicion());
```

Si se cumple la condición anterior (es decir, si tanto el área OCDE como su tipo de validación están presentes), se hace una llamada al servicio `proyectoServiceFacade.obtenerValoresDefinicion` con el tipo de definición de validación (`getTipoDefinicion()`) como parámetro. Esto devolverá una lista de disciplinas que se almacenan en `listaDisciplinasOcde`.

5. Manejo de Excepciones:

```
} catch (Exception ex) {  
    LOG.error("Error al consultar la lista de disciplinas OCDE", ex);  
}
```

Si ocurre alguna excepción durante el proceso de obtención de las disciplinas OCDE, se captura y se registra un mensaje de error en el log.

- Validar obligatoriedad en el formulario al Validar y Enviar Proyecto

```
if (convocatoriaMenu.getFuncion().equals("FOR1155")) { //Validaciones PRY - Gran Área OCDE,  
Área OCDE y Disciplina OCDE  
    if (proyecto.getGranAreaOcdeId() == null) {  
        descripcion = descripcion + " ingresar una gran área OCDE, ";  
    }  
    if (proyecto.getAreaOcdeId() == null) {  
        descripcion = descripcion + " ingresar un área OCDE";  
    }  
    if (proyecto.getDisciplinaOcdeId() == null) {  
        descripcion = descripcion + " , ingresar una disciplina OCDE";  
    }  
}
```

- Internacionalización en Español e Inglés

```
##### ES  
formulario.proyectos.modificar.disciplinaOcde.label    = Disciplina OCDE de conocimiento  
formulario.proyectos.modificar.disciplinaOcde.required = Debe seleccionar la disciplina OCDE  
de conocimiento  
  
##### EN  
formulario.proyectos.modificar.disciplinaOcde.label    = OECD Discipline of Knowledge  
formulario.proyectos.modificar.disciplinaOcde.required = You must select the OECD discipline  
of knowledge
```

Formulario de Beneficios Tributarios

Campos deCodigo BPIN, numero de Contrato y Disciplina OCDE por Proyecto

- Creación nuevo Campo Disciplina OCDE por proyecto, asignación de funciones y derechos para la nueva pantalla

```
insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
values (null, 'CAMPOSPRY', '470', 'DISCIPLINA (BT)', 'S', 'CARACTER', null, null, null, null,
null, 'SIGP_WEB', to_date('21-05-2024', 'dd-mm-yyyy'), null, null, null, null, null, null,
null, null);
```

- Crear nuevos campos en la tabla de **SGP_PROYECTOS** en la base de datos Sigg

```
-- Add/modify columns
alter table SGP_PROYECTOS add registra_codigo_bpın VARCHAR2(1);
alter table SGP_PROYECTOS add codigo_bpın VARCHAR2(30);
alter table SGP_PROYECTOS add registra_numero_contrato VARCHAR2(1);
alter table SGP_PROYECTOS add numero_contrato VARCHAR2(30);
-- Add comments to the columns
comment on column SGP_PROYECTOS.registra_codigo_bpın
is 'Indica si el proyecto registra código BPIN (S/N)';
comment on column SGP_PROYECTOS.codigo_bpın
is 'Código BPIN registrado por el proyecto';
comment on column SGP_PROYECTOS.registra_numero_contrato
is 'Indica si el proyecto registra número de contrato (S/N)';
comment on column SGP_PROYECTOS.numero_contrato
is 'Número contrato registrado por el proyecto';

-- Create/Recreate check constraints
alter table SGP_PROYECTOS
add constraint SGPPRY_REGISTRA_COD_BPIN_CK
check (REGISTRA_CODIGO_BPIN IN ('S' /*Si*/, 'N' /*No*/));
alter table SGP_PROYECTOS
add constraint SGPPRY_REGISTRA_NUM_CONTR_CK
check (REGISTRA_NUMERO_CONTRATO IN ('S' /*Si*/, 'N' /*No*/));

-- Add/modify columns
alter table SGP_PROYECTOS add disciplina_ocde_id NUMBER(15);
-- Add comments to the columns
comment on column SGP_PROYECTOS.disciplina_ocde_id
is 'Indica la Disciplina para la Organización para la Cooperación y el Desarrollo Económico
(OCDE) a la cual apunta el proyecto';
-- Create/Recreate indexes
create index SGPPRY_ADMVDEF_DISC_OCDE_FK_I on SGP_PROYECTOS (disciplina_ocde_id)
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
initial 64K
```

```
next 1M
minextents 1
maxextents unlimited
);
-- Create/Recreate primary, unique and foreign key constraints
alter table SGP_PROYECTOS
add constraint SGPPRY_ADMVDEF_DISC_OCDE_FK foreign key (DISCIPLINA_OCDE_ID)
references adm_valores_definiciones (VALOR_DEFINICION_ID);
```

- Se mapean nuevos campos en entidad **Proyecto.java** de la nueva tabla

```
@Column(name = "REGISTRA_CODIGO_BPIN")
private String registraCodigoBPIN;
@Column(name = "CODIGO_BPIN")
private String codigoBPIN;
@Column(name = "REGISTRA_NUMERO_CONTRATO")
private String registraNumeroContrato;
@Column(name = "NUMERO_CONTRATO")
private String numeroContrato;

@JoinColumn(name = "DISCIPLINA_OCDE_ID", referencedColumnName = "VALOR_DEFINICION_ID")
@ManyToOne
private ValorDefinicion disciplinaOcdeId;

public String getRegistraCodigoBPIN() {
    return registraCodigoBPIN;
}

public void setRegistraCodigoBPIN(String registraCodigoBPIN) {
    this.registraCodigoBPIN = registraCodigoBPIN;
}

public String getCodigoBPIN() {
    return codigoBPIN;
}

public void setCodigoBPIN(String codigoBPIN) {
    this.codigoBPIN = codigoBPIN;
}

public String getRegistraNumeroContrato() {
    return registraNumeroContrato;
}

public void setRegistraNumeroContrato(String registraNumeroContrato) {
    this.registraNumeroContrato = registraNumeroContrato;
}

public String getNumeroContrato() {
    return numeroContrato;
}

public void setNumeroContrato(String numeroContrato) {
    this.numeroContrato = numeroContrato;
}

public ValorDefinicion getDisciplinaOcdeId() {
    return disciplinaOcdeId;
}
```

```
}  
  
public void setDisciplinaOcdeId(ValorDefinicion disciplinaOcdeId) {  
    this.disciplinaOcdeId = disciplinaOcdeId;  
}
```

- Se mapea nuevos campos en **ProyectoDTO.java** de la nueva tabla

```
@Bind(name = "registraCodigoBPIN")  
private String registraCodigoBPIN;  
@Bind(name = "codigoBPIN")  
private String codigoBPIN;  
@Bind(name = "registraNumeroContrato")  
private String registraNumeroContrato;  
@Bind(name = "numeroContrato")  
private String numeroContrato;  
  
@Bind(name = "disciplinaOcdeId", returnType = ValorDefinicionDTO.class, type =  
ValorDefinicion.class)  
private ValorDefinicionDTO disciplinaOcdeId;  
  
public String getRegistraCodigoBPIN() {  
    return registraCodigoBPIN;  
}  
  
public void setRegistraCodigoBPIN(String registraCodigoBPIN) {  
    this.registraCodigoBPIN = registraCodigoBPIN;  
}  
  
public String getCodigoBPIN() {  
    return codigoBPIN;  
}  
  
public void setCodigoBPIN(String codigoBPIN) {  
    this.codigoBPIN = codigoBPIN;  
}  
  
public String getRegistraNumeroContrato() {  
    return registraNumeroContrato;  
}  
  
public void setRegistraNumeroContrato(String registraNumeroContrato) {  
    this.registraNumeroContrato = registraNumeroContrato;  
}  
  
public String getNumeroContrato() {  
    return numeroContrato;  
}  
  
public void setNumeroContrato(String numeroContrato) {  
    this.numeroContrato = numeroContrato;  
}  
  
public ValorDefinicionDTO getDisciplinaOcdeId() {  
    return disciplinaOcdeId;  
}  
  
public void setDisciplinaOcdeId(ValorDefinicionDTO disciplinaOcdeId) {  
    this.disciplinaOcdeId = disciplinaOcdeId;  
}
```



```
}

```

- Se agrega en pantalla de información general del proyecto, en la pestaña de Información Basica los campos de Código BPIN y Código de registro

Código BPIN

Para permitir controlar el registro del código BPIN (Banco de Proyectos de Inversión Nacional) se utilizó un componente **toggleSwitch** (interruptor de activación/desactivación). Cuando el usuario activa el interruptor, se muestra el campo de texto donde puede ingresar el código.

Explicación Detallada del Código

1. Campo toggleSwitch para Registrar el Código BPIN:

```
<p:toggleSwitch id="codigoBPINBool"
    value="#{appSessController.proyectoSelected.registraCodigoBPINSINO}"
    required="false">
    <f:passThroughAttribute
        name="aria-label"
        value="#{msg['formulario.estimulos.proyectos.codigoBPINBoolean.label.placeholder']}"
    />
    <p:ajax event="change"
        update="panelCodigoBPIN"

oncomplete='${'##{proyectoSessController.modulo}Form\\\\\\\\codigoBPINBool').focus();'
    />
</p:toggleSwitch>
```

- **p:toggleSwitch**: Este componente de PrimeFaces muestra un interruptor que permite seleccionar si se registra o no el código BPIN.
- **value**: El valor del interruptor está enlazado al atributo **registraCodigoBPINSINO** en **proyectoSelected** dentro del **appSessController**. Este atributo probablemente es un booleano o similar.
- **p:ajax**: Este evento ajax se dispara cada vez que cambia el valor del toggleSwitch:
 - **update="panelCodigoBPIN"**: Actualiza el **panelCodigoBPIN**, que contiene el campo para ingresar el código BPIN.
 - **oncomplete**: Ejecuta un script de JavaScript para volver a enfocar el **toggleSwitch** después de la actualización.

2. Campo para Ingresar el Código BPIN:

```
<h:panelGroup layout="block" id="panelCodigoBPIN">
```

```
<ui:fragment rendered="#{appSessController.proyectoSelected.registraCodigoBPINSINO}">
  <div class="col-lg-6">
    <div class="form-group">
      <p:outputLabel
value="#{msg['formulario.estimulos.proyectos.codigoBPIN.label']}"
      for="codigoBPIN"
      styleClass="col-lg-4 control-label"/>
      <div class="col-lg-8">
        <p:inputTextarea id="codigoBPIN"
          required="true"

value="#{appSessController.proyectoSelected.codigoBPIN}"

requiredMessage="#{msg['formulario.estimulos.proyectos.codigoBPIN.required']}"
          styleClass="form-control inputArea"

placeholder="#{msg['formulario.estimulos.proyectos.codigoBPIN.placeholder']}"
          counter="displayCodigoBPIN"          maxLength="30"
counterTemplate="{0} #{msg['sb.mensajes.caracteres.label']}"
          rows="1"
        />
        <h:outputText id="displayCodigoBPIN" style="font-style: italic" />
      </div>
    </div>
  </div>
</ui:fragment>
</h:panelGroup>
```

- **h:panelGroup**: Contenedor para el panel de entrada del código BPIN.
- **ui:fragment**: El fragmento se renderiza condicionalmente cuando **registraCodigoBPINSINO** es verdadero. Si el **toggleSwitch** está activado, este fragmento se muestra.
- **p:inputTextarea**: Este componente permite al usuario ingresar el código BPIN:
 - **required="true"**: Marca el campo como obligatorio.
 - **value**: Enlaza el valor del área de texto al atributo **codigoBPIN** en el **proyectoSelected**.
 - **counter y maxLength**: Límita el campo a 30 caracteres y muestra un contador en **displayCodigoBPIN**.

Número Contrato

Este código es similar al del registro del código BPIN, permite a los usuarios activar o desactivar la entrada del número de contrato usando un **toggleSwitch**. Cuando el usuario activa el interruptor, se muestra el campo de entrada del número de contrato, con validación de obligatoriedad y un contador de caracteres.

Explicación del Código

1. Campo toggleSwitch para Registrar el Número de Contrato:

```
<p:toggleSwitch id="numeroContratoBool"
    value="#{appSessController.proyectoSelected.registraNumeroContratoSINO}"
    required="false">
    <f:passThroughAttribute
        name="aria-label"

value="#{msg['formulario.estimulos.proyectos.numeroContratoBoolean.label.placeholder']}"
    />
    <p:ajax event="change"
        update="panelNumeroContrato"

oncomplete="$('##{projectoSessController.modulo}Form\\\\\\\\numeroContratoBool').focus();"
    />
</p:toggleSwitch>
```

- **p:toggleSwitch:** Este componente muestra un interruptor para decidir si se debe registrar el número de contrato.
- **value:** El estado del interruptor está enlazado a la propiedad **registraNumeroContratoSINO** en **proyectoSelected** dentro del **appSessController**. Este valor determina si se muestra el campo de número de contrato.
- **p:ajax:** Configura un evento ajax que:
 - **update="panelNumeroContrato":** Actualiza el **panelNumeroContrato**, que contiene el campo para ingresar el número de contrato, cada vez que cambia el estado del interruptor.
 - **oncomplete:** Reenfoca el interruptor después de la actualización.

2. Campo para Ingresar el Número de Contrato:

```
<h:panelGroup layout="block" id="panelNumeroContrato">
    <ui:fragment
        rendered="#{appSessController.proyectoSelected.registraNumeroContratoSINO}">
        <div class="col-lg-6">
            <div class="form-group">
                <p:outputLabel
                    value="#{msg['formulario.estimulos.proyectos.numeroContrato.label']}"
                    for="numeroContrato"
                    styleClass="col-lg-4 control-label"/>
                <div class="col-lg-8">
                    <p:inputTextarea id="numeroContrato"
                        required="true"

                    value="#{appSessController.proyectoSelected.numeroContrato}"

                    requiredMessage="#{msg['formulario.estimulos.proyectos.numeroContrato.required']}"
```

```

                                styleClass="form-control inputArea"

placeholder="#{msg['formulario.estimulos.proyectos.numeroContrato.placeholder']}"
                                counter="displayNumeroContrato"           maxLength="30"
counterTemplate="{0} #{msg['sb.mensajes.caracteres.label']}"
                                rows="1"
                                />
                                <h:outputText id="displayNumeroContrato" style="font-style: italic" />
                                </div>
                                </div>
                                </div>
                                </ui:fragment>
</h:panelGroup>

```

- **h:panelGroup**: Contenedor para el panel de entrada del número de contrato.
- **ui:fragment**: Renderiza el campo de texto solo si **registraNumeroContratoSINO** es verdadero.
- **p:inputTextarea**: Proporciona un área de texto para el número de contrato:
 - **required="true"**: Indica que el campo es obligatorio.
 - **value**: El valor está enlazado al atributo **numeroContrato** en **proyectoSelected**.
 - **counter y maxLength**: Establece un límite de 30 caracteres y muestra un contador de caracteres en **displayNumeroContrato**.

- Se agrega en pantalla de información general del proyecto, en la pestaña de Deduccion/Excencion el campo de Disciplina OCDE

Disciplina OCDE

Este campo permite al usuario seleccionar una disciplina OCDE de una lista filtrable y con estilo personalizado. La validación de obligatoriedad y el estado editable están controlados por los atributos y mensajes configurados.

Explicación del Código

1. Condición de Renderizado

```
<ui:fragment rendered="#{proyectoSessController.renderCampo('470', 'CAMPOSPRY')}">
```

- Este fragmento solo se renderiza si la condición **renderCampo('470', 'CAMPOSPRY')** en **proyectoSessController** es verdadera.

2. Contenedor del Campo de Disciplina OCDE

```
<h:panelGroup id="panelDisciplinasOCDE" layout="block">
```

- Agrupa los elementos del campo en un contenedor para poder actualizarlos o manipularlos como un bloque.

3. Etiqueta del Campo

```
<p:outputLabel value="#{msg['formulario.estimulos.proyecto.deducion.disciplinaOcde']}"  
styleClass="col-lg-2 control-label"  
for="disciplinaOcde-discOCDE"/>
```

- La etiqueta se enlaza con el menú desplegable **disciplinaOcde-discOCDE** y usa **msg** para mostrar el texto localizable.

4. Menú de Selección de Disciplina OCDE

```
<p:selectOneMenu id="disciplinaOcde-discOCDE"  
value="#{appSessController.proyectoSelected.disciplinaOcdeId}"  
required="true"  
  
requiredMessage="#{msg['formulario.estimulos.proyecto.deducion.disciplinaOcde.required']}"  
  
disabled="#{!proyectoConvController.renderRegistroProyecto()}"  
converter="valorDefinicionConverter1"  
style="width:100%;"  
autoWidth="false"  
panelStyle="width:180px"  
effect="fade"  
filter="true"  
filterMatchMode="contains">
```

- **value**: Está enlazado a **disciplinaOcdeId** dentro de **proyectoSelected** en **appSessController**, donde se almacena la disciplina seleccionada.
- **required="true"**: Hace que el campo sea obligatorio y muestra **requiredMessage** en caso de no completarlo.
- **disabled**: Controla si el campo es editable, basado en **renderRegistroProyecto()** en **proyectoConvController**.
- **converter="valorDefinicionConverter1"**: Usa un convertidor para manejar la conversión de los objetos seleccionados.

- **filter="true" y filterMatchMode="contains"**: Habilita el filtro para buscar opciones que contengan el texto ingresado.

5. Opciones del Menú

```
<f:selectItem itemLabel="#{msg['sb.selectOneMenu.registro']}" />
<f:selectItems value="#{proyectoSessController.listaDisciplinasOcde}"
               var="disciplinaOcde"
               itemDescription="#{disciplinaOcde.descripcionValorDefinicion}"
               itemLabel="#{disciplinaOcde.descripcionValorDefinicion}"
               itemValue="#{disciplinaOcde}"/>
```

- **f:selectItem**: Añade una opción por defecto al menú desplegable.
- **f:selectItems**: Utiliza la lista **listaDisciplinasOcde** en **proyectoSessController** para generar opciones. **itemLabel** y **itemDescription** muestran el valor de **descripcionValorDefinicion** para cada disciplina, mientras que **itemValue** representa la opción seleccionada.
- Validar obligatoriedad en el formulario al Validar y Enviar Proyecto

```
if (renderCampo("470", "CAMPOSPRY", mapaCampos) && proyecto.getDisciplinaOcdeId() == null) {
    descripcion = descripcion + "\n * Ingresar la disciplina OCDE en la pestaña " +
    proyecto.getConvocatoriaId().getTipoTramiteId().getDescripcionValorDefinicion();
}
```

Este código agrega un mensaje en la variable **descripcion** para recordar al usuario que debe ingresar la disciplina OCDE, pero solo si:

- El campo "Disciplina OCDE" debe mostrarse (**renderCampo(...) es true**).
- El campo "Disciplina OCDE" aún no ha sido completado (**proyecto.getDisciplinaOcdeId() == null**).

El mensaje también indica la pestaña específica en la que se debe ingresar este dato, basándose en el tipo de trámite de la convocatoria del proyecto.

- Internacionalización en Español e Inglés

```
##### ES
formulario.estimulos.proyectos.codigoBPINBoolean.label = ¿Registra Código BPIN?
```

```

formulario.estimulos.proyectos.codigoBPINBoolean.label.placeholder = Seleccione si registra
código BPIN
formulario.estimulos.proyectos.codigoBPIN.label = Código BPIN
formulario.estimulos.proyectos.codigoBPIN.placeholder = Ingrese Código BPIN
formulario.estimulos.proyectos.codigoBPIN.required = Debe ingresar el código
BPIN
formulario.estimulos.proyectos.numeroContratoBoolean.label = ¿Registra Número Contrato?
formulario.estimulos.proyectos.numeroContratoBoolean.placeholder = Seleccione si registra
número de contrato
formulario.estimulos.proyectos.numeroContrato.label = Número Contrato
formulario.estimulos.proyectos.numeroContrato.placeholder = Ingrese Número Contrato
formulario.estimulos.proyectos.numeroContrato.required = Debe ingresar el número
de contrato

formulario.estimulos.proyecto.deducccion.disciplinaOcde = Disciplina OCDE
formulario.estimulos.proyecto.deducccion.disciplinaOcde.required = Debe seleccionar la
disciplina OCDE al cual apunta su proyecto

##### EN
formulario.estimulos.proyectos.codigoBPINBoolean.label = Do you register BPIN
Code?
formulario.estimulos.proyectos.codigoBPINBoolean.label.placeholder = Select if you register
BPIN Code
formulario.estimulos.proyectos.codigoBPIN.label = BPIN Code
formulario.estimulos.proyectos.codigoBPIN.placeholder = Enter BPIN Code
formulario.estimulos.proyectos.codigoBPIN.required = BPIN code is required
formulario.estimulos.proyectos.numeroContratoBoolean.label = Do you register Contract
Number?
formulario.estimulos.proyectos.numeroContrato.label = Contract Number
formulario.estimulos.proyectos.numeroContratoBoolean.placeholder = Select if you register
Contract Number
formulario.estimulos.proyectos.numeroContrato.placeholder = Enter Contract Number
formulario.estimulos.proyectos.numeroContrato.required = Contract number is
required

formulario.estimulos.proyecto.deducccion.disciplinaOcde = Discipline OCDE
formulario.estimulos.proyecto.deducccion.disciplinaOcde.required = You must select the OECD
discipline to which your project points

```

Configurar Cantidad Máxima de Evaluaciones Asignadas por Evaluador en una Convocatoria.

Para este requerimiento se solicita implementar la funcionalidad de configurar por convocatoria la cantidad máxima de evaluaciones de proyectos que pueden ser asignadas a un evaluador en la etapa de evaluación.

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

- Se crea nuevo campo de max_proyectos_evaluador en la tabla sgp_convocatorias que permita indicar la cantidad máxima de evaluaciones de proyectos para ser asignados por evaluador
- - Se mapea campo en Entidad y DTO
- - Se agrega campo en pantalla de configuración de convocatorias
- - Se agregan validaciones al registrar una evaluación de proyectos, que permita indicar si el evaluador ya excedió el límite de proyectos máximos permitidos para ser asignados en la convocatoria. Creación de métodos de consulta de los proyectos asignados por convocatoria. Validación implementada en los módulos de proyectos y programas y beneficios tributarios
- - Se agrega internacionalización en español e inglés

Configurar cantidad máxima de evaluadores por proyecto

- creación de nuevo campo en la tabla SGP_CONVOCATORIAS

```
-- Add/modify columns
alter table SGP_CONVOCATORIAS add max_proyectos_evaluador NUMBER(2);
-- Add comments to the columns
comment on column SGP_CONVOCATORIAS.max_proyectos_evaluador
is 'Indica la cantidad máxima de proyectos para ser asignados por evaluador';
```

- Mapear nuevo campo en la entidad de la convocatoria (Convocatoria.java)

```
@Column(name = "MAX_PROYECTOS_EVALUADOR")
private Short maxProyectosEvaluador;

public Short getMaxProyectosEvaluador() {
    return maxProyectosEvaluador;
}

public void setMaxProyectosEvaluador(Short maxProyectosEvaluador) {
    this.maxProyectosEvaluador = maxProyectosEvaluador;
}
```

- Mapear nuevo campo en DTO de la convocatoria (ConvocatoriaDTO.java)

```
@Bind(name = "maxProyectosEvaluador")
private Short maxProyectosEvaluador;

public Short getMaxProyectosEvaluador() {
    return maxProyectosEvaluador;
}
```



```
public void setMaxProyectosEvaluador(Short maxProyectosEvaluador) {
    this.maxProyectosEvaluador = maxProyectosEvaluador;
}
```

Este código define una propiedad **maxProyectosEvaluador**, mapeada a la columna **MAX_PROYECTOS_EVALUADOR** en la base de datos, con métodos de acceso (**getMaxProyectosEvaluador**) y modificación (**setMaxProyectosEvaluador**). Esta propiedad es de tipo **Short** y se utiliza para almacenar el número máximo de proyectos que puede evaluar un evaluador.

- En la pantalla de configuración de la convocatoria, se crea campo de texto numerico para registrar el valor configurado

```
<!-- N° Máximo de Evaluadores por Proyecto -->
<div class="col-lg-6">
    <div class="form-group">
        <p:outputLabel
value="#{msg['gestion.configuracion.convocatorias.maxProyectosEvaluador.label']}"
                                styleClass="col-lg-6 control-label"
                                style="text-align: right"
                                for="maxEvaluadoresProyecto"/>

        <div class="col-lg-6">
            <p:inputNumber id="maxEvaluadoresProyecto"

value="#{convocatoriaSessController.selected.maxProyectosEvaluador}"
                                required="false"

requiredMessage="#{msg['gestion.configuracion.convocatorias.maxProyectosEvaluador.required']}"
                                />

            </div>
        </div>
    </div>
</div>
```

Este campo de configuración permite al usuario definir el número máximo de evaluadores por proyecto en la pantalla de configuración. Los controles de entrada limitan los valores a un máximo de dos dígitos y hasta 99, con validaciones y mensajes personalizables.

1. Etiqueta de Campo (p:outputLabel)

- **p:outputLabel** muestra una etiqueta con el texto de **maxProyectosEvaluador.label** obtenido del archivo de mensajes (**msg**).

- La etiqueta tiene un estilo **CSS** aplicado (**control-label** y alineación a la derecha) y está asociada al campo **maxEvaluadoresProyecto** mediante el atributo **for**.

2. Campo de Entrada Numérica (p:inputNumber)

- **p:inputNumber** permite al usuario ingresar solo valores numéricos.
- **value** está enlazado a **maxProyectosEvaluador** en **convocatoriaSessController.selected**, lo que significa que el número ingresado se almacena en la propiedad **maxProyectosEvaluador** de la entidad seleccionada.
- **maxlength="2"** limita el campo a un máximo de 2 dígitos.
- **maxValue="99"** establece el valor máximo permitido en 99.
- **required="false"** indica que el campo no es obligatorio, pero si se cambiara a **true**, el mensaje de error de validación vendría de **maxProyectosEvaluador.required** en el archivo de mensajes.

Validaciones al asignar evaluaciones en los proyectos

- Agregar lógica que permita validar que la cantidad de proyectos asignados al evaluador no excede la permitida por la convocatoria

Método validarProyectosAsignadosEvaluador

```
/**
 * Método que permite validar que la cantidad de proyectos asignados a un
 * evaluador, no excede el máximo permitido configurado por la convocatoria
 *
 * @param proyectoId
 * @throws SystemException
 */
private void validarProyectosAsignadosEvaluador(ProyectoEvaluacion evaluacion) throws
SystemException {

    Convocatoria convocatoria = evaluacion.getProyectoId().getConvocatoriaId();
    EvaluadorScienti evaluadorScienti = evaluacion.getEvaluadorScientiId();

    if (convocatoria != null
        && convocatoria.getConvocatoriaId() != null
        && convocatoria.getMaxProyectosEvaluador() != null
        && evaluadorScienti.getEvaluadorScientiId() != null) {
        Map<String, Object> filters = new HashMap<>();
        filters.put("convocatoriaId", convocatoria.getConvocatoriaId());
```

```
filters.put("evaluadorScientiId", evaluadorScienti.getEvaluadorScientiId());
List<ProyectoEvaluacion> proyectosEvaluadorList = this.proyectosEvaluacionList(-
1, -1, "proyectoId.proyectoId", "A", filters);
if (proyectosEvaluadorList != null && !proyectosEvaluadorList.isEmpty()) {
    if (proyectosEvaluadorList.contains(evaluacion)) {
        proyectosEvaluadorList.remove(evaluacion);
    }
    if (!proyectosEvaluadorList.isEmpty() && proyectosEvaluadorList.size()
    >= convocatoria.getMaxProyectosEvaluador()) {
        SystemException error = new
        SystemException("#{msg['gestion.proyectos.evaluacion.evaluadores.msgError.maxProyectosEvalua
        dor']}");
        error.getParameters().add(convocatoria.getMaxProyectosEvaluador().toString());
        throw error;
    }
}
```

Este método, **validarProyectosAsignadosEvaluador**, realiza una validación para asegurarse de que el número de proyectos asignados a un evaluador no supere el límite configurado en la convocatoria. Aquí está el desglose de cómo funciona el método:

1. Parámetros y Objetivo del Método

- Toma un parámetro **ProyectoEvaluacion** evaluacion, que contiene la información de la evaluación, incluido el proyecto y el evaluador.
- La finalidad del método es lanzar una excepción si el evaluador ya tiene asignados el número máximo de proyectos permitido en la convocatoria.

2. Obtención de Datos de Convocatoria y Evaluador

```
Convocatoria convocatoria = evaluacion.getProyectoId().getConvocatoriaId();
EvaluadorScienti evaluadorScienti = evaluacion.getEvaluadorScientiId();
```

- Se obtienen la convocatoria asociada al proyecto y el evaluador correspondiente a la evaluación.

3. Verificación de Datos Requeridos

```
if (convocatoria != null && convocatoria.getConvocatoriaId() != null
    && convocatoria.getMaxProyectosEvaluador() != null
    && evaluadorScienti.getEvaluadorScientiId() != null) {
```

- Comprueba que la convocatoria, el ID de la convocatoria, el límite de proyectos (**maxProyectosEvaluador**) y el ID del evaluador no sean nulos.

4. Filtrado de Proyectos ya Asignados al Evaluador

```
Map<String, Object> filters = new HashMap<>();
filters.put("convocatoriaId", convocatoria.getConvocatoriaId());
filters.put("evaluadorScientiId", evaluadorScienti.getEvaluadorScientiId());
List<ProyectoEvaluacion> proyectosEvaluadorList = this.proyectosEvaluacionList(-1, -1,
"projectId.projectoId", "A", filters);
```

- Utiliza un **Map** de filtros para obtener la lista de proyectos en los que el evaluador ya está asignado dentro de la convocatoria actual.
- Llama a **proyectosEvaluacionList** con estos filtros para recuperar los proyectos asignados al evaluador.

5. Remoción de la Evaluación Actual de la Lista

```
if (proyectosEvaluadorList.contains(evaluacion)) {
    proyectosEvaluadorList.remove(evaluacion);
}
```

- Si la lista de proyectos contiene la evaluación actual (cuando se edita un proyecto existente), la elimina para evitar contarla dos veces.

6. Validación del Límite Máximo de Proyectos

```
if (!proyectosEvaluadorList.isEmpty() && proyectosEvaluadorList.size() >=
convocatoria.getMaxProyectosEvaluador()) {
    SystemException error = new
SystemException("#{msg['gestion.proyectos.evaluacion.evaluadores.msgError.maxProyectosEvalua
dor']}");
    error.getParameters().add(convocatoria.getMaxProyectosEvaluador().toString());
    throw error;
}
```

- Comprueba si la lista de proyectos, excluyendo la evaluación actual, alcanza o supera el máximo permitido (**maxProyectosEvaluador**).
- Si se supera el límite, lanza una **SystemException** con un mensaje de error que incluye el número máximo de proyectos permitido.

La anterior validación fue implementada para proyectos/programas y para proyectos de beneficios tributarios en los métodos de guardar y editar evaluacion del proyecto,

- Internacionalizacion en Español e Ingles

```
#ES
gestion.configuracion.convocatorias.maxProyectosEvaluador.label          = N° Máximo de
Proyectos Asignados por Evaluador
gestion.configuracion.convocatorias.maxProyectosEvaluador.required      = Debe indicar la
cantidad máxima de proyectos aignados por evaluador

gestion.proyectos.evaluacion.evaluadores.msgError.cierreEvaluacion      = No puede asignar o
modificar evaluadores al proyecto ya que la evaluación se encuentra cerrada
gestion.proyectos.evaluacion.evaluadores.msgError.conflictoIntereses     = No puede asignar el
evaluador a este proyecto, ya que hay un conflicto de intereses de por medio
gestion.proyectos.evaluacion.evaluadores.msgError.maxProyectosEvaluador = El evaluador ya ha
alcanzado el límite máximo de {0} evaluaciones permitidas en esta convocatoria y no puede ser
asignado a este proyecto.

#EN
gestion.configuracion.convocatorias.maxProyectosEvaluador.label          = Maximum Number of
Projects Assigned per Evaluator
gestion.configuracion.convocatorias.maxProyectosEvaluador.required      = You must indicate
the maximum number of projects assigned per evaluator

gestion.proyectos.evaluacion.evaluadores.msgError.cierreEvaluacion      = You cannot assign
or modify evaluators to the project since the evaluation is closed
gestion.proyectos.evaluacion.evaluadores.msgError.conflictoIntereses     = You cannot assign
the evaluator to this project, as there is a conflict of interest involved
gestion.proyectos.evaluacion.evaluadores.msgError.maxProyectosEvaluador = The evaluator has
already reached the maximum limit of {0} evaluations allowed in this call and cannot be
assigned to this project.
```

Configurar Acceso Funcionarios Evaluación Proyectos por Convocatoria.

Para este requerimiento se solicitó la configuración de funcionarios que pueden acceder al módulo de evaluación por convocatoria. así mismo, permitir asignar que proyectos pueden o no acceder. En caso de que un funcionario sea configurado, pero no asignado un proyecto específico, podrá ver todos los proyectos de esa convocatoria.

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

- Se crea nueva pantalla de configuración de funcionarios que pueden acceder al módulo de evaluación por convocatoria. Así mismo, permitir asignar que proyectos pueden o no

acceder. En caso de que un funcionario sea configurado, pero no asignado un proyecto específico, podrá ver todos los proyectos de esa convocatoria

- Mapeo de Entidad y DTO de la nueva tabla SGP_CONVOCATORIAS_FUNCIONARIOS
- Mapeo de Entidad y DTO de la nueva tabla SGP_CONV_FUNCIONARIO_PROYECTOS
- Creación de métodos CRUD en ejb facade, ejb service y store manager como consulta en criteria builder para Convocatorias funcionarios
- Creación de métodos CRUD en ejb facade, ejb service y store manager como consulta en criteria builder para asignar proyectos de evaluación a funcionarios
- Creación de controlador de vista y de conversión para interactuar con la nueva pantalla de configuración
- Properties en español e inglés
- adición de la nueva pantalla al menú de configuración específico por convocatoria
- En el módulo de evaluación de proyectos y programas, se agrega lógica para mostrar aquellos proyectos asignados a un funcionario de Minciencias. Modificación de consulta en criteria builder para tener en cuenta la configuración de convocatorias funcionarios y asignación de proyectos. No aplica para usuarios del grupo SGPS_ADMINISTRADOR
- En el módulo de evaluación de beneficios tributarios se agrega lógica para mostrar aquellos proyectos asignados a un funcionario de Minciencias. Modificación de consulta en criteria builder para tener en cuenta la configuración de convocatorias funcionarios y asignación de proyectos. No aplica para usuarios del grupo SGPS_ADMINISTRADOR
- Se actualiza tema de primefaces, pasando de bootstrap a nova-light
- Se actualizan estilos CSS del aplicativo

Creación pantalla de Configuración de Funcionarios por Convocatoria

La pantalla de Configuración de funcionarios permite administrar los funcionarios asignados para la evaluación de proyectos en una convocatoria específica. Esta pantalla facilita asignar proyectos a evaluadores, gestionar el estado de cada proyecto asignado y buscar proyectos según los criterios de la convocatoria.

- Creación nuevo menú de configuración de funcionarios de evaluación por Convocatoria

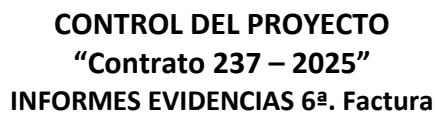
```
--funciones
insert into sb_funciones (FUNCION_ID, APLICACION, FUNCION, NOMBRE_FUNCION, TIPO_FUNCION, MENU,
OPCION, URL, URL_AYUDA, FUNCION_PADRE, ICONO, USUARIO_CREADOR, FECHA_CREACION,
USUARIO_MODIFICADOR, FECHA_MODIFICACION)
values (null, 'SIGP_WEB', 'GEST1340', 'Funcionarios de Evaluación por Convocatorias', 'F',
null, null, '/app/configuracion/convocatorias/funcionariosConvocatorias.xhtml', null,
'GEST1050', null, 'SIGP_WEB', to_date('04-06-2024', 'dd-mm-yyyy'), null, null);
```

```
--derechos
insert into sb_derechos_funciones (DERECHO_FUNCION_ID, APLICACION, FUNCION, GRUPO, ESTADO,
USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION)
values (null, 'SIGP_WEB', 'GEST1340', 'SGPS_ADMINISTRADOR', 'H', 'SIGP_WEB', to_date('04-06-
2024', 'dd-mm-yyyy'), null, null);
```

Tabla para asignacion de funcionarios por convocatoria

- Crear nueva tabla de configuración de grandes áreas por convocatoria (**SGP_CONVOCATORIAS_FUNCIONARIOS**) en la base de datos sigp

```
-- Create table
create table SGP_CONVOCATORIAS_FUNCIONARIOS
(
  convocatoria_funcionario_id NUMBER(15) not null,
  convocatoria_id            NUMBER(15) not null,
  usuario_id                 NUMBER(15) not null,
  usuario_creador            VARCHAR2(30),
  fecha_creacion              DATE,
  usuario_modificador         VARCHAR2(30),
  fecha_modificacion          DATE
)
tablespace DATOS
pctfree 10
initrans 1
maxtrans 255
storage
(
  initial 64K
  next 1M
  minextents 1
  maxextents unlimited
);
-- Add comments to the table
comment on table SGP_CONVOCATORIAS_FUNCIONARIOS
  is 'Almacena la informacion de los funcionarios que intervienen en una convocatoria';
-- Add comments to the columns
comment on column SGP_CONVOCATORIAS_FUNCIONARIOS.convocatoria_funcionario_id
  is 'Identificador llave primaria';
comment on column SGP_CONVOCATORIAS_FUNCIONARIOS.convocatoria_id
  is 'Identificador de la convocatoria';
comment on column SGP_CONVOCATORIAS_FUNCIONARIOS.usuario_id
  is 'Identificador del usuario del funcionario';
comment on column SGP_CONVOCATORIAS_FUNCIONARIOS.usuario_creador
  is 'Usuario creador del registro';
comment on column SGP_CONVOCATORIAS_FUNCIONARIOS.fecha_creacion
  is 'Fecha de creacion del registro';
comment on column SGP_CONVOCATORIAS_FUNCIONARIOS.usuario_modificador
  is 'Usuario modificador del registro';
comment on column SGP_CONVOCATORIAS_FUNCIONARIOS.fecha_modificacion
  is 'Fecha de modificacion del registro';
-- Create/Recreate indexes
create index SGPCONVFUNC_USUA_FK_I on SGP_CONVOCATORIAS_FUNCIONARIOS (USUARIO_ID)
```



```

tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
-- Create/Recreate primary, unique and foreign key constraints
alter table SGP_CONVOCATORIAS_FUNCIONARIOS
add constraint SGPCONVFUNC_PK primary key (CONVOCATORIA_FUNCIONARIO_ID)
using index
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
alter table SGP_CONVOCATORIAS_FUNCIONARIOS
add constraint SGPCONVFUNC_UK unique (CONVOCATORIA_ID, USUARIO_ID)
using index
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
alter table SGP_CONVOCATORIAS_FUNCIONARIOS
add constraint SGPCONVFUNC_USUA_FK foreign key (USUARIO_ID)
references HERCULES.SB_USUARIOS (USUARIO_ID);

```

- Crear secuencia (**SGP_CONV_FUNCIONARIO_ID_SEQ**) asociada a la tabla

[illegible]

- Crear nueva entidad de la tabla **SGP_CONVOCATORIAS_FUNCIONARIOS** en la base de datos sigp

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.sb.colciencias.gestion.business.domain;

import com.sb.colciencias.gestion.business.domain.security.Usuario;
import com.sb.colciencias.gestion.business.valueObjects.AppSessionVO;
import java.io.Serializable;
import java.util.Date;
import java.util.List;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

/**
 *
 * @author lsuarez
 */
@Entity
@Table(name = "SGP_CONVOCATORIAS_FUNCIONARIOS", catalog = "", schema = "SIGP_WEB")
@NamedQueries({
    @NamedQuery(name = "ConvocatoriaFuncionario.findAll", query = "SELECT c FROM ConvocatoriaFuncionario c")
    , @NamedQuery(name = "ConvocatoriaFuncionario.findByConvocatoriaFuncionarioId", query = "SELECT c FROM ConvocatoriaFuncionario c WHERE c.convocatoriaFuncionarioId = :convocatoriaFuncionarioId")
    , @NamedQuery(name = "ConvocatoriaFuncionario.findByConvocatoriaId", query = "SELECT c FROM ConvocatoriaFuncionario c WHERE c.convocatoriaId.convocatoriaId = :convocatoriaId")
    , @NamedQuery(name = "ConvocatoriaFuncionario.findByUsuarioId", query = "SELECT c FROM ConvocatoriaFuncionario c WHERE c.usuarioId.usuarioId = :usuarioId")
    , @NamedQuery(name = "ConvocatoriaFuncionario.findByUsuarioCreador", query = "SELECT c FROM ConvocatoriaFuncionario c WHERE c.usuarioCreador = :usuarioCreador")
    , @NamedQuery(name = "ConvocatoriaFuncionario.findByFechaCreacion", query = "SELECT c FROM ConvocatoriaFuncionario c WHERE c.fechaCreacion = :fechaCreacion")
    , @NamedQuery(name = "ConvocatoriaFuncionario.findByUsuarioModificador", query = "SELECT c FROM ConvocatoriaFuncionario c WHERE c.usuarioModificador = :usuarioModificador")
})

```

```
, @NamedQuery(name = "ConvocatoriaFuncionario.findByFechaModificacion", query = "SELECT c
FROM ConvocatoriaFuncionario c WHERE c.fechaModificacion = :fechaModificacion"))
public class ConvocatoriaFuncionario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"SGP_CONV_FUNCIONARIO_ID_SEQ")
    @SequenceGenerator(name = "SGP_CONV_FUNCIONARIO_ID_SEQ", sequenceName =
"SGP_CONV_FUNCIONARIO_ID_SEQ", allocationSize = 1)
    @Basic(optional = false)
    @NotNull
    @Column(name = "CONVOCATORIA_FUNCIONARIO_ID")
    private Long convocatoriaFuncionarioId;
    @Size(max = 30)
    @Column(name = "USUARIO_CREADOR")
    private String usuarioCreador;
    @Column(name = "FECHA_CREACION")
    @Temporal(TemporalType.TIMESTAMP)
    private Date fechaCreacion;
    @Size(max = 30)
    @Column(name = "USUARIO_MODIFICADOR")
    private String usuarioModificador;
    @Column(name = "FECHA_MODIFICACION")
    @Temporal(TemporalType.TIMESTAMP)
    private Date fechaModificacion;

    //Joins
    @JoinColumn(name = "CONVOCATORIA_ID", referencedColumnName = "CONVOCATORIA_ID")
    @ManyToOne
    private Convocatoria convocatoriaId;
    @JoinColumn(name = "USUARIO_ID", referencedColumnName = "USUARIO_ID")
    @ManyToOne
    private Usuario usuarioId;

    //Listas
    @OneToMany(cascade = {CascadeType.ALL}, mappedBy = "convocatoriaFuncionarioId", fetch =
FetchType.LAZY, orphanRemoval = true)
    private List<ConvocatoriaFuncionarioProyecto> proyectosFuncionarioList;

    //Auditoria
    @Transient
    private AppSessionVO appSessionVO;

    public ConvocatoriaFuncionario() {
    }

    public ConvocatoriaFuncionario(Long convocatoriaFuncionarioId) {
        this.convocatoriaFuncionarioId = convocatoriaFuncionarioId;
    }

    public Long getConvocatoriaFuncionarioId() {
        return convocatoriaFuncionarioId;
    }

    public void setConvocatoriaFuncionarioId(Long convocatoriaFuncionarioId) {
        this.convocatoriaFuncionarioId = convocatoriaFuncionarioId;
    }

    public String getUsuarioCreador() {
        return usuarioCreador;
    }
```

```
}

public void setUsuarioCreador(String usuarioCreador) {
    this.usuarioCreador = usuarioCreador;
}

public Date getFechaCreacion() {
    return fechaCreacion;
}

public void setFechaCreacion(Date fechaCreacion) {
    this.fechaCreacion = fechaCreacion;
}

public String getUsuarioModificador() {
    return usuarioModificador;
}

public void setUsuarioModificador(String usuarioModificador) {
    this.usuarioModificador = usuarioModificador;
}

public Date getFechaModificacion() {
    return fechaModificacion;
}

public void setFechaModificacion(Date fechaModificacion) {
    this.fechaModificacion = fechaModificacion;
}

public Convocatoria getConvocatoriaId() {
    return convocatoriaId;
}

public void setConvocatoriaId(Convocatoria convocatoriaId) {
    this.convocatoriaId = convocatoriaId;
}

public Usuario getUsuarioId() {
    return usuarioId;
}

public void setUsuarioId(Usuario usuarioId) {
    this.usuarioId = usuarioId;
}

public List<ConvocatoriaFuncionarioProyecto> getProyectosFuncionarioList() {
    return proyectosFuncionarioList;
}

public void setProyectosFuncionarioList(List<ConvocatoriaFuncionarioProyecto>
proyectosFuncionarioList) {
    this.proyectosFuncionarioList = proyectosFuncionarioList;
}

public AppSessionVO getAppSessionVO() {
    return appSessionVO;
}

public void setAppSessionVO(AppSessionVO appSessionVO) {
    this.appSessionVO = appSessionVO;
}
```

```

    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (convocatoriaFuncionarioId != null ? convocatoriaFuncionarioId.hashCode() :
0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof ConvocatoriaFuncionario)) {
            return false;
        }
        ConvocatoriaFuncionario other = (ConvocatoriaFuncionario) object;
        if ((this.convocatoriaFuncionarioId == null && other.convocatoriaFuncionarioId !=
null) || (this.convocatoriaFuncionarioId != null &&
!this.convocatoriaFuncionarioId.equals(other.convocatoriaFuncionarioId))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "com.sb.colciencias.gestion.business.domain.ConvocatoriaFuncionario[
convocatoriaFuncionarioId=" + convocatoriaFuncionarioId + " ]";
    }
}

```

- Crear DTO representativo de la Entidad

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.sb.colciencias.gestion.business.transfer;

import com.sb.colciencias.gestion.business.domain.*;
import com.sb.colciencias.gestion.business.domain.security.Usuario;
import com.sb.colciencias.gestion.business.util.annotation.Bind;
import com.sb.colciencias.gestion.business.valueObjects.AppSessionVO;
import java.io.Serializable;
import java.util.Date;
import java.util.List;

/**
 *
 * @author lsuarez
 */
public class ConvocatoriaFuncionarioDTO implements Serializable {

    private static final long serialVersionUID = 1L;

```

```

@Bind(name = "convocatoriaFuncionarioId")
private Long convocatoriaFuncionarioId;
@Bind(name = "usuarioCreador")
private String usuarioCreador;
@Bind(name = "fechaCreacion")
private Date fechaCreacion;
@Bind(name = "fechaModificacion")
private Date fechaModificacion;
@Bind(name = "usuarioModificador")
private String usuarioModificador;

//Joins
@Bind(name = "convocatoriaId", returnType = ConvocatoriaDTO.class, type =
Convocatoria.class)
private ConvocatoriaDTO convocatoriaId;
@Bind(name = "usuarioId", returnType = UsuarioDTO.class, type = Usuario.class)
private UsuarioDTO usuarioId;

//Listas
@Bind(name = "proyectosFuncionarioList", returnType =
ConvocatoriaFuncionarioProyectoDTO.class, type = ConvocatoriaFuncionarioProyecto.class)
private List<ConvocatoriaFuncionarioProyectoDTO> proyectosFuncionarioList;

//Auditoria
@Bind(name = "appSessionVO")
private AppSessionVO appSessionVO;

public ConvocatoriaFuncionarioDTO() {
}

public ConvocatoriaFuncionarioDTO(Long convocatoriaFuncionarioId) {
    this.convocatoriaFuncionarioId = convocatoriaFuncionarioId;
}

public Long getConvocatoriaFuncionarioId() {
    return convocatoriaFuncionarioId;
}

public void setConvocatoriaFuncionarioId(Long convocatoriaFuncionarioId) {
    this.convocatoriaFuncionarioId = convocatoriaFuncionarioId;
}

public String getUsuarioCreador() {
    return usuarioCreador;
}

public void setUsuarioCreador(String usuarioCreador) {
    this.usuarioCreador = usuarioCreador;
}

public Date getFechaCreacion() {
    return fechaCreacion;
}

public void setFechaCreacion(Date fechaCreacion) {
    this.fechaCreacion = fechaCreacion;
}

public String getUsuarioModificador() {
    return usuarioModificador;
}

```

```

public void setUsuarioModificador(String usuarioModificador) {
    this.usuarioModificador = usuarioModificador;
}

public Date getFechaModificacion() {
    return fechaModificacion;
}

public void setFechaModificacion(Date fechaModificacion) {
    this.fechaModificacion = fechaModificacion;
}

public ConvocatoriaDTO getConvocatoriaId() {
    return convocatoriaId;
}

public void setConvocatoriaId(ConvocatoriaDTO convocatoriaId) {
    this.convocatoriaId = convocatoriaId;
}

public UsuarioDTO getUsuarioId() {
    return usuarioId;
}

public void setUsuarioId(UsuarioDTO usuarioId) {
    this.usuarioId = usuarioId;
}

public List<ConvocatoriaFuncionarioProyectoDTO> getProyectosFuncionarioList() {
    return proyectosFuncionarioList;
}

public void setProyectosFuncionarioList(List<ConvocatoriaFuncionarioProyectoDTO>
proyectosFuncionarioList) {
    this.proyectosFuncionarioList = proyectosFuncionarioList;
}

public AppSessionVO getAppSessionVO() {
    return appSessionVO;
}

public void setAppSessionVO(AppSessionVO appSessionVO) {
    this.appSessionVO = appSessionVO;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (convocatoriaFuncionarioId != null ? convocatoriaFuncionarioId.hashCode() :
0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    if (!(object instanceof ConvocatoriaFuncionarioDTO)) {
        return false;
    }
    ConvocatoriaFuncionarioDTO other = (ConvocatoriaFuncionarioDTO) object;

```

```

        if ((this.convocatoriaFuncionarioId == null && other.convocatoriaFuncionarioId !=
null) || (this.convocatoriaFuncionarioId != null &&
!this.convocatoriaFuncionarioId.equals(other.convocatoriaFuncionarioId))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "com.sb.colciencias.gestion.business.transfer.ConvocatoriaFuncionarioDTO[
convocatoriaFuncionarioId=" + convocatoriaFuncionarioId + " ]";
    }
}

```

Tabla de asignación de proyectos por funcionario

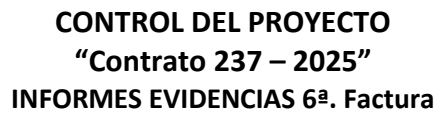
- Crear nueva tabla de configuración de proyectos por funcionario (**SGP_CONV_FUNCIONARIO_PROYECTOS**) en la base de datos sigp

```

-- Create table
create table SGP_CONV_FUNCIONARIO_PROYECTOS
(
    conv_funcionario_proyecto_id NUMBER(15) not null,
    convocatoria_funcionario_id  NUMBER(15) not null,
    proyecto_id                  NUMBER(15) not null,
    usuario_creador               VARCHAR2(30),
    fecha_creacion                DATE,
    usuario_modificador           VARCHAR2(30),
    fecha_modificacion            DATE
)
tablespace DATOS
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
-- Add comments to the table
comment on table SGP_CONV_FUNCIONARIO_PROYECTOS
is 'Almacena la informacion de los proyectos asignados a funcionarios que intervienen en
una convocatoria';
-- Add comments to the columns
comment on column SGP_CONV_FUNCIONARIO_PROYECTOS.conv_funcionario_proyecto_id
is 'Identificador llave primaria';
comment on column SGP_CONV_FUNCIONARIO_PROYECTOS.convocatoria_funcionario_id
is 'Identificador de la convocatoria funcionario';
comment on column SGP_CONV_FUNCIONARIO_PROYECTOS.proyecto_id
is 'Identificador del proyecto';

```

```
comment on column SGP_CONV_FUNCIONARIO_PROYECTOS.usuario_creador
is 'Usuario creador del registro';
comment on column SGP_CONV_FUNCIONARIO_PROYECTOS.fecha_creacion
is 'Fecha de creacion del registro';
comment on column SGP_CONV_FUNCIONARIO_PROYECTOS.usuario_modificador
is 'Usuario modificador del registro';
comment on column SGP_CONV_FUNCIONARIO_PROYECTOS.fecha_modificacion
is 'Fecha de modificacion del registro';
-- Create/Recreate indexes
create index SGPCONVFUNCPRY_SGPCONFUNC_FK_I on SGP_CONV_FUNCIONARIO_PROYECTOS
(CONVOCATORIA_FUNCIONARIO_ID)
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
create index SGPCONVFUNCPRY_SGPPRY_FK_I on SGP_CONV_FUNCIONARIO_PROYECTOS (PROYECTO_ID)
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
-- Create/Recreate primary, unique and foreign key constraints
alter table SGP_CONV_FUNCIONARIO_PROYECTOS
add constraint SGPCONVFUNCPRY_PK primary key (CONV_FUNCIONARIO_PROYECTO_ID)
using index
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
);
alter table SGP_CONV_FUNCIONARIO_PROYECTOS
add constraint SGPCONVFUNCPRY_UK unique (CONVOCATORIA_FUNCIONARIO_ID, PROYECTO_ID)
using index
tablespace INDICES
pctfree 10
initrans 2
maxtrans 255
storage
(
    initial 64K
    next 1M
    minextents 1
```

```

maxextents unlimited
);
alter table SGP_CONV_FUNCIONARIO_PROYECTOS
add constraint SGPCONVFUNCPRY_SGPCONFUNK_FK foreign key (CONVOCATORIA_FUNCIONARIO_ID)
references SGP_CONVOCATORIAS_FUNCIONARIOS (CONVOCATORIA_FUNCIONARIO_ID);
alter table SGP_CONV_FUNCIONARIO_PROYECTOS
add constraint SGPCONVFUNCPRY_SGPPRY_FK foreign key (PROYECTO_ID)
references SGP_PROYECTOS (PROYECTO_ID);

```

- Crear secuencia (**SGP_CONV_FUNC_PROYECTO_ID_SEQ**) asociada a la tabla

[illegible]

- Crear nueva entidad de la tabla **SGP_CONV_FUNCIONARIO_PROYECTOS** en la base de datos sigp

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.sb.colciencias.gestion.business.domain;

import com.sb.colciencias.gestion.business.valueObjects.AppSessionVO;
import java.io.Serializable;
import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

/**
 *
 * @author lsuarez

```

```

*/
@Entity
@Table(name = "SGP_CONV_FUNCIONARIO_PROYECTOS", catalog = "", schema = "SIGP_WEB")
@NamedQueries({
    @NamedQuery(name = "ConvocatoriaFuncionarioProyecto.findAll", query = "SELECT c FROM
ConvocatoriaFuncionarioProyecto c")
    , @NamedQuery(name =
"ConvocatoriaFuncionarioProyecto.findByConvocatoriaFuncionarioProyectoId", query = "SELECT c
FROM ConvocatoriaFuncionarioProyecto c WHERE c.convocatoriaFuncionarioProyectoId =
:convocatoriaFuncionarioProyectoId")
    , @NamedQuery(name = "ConvocatoriaFuncionarioProyecto.findByConvocatoriaFuncionarioId",
query = "SELECT c FROM ConvocatoriaFuncionarioProyecto c WHERE
c.convocatoriaFuncionarioId.convocatoriaFuncionarioId = :convocatoriaFuncionarioId")
    , @NamedQuery(name = "ConvocatoriaFuncionarioProyecto.findByProyectoId", query = "SELECT
c FROM ConvocatoriaFuncionarioProyecto c WHERE c.proyectoId.proyectoId = :proyectoId")
    , @NamedQuery(name = "ConvocatoriaFuncionarioProyecto.findByUsuarioCreador", query =
"SELECT c FROM ConvocatoriaFuncionarioProyecto c WHERE c.usuarioCreador = :usuarioCreador")
    , @NamedQuery(name = "ConvocatoriaFuncionarioProyecto.findByFechaCreacion", query =
"SELECT c FROM ConvocatoriaFuncionarioProyecto c WHERE c.fechaCreacion = :fechaCreacion")
    , @NamedQuery(name = "ConvocatoriaFuncionarioProyecto.findByUsuarioModificador", query =
"SELECT c FROM ConvocatoriaFuncionarioProyecto c WHERE c.usuarioModificador =
:usuarioModificador")
    , @NamedQuery(name = "ConvocatoriaFuncionarioProyecto.findByFechaModificacion", query =
"SELECT c FROM ConvocatoriaFuncionarioProyecto c WHERE c.fechaModificacion =
:fechaModificacion"))
public class ConvocatoriaFuncionarioProyecto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"SGP_CONV_FUNC_PROYECTO_ID_SEQ")
    @SequenceGenerator(name = "SGP_CONV_FUNC_PROYECTO_ID_SEQ", sequenceName =
"SGP_CONV_FUNC_PROYECTO_ID_SEQ", allocationSize = 1)
    @Basic(optional = false)
    @NotNull
    @Column(name = "CONV_FUNCIONARIO_PROYECTO_ID")
    private Long convocatoriaFuncionarioProyectoId;
    @Size(max = 30)
    @Column(name = "USUARIO_CREADOR")
    private String usuarioCreador;
    @Column(name = "FECHA_CREACION")
    @Temporal(TemporalType.TIMESTAMP)
    private Date fechaCreacion;
    @Size(max = 30)
    @Column(name = "USUARIO_MODIFICADOR")
    private String usuarioModificador;
    @Column(name = "FECHA_MODIFICACION")
    @Temporal(TemporalType.TIMESTAMP)
    private Date fechaModificacion;

    //Joins
    @JoinColumn(name = "CONVOCATORIA_FUNCIONARIO_ID", referencedColumnName =
"CONVOCATORIA_FUNCIONARIO_ID")
    @ManyToOne
    private ConvocatoriaFuncionario convocatoriaFuncionarioId;
    @JoinColumn(name = "PROYECTO_ID", referencedColumnName = "PROYECTO_ID")
    @ManyToOne
    private Proyecto proyectoId;

    //Auditoria
    @Transient

```

```
private AppSessionVO appSessionVO;

public ConvocatoriaFuncionarioProyecto() {
}

public ConvocatoriaFuncionarioProyecto(Long convocatoriaFuncionarioProyectoId) {
    this.convocatoriaFuncionarioProyectoId = convocatoriaFuncionarioProyectoId;
}

public Long getConvocatoriaFuncionarioProyectoId() {
    return convocatoriaFuncionarioProyectoId;
}

public void setConvocatoriaFuncionarioProyectoId(Long convocatoriaFuncionarioProyectoId)
{
    this.convocatoriaFuncionarioProyectoId = convocatoriaFuncionarioProyectoId;
}

public String getUsuarioCreador() {
    return usuarioCreador;
}

public void setUsuarioCreador(String usuarioCreador) {
    this.usuarioCreador = usuarioCreador;
}

public Date getFechaCreacion() {
    return fechaCreacion;
}

public void setFechaCreacion(Date fechaCreacion) {
    this.fechaCreacion = fechaCreacion;
}

public String getUsuarioModificador() {
    return usuarioModificador;
}

public void setUsuarioModificador(String usuarioModificador) {
    this.usuarioModificador = usuarioModificador;
}

public Date getFechaModificacion() {
    return fechaModificacion;
}

public void setFechaModificacion(Date fechaModificacion) {
    this.fechaModificacion = fechaModificacion;
}

public ConvocatoriaFuncionario getConvocatoriaFuncionarioId() {
    return convocatoriaFuncionarioId;
}

public void setConvocatoriaFuncionarioId(ConvocatoriaFuncionario
convocatoriaFuncionarioId) {
    this.convocatoriaFuncionarioId = convocatoriaFuncionarioId;
}

public Proyecto getProyectoId() {
    return proyectoId;
}
```

```

    }

    public void setProyectoId(Proyecto proyectoId) {
        this.proyectoId = proyectoId;
    }

    public AppSessionVO getAppSessionVO() {
        return appSessionVO;
    }

    public void setAppSessionVO(AppSessionVO appSessionVO) {
        this.appSessionVO = appSessionVO;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (convocatoriaFuncionarioProyectoId != null ?
convocatoriaFuncionarioProyectoId.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof ConvocatoriaFuncionarioProyecto)) {
            return false;
        }
        ConvocatoriaFuncionarioProyecto other = (ConvocatoriaFuncionarioProyecto) object;
        if ((this.convocatoriaFuncionarioProyectoId == null &&
other.convocatoriaFuncionarioProyectoId != null) || (this.convocatoriaFuncionarioProyectoId
!= null &&
!this.convocatoriaFuncionarioProyectoId.equals(other.convocatoriaFuncionarioProyectoId))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "com.sb.colciencias.gestion.business.domain.ConvocatoriaFuncionarioProyecto[
convocatoriaFuncionarioProyectoId=" + convocatoriaFuncionarioProyectoId + " ]";
    }
}

```

- Crear nuevo DTO representativo de la entidad

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.sb.colciencias.gestion.business.transfer;

import com.sb.colciencias.gestion.business.domain.*;
import com.sb.colciencias.gestion.business.util.annotation.Bind;

```

```
import com.sb.colciencias.gestion.business.valueObjects.AppSessionVO;
import java.io.Serializable;
import java.util.Date;

/**
 *
 * @author lsuarez
 */
public class ConvocatoriaFuncionarioProyectoDTO implements Serializable {

    private static final long serialVersionUID = 1L;

    @Bind(name = "convocatoriaFuncionarioProyectoId")
    private Long convocatoriaFuncionarioProyectoId;
    @Bind(name = "usuarioCreador")
    private String usuarioCreador;
    @Bind(name = "fechaCreacion")
    private Date fechaCreacion;
    @Bind(name = "fechaModificacion")
    private Date fechaModificacion;
    @Bind(name = "usuarioModificador")
    private String usuarioModificador;

    //Joins
    @Bind(name = "convocatoriaFuncionarioId", returnType = ConvocatoriaFuncionarioDTO.class,
type = ConvocatoriaFuncionario.class)
    private ConvocatoriaFuncionarioDTO convocatoriaFuncionarioId;
    @Bind(name = "proyectoId", returnType = ProyectoDTO.class, type = Proyecto.class)
    private ProyectoDTO proyectoId;

    //Auditoria
    @Bind(name = "appSessionVO")
    private AppSessionVO appSessionVO;

    public ConvocatoriaFuncionarioProyectoDTO() {
    }

    public ConvocatoriaFuncionarioProyectoDTO(Long convocatoriaFuncionarioProyectoId) {
        this.convocatoriaFuncionarioProyectoId = convocatoriaFuncionarioProyectoId;
    }

    public Long getConvocatoriaFuncionarioProyectoId() {
        return convocatoriaFuncionarioProyectoId;
    }

    public void setConvocatoriaFuncionarioProyectoId(Long convocatoriaFuncionarioProyectoId)
    {
        this.convocatoriaFuncionarioProyectoId = convocatoriaFuncionarioProyectoId;
    }

    public String getUsuarioCreador() {
        return usuarioCreador;
    }

    public void setUsuarioCreador(String usuarioCreador) {
        this.usuarioCreador = usuarioCreador;
    }

    public Date getFechaCreacion() {
        return fechaCreacion;
    }
}
```

```

public void setFechaCreacion(Date fechaCreacion) {
    this.fechaCreacion = fechaCreacion;
}

public String getUsuarioModificador() {
    return usuarioModificador;
}

public void setUsuarioModificador(String usuarioModificador) {
    this.usuarioModificador = usuarioModificador;
}

public Date getFechaModificacion() {
    return fechaModificacion;
}

public void setFechaModificacion(Date fechaModificacion) {
    this.fechaModificacion = fechaModificacion;
}

public ConvocatoriaFuncionarioDTO getConvocatoriaFuncionarioId() {
    return convocatoriaFuncionarioId;
}

public void setConvocatoriaFuncionarioId(ConvocatoriaFuncionarioDTO
convocatoriaFuncionarioId) {
    this.convocatoriaFuncionarioId = convocatoriaFuncionarioId;
}

public ProyectoDTO getProyectoId() {
    return proyectoId;
}

public void setProyectoId(ProyectoDTO proyectoId) {
    this.proyectoId = proyectoId;
}

public AppSessionVO getAppSessionVO() {
    return appSessionVO;
}

public void setAppSessionVO(AppSessionVO appSessionVO) {
    this.appSessionVO = appSessionVO;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (convocatoriaFuncionarioProyectoId != null ?
convocatoriaFuncionarioProyectoId.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    if (!(object instanceof ConvocatoriaFuncionarioProyectoDTO)) {
        return false;
    }
}

```

```
ConvocatoriaFuncionarioProyectoDTO other = (ConvocatoriaFuncionarioProyectoDTO)
object;
if ((this.convocatoriaFuncionarioProyectoId == null &&
other.convocatoriaFuncionarioProyectoId != null) || (this.convocatoriaFuncionarioProyectoId
!= null &&
!this.convocatoriaFuncionarioProyectoId.equals(other.convocatoriaFuncionarioProyectoId))) {
    return false;
}
return true;
}

@Override
public String toString() {
    return
"com.sb.colciencias.gestion.business.transfer.ConvocatoriaFuncionarioProyectoDTO["
convocatoriaFuncionarioProyectoId=" + convocatoriaFuncionarioProyectoId + " ]";
}
}
```

- Adición del menú **"Funcionarios por Convocatoria"** en la plantilla del menú del aplicativo de gestión

```
<!-- Funcionario Evaluación -->
<p:menuitem value="#{msg['gestion.configuracion.convocatoria.funcionarios.button']}"
            icon="fa fa-users"
outcome="/app/configuracion/convocatorias/funcionariosConvocatorias.xhtml"
            rendered="#{SeguridadSessController.mapaMenu.containsKey('GEST1340')}"
/>
```

El botón "Funcionarios Evaluación" permite acceder a la configuración de los funcionarios de evaluación para una convocatoria en particular. Este botón solo se muestra bajo ciertas condiciones, lo cual ayuda a controlar el acceso según el tipo de convocatoria y los permisos de usuario.

Este botón, representado por el componente **<p:menuitem>**, está diseñado para redirigir al usuario a la página **funcionariosConvocatorias.xhtml** cuando se selecciona. A continuación, se explica el código:

Explicación de los Atributos

5. Etiqueta del Botón:

- **value="#{msg['gestion.configuracion.convocatoria.funcionarios.button']}"**
- Este atributo **value** asigna el texto que se muestra en el botón. El texto se obtiene de un archivo de mensajes (por ejemplo, un archivo de propiedades) mediante la

clave **gestion.configuracion.convocatoria.funcionarios.button**, permitiendo soporte para múltiples idiomas.

6. Ícono:

- **icon="fa fa- users"**
- Asigna un icono al botón utilizando la clase **fa fa- users** de **FontAwesome**. Esto agrega un ícono de cubos al botón, visualmente representativo de las "funcionarios evaluación".

7. Redirección a una Página:

- **outcome="/app/configuracion/convocatorias/funcionariosConvocatorias.xhtml"**
- Este atributo **outcome** define la página de destino a la que se redirige el usuario al hacer clic en el botón, en este caso, a **funcionariosConvocatorias.xhtml**, que contiene la configuración de los funcionarios de evaluación de la convocatoria.

8. Condición para Renderizar el Botón:

- **rendered="#{SeguridadSessController.mapaMenu.containsKey('GEST1340')}"**
 - Este atributo **rendered** establece las condiciones bajo las cuales el botón será visible para el usuario. La condición evalúa un criterio:
 - **Permiso de Usuario:** El usuario debe tener acceso al menú identificado por la clave '**GEST1340**' en el mapaMenu del controlador de sesión de seguridad (**SeguridadSessController**). Esto permite que solo los usuarios autorizados puedan ver el botón.
- Creación Controlador de Vista **ConvocatoriaGranAreaViewController.java**, encargado de mostrar la información (listado de registros de TRLs) de convocatorias.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.sb.colciencias.gestion.web.controller.configuracion.view;
```



```
import
com.sb.colciencias.gestion.business.facade.beneficiosTributarios.BeneficiosServiceFacade;
import com.sb.colciencias.gestion.business.facade.convocatoria.ConvocatoriaServiceFacade;
import com.sb.colciencias.gestion.business.facade.proyectos.ProyectosServiceFacade;
import com.sb.colciencias.gestion.business.transfer.ConvocatoriaDTO;
import com.sb.colciencias.gestion.business.transfer.ConvocatoriaFuncionarioDTO;
import com.sb.colciencias.gestion.business.transfer.ConvocatoriaFuncionarioProyectoDTO;
import com.sb.colciencias.gestion.business.transfer.ProyectoDTO;
import com.sb.colciencias.gestion.business.transfer.UsuarioDTO;
import com.sb.colciencias.gestion.business.transfer.ValorDefinicionDTO;
import com.sb.colciencias.gestion.web.controller.application.ConfigApplController;
import com.sb.colciencias.gestion.web.session.ColcienciasSessController;
import com.sb.standard.web.controller.session.GenericSessionController;
import java.io.Serializable;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.ejb.EJB;
import javax.inject.Inject;
import javax.inject.Named;
import javax.annotation.PostConstruct;
import org.omnifaces.cdi.ViewScoped;
import org.primefaces.model.LazyDataModel;
import org.primefaces.model.SortOrder;
import org.primefaces.model.FilterMeta;
import java.util.TreeMap;

/**
 *
 * @author lsuarez
 */
@Named(value = "convocatoriaFuncionarioViewController")
@ViewScoped
public class ConvocatoriaFuncionarioViewController extends
GenericSessionController<ConvocatoriaFuncionarioDTO> implements Serializable {

    @EJB
    private ConvocatoriaServiceFacade convocatoriaServiceFacade;
    @EJB
    private BeneficiosServiceFacade beneficiosServiceFacade;
    @EJB
    private ProyectosServiceFacade proyectosServiceFacade;
    @Inject
    private ColcienciasSessController colcienciasSessController;
    @Inject
    private ConfigApplController configApplController;

    private LazyDataModel<ConvocatoriaFuncionarioDTO> model;
    private List<ProyectoDTO> selectedProyectos;
    private ConvocatoriaDTO convocatoriaSelected;
    private List<UsuarioDTO> usuariosList;
    //
    ConvocatoriaFuncionarioDTO convocatoriaFuncionarioSelected;
    //
    private List<ProyectoDTO> fullProyectos;
    private List<ConvocatoriaFuncionarioProyectoDTO> listaConvocatoriaFuncionarioProyectos;
    private List<ConvocatoriaFuncionarioProyectoDTO>
listaConvocatoriaFuncionarioProyectosFilter;
    //
    private Map<String, ValorDefinicionDTO> listaEstados;
```

```

private static final org.apache.log4j.Logger LOG =
org.apache.log4j.Logger.getLogger(ConvocatoriaFuncionarioViewController.class);

public ConvocatoriaFuncionarioViewController() {
    super(ConvocatoriaFuncionarioDTO.class);
    convocatoriaSelected = new ConvocatoriaDTO();
}

@PostConstruct
public void cargarListas() {

    Map<String, Object> mapReferencias = (Map<String, Object>)
colcienciasSessController.getUtilObject();
    convocatoriaSelected = (ConvocatoriaDTO) mapReferencias.get("convocatoriaSelected");

    listaEstados = configApplController.getMapValorDefinicion().get("ESTADO");

    this.model = new LazyDataModel<ConvocatoriaFuncionarioDTO>() {

        @Override
        public List<ConvocatoriaFuncionarioDTO> load(int first, int pageSize, String
sortField, SortOrder sortOrder, Map<String, FilterMeta> filterBy) {
            Map<String, Object> filters = new TreeMap<>();
            filterBy.forEach((k, v) -> {
                if (k.equals("globalFilter")) {
                    if (v.getFilterValue() != null &&
!v.getFilterValue().toString().isEmpty()) {
                        filters.put(k, v.getFilterValue());
                    }
                }
            });

            try {
                //Filtro por Convocatoria ID
                filters.put("convocatoriaId", convocatoriaSelected.getConvocatoriaId());

                int count =
convocatoriaServiceFacade.convocatoriaFuncionariosListCount(filters);
                if (count > 0) {
                    list = convocatoriaServiceFacade.convocatoriaFuncionariosList(first,
pageSize, sortField, sortOrder.equals(SortOrder.ASCENDING) ? "A" : "D", filters);
                }
                model.setRowCount(count);
            } catch (Exception ex) {
                LOG.error("Error al cargar el listado de funcionarios por convocatoria ",
ex);
            }
            return list;
        }

        @Override
        public Object getRowKey(ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO) {
            return convocatoriaFuncionarioDTO.getConvocatoriaFuncionarioId();
        }

        @Override
        public ConvocatoriaFuncionarioDTO getRowData(String rowKey) {

            try {

```

```
//Se consulta en la base de datos para mantener independiente el objeto
que trae en la lista principal y no lo actualice por referencia
ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
convocatoriaServiceFacade.queryConvocatoriaFuncionario(Long.parseLong(rowKey));
return convocatoriaFuncionarioDTO;
} catch (Exception ex) {
LOG.error("no se pudo cargar el funcionario convocatoria ( id => " +
rowKey + ")", ex);
}
return null;
}
};
}

@Override
public LazyDataModel<ConvocatoriaFuncionarioDTO> getModel() {
return model;
}

@Override
public void setModel(LazyDataModel<ConvocatoriaFuncionarioDTO> model) {
this.model = model;
}

public ConfigApplController getConfigApplController() {
return configApplController;
}

public void setConfigApplController(ConfigApplController configApplController) {
this.configApplController = configApplController;
}

public ConvocatoriaDTO getConvocatoriaSelected() {
return convocatoriaSelected;
}

public void setConvocatoriaSelected(ConvocatoriaDTO convocatoriaSelected) {
this.convocatoriaSelected = convocatoriaSelected;
}

public List<UsuarioDTO> getUsuariosList() {
return usuariosList;
}

public void setUsuariosList(List<UsuarioDTO> usuariosList) {
this.usuariosList = usuariosList;
}

public Map<String, ValorDefinicionDTO> getListaEstados() {
return listaEstados;
}

public void setListaEstados(Map<String, ValorDefinicionDTO> listaEstados) {
this.listaEstados = listaEstados;
}

public List<ProyectoDTO> getFullProyectos() {
return fullProyectos;
}

public void setFullProyectos(List<ProyectoDTO> fullProyectos) {
```

```

        this.fullProyectos = fullProyectos;
    }

    public List<ConvocatoriaFuncionarioProyectoDTO>
getListaConvocatoriaFuncionarioProyectos() {
        return listaConvocatoriaFuncionarioProyectos;
    }

    public void
setListaConvocatoriaFuncionarioProyectos(List<ConvocatoriaFuncionarioProyectoDTO>
listaConvocatoriaFuncionarioProyectos) {
        this.listaConvocatoriaFuncionarioProyectos = listaConvocatoriaFuncionarioProyectos;
    }

    public List<ConvocatoriaFuncionarioProyectoDTO>
getListaConvocatoriaFuncionarioProyectosFilter() {
        return listaConvocatoriaFuncionarioProyectosFilter;
    }

    public void
setListaConvocatoriaFuncionarioProyectosFilter(List<ConvocatoriaFuncionarioProyectoDTO>
listaConvocatoriaFuncionarioProyectosFilter) {
        this.listaConvocatoriaFuncionarioProyectosFilter
        =
        listaConvocatoriaFuncionarioProyectosFilter;
    }

    public List<ProyectoDTO> getSelectedProyectos() {
        return selectedProyectos;
    }

    public void setSelectedProyectos(List<ProyectoDTO> selectedProyectos) {
        this.selectedProyectos = selectedProyectos;
    }

    public ConvocatoriaFuncionarioDTO getConvocatoriaFuncionarioSelected() {
        return convocatoriaFuncionarioSelected;
    }

    public void setConvocatoriaFuncionarioSelected(ConvocatoriaFuncionarioDTO
convocatoriaFuncionarioSelected) {
        this.convocatoriaFuncionarioSelected = convocatoriaFuncionarioSelected;
    }

    public List<ProyectoDTO> buscarProyectos(String query) {
        try {
            if (query != null && !query.isEmpty()) {
                // System.out.println("entra a buscarProyectos");

                Boolean isConvEstimulos = convocatoriaSelected.getConvEstimulos() != null &&
convocatoriaSelected.getConvEstimulos().equals("S");

                // Implementa la lógica para buscar proyectos basados en la consulta
                List<Long> estadosFiltro = Arrays.asList(
                    listaEstados.get("REVISADO").getValorDefinicionId(),
                    listaEstados.get("EVALUADO").getValorDefinicionId(),
                    listaEstados.get("SEGUIMIENTO").getValorDefinicionId(),
                    listaEstados.get("NEGADO").getValorDefinicionId(),
                    listaEstados.get("APROBADO").getValorDefinicionId()
                );

                ProyectoDTO proyectoFilter = new ProyectoDTO();
            }
        }
    }

```

```
        proyectoFilter.setConvocatoriaId(convocatoriaSelected);

        Map<String, Object> filters = new HashMap();
        filters.put("query", query);

        int count = 0;
        if (isConvEstimulos) {
            // Proyectos de Estimulos Tributarios
            count = beneficiosServiceFacade.countProyectos(filters, estadosFiltro,
proyectoFilter, true);
            // System.out.println("fullProyectos ESTIMULOS COUNT: " + count);
            if (count > 0) {
                return beneficiosServiceFacade.obtenerTodosProyectos(-1, -1,
"proyectoId", "A", filters, estadosFiltro, proyectoFilter);
            }
        } else {
            // Proyectos y Programas
            count = proyectosServiceFacade.countProyectos(filters, estadosFiltro,
proyectoFilter, false);
            // System.out.println("fullProyectos PROYECTOS COUNT: " + count);
            if (count > 0) {
                return proyectosServiceFacade.obtenerTodosProyectos(-1, -1,
"proyectoId", "A", filters, estadosFiltro, proyectoFilter);
            }
        }
    }

    } catch (Exception ex) {
        LOG.error("Error al buscar los proyectos por autocomplete ", ex);
    }
    return null;
}
}
```

El controlador **ConvocatoriaFuncionarioViewController** maneja la lógica de carga, filtrado, y búsqueda de proyectos y evaluadores, utilizando PrimeFaces y componentes de Omnifaces. Este controlador también se comunica con los *facades* de servicios que interactúan con la base de datos para obtener, filtrar, y manipular datos de convocatorias, funcionarios y proyectos.

Funciones Principales

1. Carga de Datos en la Inicialización

El método **cargarListas()** se ejecuta en la fase **@PostConstruct**, cargando inicialmente los datos requeridos:

- **Convocatoria seleccionada:** Obtiene la convocatoria actual de la sesión.
- **Estados:** Carga el mapa de estados de proyecto configurado para la aplicación.

La carga de datos se realiza en un modelo de datos **LazyDataModel**, que optimiza la visualización al cargar solo los registros necesarios para cada página.

2. Modelo de Datos Diferido (LazyDataModel)

Se utiliza un modelo de datos diferido (**LazyDataModel<ConvocatoriaFuncionarioDTO>**) para gestionar la paginación y carga eficiente de funcionarios de evaluación por convocatoria:

- **Método load:** Este método se invoca para cargar una página de datos a partir de filtros y ordenamientos definidos.
- **Filtros:** Se define un filtro por **convocatoriald** y **globalFilter**, lo que permite limitar la lista de funcionarios a los de una convocatoria específica y aplicar un filtro global a los campos.

3. Selección y Gestión de Proyectos

La pantalla permite buscar y seleccionar proyectos mediante el método **buscarProyectos**, que permite:

- **Filtro de Estado:** Limita los proyectos según sus estados relevantes (Revisado, Evaluado, Seguimiento, Negado, Aprobado).
- **Condición de Tipo de Convocatoria:** Si la convocatoria es de tipo estímulos tributarios, la búsqueda se realiza mediante el servicio **BeneficiosServiceFacade**; en caso contrario, se usa **ProyectosServiceFacade**.
- **Parámetros de búsqueda:** Filtra proyectos con base en una palabra clave (query) ingresada por el usuario.

4. Asignación de Proyectos a Evaluadores

La asignación de proyectos se controla mediante las listas **selectedProyectos** y **fullProyectos**. Al seleccionar uno o más proyectos para un funcionario, el sistema actualiza estos elementos en la base de datos.

Servicios Inyectados

Los *facades* de servicio (**ConvocatoriaServiceFacade**, **BeneficiosServiceFacade**, y **ProyectosServiceFacade**) proporcionan la conexión con la capa de negocio y base de datos, gestionando las operaciones **CRUD** necesarias para la pantalla. El uso de inyección de dependencias (**@EJB** y **@Inject**) permite que el controlador acceda a estos servicios sin necesidad de instanciarlos directamente.

Registro de Errores

El controlador emplea **LOG.error** para registrar cualquier excepción o error durante la carga de datos, búsqueda, y asignación de proyectos. Esto facilita la detección y análisis de problemas en tiempo de ejecución.

Notas Adicionales:

- **Seguridad y Sesión:** El controlador usa **ColcienciasSessController** para obtener datos de la sesión actual, asegurando que la información de la convocatoria y del usuario esté disponible durante la configuración.
- **Estados del Proyecto:** La variable **listaEstados** mantiene un mapeo de estados que se usan para filtrar proyectos de acuerdo con las definiciones de la aplicación.
- Creación Controlador de Conversación **ConvocatoriaFuncionarioConvController.java**, el cual permite la comunicación e interacción de la pantalla con el usuario y funcionalidades de **CRUD** (Create, Read, Update y Delete).

```
/**
 *
 * @author lsuarez
 */
@Named(value = "convocatoriaFuncionarioConvController")
@ConversationScoped
public class ConvocatoriaFuncionarioConvController extends
GenericConversationController<ConvocatoriaFuncionarioDTO> implements Serializable {

    @EJB
    private ConvocatoriaServiceFacade convocatoriaServiceFacade;
    @EJB
    private AdministracionServiceFacade administracionServiceFacade;
    @Inject
    private ConvocatoriaFuncionarioViewController convocatoriaFuncionarioViewController;
    @Inject
    private SeguridadSessController seguridadSessController;
    @Inject
    private ColcienciasSessController colcienciasSessController;

    private final String programa = "GEST1340";

    private static final org.apache.log4j.Logger LOG =
org.apache.log4j.Logger.getLogger(ConvocatoriaFuncionarioConvController.class);

    public ConvocatoriaFuncionarioConvController() {
        this.selected = new ConvocatoriaFuncionarioDTO();
    }
}
```

```

@Override
public void guardar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
convocatoriaFuncionarioViewController.getSelected();

convocatoriaFuncionarioDTO.setUsuarioCreador(seguridadSessController.getSelected().getUsuario());

        convocatoriaFuncionarioDTO.setFechaCreacion(new Date());

convocatoriaFuncionarioDTO.setConvocatoriaId(convocatoriaFuncionarioViewController.getConvocatoriaSelected());
        //Auditoria
        AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
        appSessionVO.setPrograma(programa);
        convocatoriaFuncionarioDTO.setAppSessionVO(appSessionVO);
        convocatoriaFuncionarioDTO
convocatoriaServiceFacade.guardarConvocatoriaFuncionario(convocatoriaFuncionarioDTO);
        guardarProyectosAsignados(convocatoriaFuncionarioDTO);

        convocatoriaFuncionarioViewController.setSelected(new
ConvocatoriaFuncionarioDTO());
        convocatoriaFuncionarioViewController.cargarListas();
        //se dehabilita el boton de editar / eliminar ya que no habra nada seleccionado
despues de la operación
        this.onRowUnselect(null);
        this.obtenerPestana = 0;
    } catch (SystemException ex) {
        message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
        LOG.error("Error al guardar el funcionario por convocatoria ", ex);
    } catch (Exception ex) {
        message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
        LOG.error("Error al guardar el funcionario por convocatoria ", ex);
    } finally {
        FacesContext.getCurrentInstance().addMessage(null, message);
    }
}

@Override
public void eliminar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
convocatoriaFuncionarioViewController.getSelected();
        //Auditoria
        AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
        appSessionVO.setPrograma(programa);
        convocatoriaFuncionarioDTO.setAppSessionVO(appSessionVO);

convocatoriaServiceFacade.eliminarConvocatoriaFuncionario(convocatoriaFuncionarioDTO);

        convocatoriaFuncionarioViewController.setSelected(new
ConvocatoriaFuncionarioDTO());
    }
}

```



```

        convocatoriaFuncionarioViewController.cargarListas();
        this.onRowUnselect(null);
    } catch (Exception ex) {
        message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
        LOG.error("Error al eliminar el funcionario de la Convocatoria ", ex);
    } finally {
        FacesContext.getCurrentInstance().addMessage(null, message);
    }
}

@Override
public void editar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
convocatoriaFuncionarioViewController.getSelected();

convocatoriaFuncionarioDTO.setUsuarioModificador(seguridadSessController.getSelected().getUs
uario());
        convocatoriaFuncionarioDTO.setFechaModificacion(new Date());

convocatoriaFuncionarioDTO.setConvocatoriaId(convocatoriaFuncionarioViewController.getConvoc
atoriaSelected());
        //Auditoria
        AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
        appSessionVO.setProgram(programa);
        convocatoriaFuncionarioDTO.setAppSessionVO(appSessionVO);

convocatoriaServiceFacade.editarConvocatoriaFuncionario(convocatoriaFuncionarioDTO);
        guardarProyectosAsignados(convocatoriaFuncionarioDTO);

        convocatoriaFuncionarioViewController.setSelected(new
ConvocatoriaFuncionarioDTO());
        convocatoriaFuncionarioViewController.cargarListas();
        //se deshabilita el boton de editar / eliminar ya que no habra nada seleccionado
despues de la operación
        this.onRowUnselect(null);
        this.obtenerPestana = 0;
    } catch (SystemException ex) {
        message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
        LOG.error("Error al editar el funcionario por convocatoria ", ex);
    } catch (Exception ex) {
        message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
        LOG.error("Error al editar el funcionario de la Convocatoria ", ex);
    } finally {
        FacesContext.getCurrentInstance().addMessage(null, message);
    }
}

@Override
public void buscar(ActionEvent ae) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
}

```

```
@Override
public void mostrar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    //Obtengo la tabla+
    String modulo = convocatoriaFuncionarioViewController.getModulo();
    DataTable dataTable = (DataTable)
FacesContext.getCurrentInstance().getViewRoot().findComponent(modulo + "Form:" + modulo +
"TabList:" + modulo + "Table");
    //limpio los filtros
    dataTable.setFilterBy(null);
    //Se actualiza el formulario que contiene la tabla
    PrimeFaces.current().ajax().update(modulo + "Form");
    //Se muestra el growl con el mensaje
    FacesContext.getCurrentInstance().addMessage(null, message);

    convocatoriaFuncionarioViewController.cargarListas();
    this.onRowUnselect(null);
}

@Override
public void goNuevo(ActionEvent ae) {
    convocatoriaFuncionarioViewController.setSelected(new ConvocatoriaFuncionarioDTO());
    convocatoriaFuncionarioViewController.modaGuardar();
    this.obtenerPestana = 1;
}

@Override
public void goEditar(ActionEvent ae) {
    convocatoriaFuncionarioViewController.modaEditar();

    this.completeUsers(convocatoriaFuncionarioViewController.getSelected().getUsuarioId().getDes
cripcionUsuario());
    this.obtenerPestana = 1;
}

@Override
public void goBuscar(int i) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
}

@Override
public void goEliminar() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
}

@Override
public void onRowSelect(SelectEvent event) {
    super.onRowSelect(event);
    convocatoriaFuncionarioViewController.setSelected(this.selected);
    cargarProyectos();
    convocatoriaFuncionarioViewController.setActivaEditar(true);
    convocatoriaFuncionarioViewController.modaEditar();
}

@Override
public void onRowUnselect(UnselectEvent event) {
    super.onRowUnselect(event);
}
```

```
convocatoriaFuncionarioViewController.setSelected(new ConvocatoriaFuncionarioDTO());
convocatoriaFuncionarioViewController.setActivaEditar(false);
convocatoriaFuncionarioViewController.modaGuardar();
selected = null;
}

public void volverConvocatorias() {
    Map<String, Object> mapReferencias = new HashMap<>();
    colcienciasSessController.setUtilObject(mapReferencias);

    this.redirectPage("app/configuracion/convocatorias/cortesConvocatorias.xhtml");
}

/**
 * Metodo autocompletar que carga el listado de usuarios internos del
 * sistema sigp
 *
 * @param query
 * @return
 */
public List<UsuarioDTO> completeUsers(String query) {
    List<UsuarioDTO> usuarios = new ArrayList<>();
    Map<String, Object> filters = new HashMap<>();
    filters.put("query", query);
    filters.put("atributo5", "I");
    filters.put("vigente", "S");
    try {
        usuarios = administracionServiceFacade.usuariosList(-1, -1, "descripcionUsuario",
"A", filters);
        convocatoriaFuncionarioViewController.setUsuariosList(usuarios);
    } catch (Exception ex) {
        LOG.error("Error al consultar el autocompletar de usuarios ", ex);
    }
    return usuarios;
}

public void cargarProyectosAsignados(ConvocatoriaFuncionarioDTO
convocatoriaFuncionarioDTO) {

convocatoriaFuncionarioViewController.setConvocatoriaFuncionarioSelected(convocatoriaFuncion
arioDTO);
    try {
        List<ConvocatoriaFuncionarioProyectoDTO> listaProyectos = new ArrayList();
        Map<String, Object> filters = new HashMap<>();
        filters.put("convocatoriaFuncionarioId",
convocatoriaFuncionarioDTO.getConvocatoriaFuncionarioId());
        int count =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosListCount(filters);
        if (count > 0) {
            listaProyectos =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosList(-1,
"proyectoId.proyectoId", "A", filters);
        }

convocatoriaFuncionarioViewController.setListaConvocatoriaFuncionarioProyectos(listaProyecto
s);

convocatoriaFuncionarioViewController.setListaConvocatoriaFuncionarioProyectosFilter(listaPr
oyectos);
    } catch (Exception ex) {
        LOG.error("Error al cargar los proyectos asignados del funcionario ", ex);
    }
}
```

```

    }
}

public void cargarProyectos() {
    try {
        if (convocatoriaFuncionarioViewController.getSelected() != null &&
convocatoriaFuncionarioViewController.getSelected().getUsuarioId() != null) {
            List<ProyectoDTO> proyectosAsignados = new ArrayList<>();

            Map<String, Object> filters = new HashMap<>();
            filters.put("usuarioId",
convocatoriaFuncionarioViewController.getSelected().getUsuarioId().getUsuarioId());
            filters.put("convocatoriaId",
convocatoriaFuncionarioViewController.getConvocatoriaSelected().getConvocatoriaId());
            int count =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosListCount(filters);
            // System.out.println("convocatoriaFuncionarioProyectosListCount: " + count);
            if (count > 0) {
                List<ConvocatoriaFuncionarioProyectoDTO> proyectosFuncionarioList =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosList(-1,
"proyectoId.proyectoId", "A", filters);
                if (proyectosFuncionarioList != null &&
!proyectosFuncionarioList.isEmpty()) {
                    proyectosFuncionarioList.forEach((funcionarioProyectoDTO) -> {
                        proyectosAsignados.add(funcionarioProyectoDTO.getProyectoId());
                    });
                }
            }
        }

convocatoriaFuncionarioViewController.setSelectedProyectos(proyectosAsignados);
    } catch (Exception ex) {
        LOG.error("Error al cargar los proyectos asignados del funcionario seleccionado
", ex);
    }
}

public void guardarProyectosAsignados(ConvocatoriaFuncionarioDTO
convocatoriaFuncionarioDTO) throws Exception {
    System.out.println("entra a guardarProyectosAsignados");
    //Eliminamos los proyectos existentes
    eliminarProyectosAsignados();

    //Registramos los proyectos asignados
    List<ProyectoDTO> nuevosProyectos =
convocatoriaFuncionarioViewController.getSelectedProyectos();

    if (nuevosProyectos != null && !nuevosProyectos.isEmpty()) {
        for (ProyectoDTO proyectoDTO : nuevosProyectos) {

            //Auditoria
            AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
            appSessionVO.setProgram(programa);
            ConvocatoriaFuncionarioProyectoDTO funcionarioProyectoAdd =
convocatoriaServiceFacade.obtenerConvocatoriaFuncionarioProyecto(convocatoriaFuncionarioDTO.
getConvocatoriaFuncionarioId(), proyectoDTO.getProyectoId());
            //Se edita el registro
            if (funcionarioProyectoAdd != null &&
funcionarioProyectoAdd.getConvocatoriaFuncionarioProyectoId() != null) {
                // System.out.println("proyectoDTO Editar " + proyectoDTO);
                funcionarioProyectoAdd.setAppSessionVO(appSessionVO);
            }
        }
    }
}

```

```

funcionarioProyectoAdd.setUsuarioModificador(seguridadSessController.getSelected().getUsuario());
funcionarioProyectoAdd.setFechaModificacion(new Date());

convocatoriaServiceFacade.editarConvocatoriaFuncionarioProyecto(funcionarioProyectoAdd);
} else { //Se guarda el registro
//      System.out.println("proyectoDTO Guardar " + proyectoDTO);
ConvocatoriaFuncionarioProyectoDTO funcionarioProyectoDTO = new
ConvocatoriaFuncionarioProyectoDTO();
funcionarioProyectoDTO.setProyectoId(proyectoDTO);

funcionarioProyectoDTO.setConvocatoriaFuncionarioId(convocatoriaFuncionarioDTO);
funcionarioProyectoDTO.setAppSessionVO(appSessionVO);

funcionarioProyectoDTO.setUsuarioCreador(seguridadSessController.getSelected().getUsuario());
//Usuario
funcionarioProyectoDTO.setFechaCreacion(new Date());

convocatoriaServiceFacade.guardarConvocatoriaFuncionarioProyecto(funcionarioProyectoDTO);
}

}

}

}

public void eliminarProyectosAsignados() {
try {
Map<String, Object> filters = new HashMap<>();
filters.put("usuarioId",
convocatoriaFuncionarioViewController.getSelected().getUsuarioId().getUsuarioId());
filters.put("convocatoriaId",
convocatoriaFuncionarioViewController.getConvocatoriaSelected().getConvocatoriaId());
int count =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosListCount(filters);
if (count > 0) {
List<ConvocatoriaFuncionarioProyectoDTO> proyectosFuncionarioList =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosList(-1,
"proyectoId.proyectoId", "A", filters);
if (proyectosFuncionarioList != null && !proyectosFuncionarioList.isEmpty())
{
for (ConvocatoriaFuncionarioProyectoDTO funcionarioProyectoDTO :
proyectosFuncionarioList) {
//Auditoria
AppSessionVO appSessionVO =
seguridadSessController.getAppSessionVO();
appSessionVO.setProgram(programa);
funcionarioProyectoDTO.setAppSessionVO(appSessionVO);

convocatoriaServiceFacade.eliminarConvocatoriaFuncionarioProyecto(funcionarioProyectoDTO);
}
}
} catch (Exception ex) {
LOG.error("Error al eliminar los proyectos asignados del funcionario ", ex);
}
}
}

```

Este controlador de conversación (**ConvocatoriaFuncionarioConvController**) maneja la interacción para asignar funcionarios a una convocatoria, con operaciones de guardar, eliminar, editar, y cargar proyectos **asignados**.

Descripción General del Controlador

1. **Inyección de Dependencias:** Usa **@EJB** para inyectar servicios (**ConvocatoriaServiceFacade** y **AdministracionServiceFacade**) y **@Inject** para controladores de vista y seguridad (**ConvocatoriaFuncionarioViewController**, **SeguridadSessController**, **ColcienciasSessController**).
 2. **Métodos CRUD:**
 - **Guardar (guardar):** Asigna información de auditoría, guarda un funcionario en la convocatoria, y luego llama a **guardarProyectosAsignados**.
 - **Eliminar (eliminar):** Similar a guardar, pero elimina el registro seleccionado.
 - **Editar (editar):** Actualiza el funcionario de convocatoria con datos de modificación y guarda proyectos asignados.
 - **Cargar Proyectos (cargarProyectosAsignados y cargarProyectos):** Extraen y filtran los proyectos asignados al funcionario.
 3. **Manejo de Vista y Conversación:**
 - Controla elementos de la interfaz (como tablas y filtros) y usa métodos para actualizar y limpiar la selección.
 - Se encarga de mostrar mensajes de confirmación de transacción en **FacesContext** y **PrimeFaces**.
 4. **Auditoría y Seguridad:** La auditoría se realiza estableciendo datos en **AppSessionVO**, mientras que la seguridad se obtiene del controlador **SeguridadSessController**.
- Se definen métodos de crud en Facade EJB para funcionalidad de pantalla

```
//  
//
```

```
//FUNCIONARIOS POR CONVOCATORIA
```

```
//  
  
/**  
 * Consulta el listado de Funcionarios por convocatoria registrados en la  
 * base de datos  
 *  
 * @param first indica desde donde empieza la consulta  
 * @param pageSize indica el final de la consulta (first + pageSize)  
 * @param sortField indica el campo por el que se realiza el ordenamiento  
 * @param sortOrder indica el orden (ascendente o descendente)  
 * @param filters contiene los filtros al momento de solicitar el listado de  
 * Funcionarios por convocatoria  
 * @return lista de Funcionarios por convocatoria encontrados en el rango  
 * enviado y basado en los filtros enviados  
 * @throws Exception  
 */  
List<ConvocatoriaFuncionarioDTO> convocatoriaFuncionariosList(int first, int pageSize, String  
sortField, String sortOrder, Map<String, Object> filters) throws Exception;  
  
/**  
 * Cuenta la cantidad de Funcionarios por convocatoria actualmente  
 * registrados en la base de datos  
 *  
 * @param filters  
 * @return  
 * @throws Exception  
 */  
int convocatoriaFuncionariosListCount(Map<String, Object> filters) throws Exception;  
  
/**  
 * Guarda el registro de Funcionario por convocatoria en la base de datos  
 *  
 * @param convocatoriaFuncionarioDTO  
 * @return  
 * @throws Exception  
 */  
ConvocatoriaFuncionarioDTO guardarConvocatoriaFuncionario(ConvocatoriaFuncionarioDTO  
convocatoriaFuncionarioDTO) throws Exception;  
  
/**  
 * Edita el registro de Funcionario por convocatoria en la base de datos  
 *  
 * @param convocatoriaFuncionarioDTO  
 * @throws Exception  
 */  
void editarConvocatoriaFuncionario(ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO)  
throws Exception;  
  
/**  
 * Elimina el registro de Funcionario por convocatoria en la base de datos  
 *  
 * @param convocatoriaFuncionarioDTO  
 * @throws Exception  
 */  
void eliminarConvocatoriaFuncionario(ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO)  
throws Exception;  
  
/**  
 * Buscar el registro de una Funcionario por convocatoria y lo retorna  
 * basado en el id (llave primaria)  
 *  
 */
```

```
* @param convocatoriaFuncionarioId
* @return la Funcionario por convocatoria que coincide con el id enviado
* @throws Exception
*/
ConvocatoriaFuncionarioDTO queryConvocatoriaFuncionario(Long convocatoriaFuncionarioId)
throws Exception;

//
//
//
//
//FUNCIONARIOS PROYECTOS POR CONVOCATORIA
//
//
/**
 * Consulta el listado de FuncionarioProyectos por convocatoria registrados
 * en la base de datos
 *
 * @param first indica desde donde empieza la consulta
 * @param pageSize indica el final de la consulta (first + pageSize)
 * @param sortField indica el campo por el que se realiza el ordenamiento
 * @param sortOrder indica el orden (ascendente o descendente)
 * @param filters contiene los filtros al momento de solicitar el listado de
 * FuncionarioProyectos por convocatoria
 * @return lista de FuncionarioProyectos por convocatoria encontrados en el
 * rango enviado y basado en los filtros enviados
 * @throws Exception
 */
List<ConvocatoriaFuncionarioProyectoDTO> convocatoriaFuncionarioProyectosList(int first, int
pageSize, String sortField, String sortOrder, Map<String, Object> filters) throws Exception;

/**
 * Cuenta la cantidad de FuncionarioProyectos por convocatoria actualmente
 * registrados en la base de datos
 *
 * @param filters
 * @return
 * @throws Exception
 */
int convocatoriaFuncionarioProyectosListCount(Map<String, Object> filters) throws Exception;

/**
 * Guarda el registro de FuncionarioProyecto por convocatoria en la base de
 * datos
 *
 * @param convocatoriaFuncionarioProyectoDTO
 * @throws Exception
 */
void guardarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyectoDTO
convocatoriaFuncionarioProyectoDTO) throws Exception;

/**
 * Edita el registro de FuncionarioProyecto por convocatoria en la base de
 * datos
 *
 * @param convocatoriaFuncionarioProyectoDTO
 * @throws Exception
 */
void editarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyectoDTO
convocatoriaFuncionarioProyectoDTO) throws Exception;

/**
```



```

* Elimina el registro de FuncionarioProyecto por convocatoria en la base de
* datos
*
* @param convocatoriaFuncionarioProyectoDTO
* @throws Exception
*/
void eliminarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyectoDTO
convocatoriaFuncionarioProyectoDTO) throws Exception;

/**
* Buscar el registro de una FuncionarioProyecto por convocatoria y lo
* retorna basado en el id (llave primaria)
*
* @param convocatoriaFuncionarioProyectoId
* @return la FuncionarioProyecto por convocatoria que coincide con el id
* enviado
* @throws Exception
*/
ConvocatoriaFuncionarioProyectoDTO queryConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioProyectoId) throws Exception;

/**
* Buscar el registro de una FuncionarioProyecto por convocatoria y lo
* retorna basado en el id (llave primaria)
*
* @param convocatoriaFuncionarioId
* @param proyectoId
* @return la FuncionarioProyecto por convocatoria que coincide con el id
* enviado
* @throws Exception
*/
ConvocatoriaFuncionarioProyectoDTO obtenerConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioId, Long proyectoId) throws Exception;

```

- Se implementan métodos crud en facade EJB para funcionalidad de pantalla

```

//----- Configuración Funcionarios por Convocatoria -----//
@Override
public List<ConvocatoriaFuncionarioDTO> convocatoriaFuncionariosList(int first, int pageSize,
String sortField, String sortOrder, Map<String, Object> filters) throws Exception {
    ObjectConverter<ConvocatoriaFuncionarioDTO, ConvocatoriaFuncionario> converter =
    objectConverterFactory.checkOut();
    List<ConvocatoriaFuncionarioDTO> listConvocatoriaFuncionarioDTOs =
    converter.toTransferList(convocatoriaFuncionarioService.convocatoriaFuncionariosList(first,
    pageSize, sortField, sortOrder, filters), ConvocatoriaFuncionarioDTO.class);
    objectConverterFactory.checkIn(converter);
    return listConvocatoriaFuncionarioDTOs;
}

@Override
public int convocatoriaFuncionariosListCount(Map<String, Object> filters) throws Exception {
    return convocatoriaFuncionarioService.convocatoriaFuncionariosListCount(filters);
}

@Override
public ConvocatoriaFuncionarioDTO guardarConvocatoriaFuncionario(ConvocatoriaFuncionarioDTO
convocatoriaFuncionarioDTO) throws Exception {

```

```

        ObjectConverter<ConvocatoriaFuncionarioDTO, ConvocatoriaFuncionario> converter =
        objectConverterFactory.checkOut();
        convocatoriaFuncionarioDTO
        =
        converter.toTransfer(convocatoriaFuncionarioService.guardarConvocatoriaFuncionario(converter
        .toEntity(convocatoriaFuncionarioDTO, ConvocatoriaFuncionario.class)),
        ConvocatoriaFuncionarioDTO.class);
        objectConverterFactory.checkIn(converter);
        return convocatoriaFuncionarioDTO;
    }

    @Override
    public void editarConvocatoriaFuncionario(ConvocatoriaFuncionarioDTO
    convocatoriaFuncionarioDTO) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioDTO, ConvocatoriaFuncionario> converter =
        objectConverterFactory.checkOut();
        convocatoriaFuncionarioService.editarConvocatoriaFuncionario(converter.toEntity(convoca
        toriaFuncionarioDTO, ConvocatoriaFuncionario.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public void eliminarConvocatoriaFuncionario(ConvocatoriaFuncionarioDTO
    convocatoriaFuncionarioDTO) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioDTO, ConvocatoriaFuncionario> converter =
        objectConverterFactory.checkOut();
        convocatoriaFuncionarioService.eliminarConvocatoriaFuncionario(converter.toEntity(convo
        catoriaFuncionarioDTO, ConvocatoriaFuncionario.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public ConvocatoriaFuncionarioDTO queryConvocatoriaFuncionario(Long
    convocatoriaFuncionarioId) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioDTO, ConvocatoriaFuncionario> converter =
        objectConverterFactory.checkOut();
        ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO
        =
        converter.toTransfer(convocatoriaFuncionarioService.queryConvocatoriaFuncionario(convocatori
        aFuncionarioId), ConvocatoriaFuncionarioDTO.class);
        objectConverterFactory.checkIn(converter);
        return convocatoriaFuncionarioDTO;
    }

    //----- Configuración Funcionarios Proyectos por Convocatoria -----//
    @Override
    public List<ConvocatoriaFuncionarioProyectoDTO> convocatoriaFuncionarioProyectosList(int
    first, int pageSize, String sortField, String sortOrder, Map<String, Object> filters) throws
    Exception {
        ObjectConverter<ConvocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto>
        converter = objectConverterFactory.checkOut();
        List<ConvocatoriaFuncionarioProyectoDTO> listConvocatoriaFuncionarioProyectoDTOS
        =
        converter.toTransferList(convocatoriaFuncionarioProyectoService.convocatoriaFuncionarioProye
        ctosList(first, pageSize, sortField, sortOrder, filters),
        ConvocatoriaFuncionarioProyectoDTO.class);
        objectConverterFactory.checkIn(converter);
        return listConvocatoriaFuncionarioProyectoDTOS;
    }

    @Override
    public int convocatoriaFuncionarioProyectosListCount(Map<String, Object> filters) throws
    Exception {

```

```

        return
        convocatoriaFuncionarioProyectoService.convocatoriaFuncionarioProyectosListCount(filters);
    }

    @Override
    public void guardarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyectoDTO
convocatoriaFuncionarioProyectoDTO) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto>
converter = objectConverterFactory.checkOut();
        convocatoriaFuncionarioProyectoService.guardarConvocatoriaFuncionarioProyecto(converter
.toEntity(convocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public void editarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyectoDTO
convocatoriaFuncionarioProyectoDTO) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto>
converter = objectConverterFactory.checkOut();
        convocatoriaFuncionarioProyectoService.editarConvocatoriaFuncionarioProyecto(converter
.toEntity(convocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public void eliminarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyectoDTO
convocatoriaFuncionarioProyectoDTO) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto>
converter = objectConverterFactory.checkOut();
        convocatoriaFuncionarioProyectoService.eliminarConvocatoriaFuncionarioProyecto(converte
r.toEntity(convocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto.class));
        objectConverterFactory.checkIn(converter);
    }

    @Override
    public ConvocatoriaFuncionarioProyectoDTO queryConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioProyectoId) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto>
converter = objectConverterFactory.checkOut();
        ConvocatoriaFuncionarioProyectoDTO convocatoriaFuncionarioProyectoDTO
=
converter.toTransfer(convocatoriaFuncionarioProyectoService.queryConvocatoriaFuncionarioProy
ecto(convocatoriaFuncionarioProyectoId), ConvocatoriaFuncionarioProyectoDTO.class);
        objectConverterFactory.checkIn(converter);
        return convocatoriaFuncionarioProyectoDTO;
    }

    @Override
    public ConvocatoriaFuncionarioProyectoDTO obtenerConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioId, Long proyectoId) throws Exception {
        ObjectConverter<ConvocatoriaFuncionarioProyectoDTO, ConvocatoriaFuncionarioProyecto>
converter = objectConverterFactory.checkOut();
        ConvocatoriaFuncionarioProyectoDTO convocatoriaFuncionarioProyectoDTO
=
converter.toTransfer(convocatoriaFuncionarioProyectoService.obtenerConvocatoriaFuncionarioPr
oyecto(convocatoriaFuncionarioId, proyectoId), ConvocatoriaFuncionarioProyectoDTO.class);
        objectConverterFactory.checkIn(converter);
        return convocatoriaFuncionarioProyectoDTO;
    }
}

```

- Se crea interfaz **ConvocatoriaFuncionarioService.java** que define métodos de crud en Service EJB para funcionalidad de pantalla

```
@Local
public interface ConvocatoriaFuncionarioService {

    List<ConvocatoriaFuncionario> convocatoriaFuncionariosList(int first, int pageSize,
String sortField, String sortOrder, Map<String, Object> filters) throws SystemException;

    int convocatoriaFuncionariosListCount(Map<String, Object> filters) throws
SystemException;

    ConvocatoriaFuncionario guardarConvocatoriaFuncionario(ConvocatoriaFuncionario
convocatoriaFuncionario) throws SystemException;

    void editarConvocatoriaFuncionario(ConvocatoriaFuncionario convocatoriaFuncionario)
throws SystemException;

    void eliminarConvocatoriaFuncionario(ConvocatoriaFuncionario convocatoriaFuncionario)
throws SystemException;

    ConvocatoriaFuncionario queryConvocatoriaFuncionario(Long convocatoriaFuncionarioId)
throws SystemException;
}
```

- Se crea **ConvocatoriaFuncionarioServiceBean.java** donde implementan métodos crud en Service EJB para funcionalidad de pantalla

```
@Stateless
public class ConvocatoriaFuncionarioServiceBean extends StoreManagerImpl implements
ConvocatoriaFuncionarioService {

    //----- Convocatoria Funcionarios -----//
    @Override
    public List<ConvocatoriaFuncionario> convocatoriaFuncionariosList(int first, int
pageSize, String sortField, String sortOrder, Map<String, Object> filters) throws
SystemException {
        return
this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaFuncionariosList(first,
pageSize, sortField, sortOrder, filters);
    }

    @Override
    public int convocatoriaFuncionariosListCount(Map<String, Object> filters) throws
SystemException {
        return
this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaFuncionariosListCount(filte
rs);
    }

    @Override
    public ConvocatoriaFuncionario guardarConvocatoriaFuncionario(ConvocatoriaFuncionario
convocatoriaFuncionario) throws SystemException {
        return this.crear(convocatoriaFuncionario, ConvocatoriaFuncionario.class);
    }
}
```

```
@Override
public void editarConvocatoriaFuncionario(ConvocatoriaFuncionario
convocatoriaFuncionario) throws SystemException {
    this.modificar(convocatoriaFuncionario, ConvocatoriaFuncionario.class);
}

@Override
public void eliminarConvocatoriaFuncionario(ConvocatoriaFuncionario
convocatoriaFuncionario) throws SystemException {
    eliminarFuncionarioProyectos(convocatoriaFuncionario);
    this.eliminar(convocatoriaFuncionario, ConvocatoriaFuncionario.class);
}

@Override
public ConvocatoriaFuncionario queryConvocatoriaFuncionario(Long
convocatoriaFuncionarioId) throws SystemException {
    return this.buscar(convocatoriaFuncionarioId, ConvocatoriaFuncionario.class);
}

private void eliminarFuncionarioProyectos(ConvocatoriaFuncionario
convocatoriaFuncionario) throws SystemException {
    Map<String, Object> filters = new HashMap<>();
    filters.put("convocatoriaFuncionarioId",
convocatoriaFuncionario.getConvocatoriaFuncionarioId()); // Filtro por convocatoria funcionario
ID
    List<ConvocatoriaFuncionarioProyecto> funcionarioProyectosList =
this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaFuncionarioProyectosList(-
1, -1, "convocatoriaFuncionarioProyectoId", "A", filters);
    if (funcionarioProyectosList != null && !funcionarioProyectosList.isEmpty()) {
        for (ConvocatoriaFuncionarioProyecto convocatoriaFuncionarioProyecto :
funcionarioProyectosList) {
            convocatoriaFuncionarioProyecto.setAppSessionVO(convocatoriaFuncionario.getAppSessionVO());
            this.eliminar(convocatoriaFuncionarioProyecto);
        }
    }
}
```

- Se crea interfaz **ConvocatoriaFuncionarioProyectoService.java** que define métodos de crud en Service EJB para funcionalidad de pantalla

```
@Local
public interface ConvocatoriaFuncionarioProyectoService {

    List<ConvocatoriaFuncionarioProyecto> convocatoriaFuncionarioProyectosList(int first, int
pageSize, String sortField, String sortOrder, Map<String, Object> filters) throws
SystemException;

    int convocatoriaFuncionarioProyectosListCount(Map<String, Object> filters) throws
SystemException;

    void guardarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyecto
convocatoriaFuncionarioProyecto) throws SystemException;
}
```

```
void editarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyecto
convocatoriaFuncionarioProyecto) throws SystemException;

void eliminarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyecto
convocatoriaFuncionarioProyecto) throws SystemException;

ConvocatoriaFuncionarioProyecto queryConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioProyectoId) throws SystemException;

ConvocatoriaFuncionarioProyecto obtenerConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioId, Long proyectoId) throws SystemException;
}
```

- Se crea **ConvocatoriaFuncionarioProyectoServiceBean.java** donde implementan métodos crud en Service EJB para funcionalidad de pantalla

```
@Stateless
public class ConvocatoriaFuncionarioProyectoServiceBean extends StoreManagerImpl implements
ConvocatoriaFuncionarioProyectoService {

    //----- Convocatoria Funcionarios Proyectos -----
    -----//
    @Override
    public List<ConvocatoriaFuncionarioProyecto> convocatoriaFuncionarioProyectosList(int
first, int pageSize, String sortField, String sortOrder, Map<String, Object> filters) throws
SystemException {
        return
this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaFuncionarioProyectosList(fi
rst, pageSize, sortField, sortOrder, filters);
    }

    @Override
    public int convocatoriaFuncionarioProyectosListCount(Map<String, Object> filters) throws
SystemException {
        return
this.getStoreManager(ConvocatoriaStoreManager.class).convocatoriaFuncionarioProyectosListCou
nt(filters);
    }

    @Override
    public void guardarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyecto
convocatoriaFuncionarioProyecto) throws SystemException {
        this.crear(convocatoriaFuncionarioProyecto, ConvocatoriaFuncionarioProyecto.class);
    }

    @Override
    public void editarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyecto
convocatoriaFuncionarioProyecto) throws SystemException {
        this.modificar(convocatoriaFuncionarioProyecto,
ConvocatoriaFuncionarioProyecto.class);
    }

    @Override
    public void eliminarConvocatoriaFuncionarioProyecto(ConvocatoriaFuncionarioProyecto
convocatoriaFuncionarioProyecto) throws SystemException {
        this.eliminar(convocatoriaFuncionarioProyecto,
ConvocatoriaFuncionarioProyecto.class);
    }
}
```

```

    }

    @Override
    public ConvocatoriaFuncionarioProyecto queryConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioProyectoId) throws SystemException {
        return this.buscar(convocatoriaFuncionarioProyectoId,
ConvocatoriaFuncionarioProyecto.class);
    }
    @Override
    public ConvocatoriaFuncionarioProyecto obtenerConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioId, Long proyectoId) throws SystemException {
        return
this.getStoreManager(ConvocatoriaStoreManager.class).obtenerConvocatoriaFuncionarioProyecto(
convocatoriaFuncionarioId, proyectoId);
    }
}

```

- Se crean métodos en **ConvocatoriaStoreManager.java** donde se definen métodos de consulta criteria para funcionalidad de pantalla

```

//-----Funcionarios por Convocatoria -----//
List<ConvocatoriaFuncionario> convocatoriaFuncionariosList(int first, int pageSize, String
sortField, String sortOrder, Map<String, Object> filters);

int convocatoriaFuncionariosListCount(Map<String, Object> filters);

//----- Funcionarios Proyectos por Convocatoria -----//
List<ConvocatoriaFuncionarioProyecto> convocatoriaFuncionarioProyectosList(int first, int
pageSize, String sortField, String sortOrder, Map<String, Object> filters);

int convocatoriaFuncionarioProyectosListCount(Map<String, Object> filters);

ConvocatoriaFuncionarioProyecto obtenerConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioId, Long proyectoId);

```

- Se crean métodos en **ConvocatoriaStoreManagerImpl.java** donde implementan métodos de consulta en criteria para funcionalidad de pantalla

Criteria ConvocatoriaFuncionario

```

@Override
public List<ConvocatoriaFuncionario> convocatoriaFuncionariosList(int first, int pageSize,
String sortField, String sortOrder, Map<String, Object> filters) {
    cb = em.getCriteriaBuilder();
    queryConvocatoriaFuncionario = cb.createQuery(ConvocatoriaFuncionario.class);
    convocatoriaFuncionarioRoot = queryConvocatoriaFuncionario.from(ConvocatoriaFuncionario.class);
    queryConvocatoriaFuncionario.select(convocatoriaFuncionarioRoot);
    //Where
    queryConvocatoriaFuncionario.where(cb.and(filtrosConvocatoriaFuncionario(filters).toArr
ay(new Predicate[]{})));
}

```

```
//Order
if (sortOrder.equals("A")) {

    queryConvocatoriaFuncionario.orderBy(cb.asc(getConvocatoriaFuncionarioPath(sortField)))
;
} else {

    queryConvocatoriaFuncionario.orderBy(cb.desc(getConvocatoriaFuncionarioPath(sortField)))
);
}
//pagination
TypedQuery<ConvocatoriaFuncionario> tq = em.createQuery(queryConvocatoriaFuncionario);
if (pageSize >= 0) {
    tq.setMaxResults(pageSize);
}
if (first >= 0) {
    tq.setFirstResult(first);
}
return tq.getResultList();
}

@Override
public int convocatoriaFuncionariosListCount(Map<String, Object> filters) {
    CriteriaQuery<Long> qCount;
    cb = em.getCriteriaBuilder();
    qCount = cb.createQuery(Long.class);
    convocatoriaFuncionarioRoot = qCount.from(ConvocatoriaFuncionario.class);
    qCount.select(cb.count(convocatoriaFuncionarioRoot));
    //Where
    qCount.where(cb.and(filtrosConvocatoriaFuncionario(filters).toArray(new
Predicate[]{})));
    Long res = em.createQuery(qCount).getSingleResult();
    return res.intValue();
}

private List<Predicate> filtrosConvocatoriaFuncionario(Map<String, Object> filters) {
    List<Predicate> lista = new ArrayList<>();
    List<String> camposFiltro = new ArrayList<>();
    Path pathFilter;
    camposFiltro.add("usuarioId.usuario");
    camposFiltro.add("usuarioId.descripcionUsuario");
    camposFiltro.add("usuarioId.dependenciaId.nombreDependencia");

    //Filtro por convocatoria ID
    Object convocatoriaId = filters.get("convocatoriaId");
    if (convocatoriaId != null) {

        lista.add(cb.equal(getConvocatoriaFuncionarioPath("convocatoriaId.convocatoriaId"),
(Long) convocatoriaId));
    }
    //Filtro por usuario ID
    Object usuarioId = filters.get("usuarioId");
    if (usuarioId != null) {
        lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuarioId"),
(Long) usuarioId));
    }
    //Filtro por usuario
    Object usuario = filters.get("usuario");
    if (usuario != null) {
        lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuario"),
(String) usuario));
    }
}
```



```

    }

    //Filtro Global
    globalFilter(filters, camposFiltro, "getConvocatoriaFuncionarioPath", lista);

    return lista;
}

private Path<?> getConvocatoriaFuncionarioPath(String field) {
    //Joins
    Join<ConvocatoriaFuncionario, Convocatoria> convocatoriaId =
convocatoriaFuncionarioRoot.join(ConvocatoriaFuncionario_.convocatoriaId, JoinType.LEFT);
    Join<ConvocatoriaFuncionario, Usuario> usuarioId =
convocatoriaFuncionarioRoot.join(ConvocatoriaFuncionario_.usuarioId, JoinType.LEFT);
    Join<Usuario, Dependencia> dependenciaId = usuarioId.join(Usuario_.dependenciaId,
JoinType.LEFT);

    Path<?> path = null;
    if (null == field) {
        path
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.convocatoriaFuncionarioId);
    } else {
        switch (field) {
            case "convocatoriaFuncionarioId":
                path
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.convocatoriaFuncionarioId);
                break;
            case "usuarioCreador":
                path
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.usuarioCreador);
                break;
            case "fechaCreacion":
                path
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.fechaCreacion);
                break;
            case "convocatoriaId.convocatoriaId":
                path = convocatoriaId.get(Convocatoria_.convocatoriaId);
                break;
            case "convocatoriaId.descripcionConvocatoria":
                path
convocatoriaId.get(Convocatoria_.descripcionConvocatoria);
                break;
            case "usuarioId.usuarioId":
                path = usuarioId.get(Usuario_.usuarioId);
                break;
            case "usuarioId.usuario":
                path = usuarioId.get(Usuario_.usuario);
                break;
            case "usuarioId.descripcionUsuario":
                path = usuarioId.get(Usuario_.descripcionUsuario);
                break;
            case "usuarioId.estadoUsuario":
                path = usuarioId.get(Usuario_.estadoUsuario);
                break;
            case "usuarioId.dependenciaId.dependenciaId":
                path = dependenciaId.get(Dependencia_.dependenciaId);
                break;
            case "usuarioId.dependenciaId.nombreDependencia":
                path = dependenciaId.get(Dependencia_.nombreDependencia);
                break;
            default:

```

```
        break;
    }
    return path;
}
```

A continuación, se explica cada método relacionado con Criterios API que se usó para el módulo **ConvocatoriaFuncionario**. Se revisa en detalle las implementaciones de **convocatoriaFuncionariosList**, **convocatoriaFuncionariosListCount**, **filtrosConvocatoriaFuncionario**, y **getConvocatoriaFuncionarioPath**.

1. Método **convocatoriaFuncionariosList**

Este método obtiene una lista de entidades **ConvocatoriaFuncionario** con filtros, ordenamiento, y paginación. Veamos cada paso:

- **Inicialización de **CriteriaBuilder** y **CriteriaQuery**:**

```
cb = em.getCriteriaBuilder();
queryConvocatoriaFuncionario = cb.createQuery(ConvocatoriaFuncionario.class);
```

CriteriaBuilder (cb) se utiliza para construir la consulta, proporcionando operaciones de criterios. **CriteriaQuery (queryConvocatoriaFuncionario)** es la consulta específica para **ConvocatoriaFuncionario**.

- **Definición de la raíz de la consulta:**

```
convocatoriaFuncionarioRoot =
queryConvocatoriaFuncionario.from(ConvocatoriaFuncionario.class);
queryConvocatoriaFuncionario.select(convocatoriaFuncionarioRoot);
```

convocatoriaFuncionarioRoot define el punto de inicio para los atributos de la entidad **ConvocatoriaFuncionario**, similar a una tabla en SQL.

- **Aplicación de filtros:**

```
queryConvocatoriaFuncionario.where(cb.and(filtrosConvocatoriaFuncionario
(filters).toArray(new Predicate[]{})));
```

Aquí se llaman los predicados de **filtrosConvocatoriaFuncionario** y se aplican con **.where**. Los filtros son dinámicos, adaptándose a las condiciones proporcionadas en **filters**.

- **Ordenamiento:**

```
if (sortOrder.equals("A")) {  
    queryConvocatoriaFuncionario.orderBy(cb.asc(getConvocatoriaFuncionarioPath(sortField)));  
} else {  
  
    queryConvocatoriaFuncionario.orderBy(cb.desc(getConvocatoriaFuncionarioPath(sortField)));  
}
```

Dependiendo del valor de **sortOrder**, la consulta se ordena de forma ascendente (**asc**) o descendente (**desc**). El método **getConvocatoriaFuncionarioPath** se usa para obtener la ruta específica del campo de ordenamiento, adaptándose al campo seleccionado.

- **Paginación:**

```
TypedQuery<ConvocatoriaFuncionario> tq = em.createQuery(queryConvocatoriaFuncionario);  
if (pageSize >= 0) {  
    tq.setMaxResults(pageSize);  
}  
if (first >= 0) {  
    tq.setFirstResult(first);  
}
```

pageSize y **first** controlan la cantidad de registros devueltos y el índice inicial. Esto es útil para manejar grandes cantidades de datos al cargar solo una parte de los registros en cada llamada.

2. Método **convocatoriaFuncionariosListCount**

Este método calcula el número total de registros que cumplen con los filtros, útil para la paginación en el front-end:

- **Configuración de **CriteriaQuery** para contar:**

```
qCount = cb.createQuery(Long.class);  
convocatoriaFuncionarioRoot = qCount.from(ConvocatoriaFuncionario.class);  
qCount.select(cb.count(convocatoriaFuncionarioRoot));
```

Aquí, **CriteriaQuery<Long>** se configura para realizar una operación de conteo (**count**). **cb.count** cuenta el número de elementos en la raíz **convocatoriaGranAreaRoot**.

- **Aplicación de filtros:**

```
qCount.where(cb.and(filtrosConvocatoriaFuncionario(filters).toArray(new Predicate[1])));
```

Se aplican los mismos filtros que en **convocatoriaFuncionariosList**, lo cual asegura que el conteo coincida con los registros obtenidos en la lista.

3. Método **filtrosConvocatoriaFuncionario**

Este método construye la lista de **Predicates** (condiciones) basados en los filtros proporcionados:

- **Lista de campos de filtro:**

```
List<String> camposFiltro = new ArrayList<>();
camposFiltro.add("usuarioId.usuario");
camposFiltro.add("usuarioId.descripcionUsuario");
camposFiltro.add("usuarioId.dependenciaId.nombreDependencia");
```

camposFiltro contiene los nombres de los campos sobre los que se aplicará un filtro global.

- **Filtro específico para convocatoriaId:**

```
Object convocatoriaId = filters.get("convocatoriaId");
if (convocatoriaId != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioPath("convocatoriaId.convocatoriaId"),
        (Long) convocatoriaId));
}
```

Si el filtro **convocatoriaId** está presente, se agrega una condición equal para buscar coincidencias exactas en **convocatoriaId**.

- **Filtro específico para usuarioId:**

```
Object usuarioId = filters.get("usuarioId");
if (usuarioId != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuarioId"),
        (Long) usuarioId));
}
```

Si el filtro **usuarioId** está presente, se agrega una condición equal para buscar coincidencias exactas en **usuarioId**.

- **Filtro específico para usuario:**

```
Object usuario = filters.get("usuario");
```

```
if (usuario != null) {  
    lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuario"), (String)  
usuario));  
}
```

Si el filtro **usuario** está presente, se agrega una condición equal para buscar coincidencias exactas en **usuario**.

- **Filtro global:**

```
globalFilter(filters, camposFiltro, "getConvocatoriaFuncionarioPath", lista);
```

globalFilter aplica un filtro más amplio, permitiendo buscar coincidencias en varios campos al mismo tiempo (como una búsqueda de texto completo).

4. Método **getConvocatoriaFuncionarioPath**

Este método es crucial para construir las rutas de los atributos en Criteria API para diferentes campos. Esto permite crear una lógica flexible para orden y filtrado en función del campo solicitado.

- **Join con otras entidades:**

```
Join<ConvocatoriaFuncionario, Convocatoria> convocatoriaId =  
convocatoriaFuncionarioRoot.join(ConvocatoriaFuncionario_.convocatoriaId, JoinType.LEFT);  
Join<ConvocatoriaFuncionario, Usuario> usuarioId =  
convocatoriaFuncionarioRoot.join(ConvocatoriaFuncionario_.usuarioId, JoinType.LEFT);  
Join<Usuario, Dependencia> dependenciaId = usuarioId.join(Usuario_.dependenciaId,  
JoinType.LEFT);
```

Aquí se establecen joins con las entidades **Convocatoria**, **Usuario** y **Dependencia** para acceder a sus campos. El **JoinType.LEFT** asegura que se incluyan registros de **ConvocatoriaFuncionario** que puedan no tener coincidencias en estas tablas relacionadas.

- **Rutas de campos específicos:**

```
switch (field) {  
    case "convocatoriaFuncionarioId":  
        path =  
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.convocatoriaFuncionarioId);  
        break;  
    // Otros casos...  
}
```

Para cada campo solicitado (**field**), el método devuelve la ruta adecuada. Esto permite que los filtros y el ordenamiento sean dinámicos y adaptables a los campos solicitados.

Criteria ConvocatoriaFuncionarioProyecto

```
//----- FUNCIONARIOS PROYECTOS POR CONVOCATORIA -
//-----
@Override
public List<ConvocatoriaFuncionarioProyecto> convocatoriaFuncionarioProyectosList(int first,
int pageSize, String sortField, String sortOrder, Map<String, Object> filters) {
    cb = em.getCriteriaBuilder();
    queryConvocatoriaFuncionarioProyecto =
cb.createQuery(ConvocatoriaFuncionarioProyecto.class);
    convocatoriaFuncionarioProyectoRoot =
queryConvocatoriaFuncionarioProyecto.from(ConvocatoriaFuncionarioProyecto.class);
    queryConvocatoriaFuncionarioProyecto.select(convocatoriaFuncionarioProyectoRoot);
    //Where
    queryConvocatoriaFuncionarioProyecto.where(cb.and(filtrosConvocatoriaFuncionarioProyecto(
o(filters).toArray(new Predicate[]{})));
    //Order
    if (sortOrder.equals("A")) {

        queryConvocatoriaFuncionarioProyecto.orderBy(cb.asc(getConvocatoriaFuncionarioProyectoPath(
sortField)));
    } else {

        queryConvocatoriaFuncionarioProyecto.orderBy(cb.desc(getConvocatoriaFuncionarioProyectoPath(
sortField)));
    }
    //pagination
    TypedQuery<ConvocatoriaFuncionarioProyecto> tq =
em.createQuery(queryConvocatoriaFuncionarioProyecto);
    if (pageSize >= 0) {
        tq.setMaxResults(pageSize);
    }
    if (first >= 0) {
        tq.setFirstResult(first);
    }
    return tq.getResultList();
}

@Override
public int convocatoriaFuncionarioProyectosListCount(Map<String, Object> filters) {
    CriteriaQuery<Long> qCount;
    cb = em.getCriteriaBuilder();
    qCount = cb.createQuery(Long.class);
    convocatoriaFuncionarioProyectoRoot =
qCount.from(ConvocatoriaFuncionarioProyecto.class);
    qCount.select(cb.count(convocatoriaFuncionarioProyectoRoot));
    //Where
    qCount.where(cb.and(filtrosConvocatoriaFuncionarioProyecto(filters).toArray(new
Predicate[]{})));
    Long res = em.createQuery(qCount).getSingleResult();
    return res.intValue();
}

private List<Predicate> filtrosConvocatoriaFuncionarioProyecto(Map<String, Object> filters) {
    List<Predicate> lista = new ArrayList<>();
```

```

List<String> camposFiltro = new ArrayList<>();
Path pathFilter;
camposFiltro.add("FuncionarioProyectoId.descripcionValorDefinicion");
camposFiltro.add("convocatoriaId.descripcionConvocatoria");

//Filtro por convocatoriaFuncionario ID
Object convocatoriaFuncionarioId = filters.get("convocatoriaFuncionarioId");
if (convocatoriaFuncionarioId != null) {

    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("convocatoriaFuncionarioId.co
nocatoriaFuncionarioId"), (Long) convocatoriaFuncionarioId));
}
//Filtro por convocatoria ID
Object convocatoriaId = filters.get("convocatoriaId");
if (convocatoriaId != null) {

    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("convocatoriaFuncionarioId.co
nocatoriaId.convocatoriaId"), (Long) convocatoriaId));
}
//Filtro por usuario ID
Object usuarioId = filters.get("usuarioId");
if (usuarioId != null) {

    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("convocatoriaFuncionarioId.us
uarioId.usuarioId"), (Long) usuarioId));
}
//Filtro por proyecto ID
Object proyectoId = filters.get("proyectoId");
if (proyectoId != null) {

    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("proyectoId.proyectoId"),
(Long) proyectoId));
}

//Filtro Global
globalFilter(filters, camposFiltro, "getConvocatoriaFuncionarioProyectoPath", lista);

return lista;
}

private Path<?> getConvocatoriaFuncionarioProyectoPath(String field) {
    //Joins
    Join<ConvocatoriaFuncionarioProyecto, ConvocatoriaFuncionario> convocatoriaFuncionarioId
    =
    convocatoriaFuncionarioProyectoRoot.join(ConvocatoriaFuncionarioProyecto_.convocatoriaFuncio
narioId, JoinType.LEFT);
    Join<ConvocatoriaFuncionario, Convocatoria> convocatoriaId
    =
    convocatoriaFuncionarioId.join(ConvocatoriaFuncionario_.convocatoriaId, JoinType.LEFT);
    Join<ConvocatoriaFuncionario, Usuario> usuarioId
    =
    convocatoriaFuncionarioId.join(ConvocatoriaFuncionario_.usuarioId, JoinType.LEFT);
    Join<ConvocatoriaFuncionarioProyecto, Proyecto> proyectoId
    =
    convocatoriaFuncionarioProyectoRoot.join(ConvocatoriaFuncionarioProyecto_.proyectoId,
JoinType.LEFT);

    Path<?> path = null;
    if (null == field) {
        path
        =
        convocatoriaFuncionarioProyectoRoot.get(ConvocatoriaFuncionarioProyecto_.convocatoriaFuncion
arioProyectoId);
    } else {
        switch (field) {

```

```

        case "convocatoriaFuncionarioProyectoId":
            path =
convocatoriaFuncionarioProyectoRoot.get(ConvocatoriaFuncionarioProyecto_.convocatoriaFuncion
arioProyectoId);
            break;
        case "usuarioCreador":
            path =
convocatoriaFuncionarioProyectoRoot.get(ConvocatoriaFuncionarioProyecto_.usuarioCreador);
            break;
        case "fechaCreacion":
            path =
convocatoriaFuncionarioProyectoRoot.get(ConvocatoriaFuncionarioProyecto_.fechaCreacion);
            break;
        case "convocatoriaFuncionarioId.convocatoriaFuncionarioId":
            path =
convocatoriaFuncionarioId.get(ConvocatoriaFuncionario_.convocatoriaFuncionarioId);
            break;
        case "convocatoriaFuncionarioId.convocatoriaId.convocatoriaId":
            path = convocatoriaId.get(Convocatoria_.convocatoriaId);
            break;
        case
"convocatoriaFuncionarioId.convocatoriaId.descripcionConvocatoria":
            path =
convocatoriaId.get(Convocatoria_.descripcionConvocatoria);
            break;
        case "convocatoriaFuncionarioId.usuarioId.usuarioId":
            path = usuarioId.get(Usuario_.usuarioId);
            break;
        case "convocatoriaFuncionarioId.usuarioId.descripcionUsuario":
            path = usuarioId.get(Usuario_.descripcionUsuario);
            break;
        case "proyectoId.proyectoId":
            path = proyectoId.get(Proyecto_.proyectoId);
            break;
        case "proyectoId.titulo":
            path = proyectoId.get(Proyecto_.titulo);
            break;
        case "proyectoId.exencion":
            path = proyectoId.get(Proyecto_.exencion);
            break;
        default:
            break;
    }
}
return path;
}

```

A continuación, se explica cada método relacionado con Criterios API que se usó para el módulo **ConvocatoriaFuncionarioProyecto**. Se revisa en detalle las implementaciones de **convocatoriaFuncionariosList**, **convocatoriaFuncionarioProyectosListCount**, **filtrosConvocatoriaFuncionarioProyecto**, y **getConvocatoriaFuncionarioProyectoPath**.

1. Método convocatoriaFuncionarioProyectosList

Este método obtiene una lista de entidades **ConvocatoriaFuncionarioProyecto** con filtros, ordenamiento, y paginación. Veamos cada paso:

- **Inicialización de CriteriaBuilder y CriteriaQuery:**

```
cb = em.getCriteriaBuilder();  
queryConvocatoriaFuncionarioProyecto  
cb.createQuery(ConvocatoriaFuncionarioProyecto.class);
```

CriteriaBuilder (cb) se utiliza para construir la consulta, proporcionando operaciones de criterios. **CriteriaQuery (queryConvocatoriaFuncionario)** es la consulta específica para **ConvocatoriaFuncionario**.

- **Definición de la raíz de la consulta:**

```
convocatoriaFuncionarioProyectoRoot  
queryConvocatoriaFuncionarioProyecto.from(ConvocatoriaFuncionarioProyecto.class);  
queryConvocatoriaFuncionarioProyecto.select(convocatoriaFuncionarioProyectoRoot);
```

convocatoriaFuncionarioProyectoRoot define el punto de inicio para los atributos de la entidad **ConvocatoriaFuncionarioProyecto**, similar a una tabla en SQL.

- **Aplicación de filtros:**

```
queryConvocatoriaFuncionarioProyecto.where(cb.and(filtrosConvocatoriaFuncionarioProyecto(filters).toArray(new Predicate[]{})));
```

Aquí se llaman los predicados de **filtrosConvocatoriaFuncionarioProyecto** y se aplican con **.where**. Los filtros son dinámicos, adaptándose a las condiciones proporcionadas en **filters**.

- **Ordenamiento:**

```
if (sortOrder.equals("A")) {  
  
queryConvocatoriaFuncionarioProyecto.orderBy(cb.asc(getConvocatoriaFuncionarioProyectoPath(sortField)));  
} else {  
  
queryConvocatoriaFuncionarioProyecto.orderBy(cb.desc(getConvocatoriaFuncionarioProyectoPath(sortField)));  
}
```

Dependiendo del valor de **sortOrder**, la consulta se ordena de forma ascendente (**asc**) o descendente (**desc**). El método **getConvocatoriaFuncionarioProyectoPath** se usa para obtener la ruta específica del campo de ordenamiento, adaptándose al campo seleccionado.

- **Paginación:**

```
TypedQuery<ConvocatoriaFuncionarioProyecto> tq =  
em.createQuery(queryConvocatoriaFuncionarioProyecto);  
if (pageSize >= 0) {  
    tq.setMaxResults(pageSize);  
}  
if (first >= 0) {  
    tq.setFirstResult(first);  
}
```

pageSize y **first** controlan la cantidad de registros devueltos y el índice inicial. Esto es útil para manejar grandes cantidades de datos al cargar solo una parte de los registros en cada llamada.

2. Método **convocatoriaFuncionarioProyectosListCount**

Este método calcula el número total de registros que cumplen con los filtros, útil para la paginación en el front-end:

- **Configuración de **CriteriaQuery** para contar:**

```
qCount = cb.createQuery(Long.class);  
convocatoriaFuncionarioProyectoRoot = qCount.from(ConvocatoriaFuncionarioProyecto.class);  
qCount.select(cb.count(convocatoriaFuncionarioProyectoRoot));
```

Aquí, **CriteriaQuery<Long>** se configura para realizar una operación de conteo (**count**). **cb.count** cuenta el número de elementos en la raíz **convocatoriaFuncionarioProyectoRoot**.

- **Aplicación de filtros:**

```
qCount.where(cb.and(filtrosConvocatoriaFuncionarioProyecto (filters).toArray(new  
Predicate[]{})));
```

Se aplican los mismos filtros que en **convocatoriaFuncionarioProyectosList**, lo cual asegura que el conteo coincida con los registros obtenidos en la lista.

3. Método `filtrosConvocatoriaFuncionarioProyecto`

Este método construye la lista de **Predicates** (condiciones) basados en los filtros proporcionados:

- **Lista de campos de filtro:**

```
List<String> camposFiltro = new ArrayList<>();
camposFiltro.add("FuncionarioProyectoId.descripcionValorDefinicion");
camposFiltro.add("convocatoriaId.descripcionConvocatoria");
```

camposFiltro contiene los nombres de los campos sobre los que se aplicará un filtro global.

- **Filtro específico para `convocatoriaFuncionarioId`:**

```
Object convocatoriaFuncionarioId = filters.get("convocatoriaFuncionarioId");
if (convocatoriaFuncionarioId != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("convocatoriaFuncionarioId.convocatoriaFuncionarioId"), (Long) convocatoriaFuncionarioId));
}
```

Si el filtro **`convocatoriaFuncionarioId`** está presente, se agrega una condición equal para buscar coincidencias exactas en **`convocatoriaFuncionarioId`**.

- **Filtro específico para `convocatoriaId`:**

```
Object convocatoriaId = filters.get("convocatoriaId");
if (convocatoriaId != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("convocatoriaFuncionarioId.convocatoriaId"), (Long) convocatoriaId));
}
```

Si el filtro **`convocatoriaId`** está presente, se agrega una condición equal para buscar coincidencias exactas en **`convocatoriaId`**.

- **Filtro específico para `usuarioId`:**

```
Object usuarioId = filters.get("usuarioId");
if (usuarioId != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("convocatoriaFuncionarioId.usuarioId.usuarioId"), (Long) usuarioId));
}
```

Si el filtro **usuariold** está presente, se agrega una condición equal para buscar coincidencias exactas en **usuariold**.

- **Filtro específico para proyectold:**

```
Object proyectoId = filters.get("proyectoId");
if (proyectoId != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioProyectoPath("proyectoId.proyectoId"),
        (Long) proyectoId));
}
```

Si el filtro **proyectold** está presente, se agrega una condición equal para buscar coincidencias exactas en **proyectold**.

- **Filtro global:**

```
globalFilter(filters, camposFiltro, "getConvocatoriaFuncionarioProyectoPath", lista);
```

globalFilter aplica un filtro más amplio, permitiendo buscar coincidencias en varios campos al mismo tiempo (como una búsqueda de texto completo).

4. Método **getConvocatoriaFuncionarioProyectoPath**

Este método es crucial para construir las rutas de los atributos en Criteria API para diferentes campos. Esto permite crear una lógica flexible para orden y filtrado en función del campo solicitado.

- **Join con otras entidades:**

```
Join<ConvocatoriaFuncionarioProyecto, ConvocatoriaFuncionario> convocatoriaFuncionarioId =
convocatoriaFuncionarioProyectoRoot.join(ConvocatoriaFuncionarioProyecto_.convocatoriaFuncio
narioId, JoinType.LEFT);
Join<ConvocatoriaFuncionario, Convocatoria> convocatoriaId =
convocatoriaFuncionarioId.join(ConvocatoriaFuncionario_.convocatoriaId, JoinType.LEFT);
Join<ConvocatoriaFuncionario, Usuario> usuarioId =
convocatoriaFuncionarioId.join(ConvocatoriaFuncionario_.usuarioId, JoinType.LEFT);
Join<ConvocatoriaFuncionarioProyecto, Proyecto> proyectoId =
convocatoriaFuncionarioProyectoRoot.join(ConvocatoriaFuncionarioProyecto_.proyectoId,
JoinType.LEFT);
```

Aquí se establecen joins con las entidades **ConvocatoriaFuncionario**, **Convocatoria**, **Usuario** y **Proyecto** para acceder a sus campos. El **JoinType.LEFT** asegura que se incluyan registros de **ConvocatoriaFuncionarioProyecto** que puedan no tener coincidencias en estas tablas relacionadas.

- **Rutas de campos específicos:**

```
switch (field) {
    case "convocatoriaFuncionarioProyectoId ":
        path = convocatoriaFuncionarioProyectoRoot.get(ConvocatoriaFuncionarioProyecto_.
convocatoriaFuncionarioProyectoId);
        break;
        // Otros casos...
}
```

Para cada campo solicitado (**field**), el método devuelve la ruta adecuada. Esto permite que los filtros y el ordenamiento sean dinámicos y adaptables a los campos solicitados.

- **Metodo `obtenerConvocatoriaFuncionarioProyecto`**

```
@Override
public ConvocatoriaFuncionarioProyecto obtenerConvocatoriaFuncionarioProyecto(Long
convocatoriaFuncionarioId, Long proyectoId) {
    Query query = em.createQuery("SELECT d "
        + " FROM ConvocatoriaFuncionarioProyecto d "
        + " WHERE d.convocatoriaFuncionarioId.convocatoriaFuncionarioId =
:convocatoriaFuncionarioId "
        + " AND d.proyectoId.proyectoId = :proyectoId ");
    query.setParameter("convocatoriaFuncionarioId", convocatoriaFuncionarioId);
    query.setParameter("proyectoId", proyectoId);

    List<ConvocatoriaFuncionarioProyecto> lista = query.getResultList();
    return lista != null && !lista.isEmpty() ? lista.get(0) : new
ConvocatoriaFuncionarioProyecto();
}
```

- **Internacionalización en español e inglés**

```
##### ES

#Configuracion - Funcionarios por Convocatoria
sb.menu.gestion.configuracion.convocatorias.funcionarios          = Funcionarios
Evaluación
gestion.configuracion.convocatoria.funcionarios.button           = Funcionarios
Evaluación
gestion.configuracion.convocatoria.funcionarios.header           = Funcionarios
Evaluación de la Convocatoria
gestion.configuracion.convocatoria.funcionarios.funcionario.header = Funcionario
gestion.configuracion.convocatoria.funcionarios.usuario.header   = Usuario
gestion.configuracion.convocatoria.funcionarios.dependencia.header = Dependencia
gestion.configuracion.convocatoria.funcionarios.proyectos.button  = Ver Proyectos
Asignados

gestion.configuracion.convocatoria.funcionarios.funcionario.label = Funcionario
gestion.configuracion.convocatoria.funcionarios.funcionario.required = Debe seleccionar
el nombre funcionario
```

gestion.configuracion.convocatoria.funcionarios.funcionario.placeholder	= Digite el nombre o el usuario del funcionario
gestion.configuracion.convocatoria.funcionarios.proyecto.info.msg	= A continuación, seleccione los proyectos que desea asignar al funcionario en el módulo de evaluación. En caso de no asignar ningún proyecto, el funcionario podrá ver todos los proyectos de la convocatoria.
gestion.configuracion.convocatoria.funcionarios.proyecto.disponibles.label	= Proyectos Disponibles
gestion.configuracion.convocatoria.funcionarios.proyecto.seleccionados.label	= Proyectos Seleccionados
gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.label	= Proyectos Asignados
gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.header	= Proyectos Asignados en Evaluación
gestion.configuracion.convocatoria.funcionarios.proyecto.required	= Debe seleccionar el proyecto que desea asignar al funcionario
gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder	= Seleccione uno o mas proyectos que desea asignar al funcionario
com.sb.persistence.constraint.ORA-00001.SGPCONVFUNC_UK	= Ya se ha configurado el funcionario seleccionado para esta convocatoria
com.sb.persistence.constraint.ORA-00001.SGPCONVFUNCPRY_UK	= Para el funcionario indicado ya se ha asignado uno de los proyectos seleccionados
##### EN	
#Configuracion - Funcionarios por Convocatoria	
sb.menu.gestion.configuracion.convocatorias.funcionarios	= Evaluation Officials
gestion.configuracion.convocatoria.funcionarios.button	= Officials Evaluation
gestion.configuracion.convocatoria.funcionarios.header	= Officials Evaluation of the Call
gestion.configuracion.convocatoria.funcionarios.funcionario.header	= Official
gestion.configuracion.convocatoria.funcionarios.usuario.header	= User
gestion.configuracion.convocatoria.funcionarios.dependencia.header	= Dependency
gestion.configuracion.convocatoria.funcionarios.proyectos.button	= View Assigned Projects
gestion.configuracion.convocatoria.funcionarios.funcionario.label	= Official
gestion.configuracion.convocatoria.funcionarios.funcionario.required	= You must select the official name
gestion.configuracion.convocatoria.funcionarios.funcionario.placeholder	= Enter the name or username of the official
gestion.configuracion.convocatoria.funcionarios.proyecto.info.msg	= Next, select the projects you want to assign to the official in the evaluation module. If no project is assigned, the official will be able to see all the projects in the call.
gestion.configuracion.convocatoria.funcionarios.proyecto.disponibles.label	= Available Projects
gestion.configuracion.convocatoria.funcionarios.proyecto.seleccionados.label	= Selected Projects
gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.label	= Assigned Projects
gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.header	= Assigned Projects in Evaluation
gestion.configuracion.convocatoria.funcionarios.proyecto.required	= You must select the project you want to assign to the official
gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder	= Select one or more projects that you want to assign to the official
com.sb.persistence.constraint.ORA-00001.SGPCONVFUNC_UK	= The official selected for this call has already been configured

com.sb.persistence.constraint.ORA-00001.SGPCONVFUNCPRY_UK = One of the
selected projects has already been assigned to the indicated official

- Se crea tabla de visualización de la información en la pantalla de Funcionarios

```
<!--PESTAÑA TABLE -->
<p:tab title="#{msg['gestion.configuracion.convocatoria.funcionarios.header']}">
  <!-- TABLA -->
  <div class="row">
    <div class="col-lg-12" style="overflow-x: scroll;">
      <p:dataTable id="#{convocatoriaFuncionarioViewController.modulo}Table"
        var = "convFuncionario"

value="#{convocatoriaFuncionarioViewController.model}"

rowKey="#{convFuncionario.convocatoriaFuncionarioId}"

selection="#{convocatoriaFuncionarioConvController.selected}"
              selectionMode="single"
              paginator="true"
              rows="10"
              lazy="true"
              reflow="true"

sortBy="#{convFuncionario.usuarioId.descripcionUsuario}"
              sortOrder="ASCENDING"
              currentPageReportTemplate="Total de registros:
{totalRecords}"
              paginatorTemplate="{CurrentPageReport}
{FirstPageLink} {PreviousPageLink} {PageLinks} {NextPageLink} {LastPageLink}
{RowsPerPageDropdown}"
              paginatorPosition="bottom"

emptyMessage="#{msg['sb.datatable.registro.empty']}"

widgetVar="items#{convocatoriaFuncionarioViewController.modulo}">
      <f:facet name="header">
        <ui:decorate
template="../../../../template/_tableHeader.xhtml">
          <ui:define name="comandos">
            <ui:param name="sessionController"
value="#{convocatoriaFuncionarioViewController}" />
            <ui:param name="conversationController"
value="#{convocatoriaFuncionarioConvController}" />
            <ui:param name="btnBusqueda" value="N" />
            <ui:param name="buscarVisible"
value="S" />
          </ui:define>
        </ui:decorate>
      </f:facet>
      <p:ajax event="rowSelect"

listener="#{convocatoriaFuncionarioConvController.onRowSelect}"

update="#{convocatoriaFuncionarioViewController.modulo}Table"
              process="@this"/>
      <p:ajax event="rowUnselect"
```

```

        listener="#{convocatoriaFuncionarioConvController.onRowUnselect}"

        update="#{convocatoriaFuncionarioViewController.modulo}Table"
                                process="@this"/>
                <p:column
headerText="#{msg['gestion.configuracion.convocatoria.funcionarios.usuario.header']}"
filterBy="#{convFuncionario.usuarioId.usuario}"
                                filterMatchMode="contains"
                                filterStyle="display:none;"
                                width="15%"
                                >
                        <h:outputText
value="#{convFuncionario.usuarioId.usuario}"/>
                </p:column>
                <p:column
headerText="#{msg['gestion.configuracion.convocatoria.funcionarios.funcionario.header']}"
filterBy="#{convFuncionario.usuarioId.descripcionUsuario}"
                                filterMatchMode="contains"
                                filterStyle="display:none;"
                                >
                        <h:outputText
value="#{convFuncionario.usuarioId.descripcionUsuario}"/>
                </p:column>
                <p:column
headerText="#{msg['gestion.configuracion.convocatoria.funcionarios.dependencia.header']}"
filterBy="#{convFuncionario.usuarioId.dependenciaId.nombreDependencia}"
                                filterMatchMode="contains"
                                filterStyle="display:none;"
                                >
                        <h:outputText
value="#{convFuncionario.usuarioId.dependenciaId.nombreDependencia}"/>
                </p:column>
                <p:column style="text-align: center;"
                                width="20%"
                                >
                        <p:commandLink id="btnProyectosAsignados"

oncomplete="PF('projectsTable').clearFilters();          PF('projectsTable').unselectAllRows();
$('#dlgProyectosAsignados').modal();"
                                styleClass="btn btn-primary"
                                process="@this"

actionListener="#{convocatoriaFuncionarioConvController.cargarProyectosAsignados(convFuncionario)}"

update="formProyectosAsignados, formProyectosAsignados:proyectosAsignadosTable"
                                >
                        <i
                                class="fa
                                fa-tasks"></i>
#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.label']}
                </p:commandLink>
        </p:column>
    </p:dataTable>
</div>
</div>
</p:tab>

```


Pestaña de Visualización de Datos en el Módulo de Funcionarios por Convocatoria

La pestaña de visualización de datos permite mostrar y gestionar los funcionarios asignados a cada convocatoria. A continuación, se detallan los componentes y configuraciones clave de la tabla utilizada en esta pantalla:

- **Tabla de Datos (p:dataTable):** Se utiliza un **p:dataTable** para listar los funcionarios con paginación, filtros y ordenación. La tabla está configurada en modo **"lazy"** para cargar los datos de manera eficiente, una página a la vez.
 - **Propiedades Principales:**
 - **value:** Enlazado al modelo de datos **convocatoriaFuncionarioViewController.model**.
 - **rowKey:** Define una clave única para cada fila, basada en el ID del funcionario (**convocatoriaFuncionarioId**).
 - **selection** y **selectionMode:** Permiten la selección de una única fila para realizar acciones específicas sobre un funcionario seleccionado.
 - **paginator** y **rows:** Configuran la paginación para mostrar 10 registros por página.
 - **sortBy** y **sortOrder:** Ordenan inicialmente los registros en base a la descripción del usuario.
- **Encabezado de la Tabla:** El encabezado incluye una plantilla común **ui:decorate** que utiliza parámetros como **btnBusqueda** y **buscarVisible** para manejar la visibilidad de los botones de búsqueda y acciones.
- **Eventos de Selección (p:ajax):**
 - **rowSelect** y **rowUnselect:** Al seleccionar o deseleccionar una fila, se ejecutan los métodos **onRowSelect** y **onRowUnselect** en el controlador **convocatoriaFuncionarioConvController**, actualizando la tabla según sea necesario.
- **Columnas:**
 - **Usuario:** Muestra el usuario asociado al funcionario, con filtrado por coincidencia (**filterMatchMode="contains"**).

- **Funcionario:** Muestra la descripción del usuario, también con filtrado por coincidencia.
- **Dependencia:** Despliega el nombre de la dependencia asociada al usuario.
- **Proyectos Asignados:** Contiene un botón de comando (**p:commandLink**) que permite abrir un cuadro de diálogo modal para ver los proyectos asignados al funcionario seleccionado. El botón ejecuta el método **cargarProyectosAsignados** y actualiza el contenido del diálogo **formProyectosAsignados**.

- **Diálogo Modal de Proyectos Asignados**

```
<!-- FORM PROYECTOS ASIGNADOS -->
<h:form id="formProyectosAsignados">
  <div class="modal inmodal" id="dlgProyectosAsignados" tabindex="-1" role="dialog" aria-
hidden="true" data-backdrop="static" data-keyboard="false">
    <div class="modal-dialog modal-lg">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-
dismiss="modal"><span
only">Close</span></button>
          <button type="button" class="close" data-
aria-hidden="true">&times;</span><span class="sr-
only">Close</span></button>
          <i class="fa fa-check-square-o modal-icon"></i>
        </div>
        <div class="modal-body form-horizontal">
          <div class="panel panel-info">
            <div class="panel-heading" style="font-size:
18px; text-align: center;">
              #{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.header']}
            </div>
            <div class="panel-body">
              <div class="row">
                <div class="col-lg-12">
                  <dl class="dl-horizontal">
                    <dt>#{msg['gestion.configuracion.convocatoria.funcionarios.usuario.header']}</dt>
                    <dd>#{convocatoriaFuncionarioViewController.convocatoriaFuncionarioSelected.usuarioId.u
suario}</dd>
                    <dt>#{msg['gestion.configuracion.convocatoria.funcionarios.funcionario.header']}</dt>
                    <dd>#{convocatoriaFuncionarioViewController.convocatoriaFuncionarioSelected.usuarioId.d
escripcionUsuario}</dd>
                    <dt>#{msg['gestion.configuracion.convocatoria.funcionarios.dependencia.header']}</dt>
                    <dd>#{convocatoriaFuncionarioViewController.convocatoriaFuncionarioSelected.usuarioId.d
ependenciaId.nombreDependencia}</dd>
                  </dl>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</h:form>
```

```

                                </div>
                                </div>
                                <p:dataTable
id="proyectosAsignadosTable"
                                var
="convFuncionarioPry"
value="#{convocatoriaFuncionarioViewController.listaConvocatoriaFuncionarioProyectos}"
rowKey="#{convFuncionarioPry.convocatoriaFuncionarioProyectoId}"
selectionMode="single"
paginator="true"
                                rows="5"
                                lazy="false"
                                reflow="true"

sortBy="#{convFuncionarioPry.proyectoId.proyectoId}"
sortOrder="ASCENDING"
currentPageReportTemplate="Total de registros: {totalRecords}"
paginatorTemplate="{CurrentPageReport} {FirstPageLink} {PreviousPageLink} {PageLinks}
{NextPageLink} {LastPageLink} {RowsPerPageDropdown}"
paginatorPosition="bottom"
emptyMessage="#{msg['sb.datatable.registro.empty']}"
widgetVar="projectsTable"
filteredValue="#{convocatoriaFuncionarioViewController.listaConvocatoriaFuncionarioProyectos
Filter}"
                                >
                                <f:facet name="header">
                                <p:outputPanel
style="text-align: right;">
                                <h:outputText
value="#{msg['sb.component.tableHeader.buscar']}" />
                                <p:inputText
id="globalFilter" placeholder="#{msg['sb.component.tableHeader.buscar.placeholder']}"
onkeyup="PF('projectsTable').filter()" style="width:300px"/>
                                </p:outputPanel>
                                </f:facet>
                                <p:column
headerText="#{msg['gestion.proyectos.evaluacion.evaluadores.codigoProyecto']}"
filterBy="#{convFuncionarioPry.proyectoId.proyectoId}"
filterMatchMode="contains"
filterStyle="display:none;"
                                style="text-
align: center;"
                                width="20%"
                                >
                                <h:outputText
value="#{convFuncionarioPry.proyectoId.proyectoId}"/>

```

```

                                                                    </p:column>
                                                                    <p:column
headerText="#{msg['gestion.proyectos.evaluacion.proyectos.titulo']}"
filterBy="#{convFuncionarioPry.proyectoId.titulo}"
filterMatchMode="contains"
filterStyle="display:none;"
                                                                    width="80%"
                                                                    >
                                                                    <h:outputText
value="#{convFuncionarioPry.proyectoId.titulo}"/>
                                                                    </p:column>
                                                                    </p:dataTable>
                                                                    </div>
                                                                    </div>
                                                                    </div>
                                                                    <div class="modal-footer">
                                                                    <button type="button" class="btn btn-white" data-
dismiss="modal">#{msg['sb.component.boton.cancelar']}</button>
                                                                    </div>
                                                                    </div>
                                                                    </div>
                                                                    </div>
                                                                    </div>
                                                                    </h:form>

```

Este diálogo modal muestra los proyectos asignados a un funcionario seleccionado en el módulo de configuración de convocatorias. A continuación, se detallan los componentes y configuraciones clave:

- **Estructura del Diálogo (div con clase modal inmodal):** La estructura principal del diálogo utiliza la clase modal inmodal de Bootstrap y contiene configuraciones que lo hacen de tamaño grande (**modal-lg**), con propiedades para evitar que el usuario lo cierre accidentalmente (**data-backdrop="static"** y **data-keyboard="false"**).
- **Encabezado del Diálogo:**
 - Incluye un botón de cierre con íconos visuales y una referencia al texto de cierre para accesibilidad.
 - Utiliza un ícono de fuente (**fa fa-check-square-o modal-icon**) para distinguir visualmente el encabezado.
- **Cuerpo del Diálogo:**
 - **Información del Funcionario (dl-horizontal):** Muestra los datos clave del funcionario seleccionado, como el nombre de usuario, el nombre completo y la

dependencia, utilizando el modelo **convocatoriaFuncionarioViewController.convocatoriaFuncionarioSelected**.

- **Tabla de Proyectos Asignados (p:dataTable):** Muestra los proyectos asignados al funcionario en una tabla con paginación y opciones de filtrado global.

- **Propiedades de la Tabla:**

- **value:** Enlazada a **convocatoriaFuncionarioViewController.listaConvocatoriaFuncionarioProyectos**, que contiene los proyectos asignados.
- **rowKey:** Define una clave única para cada proyecto basado en el ID **convocatoriaFuncionarioProyectoId**.
- **selectionMode:** Configurada en **"single"** para permitir la selección de un único proyecto.
- **paginator** y **rows:** Configurada para mostrar cinco registros por página.
- **sortBy** y **sortOrder:** Ordenan inicialmente la lista de proyectos por el ID del proyecto (**proyectoId**) en orden ascendente.
- **filteredValue:** Asigna un filtro a la lista de proyectos (**listaConvocatoriaFuncionarioProyectosFilter**).

- **Encabezado de la Tabla:**

- Incluye un campo de filtro global (**p:inputText**) que permite realizar búsquedas en toda la tabla.

- **Columnas de la Tabla:**

- **Código de Proyecto:** Muestra el código único del proyecto, con opciones de filtrado ocultas.
- **Título del Proyecto:** Muestra el título del proyecto y también incluye opciones de filtrado ocultas.

- **Pie del Diálogo (modal-footer):** Incluye un botón de "Cancelar" (**btn-white**) que permite cerrar el diálogo sin realizar ninguna acción.

- Se crea formulario de captura de datos

```
<!--PESTAÑA GUARDAR/EDITAR -->
<p:tab title="#{msg['sb.component.tabview.guardarEditar']}">
  <p:panel id="formularioEditarCrear">
    <div class="ibox-content">
      <div class="form-horizontal">
        <div class="row">
          <div class="col-lg-12">
            <div class="form-group">
              <p:outputLabel
value="#{msg['gestion.configuracion.convocatoria.funcionarios.funcionario.label']}"
styleClass="col-lg-3 control-label"
for="user-
CF"
align: right"/>
              <div class="col-lg-8">
                <p:autoComplete id="user-CF"

                value="#{convocatoriaFuncionarioViewController.selected.usuarioId}"

                converter="usuarioConverter"

                completeMethod="#{convocatoriaFuncionarioConvController.completeUsers}"

                var="user"

                placeholder="#{msg['gestion.configuracion.convocatoria.funcionarios.funcionario.placeho
lder']}"

                forceSelection="true"

                queryDelay="1000"

                minQueryLength="3"

                itemLabel="#{user.usuarioDescripcion}"

                itemValue="#{user}"

                style="width: 100%"

                inputStyleClass="form-control"

                required="true"

                requiredMessage="#{msg['gestion.configuracion.convocatoria.funcionarios.funcionario.req
uired']}"

                >
              </p:autoComplete>
            </div>
          </div>
        </div>
      </div>
      <p:separator/>
    </div class="row">
  </p:panel>
</p:tab>
```

```

        <div class="col-lg-12">
            <div class="alert alert-info alert-dismissible">
                <b> <i class="fa fa-info-circle"></i>
            </b>
            #{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.info.msg']}
            </b>
        </div>
    </div>
</div>
<div class="panel-body">
    <p:growl id="msg" showDetail="true"
skipDetailIfEqualsSummary="true" escape="false"/>
    <div class="row">
        <div class="col-lg-12">
            <div class="form-group">
                <p:outputLabel
value="#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.label']}"
styleClass="col-lg-3 control-label"
for="autoCompleteProyectos"
            </p:outputLabel>
            <div class="col-lg-8">
                <p:autoComplete
id="autoCompleteProyectos"
                value="#{convocatoriaFuncionarioViewController.selectedProyectos}"
                completeMethod="#{convocatoriaFuncionarioViewController.buscarProyectos}"
                var="proyecto"
                itemLabel="#{proyecto.proyectoId} - #{proyecto.titulo}"
                itemValue="#{proyecto}"
                multiple="true"
                forceSelection="true"
                size="50"
                dropdown="true"
                scrollHeight="300"
                inputStyleClass="form-control"
                style="width: 100%"
                minQueryLength="3"
                required="false"
                requiredMessage="#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder']}"
                placeholder="#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder']}"
            </p:autoComplete>
            </div>
        </div>
    </div>
</div>

```

```

        converter="proyectoConverter"
    >
event="itemSelect" update="msg" />
<p:ajax
</p:ajax>
</p:autoComplete>
</div>
</div>
</div>
</div>
</div>
<br/>
<div class="hr-line-dashed"></div>
<ui:decorate template="../../template/_formButtons.xhtml"
rendered="false">
    <ui:define name="comandos">
        <ui:param name="sessionController"
value="#{convocatoriaFuncionarioViewController}" />
        <ui:param name="conversationController"
value="#{convocatoriaFuncionarioConvController}" />
    </ui:define>
    </ui:decorate>
</div>
</div>
</p:panel>
</p:tab>

```

Formulario de Captura de Datos

Este formulario, ubicado dentro de la pestaña "Guardar/Editar", permite seleccionar y asignar proyectos a un funcionario en el módulo de configuración de convocatorias. A continuación, se detallan los componentes y sus configuraciones principales:

- **Estructura General:**
 - El formulario está envuelto en un panel PrimeFaces (**p:panel**) para agrupar y organizar los campos de entrada de datos.
- **Campo de Selección de Funcionario (p:autoComplete):**
 - **Label:** Se utiliza **p:outputLabel** para mostrar la etiqueta del campo, con alineación a la derecha.
 - **Componente p:autoComplete:**
 - **value:** Enlazado a **convocatoriaFuncionarioViewController.selected.usuariold**, donde se almacena el usuario seleccionado.

- **converter:** Usa un conversor personalizado (**usuarioConverter**) para convertir objetos **Usuario** entre la presentación y el modelo.
- **completeMethod:** Define el método **convocatoriaFuncionarioConvController.completeUsers** para autocompletar sugerencias de usuarios basadas en el texto ingresado.
- **var, itemLabel y itemValue:** Controlan la visualización de las sugerencias y el valor seleccionado.
- **forceSelection:** Requiere que el usuario seleccione uno de los valores sugeridos.
- **required:** Indica que el campo es obligatorio, y muestra un mensaje de error personalizado si el usuario no selecciona un funcionario.
- **minQueryLength y queryDelay:** Configuran el tiempo de espera y el mínimo de caracteres antes de realizar la búsqueda.
- **Mensaje Informativo:**
 - **Componente alert:** Incluye un mensaje de tipo alerta (**alert-info**) que proporciona información adicional relevante para el usuario sobre la asignación de proyectos.
- **Campo de Selección de Proyectos (p:autoComplete):**
 - **Label:** Se usa **p:outputLabel** para la etiqueta de este campo, alineado con el control **col-lg-3 control-label**.
 - **Componente p:autoComplete:**
 - **value:** Vinculado a **convocatoriaFuncionarioViewController.selectedProyectos**, donde se almacenan los proyectos seleccionados.
 - **completeMethod:** Método **convocatoriaFuncionarioViewController.buscarProyectos** para autocompletar proyectos en base al texto ingresado.
 - **multiple:** Permite seleccionar múltiples proyectos.

- **itemLabel** y **itemValue**: Controlan la visualización de cada proyecto en el formato proyectold - título.
 - **forceSelection**, **dropdown** y **scrollHeight**: Configuran la selección de valores sugeridos, así como el estilo y funcionalidad del menú desplegable.
 - **converter**: Usa un conversor personalizado (**proyectoConverter**) para convertir entre la presentación y el modelo.
 - **placeholder**: Muestra un texto de sugerencia en el campo.
 - **p:ajax**: Permite actualizar el componente **msg** en el evento **itemSelect**, mostrando un mensaje de confirmación o alerta al seleccionar un proyecto.
- **Botones de Acción (ui:decorate)**:
 - **Plantilla**: Usa un archivo de plantilla externo (**_formButtons.xhtml**) para los botones de acción, como guardar o cancelar, manteniendo la consistencia y reusabilidad en toda la aplicación.

Ajuste Acceso Evaluación de Proyectos/Programas por Funcionarios Internos

Para que los funcionarios internos de Minciencias puedan acceder a los proyectos que están en etapa de evaluación, se debe validar que estos solo puedan ver aquellos proyectos que tienen asignados y permitidos gestionar su evaluación. Se agregan validaciones a los métodos de consulta de proyectos para tener en cuenta esta lógica.

- Metodo **LazyDataModel** en controlador de vista **ProyectoEvaluacionSessController**

```
@PostConstruct
public void init() {
    this.model = new LazyDataModel<ProyectoDTO>() {
        @Override
        public List<ProyectoDTO> load(int first, int pageSize, String sortField,
SortOrder sortOrder, Map<String, FilterMeta> filterBy) {
            Map<String, Object> filters = new TreeMap<>();
            filterBy.forEach((k, v) -> {
                if (k.equals("globalFilter")) {
                    if (v.getFilterValue() != null &&
!v.getFilterValue().toString().isEmpty()) {
                        filters.put(k, v.getFilterValue());
                    }
                }
            });
        }
    };
}
```

```

    }
    });
    List<ProyectoDTO> proyectoList = new ArrayList<>();
    try {
        List<Long> estadosFiltro = new ArrayList<>();

        estadosFiltro.add(listaEstados.get("REVISADO").getValorDefinicionId());
        estadosFiltro.add(listaEstados.get("EVALUADO").getValorDefinicionId());
        estadosFiltro.add(listaEstados.get("SEGUIMIENTO").getValorDefinicionId());
        estadosFiltro.add(listaEstados.get("NEGADO").getValorDefinicionId());
        estadosFiltro.add(listaEstados.get("APROBADO").getValorDefinicionId());
        // Filtro por usuario conectado (NO APLICA PARA ADMINISTRADORES)
        if
(!seguridadSessController.getListaGruposUsuario().contains("SGPS_ADMINISTRADOR")) {
            System.out.println("// Filtro por usuario conectado");
            filters.put("usuarioConvocatoriaFuncionario",
seguridadSessController.getSelected().getUsuario());
        }
        int count = proyectosServiceFacade.countProyectos(filters,
estadosFiltro, proyectoFilter, false);
        if (count > 0) {
            proyectoList
proyectosServiceFacade.obtenerTodosProyectos(first, pageSize, sortField,
sortOrder.equals(SortOrder.ASCENDING) ? "A" : "D", filters, estadosFiltro, proyectoFilter);
        }
        model.setRowCount(count);
    } catch (Exception ex) {
        LOG.error("Error al cargar el listado de proyectos en
evaluacion", ex);
    }
    return proyectoList;
}

};
}

```

En este código, se ha implementado un filtro en el método **load** de **LazyDataModel** para que solo los proyectos correspondientes al funcionario conectado sean visibles, excluyendo a los administradores.

Explicación del Código

1. Inicialización del LazyDataModel:

- En el método **@PostConstruct init()**, se crea una instancia de **LazyDataModel<ProyectoDTO>** para gestionar la carga de los proyectos de manera paginada y eficiente.

2. Método load:

- Este método sobrescribe el método **load** de **LazyDataModel** y se llama automáticamente cuando se requiere cargar un conjunto específico de datos, por ejemplo, al navegar entre páginas o aplicar filtros en la tabla.

3. Filtrado Global (globalFilter):

- Utiliza **filterBy** para recorrer todos los filtros aplicados. Si se encuentra una clave **globalFilter**, se agrega su valor al **filters** si no está vacío.

4. Definición de Estados Permitidos:

- Se crea una lista **estadosFiltro** que incluye solo los estados de proyectos relevantes para la evaluación (por ejemplo, "REVISADO", "EVALUADO", etc.). Esto asegura que solo los proyectos en estos estados serán consultados.

5. Filtro por Usuario Conectado:

- Se aplica una condición para excluir administradores (usuarios que pertenecen al grupo **SGPS_ADMINISTRADOR**) de esta restricción.
- Si el usuario conectado no es un administrador, se añade un filtro específico a **filters** para limitar los resultados al usuario en sesión:

```
filters.put("usuarioConvocatoriaFuncionario",  
seguridadSessController.getSelected().getUsuario());
```

- Aquí, **seguridadSessController.getSelected().getUsuario()** obtiene el usuario actualmente conectado para filtrar los proyectos asociados únicamente a él.

6. Consulta de Cantidad de Proyectos:

- Se llama a **proyectosServiceFacade.countProyectos** pasando los filtros (**filters**), los estados permitidos (**estadosFiltro**) y otros parámetros para obtener el número total de proyectos que cumplen los criterios.

7. Consulta Paginada de Proyectos:

- Si la cantidad de proyectos (**count**) es mayor que cero, se llama a **proyectosServiceFacade.obtenerTodosProyectos**, que realiza la consulta paginada de proyectos con los filtros aplicados.

- El resultado se almacena en proyectoList y la cantidad de proyectos (**count**) se asigna a **model.setRowCount(count)** para el manejo de la paginación.

Consideraciones

- **Exclusión de Administradores:**
 - La lógica que excluye a los administradores permite que estos usuarios vean todos los proyectos, sin restricciones basadas en el usuario conectado.
- **Uso de LazyDataModel:**
 - **LazyDataModel** facilita la carga de datos en pequeños conjuntos (**pageSize**), lo cual mejora el rendimiento y evita cargar todos los registros al mismo tiempo.
- Ajuste método **CriteriaBuilder** de consulta de proyectos en **ProyectosStoreManagerImpl**, agregando el filtro por usuario convocatoria funcionario asignado

```
//Filtro por usuario convocatoria funcionario asignado
Object usuarioConvocatoriaFuncionario = filters.get("usuarioConvocatoriaFuncionario");
if (usuarioConvocatoriaFuncionario != null) {
    String usuarioFuncionario = (String) usuarioConvocatoriaFuncionario;
    /**
     * Manejar el escenario donde un funcionario asignado a una
     * convocatoria puede ver todos los proyectos si no tiene
     * proyectos específicos asignados, pero solo puede ver esos
     * proyectos específicos si tiene asignaciones, podemos usar la
     * siguiente lógica:
     */
    List<Long> listaIdsProyectos = obtenerProyectosFuncionarios(usuarioFuncionario);
    List<Long> listaConvocatoriasSinProyectos =
obtenerConvocatoriasSinProyectosAsignados(usuarioFuncionario);

    if ((listaIdsProyectos != null && !listaIdsProyectos.isEmpty())
        || (listaConvocatoriasSinProyectos != null &&
!listaConvocatoriasSinProyectos.isEmpty())) {

        Predicate predicateProyectos =
projecto.get(Projecto_.projectId).in(listaIdsProyectos);
        Predicate predicateConvocatorias =
projectoConvocatoria.get(Convocatoria_.convocatoriaId).in(listaConvocatoriasSinProyectos);

        if (listaIdsProyectos != null && !listaIdsProyectos.isEmpty() &&
listaConvocatoriasSinProyectos != null && !listaConvocatoriasSinProyectos.isEmpty()) {
            lista.add(cb.or(predicateProyectos, predicateConvocatorias));
        } else if (listaIdsProyectos != null && !listaIdsProyectos.isEmpty()) {
            lista.add(predicateProyectos);
        } else {
            lista.add(predicateConvocatorias);
        }
    }
}
```

```
    } else {  
        // el funcionario no puede ver ningun proyecto  
        lista.add(cb.isNull(proyecto.get("proyectoId")));  
    }  
}
```

El filtro por "**usuarioConvocatoriaFuncionario**" en este código está diseñado para limitar los proyectos que un funcionario puede ver en función de sus asignaciones. La idea principal es permitir que un funcionario vea solo los proyectos específicos asignados a él o todos los proyectos de una convocatoria si no tiene asignaciones específicas.

1. Obtener IDs de Proyectos y Convocatorias

- Se invocan dos métodos:
 - **obtenerProyectosFuncionarios(usuarioFuncionario)**: obtiene una lista de IDs de proyectos específicos asignados al funcionario.
 - **obtenerConvocatoriasSinProyectosAsignados(usuarioFuncionario)**: obtiene una lista de convocatorias en las que el funcionario tiene permiso general pero sin asignaciones específicas de proyectos.

Estas listas (**listIdsProyectos** y **listaConvocatoriasSinProyectos**) determinan los proyectos o convocatorias a los que el funcionario tiene acceso.

2. Construcción del Filtro

- Si al menos una de las listas (**listIdsProyectos** o **listaConvocatoriasSinProyectos**) tiene elementos, se construyen los siguientes Predicate:
 - **predicateProyectos**: usa **proyecto.get(Proyecto_.projectId).in(listIdsProyectos)**, limitando los resultados a los IDs de proyectos asignados al funcionario.
 - **predicateConvocatorias**: utiliza **proyectoConvocatoria.get(Convocatoria_.convocatoriaId).in(listaConvocatoriasSinProyectos)**, limitando los resultados a los IDs de convocatorias sin asignaciones de proyectos específicas para el funcionario.

Luego, se usa una condición OR para combinar ambos predicados:

- **Condición OR:** si existen tanto **listaldsProyectos** como **listaConvocatoriasSinProyectos**, el funcionario verá todos los proyectos específicos asignados o todos los proyectos de las convocatorias en las que tiene acceso.
- **Condiciones Individuales:**
 - Si solo hay proyectos específicos (**listaldsProyectos**), solo se usa **predicateProyectos**.
 - Si solo hay convocatorias (**listaConvocatoriasSinProyectos**), solo se usa **predicateConvocatorias**.

3. Restricción para Funcionarios sin Asignaciones

- Si el funcionario no tiene proyectos ni convocatorias asignadas, el código agrega **cb.isNull(proyecto.get("proyectold"))** a la lista de predicados. Esta condición asegura que el funcionario **no pueda ver ningún proyecto** si no tiene permisos asignados.

Método obtenerProyectosFuncionarios

```
private List<Long> obtenerProyectosFuncionarios(String usuarioFuncionario) {  
    Query query = em.createQuery("SELECT cfp.proyectoId.proyectoId FROM  
ConvocatoriaFuncionarioProyecto cfp "  
        + " WHERE cfp.convocatoriaFuncionarioId.usuarioId.usuario = :usuario"  
        + " ORDER BY cfp.proyectoId.proyectoId");  
    query.setParameter("usuario", usuarioFuncionario);  
    return query.getResultList();  
}
```

- Este método obtiene una lista de IDs de proyectos (proyectold) que están específicamente asignados al funcionario.
- La consulta usa la entidad ConvocatoriaFuncionarioProyecto y selecciona los proyectos a los que el funcionario tiene acceso directo a través de la relación convocatoriaFuncionarioId.usuarioId.

- **Consulta JPQL:**

Esta consulta asegura que solo se obtengan los proyectos específicos que fueron asignados al funcionario en particular, y se ordenan por proyectold para mantener un orden predecible.

Método obtenerConvocatoriasSinProyectosAsignados

```
private List<Long> obtenerConvocatoriasSinProyectosAsignados(String usuarioFuncionario) {  
    Query query = em.createQuery("SELECT cf.convocatoriaId.convocatoriaId FROM  
ConvocatoriaFuncionario cf "  
        + " WHERE cf.usuarioId.usuario = :usuario"  
        + " AND NOT EXISTS ("  
        + "     SELECT 1 FROM ConvocatoriaFuncionarioProyecto cfp"  
        + "     WHERE cfp.convocatoriaFuncionarioId = cf)"  
        + " ORDER BY cf.convocatoriaId.convocatoriaId");  
    query.setParameter("usuario", usuarioFuncionario);  
    return query.getResultList();  
}
```

- Este método obtiene una lista de convocatorias (**convocatoriaId**) para las cuales el funcionario tiene acceso general, pero sin proyectos específicos asignados.
- Utiliza la entidad **ConvocatoriaFuncionario** para obtener las convocatorias en las que el funcionario está registrado.
- El filtro **NOT EXISTS** excluye convocatorias donde ya existen proyectos específicos asignados al funcionario en ConvocatoriaFuncionarioProyecto.
- **Consulta JPQL:**

Esta consulta devuelve convocatorias en las que el funcionario tiene permiso general, pero donde no hay proyectos específicos asignados. Esto permite que el funcionario vea todos los proyectos en esas convocatorias.

Cómo se Integran en el Filtro

Al combinar los resultados de estos métodos en el filtro:

- Si **obtenerProyectosFuncionarios** retorna IDs, el funcionario podrá ver solo esos proyectos específicos.
- Si **obtenerConvocatoriasSinProyectosAsignados** retorna convocatorias, el funcionario verá todos los proyectos de esas convocatorias, dado que no tiene proyectos específicos asignados en ellas.
- Si ambas listas tienen datos, el filtro permite acceso a los proyectos específicos **y** a los proyectos de las convocatorias sin asignaciones específicas.

Esto garantiza que los funcionarios puedan acceder únicamente a los proyectos asignados de manera granular, cumpliendo con los permisos definidos.

Ajuste Acceso Evaluacion de Beneficios Tributarios por Funcionarios Internos

Para que los funcionarios internos de Minciencias puedan acceder a los proyectos de beneficios tributarios que están en etapa de evaluación, se debe validar que estos solo puedan ver aquellos proyectos que tienen asignados y permitidos gestionar su evaluación. Se agregan validaciones a los métodos de consulta de proyectos para tener en cuenta esta lógica.

- Metodo **LazyDataModel** en controlador de vista
ProyectoBeneficiosEvaluadoresSessController

```
@PostConstruct
public void init() {
    this.model = new LazyDataModel<ProyectoDTO>() {

        @Override
        public List<ProyectoDTO> load(int first, int pageSize, String sortField,
SortOrder sortOrder, Map<String, FilterMeta> filterBy) {
            Map<String, Object> filters = new TreeMap<>();
            filterBy.forEach((k, v) -> {
                if (k.equals("globalFilter")) {
                    if (v.getFilterValue() != null &&
!v.getFilterValue().toString().isEmpty()) {
                        filters.put(k, v.getFilterValue());
                    }
                }
            });
            List<ProyectoDTO> proyectoList = new ArrayList<>();
            try {
                List<Long> estadosFiltro = new ArrayList<>();

                estadosFiltro.add(listaEstados.get("REVISADO").getValorDefinicionId());
                estadosFiltro.add(listaEstados.get("EVALUADO").getValorDefinicionId());
                estadosFiltro.add(listaEstados.get("SEGUIMIENTO").getValorDefinicionId());
                estadosFiltro.add(listaEstados.get("NEGADO").getValorDefinicionId());
                estadosFiltro.add(listaEstados.get("APROBADO").getValorDefinicionId());
                // Filtro por usuario conectado (NO APLICA PARA ADMINISTRADORES)
                if
(!seguridadSessController.getListasGruposUsuario().contains("SGPS_ADMINISTRADOR")) {
                    System.out.println("// Filtro por usuario conectado");
                    filters.put("usuarioConvocatoriaFuncionario",
seguridadSessController.getSelected().getUsuario());
                }
                int count = beneficiosServiceFacade.countProyectos(filters,
estadosFiltro, proyectoFilter, true);
                if (count > 0) {
                    proyectoList
                    =
beneficiosServiceFacade.obtenerProyectos(first, pageSize, sortField,
sortOrder.equals(SortOrder.ASCENDING) ? "A" : "D", filters, estadosFiltro, proyectoFilter);
                }
            }
        }
    };
}
```

```
        model.setRowCount(count);
    } catch (Exception ex) {

        Logger.getLogger(ProyectoDecisionSessController.class.getName()).log(Level.SEVERE,
        null, ex);
    }
    return proyectoList;
}
};
}
```

En este código, se ha implementado un filtro en el método **load** de **LazyDataModel** para que solo los proyectos correspondientes al funcionario conectado sean visibles, excluyendo a los administradores.

Explicación del Código

8. Inicialización del LazyDataModel:

- En el método **@PostConstruct init()**, se crea una instancia de **LazyDataModel<ProyectoDTO>** para gestionar la carga de los proyectos de manera paginada y eficiente.

9. Método load:

- Este método sobrescribe el método **load** de **LazyDataModel** y se llama automáticamente cuando se requiere cargar un conjunto específico de datos, por ejemplo, al navegar entre páginas o aplicar filtros en la tabla.

10. Filtrado Global (globalFilter):

- Utiliza **filterBy** para recorrer todos los filtros aplicados. Si se encuentra una clave **globalFilter**, se agrega su valor al **filters** si no está vacío.

11. Definición de Estados Permitidos:

- Se crea una lista **estadosFiltro** que incluye solo los estados de proyectos relevantes para la evaluación (por ejemplo, "REVISADO", "EVALUADO", etc.). Esto asegura que solo los proyectos en estos estados serán consultados.

12. Filtro por Usuario Conectado:

- Se aplica una condición para excluir administradores (usuarios que pertenecen al grupo **SGPS_ADMINISTRADOR**) de esta restricción.

- Si el usuario conectado no es un administrador, se añade un filtro específico a **filters** para limitar los resultados al usuario en sesión:

```
filters.put("usuarioConvocatoriaFuncionario",  
seguridadSessController.getSelected().getUsuario());
```

- Aquí, **seguridadSessController.getSelected().getUsuario()** obtiene el usuario actualmente conectado para filtrar los proyectos asociados únicamente a él.

13. Consulta de Cantidad de Proyectos:

- Se llama a **beneficiosServiceFacade.countProyectos** pasando los filtros (**filters**), los estados permitidos (**estadosFiltro**) y otros parámetros para obtener el número total de proyectos que cumplen los criterios.

14. Consulta Paginada de Proyectos:

- Si la cantidad de proyectos (**count**) es mayor que cero, se llama a **beneficiosServiceFacade.obtenerTodosProyectos**, que realiza la consulta paginada de proyectos con los filtros aplicados.
- El resultado se almacena en **proyectoList** y la cantidad de proyectos (**count**) se asigna a **model.setRowCount(count)** para el manejo de la paginación.

Consideraciones

- **Exclusión de Administradores:**
 - La lógica que excluye a los administradores permite que estos usuarios vean todos los proyectos, sin restricciones basadas en el usuario conectado.
- **Uso de LazyDataModel:**
 - **LazyDataModel** facilita la carga de datos en pequeños conjuntos (**pageSize**), lo cual mejora el rendimiento y evita cargar todos los registros al mismo tiempo.
- Ajuste método **CriteriaBuilder** de consulta de proyectos en **BeneficiosStoreManagerImpl**, agregando el filtro por usuario convocatoria funcionario asignado

```
//Filtro por usuario convocatoria funcionario asignado
Object usuarioConvocatoriaFuncionario = filters.get("usuarioConvocatoriaFuncionario");
if (usuarioConvocatoriaFuncionario != null) {
    String usuarioFuncionario = (String) usuarioConvocatoriaFuncionario;
    /**
     * Manejar el escenario donde un funcionario asignado a una
     * convocatoria puede ver todos los proyectos si no tiene
     * proyectos específicos asignados, pero solo puede ver esos
     * proyectos específicos si tiene asignaciones, podemos usar la
     * siguiente lógica:
     */
    List<Long> listaIdsProyectos = obtenerProyectosFuncionarios(usuarioFuncionario);

    List<Long> listaConvocatoriasSinProyectos =
obtenerConvocatoriasSinProyectosAsignados(usuarioFuncionario);

    if ((listaIdsProyectos != null && !listaIdsProyectos.isEmpty())
        || (listaConvocatoriasSinProyectos != null &&
!listaConvocatoriasSinProyectos.isEmpty())) {

        Predicate predicateProyectos =
projecto.get(Proyecto_.projectId).in(listaIdsProyectos);
        Predicate predicateConvocatorias =
projectoConvocatoria.get(Convocatoria_.convocatoriaId).in(listaConvocatoriasSinProyectos);

        if (listaIdsProyectos != null && !listaIdsProyectos.isEmpty() &&
listaConvocatoriasSinProyectos != null && !listaConvocatoriasSinProyectos.isEmpty()) {
            lista.add(cb.or(predicateProyectos, predicateConvocatorias));
        } else if (listaIdsProyectos != null && !listaIdsProyectos.isEmpty()) {
            lista.add(predicateProyectos);
        } else {
            lista.add(predicateConvocatorias);
        }
    } else {
        // el funcionario no puede ver ningun proyecto
        lista.add(cb.isNull(projecto.get("projectId")));
    }
}
```

El filtro por "**usuarioConvocatoriaFuncionario**" en este código está diseñado para limitar los proyectos que un funcionario puede ver en función de sus asignaciones. La idea principal es permitir que un funcionario vea solo los proyectos específicos asignados a él o todos los proyectos de una convocatoria si no tiene asignaciones específicas.

1. Obtener IDs de Proyectos y Convocatorias

- Se invocan dos métodos:
 - **obtenerProyectosFuncionarios(usuarioFuncionario)**: obtiene una lista de IDs de proyectos específicos asignados al funcionario.

- **obtenerConvocatoriasSinProyectosAsignados(usuarioFuncionario):** obtiene una lista de convocatorias en las que el funcionario tiene permiso general pero sin asignaciones específicas de proyectos.

Estas listas (**listaldsProyectos** y **listaConvocatoriasSinProyectos**) determinan los proyectos o convocatorias a los que el funcionario tiene acceso.

2. Construcción del Filtro

- Si al menos una de las listas (**listaldsProyectos** o **listaConvocatoriasSinProyectos**) tiene elementos, se construyen los siguientes Predicate:
 - **predicateProyectos:** usa **proyecto.get(Proyecto_.projectId).in(listaldsProyectos)**, limitando los resultados a los IDs de proyectos asignados al funcionario.
 - **predicateConvocatorias:** utiliza **proyectoConvocatoria.get(Convocatoria_.convocatoriald).in(listaConvocatoriasSinProyectos)**, limitando los resultados a los IDs de convocatorias sin asignaciones de proyectos específicas para el funcionario.

Luego, se usa una condición **OR** para combinar ambos predicados:

- **Condición OR:** si existen tanto **listaldsProyectos** como **listaConvocatoriasSinProyectos**, el funcionario verá todos los proyectos específicos asignados o todos los proyectos de las convocatorias en las que tiene acceso.
- **Condiciones Individuales:**
 - Si solo hay proyectos específicos (**listaldsProyectos**), solo se usa **predicateProyectos**.
 - Si solo hay convocatorias (**listaConvocatoriasSinProyectos**), solo se usa **predicateConvocatorias**.

3. Restricción para Funcionarios sin Asignaciones

- Si el funcionario no tiene proyectos ni convocatorias asignadas, el código agrega **cb.isNull(proyecto.get("projectId"))** a la lista de predicados. Esta condición asegura que el funcionario **no pueda ver ningún proyecto** si no tiene permisos asignados.

Método obtenerProyectosFuncionarios

```
private List<Long> obtenerProyectosFuncionarios(String usuarioFuncionario) {
    Query query = em.createQuery("SELECT cfp.proyectoId.proyectoId FROM
ConvocatoriaFuncionarioProyecto cfp "
        + " WHERE cfp.convocatoriaFuncionarioId.usuarioId.usuario = :usuario"
        + " ORDER BY cfp.proyectoId.proyectoId");
    query.setParameter("usuario", usuarioFuncionario);
    return query.getResultList();
}
```

- Este método obtiene una lista de IDs de proyectos (proyectoId) que están específicamente asignados al funcionario.
- La consulta usa la entidad ConvocatoriaFuncionarioProyecto y selecciona los proyectos a los que el funcionario tiene acceso directo a través de la relación convocatoriaFuncionarioId.usuarioId.
- **Consulta JPQL:**

Esta consulta asegura que solo se obtengan los proyectos específicos que fueron asignados al funcionario en particular, y se ordenan por proyectoId para mantener un orden predecible.

Método obtenerConvocatoriasSinProyectosAsignados

```
private List<Long> obtenerConvocatoriasSinProyectosAsignados(String usuarioFuncionario) {
    Query query = em.createQuery("SELECT cf.convocatoriaId.convocatoriaId FROM
ConvocatoriaFuncionario cf "
        + " WHERE cf.usuarioId.usuario = :usuario"
        + " AND NOT EXISTS ("
        + "     SELECT 1 FROM ConvocatoriaFuncionarioProyecto cfp"
        + "     WHERE cfp.convocatoriaFuncionarioId = cf)"
        + " ORDER BY cf.convocatoriaId.convocatoriaId");
    query.setParameter("usuario", usuarioFuncionario);
    return query.getResultList();
}
```

- Este método obtiene una lista de convocatorias (**convocatoriaId**) para las cuales el funcionario tiene acceso general, pero sin proyectos específicos asignados.
- Utiliza la entidad **ConvocatoriaFuncionario** para obtener las convocatorias en las que el funcionario está registrado.
- El filtro **NOT EXISTS** excluye convocatorias donde ya existen proyectos específicos asignados al funcionario en **ConvocatoriaFuncionarioProyecto**.

- **Consulta JPQL:**

Esta consulta devuelve convocatorias en las que el funcionario tiene permiso general, pero donde no hay proyectos específicos asignados. Esto permite que el funcionario vea todos los proyectos en esas convocatorias.

Cómo se Integran en el Filtro

Al combinar los resultados de estos métodos en el filtro:

- Si `obtenerProyectosFuncionarios` retorna IDs, el funcionario podrá ver solo esos proyectos específicos.
- Si `obtenerConvocatoriasSinProyectosAsignados` retorna convocatorias, el funcionario verá todos los proyectos de esas convocatorias, dado que no tiene proyectos específicos asignados en ellas.
- Si ambas listas tienen datos, el filtro permite acceso a los proyectos específicos y a los proyectos de las convocatorias sin asignaciones específicas.

Esto garantiza que los funcionarios puedan acceder únicamente a los proyectos asignados de manera granular, cumpliendo con los permisos definidos.

Reclamaciones Panel Evaluacion Proyectos

Para este requerimiento se solicitó implementar en el módulo de evaluación de proyectos/programas la sección de "Reclamaciones" cuando el tipo de evaluación sea "Panel", permitiendo indicar si presenta fallo judicial y la opción de adjuntar documento del fallo

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

- Se mapea nuevo campo "**FALLO_JUDICIAL**" en la tabla **PRY_EVALUACIONES**
- En el módulo de evaluación de proyectos, crear nueva sección de reclamaciones, carga y descarga de documento de fallo judicial
- Se crean etiquetas de internacionalización
- Creación de nuevo campo en la tabla **PRY_EVALUACIONES**

-- Add/modify columns

```
alter table PRY_EVALUACIONES add fallo_judicial VARCHAR2(1);
-- Add comments to the columns
comment on column PRY_EVALUACIONES.fallo_judicial
  is 'Indica si el usuario presenta fallo judicial (S/N)';
-- Create/Recreate check constraints
alter table PRY_EVALUACIONES
  add constraint PRYEVA_FALLO_JUDICIAL_CK
  check (FALLO_JUDICIAL IN ('S'/*Si*/, 'N'/*No*/));
```

- creación de nuevo valor definición para el tipo de documento de reclamaciones

```
-- Nuevo documento de reclamaciones
insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
values (null, 'TIPDOCREC', '520', 'Documento Adjunto Reclamaciones del panel', 'S', null,
null, null, null, 'N', null, 'SIGP_WEB', to_date('02-07-2024', 'dd-mm-yyyy'), null, null,
null, null, null, null, null, null);
```

- Mapear nuevo campo en la entidad de la evaluación de proyectos (ProyectoEvaluacion.java)

```
@Column(name = "FALLO_JUDICIAL")
private String falloJudicial;

public String getFalloJudicial() {
    return falloJudicial;
}

public void setFalloJudicial(String falloJudicial) {
    this.falloJudicial = falloJudicial;
}
```

- Mapear nuevo campo en DTO de la evaluación de proyectos (ProyectoEvaluacion DTO.java)

```
@Bind(name = "falloJudicial")
private String falloJudicial;

//Campos checkBox
private Boolean falloJudicialBool;

public String getFalloJudicial() {
    return falloJudicial;
}

public void setFalloJudicial(String falloJudicial) {
    if (falloJudicial != null && falloJudicial.equals("S")) {
        this.falloJudicialBool = true;
    }
}
```



```
    }  
    this.falloJudicial = falloJudicial;  
}  
  
public Boolean getFalloJudicialBool() {  
    return falloJudicialBool;  
}  
  
public void setFalloJudicialBool(Boolean falloJudicialBool) {  
    if (falloJudicialBool) {  
        this.falloJudicial = "S";  
    } else {  
        this.falloJudicial = "N";  
    }  
    this.falloJudicialBool = falloJudicialBool;  
}
```

Descripción de los Campos Mapeados

1. Campo falloJudicial

- Es un campo String que representa si el proyecto tiene un fallo judicial.
- El valor **"S"** indica que el proyecto presenta un fallo judicial, mientras que el valor **"N"** indica lo contrario.

2. Campo falloJudicialBool

- Este campo Boolean representa de forma simplificada si el proyecto presenta fallo judicial. Es un "checkbox" en la interfaz que el usuario puede marcar o desmarcar.
- Si el campo falloJudicialBool es true, significa que el proyecto tiene un fallo judicial; en caso contrario, no lo tiene.

Métodos Getter y Setter

1. Getter y Setter de falloJudicial

- El método **getFalloJudicial()** retorna el valor almacenado en el campo **falloJudicial**.
- El método **setFalloJudicial(String falloJudicial)** asigna el valor proporcionado a **falloJudicial**.
- Si el valor de falloJudicial es **"S"**, también establece el valor de **falloJudicialBool** en true para sincronizar ambos campos y reflejar la selección en el "checkbox".

2. Getter y Setter de falloJudicialBool

- El método **getFalloJudicialBool()** retorna el valor de **falloJudicialBool**, utilizado principalmente para controlar el estado del "checkbox".
- El método **setFalloJudicialBool(Boolean falloJudicialBool)** establece el valor de **falloJudicialBool** y sincroniza el valor de **falloJudicial**. Si **falloJudicialBool** es true, el campo **falloJudicial** se establece en "S"; de lo contrario, se establece en "N".

Comportamiento del Código

Esta implementación permite una relación bidireccional entre **falloJudicial** y **falloJudicialBool**, asegurando que ambos campos estén sincronizados:

- Si el valor de **falloJudicial** cambia, también se actualiza **falloJudicialBool** y viceversa.
- Esta sincronización es útil para el manejo del "checkbox" en la interfaz gráfica, garantizando que el estado visual refleje siempre el valor actual de **falloJudicial**.
- Sección de **Reclamaciones** en la Interfaz de Usuario

```
<h:panelGroup layout="block" id="panelReclamaciones">
  <ui:fragment
    rendered="#{proyectoEvaluacionSessController.proyectoEvaluacionDTO.tipoEvaluacion eq 'P'}" >
    <h2>#{msg['gestion.proyectos.evaluacion.evaluadores.reclamaciones']}


361


```

```

                                or
proyectoEvaluacionSessController.selected.estadoProyectoId.descripcionValorDefinicion !=
'REVISADO'}"

                                >
                                <p:ajax                                event="change"
process="@this" update="botonesAdjuntoReclamaciones" />
                                </h:selectBooleanCheckbox>
                                <script>
                                    $(document).ready(function () {
                                        try {
                                            var elems =
Array.prototype.slice.call(document.querySelectorAll('.js-switchFallo'));
                                            elems.forEach(function (html) {
                                                var
switchery = new Switchery(html, {color: '#1AB394', size: 'small'});
                                                });
                                            } catch (e) {
                                                console.log('%c' + e, 'background: #222; color: #bada55');
                                            }
                                        });
                                </script>
                                </div>
                                </div>
                                </div>
                                <p:outputPanel id="botonesAdjuntoReclamaciones">
                                    <div class="row">
                                        <div class="col-lg-12">
                                            <div class="form-group">
                                                <p:outputLabel
value="#{msg['gestion.proyectos.evaluacion.evaluadores.reclamaciones.adjuntoFalloJudicial']}"
*"
styleClass="col-lg-4 control-label"

                                                </div>
                                                <div class="col-lg-8">
                                                    <p:commandLink
title="#{msg['sb.component.boton.adjuntar.add']}"
oncomplete="$('#adjuntarReclamacionDlg').modal('show')"
process="@this"
update="formAddReclamacion"
styleClass="btn btn-primary" style="font-size: 10px;"
disabled="#{proyectoEvaluacionSessController.selected.cierreEvaluacionSINO
                                or
proyectoEvaluacionSessController.selected.estadoProyectoId.descripcionValorDefinicion !=
'REVISADO'
                                or
!proyectoEvaluacionSessController.proyectoEvaluacionDTO.falloJudicialBool}"
                                >

```

```

upload"></i>
<i class="fa fa-
</p:commandLink>
<p:commandLink
title="#{msg['sb.component.boton.download']}"
process="@this"
update="@this"
onclick="PrimeFaces.monitorDownload(start, stop)"
ajax="false"
immediate="true"
actionListener="#{evaluacionConvController.descargarArchivoReclamaciones}"
value="#{proyectoEvaluacionSessController.documentoReclamaciones.nombre}"
style="margin-left: 5px; text-decoration: underline; color: #00939B;">
<p:fileDownload
value="#{proyectoEvaluacionSessController.archivoReclamaciones}"/>
</p:commandLink>
</div>
</div>
</div>
</div>
</p:outputPanel>
<p:spacer/>
<p:separator/>
</p:outputPanel>
</ui:fragment>
</h:panelGroup>

```

La sección de reclamaciones está disponible únicamente cuando el tipo de evaluación es "Panel". En esta sección, el evaluador puede indicar si el proyecto tiene un fallo judicial y, en caso afirmativo, adjuntar el documento correspondiente. Esta interfaz está diseñada en JSF utilizando componentes PrimeFaces para facilitar la interacción del usuario.

Estructura de la Interfaz

1. Panel Principal (panelReclamaciones)

- Este es un contenedor principal que incluye un fragmento condicional, visible únicamente cuando el tipo de evaluación es "Panel".
- La condición
rendered="#{proyectoEvaluacionSessController.proyectoEvaluacionDTO.tipoEvaluacion eq 'P'}" asegura que esta sección solo se muestre cuando el tipo de evaluación sea "Panel".

2. Campo "Fallo Judicial"

- Un **checkbox** (**h:selectBooleanCheckbox**) permite al usuario marcar si el proyecto presenta un fallo judicial.
- Este checkbox está enlazado al campo **falloJudicialBool** en **proyectoEvaluacionDTO**, que es un **Boolean** que refleja el estado de fallo judicial.
- **Propiedad disabled:** El checkbox se desactiva si se cumple cualquiera de las siguientes condiciones:
 - La evaluación ya ha sido cerrada (**proyectoEvaluacionSessController.selected.cierreEvaluacionSINO**).
 - El estado del proyecto no es "REVISADO" (**proyectoEvaluacionSessController.selected.estadoProyectoId.descripcionValorDefinicion != 'REVISADO'**).
- **Estilo Visual:** Se usa un estilo personalizado con **Switchery**, que convierte el checkbox en un interruptor visual.
- **Evento Ajax:** Al cambiar el estado del checkbox, se activa un evento **p:ajax** que actualiza el panel **botonesAdjuntoReclamaciones**.

3. Panel de Botones para Adjuntar y Descargar Archivos (botonesAdjuntoReclamaciones)

- Este panel contiene los botones para adjuntar y descargar documentos de fallo judicial.
- **Botón de Adjuntar Documento**
 - Este botón, representado con un icono de carga, permite abrir un diálogo modal (**adjuntarReclamacionDlg**) para subir el documento del fallo judicial.
 - **Propiedad disabled:** El botón de adjuntar documento se desactiva cuando:
 - La evaluación ya ha sido cerrada.
 - El estado del proyecto no es "REVISADO".

- El checkbox de fallo judicial (**falloJudicialBool**) no está marcado.
- **Botón de Descargar Documento**
 - Permite descargar el documento adjunto. El botón usa un componente **p:fileDownload** para gestionar la descarga de archivos de forma asincrónica.
 - **Propiedad `actionListener`:** El método **`evaluacionConvController.descargarArchivoReclamaciones`** maneja la lógica de descarga del archivo.
 - **Estilo y Texto del Botón:** El nombre del archivo que se muestra corresponde a **`proyectoEvaluacionSessController.documentoReclamaciones.nombre`**.

4. Javascript para el Estilo Switch

- El código de JavaScript convierte el checkbox de fallo judicial en un interruptor visual mediante la librería **Switchery**.
- Este script se ejecuta al cargar la página y se aplica a todos los elementos con la clase **js-switchFallo**.

Explicación del Flujo de Trabajo

1. Marcado de Fallo Judicial

- Cuando el evaluador marca el checkbox de fallo judicial, el campo **falloJudicialBool** se actualiza en el DTO, reflejando que el proyecto presenta un fallo judicial.
- Esto habilita el botón para adjuntar el documento del fallo judicial.

2. Adjuntar Documento

- Al hacer clic en el botón de adjuntar, se abre un diálogo modal (**adjuntarReclamacionDlg**) donde el usuario puede seleccionar y cargar el documento correspondiente.

3. Descargar Documento

- Si el proyecto ya tiene un documento de fallo judicial adjunto, el evaluador puede hacer clic en el botón de descarga para obtener una copia.
- La descarga del archivo se maneja en el servidor y utiliza **evaluacionConvController.descargarArchivoReclamaciones** para generar el archivo de respuesta.

- **Diálogo de Adjunto de Documento de Fallo Judicial**

```
<!-- Dialogo Adjunto RECLAMACIONES -->
<h:form enctype="multipart/form-data" id="formAddReclamacion">
  <div class="modal inmodal" id="adjuntarReclamacionDlg" tabindex="-1" role="dialog" aria-
hidden="true">
    <div class="modal-dialog">
      <div class="modal-content animated fadeIn">
        <div class="modal-header">
          <button type="button" class="close" data-
dismiss="modal"><span
aria-hidden="true">&times;</span><span class="sr-
only">Close</span></button>
          <i class="fa fa-upload modal-icon"></i>
          <h4 class="modal-
title">#{msg['sb.component.boton.adjuntar.add']}</h4>
        </div>
        <div class="modal-body">
          <p:fileUpload
listener="#{evaluacionConvController.handleFileUploadReclamaciones}"
mode="advanced"
process="@this"

sizelimit="#{SeguridadSessController.tamMaxArchivos.longValue() * 1048576}"

update="#{proyectoEvaluacionSessController.modulo}FormDetalle:msg,#{proyectoEvaluacionSessCo
ntroller.modulo}FormDetalle:botonesAdjuntoReclamaciones"

oncomplete="$('#adjuntarReclamacionDlg').modal('hide')"

cancelLabel="#{msg['sb.component.boton.cancelar']}>

uploadLabel="#{msg['sb.component.boton.adjuntar.add']}>

label="#{msg['sb.component.boton.seleccionar']}>

invalidSizeMessage="Archivo
demasiado grande, el tamaño máximo debe ser de
#{SeguridadSessController.tamMaxArchivos.longValue()}MB"
/>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-white" data-
dismiss="modal">#{msg['sb.component.boton.cancelar']}</button>
        </div>
      </div>
    </div>
  </div>
```

```

</div>

<div class="modal inmodal" id="descargarDlg" tabindex="-1" role="dialog" aria-
hidden="true">
    <div class="modal-dialog">
        <div class="modal-content animated fadeIn">
            <div class="modal-header">
                <button type="button" class="close" data-
dismiss="modal"><span
aria-hidden="true">&times;</span><span class="sr-
only">Close</span></button>
                <i class="fa fa-download modal-icon"></i>
                <h4 class="modal-
title">#{msg['sb.mensajes.descargando.label']}</h4>
            </div>
            <div class="modal-body">

        </div>
    </div>
</div>
</div>
</h:form>

```

Este diálogo se muestra cuando el usuario selecciona la opción de adjuntar un documento de fallo judicial. Permite al usuario subir un archivo, el cual se procesa y almacena en el sistema. La interfaz está estructurada como un formulario HTML dentro de un diálogo modal, utilizando componentes JSF y PrimeFaces.

Estructura de la Interfaz

1. Formulario Principal (formAddReclamacion)

- El formulario está configurado con **enctype="multipart/form-data"** para permitir la carga de archivos.
- El formulario contiene dos diálogos modales (**modal inmodal**):
 - **adjuntarReclamacionDlg** para adjuntar documentos.
 - **descargarDlg** (vacío) para indicar que el archivo se está descargando.

2. Diálogo de Adjuntar Documento (adjuntarReclamacionDlg)

- Este es el diálogo principal para subir un documento de fallo judicial.
- **Encabezado del Diálogo**
 - Incluye un botón para cerrar el diálogo (**button.close**) y un icono de carga (**fa fa-upload**) para indicar visualmente la función del diálogo.

- El título del diálogo se define como
`#{msg['sb.component.boton.adjuntar.add']}`, obtenido del archivo de mensajes para permitir localización.
- **Cuerpo del Diálogo**
 - Contiene el componente **p:fileUpload**, configurado para carga avanzada (**mode="advanced"**), y con varias opciones personalizadas:
 - **Propiedad listener:** Llama al método **evaluacionConvController.handleFileUploadReclamaciones**, que gestiona la lógica de almacenamiento y procesamiento del archivo subido.
 - **Propiedad sizeLimit:** Define el límite máximo del tamaño del archivo, calculado en megabytes con el valor de **SeguridadSessController.tamMaxArchivos.longValue()**.
 - **Propiedad update:** Actualiza los elementos **msg** y **botonesAdjuntoReclamaciones** después de la carga, asegurando que el usuario vea los cambios inmediatamente en la interfaz.
 - **Propiedad oncomplete:** Oculta el diálogo modal (**adjuntarReclamacionDlg**) una vez completada la carga.
 - **Propiedad invalidSizeMessage:** Muestra un mensaje de error personalizado si el archivo supera el tamaño permitido.
 - **Etiquetas y Botones**
 - **cancelLabel**, **uploadLabel**, y **label** especifican etiquetas personalizadas obtenidas de msg para permitir localización y accesibilidad.
 - El botón de cancelar (**cancelLabel**) permite al usuario cerrar el diálogo sin subir un archivo.

3. Diálogo de Descarga (**descargarDlg**)

- Este diálogo, aunque vacío en cuanto a contenido funcional, está diseñado para mostrar un icono y un título de descarga cuando el usuario descarga un

documento. Esto puede servir como indicativo visual de que una descarga está en proceso.

Explicación del Flujo de Trabajo

1. Inicio del Proceso de Adjunto

- El usuario abre el diálogo adjuntarReclamacionDlg haciendo clic en el botón de adjuntar en la interfaz de evaluación.

2. Subida de Archivo

- El usuario selecciona un archivo utilizando **p:fileUpload**. El sistema verifica el tamaño del archivo en función de **SeguridadSessController.tamMaxArchivos**.
- Si el archivo excede el tamaño permitido, muestra un mensaje de error (**invalidSizeMessage**).

3. Manejo del Archivo Subido

- El archivo se procesa en el **backend** mediante **evaluacionConvController.handleFileUploadReclamaciones**. Al completarse el proceso de carga:
 - Se cierra el diálogo de carga.
 - Se actualizan los componentes **msg** y **botonesAdjuntoReclamaciones** para reflejar el estado actual de la interfaz.

4. Finalización

- Al finalizar el proceso de adjunto, el usuario puede visualizar el archivo adjunto en la interfaz principal de reclamaciones.

- **Método `handleFileUploadReclamaciones`**

```
public void handleFileUploadReclamaciones(FileUploadEvent event) throws Exception {  
    InputStream is = null;  
    FacesMessage msg = new FacesMessage("Correcto", event.getFile().getFileName() + " ha sido  
cargado con éxito.");  
    try {  
        UploadedFile uploadedFile = event.getFile();
```

```

        is = uploadedFile.getInputStream();
        byte[] archivo = new byte[new Long(uploadedFile.getSize()).intValue()];
        is.read(archivo);
        DocumentoDTO documentoDTO =
        proyectoEvaluacionSessController.getDocumentoReclamaciones();

        // Validamos que el nombre del archivo no contenga caracteres especiales.
        appSessController.validarNombreArchivo(uploadedFile);

        if (documentoDTO.getDocumentoId() == null) {
            documentoDTO.setArchivo(archivo);
            documentoDTO.setNombre(uploadedFile.getFileName());
            documentoDTO.setMime(uploadedFile.getContentType());
            documentoDTO.setDocSize(uploadedFile.getSize());

            documentoDTO.setTipoOrigenId(proyectoEvaluacionSessController.getDocumentoReclamaciones
            Tipo());

            proyectoEvaluacionSessController.setDocumentoReclamaciones(documentoDTO);
        } else {
            documentoDTO.setNombre(uploadedFile.getFileName());
            documentoDTO.setMime(uploadedFile.getContentType());
            documentoDTO.setDocSize(uploadedFile.getSize());
            documentoDTO.setArchivo(archivo);

            proyectoEvaluacionSessController.setDocumentoReclamaciones(documentoDTO);
        }
        PrimeFaces.current().executeScript("#{demo2}.collapse('show');");
    } catch (IOException ex) {
        ex.printStackTrace();
        msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
        resolveExpression(ex.getMessage()));
    } catch (Exception ex) {
        ex.printStackTrace();
        msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
        resolveExpression(ex.getMessage()));
    } finally {
        try {
            FacesContext.getCurrentInstance().addMessage(null, msg);
            if (is != null) {
                is.close();
            }
        } catch (IOException ex) {
            LOG.error("Error al cerrar el inputStream del archivo ", ex);
        }
    }
}

```

Este método se encarga de manejar la carga de archivos de reclamaciones. Recibe un evento **FileUploadEvent** que contiene la información sobre el archivo cargado. El método realiza varias acciones como la validación del nombre del archivo, la lectura del archivo cargado, y la actualización de los datos del documento en el controlador.

Flujo de Trabajo y Funcionalidad

1. Inicialización y Validación Inicial

- Se crea un **FacesMessage** que muestra un mensaje de éxito, que se actualizará si la carga es exitosa.
- Se obtiene el archivo cargado desde el evento **FileUploadEvent**.

2. Obtención y Validación del Archivo

- El archivo cargado se obtiene a través del método **event.getFile()**, y se lee el contenido del archivo en un **InputStream (is)**.
- El tamaño del archivo se determina y se almacena en un arreglo de bytes (**archivo**).

3. Obtención del Documento y Validación del Nombre del Archivo

- Se obtiene un objeto **DocumentoDTO** desde **proyectoEvaluacionSessController.getDocumentoReclamaciones()**.
- El nombre del archivo es validado utilizando el método **appSessController.validarNombreArchivo(uploadedFile)**. Este método asegura que el nombre del archivo no contenga caracteres especiales u otros caracteres no permitidos.

4. Verificación de la Existencia del Documento

- Si el **documentoDTO.getDocumentId()** es **null**, significa que es la primera vez que se carga un archivo. En este caso, se asignan los valores del archivo (**archivo, nombre, mime, docSize**) al objeto **documentoDTO**, y se establece el tipo de documento mediante **setTipoOrigenId**.
- Si el **documentoDTO.getDocumentId()** no es **null**, el documento ya existe y solo se actualizan los valores del archivo, manteniendo el resto de los datos.

5. Actualización del Documento en el Controlador

- Una vez que los datos del archivo se han asignado al objeto **documentoDTO**, se actualiza el documento en el controlador de la sesión:
proyectoEvaluacionSessController.setDocumentoReclamaciones(documentoDTO).

6. Interacción con la Interfaz

- Se ejecuta un script de JavaScript (**PrimeFaces.current().executeScript(...)**) para mostrar un componente visual (acordeón) con el ID **#demo2** en la interfaz, indicando que el archivo ha sido cargado correctamente.

7. Manejo de Excepciones

- El método tiene bloques **catch** para capturar excepciones de tipo **IOException** y **Exception**, registrando el error y mostrando un mensaje de error en caso de fallos. En ambos casos, el mensaje de error es configurado para mostrar detalles de la excepción.

8. Cierre de Recursos

- Finalmente, en el bloque **finally**, se asegura que el **InputStream (is)** se cierre para liberar los recursos del sistema. También se agrega el mensaje correspondiente a la instancia de **FacesContext**, lo que permite mostrar el mensaje en la interfaz de usuario.

Ejemplo de Mensaje en Caso de Éxito o Error

- **Éxito:** "Archivo [nombre del archivo] ha sido cargado con éxito."
- **Error:** Si ocurre una excepción, el mensaje será algo como "ERROR: [detalle del error]".

Resumen de Acciones

- **Carga del archivo:** El archivo se lee y se convierte en un arreglo de bytes.
- **Validación:** Se valida el nombre del archivo para asegurar que no contenga caracteres especiales no permitidos.
- **Actualización de documento:** El archivo se asocia al documentoDTO y se actualiza en el controlador de sesión.
- **Feedback al usuario:** El sistema muestra un mensaje de éxito o error en la interfaz, y se asegura de cerrar el **InputStream** al finalizar.
- Método **descargarArchivoReclamaciones**

```
public void descargarArchivoReclamaciones() {
```

```
ByteArrayInputStream bis = new ByteArrayInputStream(  
  
    proyectoEvaluacionSessController.getDocumentoReclamaciones().getArchivo());  
proyectoEvaluacionSessController.setArchivoReclamaciones(  
    new DefaultStreamedContent(  
        bis,  
    proyectoEvaluacionSessController.getDocumentoReclamaciones().getMime(),  
    proyectoEvaluacionSessController.getDocumentoReclamaciones().getNombre()));  
}
```

Este método prepara un archivo previamente cargado para ser descargado por el usuario, convirtiéndolo en un objeto **StreamedContent** que será utilizado por un componente de descarga en la interfaz (botón).

Flujo de Trabajo y Funcionalidad

1. Obtención del Archivo desde la Sesión

- Se obtiene el archivo almacenado en la sesión a través de **proyectoEvaluacionSessController.getDocumentoReclamaciones()**. Este archivo está en formato de arreglo de bytes, el cual fue previamente cargado en el método **handleFileUploadReclamaciones**.

2. Creación del ByteArrayInputStream

- El archivo se convierte en un **ByteArrayInputStream (bis)** para poder ser transmitido como un flujo de entrada. Este flujo se utilizará para proporcionar el archivo en su formato original cuando el usuario realice la descarga.

3. Creación del StreamedContent

- Luego, se crea un objeto de tipo **StreamedContent** usando la clase **DefaultStreamedContent**. Este objeto es el que se entregará al componente de descarga en la interfaz (**p:commandLink**).
 - El flujo de datos (**bis**) es el **InputStream** que contiene los datos del archivo.
 - El tipo MIME (**mime**) se obtiene del objeto **documentoDTO**, que se refiere al tipo de archivo cargado (por ejemplo, application/pdf, application/msword, etc.).

- El nombre del archivo (**nombre**) también se toma desde el objeto **documentoDTO**, y será el nombre con el que se descargará el archivo.

4. Asignación del **StreamedContent**

- Finalmente, el **StreamedContent** recién creado se asigna a la propiedad **archivoReclamaciones** en el controlador de la sesión (**proyectoEvaluacionSessController.setArchivoReclamaciones(...)**). Este objeto estará disponible para ser utilizado por un componente en la vista, que permitirá al usuario descargar el archivo.

Resumen de las Acciones:

- **Obtención del archivo:** Se obtiene el archivo de la sesión, que fue previamente cargado y almacenado en el controlador.
- **Conversión a **InputStream**:** El archivo se convierte a un flujo de entrada (**ByteArrayInputStream**) para su transmisión.
- **Creación de **StreamedContent**:** Se crea un objeto **StreamedContent** con el flujo de entrada, el tipo MIME y el nombre del archivo.
- **Asignación al controlador:** Se asigna el **StreamedContent** al controlador para que esté disponible en la interfaz de usuario para su descarga.

Configuración de Matrices de Evaluación de Beneficios Tributarios por Tamaño Empresa

Para este requerimiento se solicitó para convocatorias de beneficios tributarios, configurar matrices de evaluación según el tamaño de la empresa (grande, mediana, pequeña o microempresa.) y el tipo de carácter del proyecto. Así mismo, mostrar en el módulo de evaluación la matriz respectiva según la configuración.

Para el desarrollo de este requerimiento, se realizaron los siguientes ajustes:

- creación de nuevos campos en pantalla de configuración de Matrices de Convocatorias de Beneficios Tributarios
- Se ajusta método de consulta y filtrado de matrices por convocatoria

- Se ajusta lógica de consulta y verificación para obtener la matriz de evaluación asignada a un proyecto, teniendo en cuenta si hay configuraciones existentes, si hay una única configuración según el tipo de carácter, si tiene habilitado o no el tamaño de empresa, en ese caso, valida por tamaño de empresa de la entidad ejecutora. Se ajustan métodos tanto de evaluación como desde la pantalla de evaluar del evaluador
- Se crean etiquetas de internacionalización

Matrices de Evaluación por Convocatoria

- creación de nuevos campos en la tabla SGP_CONVOCATORIAS_MATRICES

```
-- Add/modify columns
alter table SGP_CONVOCATORIAS_MATRICES add aplica_tamano_empresa VARCHAR2(1);
-- Add comments to the columns
comment on column SGP_CONVOCATORIAS_MATRICES.aplica_tamano_empresa
is 'Indica si la configuración aplica por el tamaño de la empresa de la entidad ejecutora
'S' o 'N'';
-- Create/Recreate check constraints
alter table SGP_CONVOCATORIAS_MATRICES
add constraint SGPCONVMATR_APLICA_TAM_EMP_CHK
check (APLICA_TAMANO_EMPRESA IN ('S'/*Si*/, 'N'/*No*/));

-- Add/modify columns
alter table SGP_CONVOCATORIAS_MATRICES add tamano_empresa_id NUMBER(15);
-- Add comments to the columns
comment on column SGP_CONVOCATORIAS_MATRICES.tamano_empresa_id
is 'Identificador del tamaño de la empresa';
-- Create/Recreate indexes
create index SGPCONVMATR_TAMEMPRESA_FK_I on SGP_CONVOCATORIAS_MATRICES (tamano_empresa_id)
tablespace indices
pctfree 10
initrans 2
maxtrans 255
storage
(
initial 64K
next 1M
minextents 1
maxextents unlimited
);
-- Create/Recreate primary, unique and foreign key constraints
alter table SGP_CONVOCATORIAS_MATRICES
add constraint SGPCONVMATR_TAMEMPRESA_FK foreign key (TAMANO_EMPRESA_ID)
references adm_valores_definiciones (VALOR_DEFINICION_ID);
```

- Actualizar Llave Unica (UK) de la tabla SGP_CONVOCATORIAS_MATRICES

```
-- Create/Recreate primary, unique and foreign key constraints
```



```
alter table SGP_CONVOCATORIAS_MATRICES
  drop constraint SGPCONVMATR_UK cascade;
alter table SGP_CONVOCATORIAS_MATRICES
  add constraint SGPCONVMATR_UK unique (CONVOCATORIA_ID, CARACTER_PROYECTO_ID,
TAMANO_EMPRESA_ID)
  using index
  tablespace DATOS
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    next 1M
    minextents 1
    maxextents unlimited
  );
```

- Mapear nuevos campos en la entidad de la convocatoria matriz (ConvocatoriaMatriz.java)

```
@Column(name = "APLICA_TAMANO_EMPRESA")
private String aplicaTamanoEmpresa;

//Joins
@JoinColumn(name = "TAMANO_EMPRESA_ID", referencedColumnName = "VALOR_DEFINICION_ID")
@ManyToOne(optional = false)
private ValorDefinicion tamanoEmpresaId;

public String getAplicaTamanoEmpresa() {
    return aplicaTamanoEmpresa;
}

public void setAplicaTamanoEmpresa(String aplicaTamanoEmpresa) {
    this.aplicaTamanoEmpresa = aplicaTamanoEmpresa;
}

public ValorDefinicion getTamanoEmpresaId() {
    return tamanoEmpresaId;
}

public void setTamanoEmpresaId(ValorDefinicion tamanoEmpresaId) {
    this.tamanoEmpresaId = tamanoEmpresaId;
}
```

- Mapear nuevos campos en DTO de la convocatoria matriz (ConvocatoriaMatriz DTO.java)

```
@Bind(name = "aplicaTamanoEmpresa")
private String aplicaTamanoEmpresa;

//Joins
@Bind(name = "tamanoEmpresaId", returnType = ValorDefinicionDTO.class, type =
ValorDefinicion.class)
private ValorDefinicionDTO tamanoEmpresaId;
```

```
//Campos checkBox
private Boolean aplicaTamanoEmpresaBool;

//Getter and Setter
public String getAplicaTamanoEmpresa() {
    return aplicaTamanoEmpresa;
}

public void setAplicaTamanoEmpresa(String aplicaTamanoEmpresa) {
    if (aplicaTamanoEmpresa != null && aplicaTamanoEmpresa.equals("S")) {
        this.aplicaTamanoEmpresaBool = true;
    }
    this.aplicaTamanoEmpresa = aplicaTamanoEmpresa;
}

public Boolean getAplicaTamanoEmpresaBool() {
    return aplicaTamanoEmpresaBool;
}

public void setAplicaTamanoEmpresaBool(Boolean aplicaTamanoEmpresaBool) {
    if (aplicaTamanoEmpresaBool) {
        this.aplicaTamanoEmpresa = "S";
    } else {
        this.aplicaTamanoEmpresa = "N";
    }
    this.aplicaTamanoEmpresaBool = aplicaTamanoEmpresaBool;
}

public ValorDefinicionDTO getTamanoEmpresaId() {
    return tamanoEmpresaId;
}

public void setTamanoEmpresaId(ValorDefinicionDTO tamanoEmpresaId) {
    this.tamanoEmpresaId = tamanoEmpresaId;
}
```

Se realiza a través de la adición de nuevos campos y su respectivo mapeo en la entidad correspondiente.

1. Descripción de los campos añadidos y mapeados

Campos creados:

1. aplicaTamanoEmpresa

- **Tipo:** String
- **Descripción:** Almacena si la matriz de evaluación aplica la configuración según el tamaño de la empresa.
- **Valores posibles:** "S" (Sí) o "N" (No).

2. **tamanoEmpresald**

- **Tipo: ValorDefinicionDTO**
- **Descripción:** Representa la asociación con un valor predefinido que indica el tamaño de la empresa (grande, mediana, pequeña o microempresa).

3. **aplicaTamanoEmpresaBool**

- **Tipo: Boolean**
- **Descripción:** Representa de manera boolean la propiedad **aplicaTamanoEmpresa**, facilitando la interacción en la lógica de negocio y en la interfaz de usuario.

2. Mapeos realizados

Declaración de campos:

- **@Bind:** Indica la relación entre el atributo de la entidad y la columna correspondiente en la base de datos.
 - **aplicaTamanoEmpresa:** Campo que indica si aplica la configuración de tamaño de empresa.
 - **tamanoEmpresald:** Relación tipo Join que asocia el tamaño de la empresa con la tabla de valores predefinidos.

3. Lógica de transformación entre String y Boolean

Se añadieron métodos **getter** y **setter** para manejar automáticamente la conversión entre el tipo String ("S"/"N") y el tipo Boolean (true/false).

- Cuando **aplicaTamanoEmpresa** toma el valor "S", el campo booleano **aplicaTamanoEmpresaBool** se configura como true.
- Si se establece el valor booleano true en **aplicaTamanoEmpresaBool**, el campo **aplicaTamanoEmpresa** se ajusta automáticamente a "S".

4. Asociación del tamaño de empresa

- **campo tamanoEmpresald:** Este campo permite seleccionar y almacenar el tamaño de la empresa asociado a la configuración, garantizando la integración con los valores predefinidos en el sistema.
- Adición de campos en el formulario de la Interfaz de Usuario

```
<div class="row">
  <!-- Configuración por Tamaño de Empresa -->
  <div class="col-lg-6">
    <div class="form-group">
      <p:outputLabel
value="#{msg['gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.label']}"
styleClass="col-lg-6 control-label"
for="aplicaTamanoEmpresa"
/>
      <div class="col-lg-6">
        <h:selectBooleanCheckbox id="aplicaTamanoEmpresa"

value="#{convocatoriaMatrizViewController.selected.aplicaTamanoEmpresaBool}"

required="false"

requiredMessage="#{msg['gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.requir
ed']}"

styleClass="js-switchTamEmpresa"

        <p:ajax event="change" process="@this"
update="panelTamanoEmpresa" />
        </h:selectBooleanCheckbox>
        <script>
          $(document).ready(function () {
            try {
              var elems =
Array.prototype.slice.call(document.querySelectorAll('.js-switchTamEmpresa'));
              elems.forEach(function (html) {
                var switchery = new
Switchery(html, {color: '#1AB394', size: 'small'});
              });
            } catch (e) {
              console.log('%c' + e, 'background: #222;
color: #bada55');
            }
          });
        </script>
      </div>
    </div>
  </div>
  <!-- Tamaño Empresa -->
  <p:outputPanel id="panelTamanoEmpresa">
    <ui:fragment
rendered="#{convocatoriaMatrizViewController.selected.aplicaTamanoEmpresaBool}">
      <div class="col-lg-6">
        <div class="form-group">
```

```

                                <p:outputLabel
value="#{msg['gestion.configuracion.convocatoria.matriz.tamanoEmpresaId.label']}"
                                styleClass="col-lg-4
control-label"
                                style="text-align: right"
                                for="tamanoEmpresaId-
#{msg['sb.convert.selectOne.tamano.ID']}" />
                                <div class="col-lg-6">
                                <p:selectOneMenu
                                id="tamanoEmpresaId-
#{msg['sb.convert.selectOne.tamano.ID']}"
value="#{convocatoriaMatrizViewController.selected.tamanoEmpresaId}"
required="true"
requiredMessage="#{msg['gestion.configuracion.convocatoria.matriz.tamanoEmpresaId.required']}"
converter="valorDefinicionConverter"
style="width:100%;"
autoWidth="false"
panelStyle="width:180px"
                                effect="fade"
                                filter="true"
filterMatchMode="contains"
                                >
                                <f:selectItem
                                itemValue="#{null}"
                                itemLabel="#{msg['sb.selectOneMenu.registro']}" />
                                <f:selectItems
                                value="#{convocatoriaMatrizViewController.mapTamanos}"
                                var="tamano"
                                itemLabel="#{tamano.descripcionValorDefinicion}"
                                itemValue="#{tamano}" />
                                </p:selectOneMenu>
                                </div>
                                </div>
                                </div>
                                </ui:fragment>
                                </p:outputPanel>
                                </div>

```

Se implementó una sección en el formulario de captura de datos para configurar matrices de evaluación basadas en el tamaño de la empresa:

1. Checkbox de activación:

Permite al usuario activar o desactivar la configuración por tamaño de empresa. Cuando está activado, se habilita un menú desplegable.

2. Menú desplegable:

Muestra una lista de tamaños de empresa disponibles para seleccionar.

1. Configuración por tamaño de empresa (h:selectBooleanCheckbox)

Código relevante:

```
<h:selectBooleanCheckbox id="aplicaTamanoEmpresa"
value="#{convocatoriaMatrizViewController.selected.aplicaTamanoEmpresaBool}"
required="false"

requiredMessage="#{msg['gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.requir
ed']}"
styleClass="js-switchTamEmpresa">
  <p:ajax event="change" process="@this" update="panelTamanoEmpresa" />
</h:selectBooleanCheckbox>
```

- **h:selectBooleanCheckbox:**

Este componente permite al usuario habilitar o deshabilitar la configuración por tamaño de empresa.

- **value:** Vincula el valor seleccionado al atributo booleano **aplicaTamanoEmpresaBool** en el controlador.
- **required:** Define si el campo es obligatorio (en este caso, no lo es).
- **p:ajax:**
 - **event="change":** Ejecuta un evento **AJAX** cuando el estado del checkbox cambia.
 - **process="@this":** Procesa únicamente este componente.
 - **update="panelTamanoEmpresa":** Actualiza dinámicamente el panel que contiene el menú desplegable para seleccionar el tamaño de la empresa.

- **Estilo visual (Switchery):**

Se utiliza el script de la librería *Switchery* para mejorar el diseño del checkbox:

```
var elems = Array.prototype.slice.call(document.querySelectorAll('.js-switchTamEmpresa'));
elems.forEach(function (html) {
  var switchery = new Switchery(html, {color: '#1AB394', size: 'small'});
});
```

2. Selección del tamaño de empresa (p:selectOneMenu)

Código relevante:

```
<ui:fragment
rendered="#{convocatoriaMatrizViewController.selected.aplicaTamanoEmpresaBool}">
    <p:selectOneMenu id="tamanoEmpresaId-#{msg['sb.convert.selectOne.tamano.ID']}"
        value="#{convocatoriaMatrizViewController.selected.tamanoEmpresaId}"
        required="true"

        requiredMessage="#{msg['gestion.configuracion.convocatoria.matriz.tamanoEmpresaId.required']}
    }"

        converter="valorDefinicionConverter"
        style="width:100%;"
        autoWidth="false"
        panelStyle="width:180px"
        effect="fade"
        filter="true"
        filterMatchMode="contains">
        <f:selectItem itemValue="#{null}" itemLabel="#{msg['sb.selectOneMenu.registro']}" />
        <f:selectItems value="#{convocatoriaMatrizViewController.mapTamanos}"
            var="tamano"
            itemLabel="#{tamano.descripcionValorDefinicion}"
            itemValue="#{tamano}" />
    </p:selectOneMenu>
</ui:fragment>
```

- **Renderizado dinámico:**

El menú desplegable solo se muestra si el checkbox **aplicaTamanoEmpresa** está habilitado (**true**):

```
<ui:fragment
rendered="#{convocatoriaMatrizViewController.selected.aplicaTamanoEmpresaBool}">
```

- **Menú desplegable (p:selectOneMenu):**

Este componente permite seleccionar un tamaño de empresa de una lista de valores definidos:

- **value:** Vincula el tamaño seleccionado al atributo tamanoEmpresald en el controlador.
- **converter:** Convierte entre el valor seleccionado y el objeto correspondiente mediante **valorDefinicionConverter**.
- **Propiedades visuales:**

- **style y panelStyle:** Definen el tamaño y estilo del menú.

- **filter="true"**: Permite filtrar los valores mostrados en la lista desplegable.

- **Opciones del menú:**

- **<f:selectItem>**: Añade una opción predeterminada con un mensaje como *"Seleccione una opción"*.
- **<f:selectItems>**: Genera dinámicamente las opciones del menú a partir de la colección **convocatoriaMatrizViewController.mapTamanos**. Cada opción muestra la descripción (**tamano.descripcionValorDefinicion**) y asocia el objeto completo como valor (tamano).

3. Estructura general del formulario

Diseño responsivo:

- La estructura utiliza clases de *Bootstrap* (**row**, **col-lg-6**, etc.) para garantizar un diseño adaptable en diferentes tamaños de pantalla.
- Adición de campos en la tabla principal de la Interfaz de Usuario

```
<p:column
headerText="#{msg['gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.header']}"
    filterBy="#{convMatriz.activo}"
    filterMatchMode="contains"
    filterStyle="display:none;"
    style="text-align: center"
    width="20%"
>
    <h:outputText styleClass="badge #{convMatriz.aplicaTamanoEmpresa eq 'S' ? 'badge-primary'
: 'badge-danger'}"
        value="#{convMatriz.aplicaTamanoEmpresa != null ?
msg['listaSINO.'.concat(convMatriz.aplicaTamanoEmpresa)] : 'No'}"/>
    <ui:fragment rendered="#{convMatriz.aplicaTamanoEmpresa eq 'S'}">
        <br/><br/>
        <div style="font-style: italic; text-align: center;">

        #{msg['gestion.configuracion.convocatoria.matriz.tamanoEmpresaId.header']}:      <b>
        #{convMatriz.tamanoEmpresaId.descripcionValorDefinicion} </b>
        </div>
    </ui:fragment>
</p:column>
```


Se añade una columna al **DataTable** principal para mostrar si una matriz de evaluación aplica configuración basada en el tamaño de la empresa, junto con la descripción del tamaño configurado cuando corresponde.

1. Descripción del código de la nueva columna

Encabezado de la columna:

- **headerText:** Indica el título de la columna. La etiqueta es extraída de los archivos de mensajes internacionales (**gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.header**).
- **filterBy:** Campo utilizado para realizar filtros en el **DataTable**. En este caso, está vinculado a la propiedad **convMatriz.activo**. Aunque el filtro está deshabilitado visualmente (**filterStyle="display:none;"**), podría habilitarse si fuese necesario.
- **style y width:** Definen la alineación y el ancho de la columna.

2. Contenido principal de la columna

Estado de la configuración (**aplicaTamanoEmpresa**):

- **styleClass:** Aplica diferentes estilos de clase CSS (**badge-primary** para "S" y **badge-danger** para cualquier otro valor).
- **value:** Muestra un mensaje basado en el valor de **aplicaTamanoEmpresa**:
 - "S": Muestra un texto correspondiente al mensaje localizado (**msg['listaSINO.S']**), por ejemplo, "Sí".
 - **Otros valores:** Muestra "No" o el mensaje correspondiente.

3. Descripción del tamaño de la empresa (si aplica)

Fragmento condicional:

- **<ui:fragment>**: Renderiza este bloque únicamente si el valor de **aplicaTamanoEmpresa** es "S".
- **Contenido:**
 - Muestra un texto indicando el tamaño de la empresa configurado.

- Utiliza **convMatriz.tamanoEmpresald.descripcionValorDefinicion** para extraer la descripción del tamaño de la empresa desde el objeto **ValorDefinicionDTO**.
- Usa etiquetas localizadas
(**msg['gestion.configuracion.convocatoria.matriz.tamanoEmpresald.header']**)
para los encabezados dinámicos.

4. Estilo visual y usabilidad

- **badge-primary:** Indica que la configuración está activa con un color destacado (normalmente azul).
- **badge-danger:** Señala configuraciones inactivas con un color de alerta (normalmente rojo).
- **Descripción en cursiva y centrada:** Mejora la visibilidad y organiza la información adicional sobre el tamaño de la empresa cuando aplica.
- Se agrega filtro para buscar por tamaño de empresa en la consulta de CriteriaBuilder de matrices por convocatoria

```
//Filtro por Tamaño Empresa ID
Object tamanoEmpresaId = filters.get("tamanoEmpresaId");
if (tamanoEmpresaId != null) {
    lista.add(cb.equal(getConvocatoriaMatrizPath("tamanoEmpresaId.valorDefinicionId"),
(Long) tamanoEmpresaId));
}
```

- Internacionalización en español e ingles

```
#ES
gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.header = Configuración por
Tamaño de Empresa
gestion.configuracion.convocatoria.matriz.tamanoEmpresald.header = Tamaño
gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.label = Configuración por
Tamaño de Empresa
gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.required = Configuración por
Tamaño de Empresa es requerido
```

gestion.configuracion.convocatoria.matriz.tamanoEmpresald.label = Tamaño
 gestion.configuracion.convocatoria.matriz.tamanoEmpresald.required = El tamaño es requerido
 com.sb.persistence.constraint.ORA-00001.SGPCONVMATR_UK = El tipo de configuración del carácter del proyecto seleccionado para esta convocatoria, se encuentra registrado en el sistema

#EN

gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.header = Configuration by Company Size

gestion.configuracion.convocatoria.matriz.tamanoEmpresald.header = Size

gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.label = Configuration by Company Size

gestion.configuracion.convocatoria.matriz.aplicaTamanoEmpresa.required = Configuration by Company Size is required

gestion.configuracion.convocatoria.matriz.tamanoEmpresald.label = Tamaño

gestion.configuracion.convocatoria.matriz.tamanoEmpresald.required = Size is required

com.sb.persistence.constraint.ORA-00001.SGPCONVMATR_UK = The type of configuration of the character of the project selected for this call is registered in the system

Ajuste de Lógica para obtener y mostrar la matriz de evaluación calificación según el tamaño de la empresa

Para implementar la nueva lógica de consultar y obtener la matriz de evaluación según el tipo de proyecto y la configuración de la convocatoria, se hace necesario lo siguiente:

- Método **irCalificaciónEvaluación**

```
/*Este metodo verifica si la matriz de evaluación
y redirige a la pantalla segun la matriz configurada para el proyecto
*/
public void irCalificaciónEvaluacion(ProyectoEvaluacionDTO proyectoEvaluacionDTO) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        if (proyectoEvaluacionDTO.getProyectoId().getCaracterProyectoId() != null) {
            proyectoBeneficiosEvaluacionCalificacionSessController.cargarListas();
            String convocatoria =
            proyectoEvaluacionDTO.getProyectoId().getConvocatoriaId().getDescripcionConvocatoria();
            /**
             * La matriz es determinada por cada evaluación al que pertenece
             */
        }
    }
}
```

```
ConvocatoriaMatrizDTO convocatoriaMatrizDTO =
obtenerValidarConvocatoriaMatriz(proyectoEvaluacionDTO.getProyectoId());
if (convocatoriaMatrizDTO != null &&
convocatoriaMatrizDTO.getMatrizEvaluacionId() != null) {

    this.redirectPage("app/beneficiosTributarios/evaluacionCalificacion.xhtml");
} else {
    throw new Exception("La Convocatoria " + convocatoria + " al cual
aplicó el proyecto que desea evaluar NO tiene Matriz de Evaluación Configurada, por tal motivo
no puede ser evaluado");
}
} else {
    throw new Exception("El proyecto con código " +
proyectoEvaluacionDTO.getProyectoId().getCodigoProyecto() + " NO tiene registrado el Caracter
del proyecto, por tal motivo no puede ser evaluado");
}

} catch (Exception ex) {
    LOG.error("Error al ir a la calificacion evaluacion ", ex);
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
} finally {
    FacesContext.getCurrentInstance().addMessage(null, message);
}
}
```

Este método realiza las siguientes tareas:

1. **Valida si el proyecto tiene configurado un carácter (caracterProyectoId):**
 - Si el carácter no está definido, lanza una excepción informando que el proyecto no puede ser evaluado.
2. **Obtiene y valida la configuración de la matriz de evaluación:**
 - Llama al método **obtenerValidarConvocatoriaMatriz** para validar si existe una matriz de evaluación configurada para la convocatoria a la que pertenece el proyecto.
 - Si la matriz no está configurada, lanza una excepción con un mensaje detallado.
3. **Carga listas necesarias para la evaluación:**
 - Llama al método **cargarListas()** en **proyectoBeneficiosEvaluacionCalificacionSessController**, lo que parece inicializar datos necesarios para la evaluación.
4. **Redirige a la pantalla de calificación de evaluación:**

- Si todo está en orden, redirige al archivo **evaluacionCalificacion.xhtml** dentro del módulo de beneficios tributarios.

5. Manejo de errores:

- Si ocurre una excepción, captura el error, lo registra en el log y muestra un mensaje en la interfaz.

- Método **obtenerValidarConvocatoriaMatriz**

```
/**
 * Metodo para obtener la configuración de la matriz registrada en la
 * convocatoria
 *
 * @param proyectoDTO
 * @return
 * @throws Exception
 */
public ConvocatoriaMatrizDTO obtenerValidarConvocatoriaMatriz(ProyectoDTO proyectoDTO) throws
Exception {
    System.out.println("*** entra a obtenerValidarConvocatoriaMatriz ***");

    ConvocatoriaMatrizDTO convocatoriaMatrizDTO = null;

    if (proyectoDTO.getCaracterProyectoId() != null && proyectoDTO.getConvocatoriaId() !=
null) {
        Map<String, Object> filters = new HashMap<>();
        filters.put("convocatoriaId",
proyectoDTO.getConvocatoriaId().getConvocatoriaId());
        filters.put("caracterProyectoId",
proyectoDTO.getCaracterProyectoId().getValorDefinicionId());
        filters.put("activo", "S");

        int count = convocatoriaServiceFacade.convocatoriaMatrizListCount(filters);
        System.out.println("count: " + count);

        if (count == 0) {
            return null;
        }

        List<ConvocatoriaMatrizDTO> convocatoriaMatrizList
convocatoriaServiceFacade.convocatoriaMatrizList(-1, -1, "convocatoriaMatrizId", "A",
filters);
        convocatoriaMatrizDTO = convocatoriaMatrizList.get(0);

        if (count > 1) {
            if (proyectoDTO.getEntidadNId().getTamanoDTO() != null) {
                filters.put("tamanoEmpresaId",
proyectoDTO.getEntidadNId().getTamanoDTO().getValorDefinicionId());
                int countTamEmpresa
convocatoriaServiceFacade.convocatoriaMatrizListCount(filters);
                System.out.println("countTamEmpresa: " + countTamEmpresa);
            }
        }
    }
}
```

```

        if (countTamEmpresa > 0) {
            convocatoriaMatrizList =
convocatoriaServiceFacade.convocatoriaMatrizList(-1, -1, "convocatoriaMatrizId", "A",
filters);
            convocatoriaMatrizDTO = convocatoriaMatrizList.get(0);
        } else {
            throw new Exception("La Convocatoria " +
proyectoDTO.getConvocatoriaId().getDescripcionConvocatoria()
+ " al cual aplicó el proyecto que desea
evaluar NO tiene Matriz de Evaluación Configurada por Tamaño de Empresa, por tal motivo no
puede ser evaluado");
        }
    } else {
        throw new Exception("La Entidad " +
proyectoDTO.getEntidadNId().getNombreEntidad()
+ " al cual esta asociado el proyecto que desea
evaluar NO ha registrado su tamaño en el formulario de Entidades, por tal motivo no puede ser
evaluado");
    }
} else if (convocatoriaMatrizDTO.getAplicaTamanoEmpresa() != null &&
convocatoriaMatrizDTO.getAplicaTamanoEmpresa().equals("S")) {
    if (proyectoDTO.getEntidadNId().getTamanoDTO() != null) {
        Long tamanoMatriz =
convocatoriaMatrizDTO.getTamanoEmpresaId().getValorDefinicionId();
        Long tamanoEntidad =
proyectoDTO.getEntidadNId().getTamanoDTO().getValorDefinicionId();

        if (!tamanoMatriz.equals(tamanoEntidad)) {
            throw new Exception("La Convocatoria " +
proyectoDTO.getConvocatoriaId().getDescripcionConvocatoria()
+ " al cual aplicó el proyecto que desea
evaluar NO tiene Matriz de Evaluación Configurada por Tamaño de Empresa, por tal motivo no
puede ser evaluado");
        }
    } else {
        throw new Exception("La Entidad " +
proyectoDTO.getEntidadNId().getNombreEntidad()
+ " al cual esta asociado el proyecto que desea
evaluar NO ha registrado su tamaño en el formulario de Entidades, por tal motivo no puede ser
evaluado");
    }
}
}
return convocatoriaMatrizDTO;
}

```

Este método valida la configuración de la matriz de evaluación para un proyecto específico y realiza las siguientes acciones:

1. Validaciones iniciales:

- Comprueba que el proyecto tenga definido tanto un carácter (**caracterProyectoId**) como una convocatoria (**convocatoriaId**).

2. Construcción de filtros:

- Define un mapa de filtros para buscar configuraciones de matrices activas asociadas a la convocatoria y el carácter del proyecto.

3. Consulta inicial de la matriz:

- Llama al servicio **convocatoriaMatrizListCount** para verificar si existen matrices configuradas.
- Si no hay configuraciones disponibles, retorna **null**.

4. Gestión de múltiples matrices:

- Si existen múltiples configuraciones, valida si la matriz aplica por tamaño de empresa (**aplicaTamanoEmpresa**):
 - Si aplica, verifica que el tamaño de la entidad asociada al proyecto coincide con el de la configuración.
 - Si no coincide o no está registrado, lanza una excepción indicando que no puede evaluarse.

5. Retorno de la matriz:

- Si se encuentra una matriz válida, la retorna.

La anterior lógica se aplicó para los módulos de evaluación y evaluar de beneficios tributarios.

18. Soporte y capacitación para nuevos requerimientos en el uso de los módulos del SIGP al personal de OTSI de MinCiencias. Y revisión de problemas encontrados al usar la aplicación. (No. Obligación Contrato: 2.2.1, 2.2.5, 2,2,6)

a. Anonimizar Reportes de Evaluación

Solicitud de requerimiento No. 153264

Descripción y alcance del Requerimiento

Anonimización de reportes del tercer evaluador para asegurar la integridad y confidencialidad de los datos y que traiga el concepto del tercer evaluador.

Se requiere un reporte para los proponentes cuando se asigna un tercer evaluador en las respectivas convocatorias que cursan en el ministerio. obtención del reporte de evaluación anonimizado. Esta acción busca garantizar el cumplimiento de los principios de transparencia, publicidad, debido proceso y el acceso a la información, derechos que protegen a los proponentes y participantes en las convocatorias públicas, abiertas y competitivas de MINCIENCIAS

Desarrollo e Implementación

Este requerimiento tiene como objetivo garantizar la confidencialidad de los evaluadores en los procesos de evaluación de proyectos, abarcando los módulos de Proyectos y Programas y Beneficios Tributarios.

Para lograr este objetivo, se realizaron ajustes en la generación de reportes, sustituyendo los nombres reales de los evaluadores por una numeración estandarizada dentro de cada proyecto, asegurando la coherencia en la identificación de los evaluadores y la protección de sus datos personales.

Los cambios implementados incluyen:

1. Anonimización de Evaluadores:

- Eliminación de los nombres reales de los evaluadores en los reportes generados.
- Asignación de un número único a cada evaluador dentro del contexto del proyecto (Ej: Evaluador 1, Evaluador 2, etc.).
- Aplicación de la numeración en la interfaz de usuario y en los reportes en formato PDF.

2. Modificaciones en la Interfaz de Evaluación:

- Incorporación de una nueva columna denominada "#" en la tabla de evaluaciones.
- Implementación de lógica en la interfaz para reflejar la numeración de evaluadores de manera consistente.

3. Actualización de Reportes:

- Ajuste en la generación de reportes mediante JasperReports, modificando los templates para reemplazar nombres de evaluadores con su respectivo número asignado.

4. Adaptación de los siguientes reportes:

- **Proyectos y Programas:** Reporte de Evaluación, Reporte de Calificación y Reporte de Regalías.
- **Beneficios Tributarios:** Reporte de Evaluación y Reporte de Calificación.

5. Ajustes en Base de Datos y Servicios:

- Modificaciones en consultas JPA (Java Persistence API) para excluir los nombres de los evaluadores y, en su lugar, asignar la numeración correspondiente.
- Implementación de lógica en los controladores de evaluación para gestionar la asignación y visualización de los números de evaluador.

6. Tecnologías Utilizadas

El desarrollo se llevó a cabo utilizando:

- Java EE y EJB para la capa de negocio.
- JSF (JavaServer Faces) y PrimeFaces para la interfaz de usuario.
- JasperReports para la generación de reportes en PDF.
- JPA con Criteria API para la gestión y recuperación de datos desde la base de datos.
- Objetivo del Documento

Módulo de Proyectos y Programas

Codificación

Componentes

Para la generación de los reportes de evaluación en el módulo de Proyectos y Programas, se utilizan varios componentes dentro de la aplicación, los cuales trabajan en conjunto para permitir la consulta, procesamiento y descarga del reporte en formato PDF con la nueva estructura de anonimización de evaluadores.

Los principales componentes involucrados en esta modificación son los siguientes:

- **Vista (View):** Representada por el archivo **evaluacion.xhtml**, que contiene la interfaz de usuario en PrimeFaces.
- **Controlador (Controller):** Representado por la clase **EvaluacionConvController.java**, encargada de manejar la lógica de negocio relacionada con la evaluación de proyectos y la generación de reportes. Se ha ajustado la lógica de generación de reportes para incluir

la nueva numeración de evaluadores y garantizar la referencia correcta a estos en los documentos generados.

- **Reportes (JasperReports Template):** Se han modificado los siguientes archivos JasperReports para implementar la anonimización y la referencia mediante numeración de evaluadores:
 - ReporteCalificaciónConcepto.jrxml
 - reporteEvaluacionProyecto.jrxml
 - ReporteEvaluacionCalificacion.jrxml
 - ReporteEvaluacionCalificacionNueva.jrxml
 - ReporteEvaluacionCalificacionProg.jrxml
 - ReporteEvaluacionRegalias.jrxml
- **Base de Datos (Database):** Contiene la información de las evaluaciones de los proyectos, almacenadas en tablas específicas. Se accede a través de consultas JPA (**Java Persistence API**) y EJB (**Enterprise JavaBeans**) para recuperar los datos necesarios para los reportes. Se han ajustado las consultas para excluir los nombres de los evaluadores y, en su lugar, obtener su identificación basada en la nueva numeración asignada en la vista.

Cada uno de estos componentes interactúa de manera coordinada para garantizar la correcta generación de los reportes en Proyectos y Programas, asegurando la anonimización de los evaluadores y la correcta referencia a través de la numeración asignada en la interfaz.

Vista (evaluacion.xhtml)

El archivo encargado de la vista es **evaluacion.xhtml**. Al ingresar a la pantalla de **Evaluación de Proyectos/Programas** y seleccionar la opción **“Evaluar”** para un proyecto, se despliegan las distintas opciones para generar los reportes de evaluación, representadas mediante botones de descarga, los cuales permiten obtener los reportes en formato PDF.

Además, dentro de la interfaz se encuentra la tabla de evaluaciones, la cual muestra la información correspondiente a cada evaluación realizada dentro del proyecto. Como parte de la actualización, se ha incorporado una nueva columna denominada **“#”**, la cual asigna a cada evaluador un número único dentro del contexto del proyecto. Esta numeración se refleja en términos de **“Evaluador 1”**,

"Evaluador 2", "Evaluador 3", ..., lo que permite identificar a los evaluadores de manera estandarizada en la interfaz y mantener la coherencia con la anonimización aplicada en los reportes generados.

Estructura de la columna "#" en la tabla de Evaluaciones

```
<h:column headerClass="column1">
  <!-- column header -->
  <f:facet name="header">#</f:facet>
  <!-- row record -->
  #{proyectoEvaluacionSessController.listaProyectoEvaluaciones.indexOf(evaluacion) + 1}
</h:column>
```

A continuación, una explicación más detallada:

- **<h:column>**: Define una columna dentro de la tabla de evaluaciones.
- **headerClass="column1"**: Aplica una clase CSS al encabezado de la columna para su estilización.
- **<f:facet name="header">#</f:facet>**: Establece el encabezado de la columna, representado por el símbolo "#".
- **#{proyectoEvaluacionSessController.listaProyectoEvaluaciones.indexOf(evaluacion) + 1}**:
 - Obtiene el índice de la evaluación dentro de la lista **listaProyectoEvaluaciones**.
 - Le suma 1 para que la numeración inicie en 1 en lugar de 0.
 - Permite asignar un número único a cada evaluador dentro del contexto del proyecto.

Controlador (EvaluacionConvController.java)

El controlador **evaluacionConvController** es el bean de respaldo (Backing Bean) de la vista de evaluación de proyectos. Está anotado con **@ConversationScoped**, lo que significa que su ciclo de vida está ligado a una conversación de JSF, permitiendo mantener el estado entre múltiples solicitudes dentro de una misma sesión de usuario.

```
@Named(value = "evaluacionConvController")
@ConversationScoped
public class EvaluacionConvController extends GenericConversationController<ProyectoEvaluacionDTO>
implements Serializable {
```

A continuación, una explicación más detallada de sus atributos:

- **@Named(value = "evaluacionConvController")**
 - Permite que el bean sea accesible desde las páginas XHTML mediante Expression Language (**`#{evaluacionConvController}`**).
 - Se le asigna un nombre explícito (**`evaluacionConvController`**) para su uso en la vista.
- **@ConversationScoped**
 - Define que el bean tiene un ciclo de vida basado en conversaciones de JSF.
 - Esto significa que el estado del bean se mantiene mientras la conversación esté activa, evitando la pérdida de datos entre peticiones.
- **extends GenericConversationController<ProyectoEvaluacionDTO>**
 - **Herencia:** **`evaluacionConvController`** extiende de **`GenericConversationController`**, lo que sugiere que hereda métodos y funcionalidades genéricas para manejar conversaciones en JSF.
 - **Parámetro Genérico:** **`ProyectoEvaluacionDTO`**, indicando que este controlador maneja objetos de tipo **`ProyectoEvaluacionDTO`**.
- **implements Serializable**
 - Requisito para que el estado del bean pueda ser serializado y almacenado en la sesión si es necesario.

En conclusión, el controlador **`EvaluacionConvController`** es un bean de respaldo (**`@ConversationScoped`**) que gestiona la lógica relacionada con la evaluación de proyectos dentro de la aplicación. Hereda funcionalidades genéricas para la gestión de entidades y mantiene la conversación activa entre peticiones, asegurando que los datos no se pierdan durante la navegación del usuario.

Dentro de este controlador, el método **`downloadReporteEvaluacionCalificacion()`** es el encargado de generar el reporte individual de cada evaluación asignada al proyecto. Cada evaluación dentro de la tabla de evaluaciones tiene un botón de descarga, el cual invoca este método para generar

y obtener el reporte correspondiente a esa evaluación en particular. A continuación, se presenta el código principal de este método:

```
public void downloadReporteEvaluacionCalificacion(Integer evaluadorId, Integer numeroEvaluador) {
    FacesContext context = FacesContext.getCurrentInstance();
    HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();
    String proyectoId =
String.valueOf(proyectoEvaluacionSessController.getSelected().getProyectoId());
    String nombreReporte = "";
    try {
        ReporteUtil reporteUtil = new ReporteUtil();
        HashMap hashMap = new HashMap();
        hashMap.put("PROYECTO_ID", proyectoId);
        hashMap.put("evaluacion_proyecto_id", String.valueOf(evaluadorId));
        byte reporte[];

        //carpeta por defecto
        reporteUtil.setNomCarpeta("ReporteEvaluacionCalificacion");
        /**
         * Se verifica si el proyecto tiene mecanismos de participacion
         * (configuracion de la matriz por mecanismo), en caso contrario se
         * verifica segun la matriz configurada en la convocatoria
         */
        if (proyectoEvaluacionSessController.getSelected().getConvocatoriaMecanismoId() == null) {
            ValorDefinicionDTO matrizEvaluacionId = null;
            //Se verifica si la matriz de evaluacion es de proyectos o de proyectos y programas, o
de regalías y en base a esto, generar el reporte
            if (proyectoEvaluacionSessController.getSelected().getExencion().equals("P")) {
                if
(proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionId()
!=
null) {
                    matrizEvaluacionId
=
proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionId();
                } else if (proyectoEvaluacionSessController.getSelected().getDependenciaId() != null
&& proyectoEvaluacionSessController.getSelected().getAreaTematicaId() !=
null) {
                    //Obtenemos la matriz de la configuracion de la tematica en la convocatoria
ConvocatoriaTematicaDTO convocatoriaTematicaDTO =
obtenerConvocatoriaTematica(proyectoEvaluacionSessController.getSelected());
                    if (convocatoriaTematicaDTO != null &&
convocatoriaTematicaDTO.getMatrizEvaluacionId() != null) {
                        matrizEvaluacionId = convocatoriaTematicaDTO.getMatrizEvaluacionId();
                    } else {
                        throw new Exception("No se ha configurado matriz de evaluación de proyectos
en la convocatoria. Por favor comuníquese con soporte.");
                    }
                } else {
                    throw new Exception("No se ha configurado matriz de evaluación de proyectos en
la convocatoria. Por favor comuníquese con soporte.");
                }
            } else { // PROGRAMA
                if
(proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionProgId() !=
null) {
                    matrizEvaluacionId
=
proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionProgId();
                } else if (proyectoEvaluacionSessController.getSelected().getDependenciaId() != null
```

```

        && proyectoEvaluacionSessController.getSelected().getAreaTematicaId() !=
null) {
        //Obtenemos la matriz de la configuracion de la tematica en la convocatoria
        ConvocatoriaTematicaDTO convocatoriaTematicaDTO =
obtenerConvocatoriaTematica(proyectoEvaluacionSessController.getSelected());
        if (convocatoriaTematicaDTO != null &&
convocatoriaTematicaDTO.getMatrizEvaluacionId() != null) {
            matrizEvaluacionId = convocatoriaTematicaDTO.getMatrizEvaluacionId();
        } else {
            throw new Exception("No se ha configurado matriz de evaluación de programas
en la convocatoria. Por favor comuníquese con soporte.");
        }
        } else {
            throw new Exception("No se ha configurado matriz de evaluación de programas en
la convocatoria. Por favor comuníquese con soporte.");
        }
    }
    // Redireccionamos a la matriz de evaluación correspondiente
    if (matrizEvaluacionId != null) {
        if (matrizEvaluacionId.getComentario() == null) {
            nombreReporte = "ReporteEvaluacionCalificacion";
        } else {
            switch (matrizEvaluacionId.getComentario()) {
                case "P":
                    nombreReporte = "ReporteEvaluacionCalificacion";
                    break;
                case "PRY":
                    nombreReporte = "ReporteEvaluacionCalificacionNueva";
                    break;
                case "REG":
                    reporteUtil.setNomCarpeta("ReporteEvaluacionRegalias");
                    nombreReporte = "ReporteEvaluacionRegalias";
                    break;
                default:
                    nombreReporte = "ReporteEvaluacionCalificacionProg";
                    break;
            }
        }
    }
    } else {
        reporteUtil.setNomCarpeta("ReporteEvaluacionRegalias");
        nombreReporte = "ReporteEvaluacionRegalias";
    }

    //Se añade el número del evaluador
    hashMap.put("NUMERO_EVALUADOR", numeroEvaluador);

    //Se setea la imagen del logo parametrizada, guardada en la base de datos
    String imagenLogoBase64 =
proyectosServiceFacade.obtenerParametroSistema("LOGIN_LOGO").getDetalle();
    String[] parts = imagenLogoBase64.split(",");
    String part1 = parts[0]; // data:image/jpeg;base64
    String imagen = parts[1]; // IMAGEN
    hashMap.put("IMG_BASE64", imagen);
    //
    reporte = reporteUtil.generarReporte(hashMap, nombreReporte, "PDF", context);
    response.setContentType("application/pdf");
    response.setHeader("Content-Disposition", "attachment;
filename=\"ReporteEvaluacionCalificacion_\"
+
proyectoEvaluacionSessController.getSelected().getProyectoId() + ".pdf\"");
    response.setContentLength(reporte.length);

```

```
ServletOutputStream outputStream = response.getOutputStream();
outputStream.write(reporte, 0, reporte.length);
outputStream.flush();
outputStream.close();
} catch (Exception ex) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
    resolveExpression(ex.getMessage()));
    LOG.error("Error al generar el reporte de evaluacion calificación", ex);

    FacesContext.getCurrentInstance().addMessage(null, message);
}
}
```

A continuación, se describe en detalle su implementación, explicando cada una de sus partes y su impacto en el proceso de generación y descarga del informe:

1. Firma del método

```
public void downloadReporteEvaluacionCalificacion(Integer evaluacionProyectoId, Integer
numeroEvaluador) { ... }
```

- **evaluadorId:** Identificador único de la evaluación seleccionada del proyecto.
- **numeroEvaluador:** Número asignado al evaluador (Ej: Evaluador 1, Evaluador 2, etc.).

2. Obtención del FacesContext y configuración de la respuesta HTTP

```
FacesContext context = FacesContext.getCurrentInstance();
HttpServletResponse response = (HttpServletResponse) context.getExternalContext().getResponse();
String proyectoId = String.valueOf(proyectoEvaluacionSessController.getSelected().getProyectoId());
String nombreReporte = "";
```

- **FacesContext.getCurrentInstance():** Obtiene la instancia actual de JSF.
- **context.getExternalContext().getResponse():** Permite acceder a la respuesta HTTP, donde se configurará el archivo PDF para descarga.
- **proyectoEvaluacionSessController.getSelected().getProyectoId():** Recupera el ID del proyecto actualmente seleccionado.
- **nombreReporte:** Variable que almacenará el nombre del reporte a generar según la matriz de evaluación.

3. Creación del HashMap con los parámetros del reporte

```
ReporteUtil reporteUtil = new ReporteUtil();
HashMap hashMap = new HashMap();
```

```
hashMap.put("PROYECTO_ID", proyectoId);
hashMap.put("evaluacion_proyecto_id", String.valueOf(evaluadorId));
```

- **ReporteUtil reporteUtil = new ReporteUtil();**: Instancia la utilidad para generar reportes.
- **HashMap hashMap = new HashMap();**: Se usa para almacenar los parámetros que se pasarán al reporte.
- **hashMap.put("PROYECTO_ID", proyectoId);**: Se añade el ID del proyecto al conjunto de parámetros.
- **hashMap.put("evaluacion_proyecto_id", String.valueOf(evaluadorId));**: Se añade el ID de la evaluación asociada al proyecto.

4. Determinación del tipo de reporte según la matriz de evaluación

Antes de generar el reporte, se determina la matriz de evaluación que debe utilizarse en función de la configuración del proyecto y la convocatoria.

```
if (proyectoEvaluacionSessController.getSelected().getConvocatoriaMecanismoId() == null) {
    ValorDefinicionDTO matrizEvaluacionId = null;

    // Determinación de la matriz según el tipo de proyecto
    if (proyectoEvaluacionSessController.getSelected().getExencion().equals("P")) {
        // Proyectos
        if
        (proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionId() !=
        null) {
            matrizEvaluacionId =
            proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionId();
        } else if (proyectoEvaluacionSessController.getSelected().getDependenciaId() != null
            && proyectoEvaluacionSessController.getSelected().getAreaTematicaId() != null) {
            ConvocatoriaTematicaDTO convocatoriaTematicaDTO =
            obtenerConvocatoriaTematica(proyectoEvaluacionSessController.getSelected());
            if (convocatoriaTematicaDTO != null && convocatoriaTematicaDTO.getMatrizEvaluacionId()
            != null) {
                matrizEvaluacionId = convocatoriaTematicaDTO.getMatrizEvaluacionId();
            } else {
                throw new Exception("No se ha configurado matriz de evaluación de proyectos en la
                convocatoria. Por favor comuníquese con soporte.");
            }
        } else {
            throw new Exception("No se ha configurado matriz de evaluación de proyectos en la
            convocatoria. Por favor comuníquese con soporte.");
        }
    } else {
        // Programas
        if
        (proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionProgId() !=
        null) {
            matrizEvaluacionId =
            proyectoEvaluacionSessController.getSelected().getConvocatoriaId().getMatrizEvaluacionProgId();
        } else if (proyectoEvaluacionSessController.getSelected().getDependenciaId() != null
```



```
        && proyectoEvaluacionSessController.getSelected().getAreaTematicaId() != null) {  
            ConvocatoriaTematicaDTO convocatoriaTematicaDTO =  
obtenerConvocatoriaTematica(proyectoEvaluacionSessController.getSelected());  
            if (convocatoriaTematicaDTO != null && convocatoriaTematicaDTO.getMatrizEvaluacionId()  
!= null) {  
                matrizEvaluacionId = convocatoriaTematicaDTO.getMatrizEvaluacionId();  
            } else {  
                throw new Exception("No se ha configurado matriz de evaluación de programas en la  
convocatoria. Por favor comuníquese con soporte.");  
            }  
        } else {  
            throw new Exception("No se ha configurado matriz de evaluación de programas en la  
convocatoria. Por favor comuníquese con soporte.");  
        }  
    }  
}
```

- Se verifica si el proyecto tiene un mecanismo de participación configurado, validando si el atributo **ConvocatoriaMecanismoId** es nulo. En caso de no existir una configuración específica, se determina la matriz de evaluación a utilizar según la convocatoria asociada al proyecto.
- Si el proyecto pertenece a la categoría **"Proyectos" (Exencion == "P")**, se intenta recuperar la matriz desde la convocatoria representada por el objeto **ConvocatoriaId**. Si esta no está definida, se busca en la configuración de la temática (**DependenciaId** y **AreaTematicaId**), accediendo al objeto **ConvocatoriaTematicaDTO** mediante la función **obtenerConvocatoriaTematica()**. Si ninguna de estas opciones devuelve una matriz válida, se lanza una excepción para evitar la generación del reporte sin una estructura de evaluación definida.
- Para los proyectos de la categoría **"Programas"**, se sigue la misma lógica, pero utilizando el atributo **MatrizEvaluacionProgId** en lugar de **MatrizEvaluacionId**, lo que permite asignar una matriz específica para programas. Si tampoco se encuentra una matriz válida en esta configuración, se intenta recuperar la matriz desde **ConvocatoriaTematicaDTO**. En caso de falla, se lanza una excepción controlada para impedir la generación de reportes sin un esquema de evaluación correctamente definido.
- Este enfoque garantiza que cada evaluación se genere con la matriz de evaluación adecuada, evitando errores en la estructura del reporte y asegurando la correcta referencia a los criterios de evaluación configurados en la convocatoria.

5. Selección del nombre del reporte según el tipo de matriz

```
if (matrizEvaluacionId != null) {
    if (matrizEvaluacionId.getComentario() == null) {
        nombreReporte = "ReporteEvaluacionCalificacion";
    } else {
        switch (matrizEvaluacionId.getComentario()) {
            case "P":
                nombreReporte = "ReporteEvaluacionCalificacion";
                break;
            case "PRY":
                nombreReporte = "ReporteEvaluacionCalificacionNueva";
                break;
            case "REG":
                reporteUtil.setNomCarpeta("ReporteEvaluacionRegalias");
                nombreReporte = "ReporteEvaluacionRegalias";
                break;
            default:
                nombreReporte = "ReporteEvaluacionCalificacionProg";
                break;
        }
    }
}
```

- Dependiendo del tipo de matriz de evaluación, se asigna un nombre diferente al reporte.
- Si la matriz no tiene un comentario, se usa el reporte **"ReporteEvaluacionCalificacion"** por defecto.
- Si pertenece a regalías ("REG"), se cambia el directorio de reportes a **"ReporteEvaluacionRegalias"**

6. Inclusión del número del evaluador en el reporte

```
hashMap.put("NUMERO_EVALUADOR", numeroEvaluador);
```

- Se agrega el número del evaluador a los parámetros del reporte.
- Este valor se usará en los reportes para reemplazar el nombre del evaluador con su número asignado (**"Evaluador 1"**, **"Evaluador 2"**, etc.).

7. Configuración de la imagen del reporte

```
String imagenLogoBase64 =
projectosServiceFacade.obtenerParametroSistema("LOGIN_LOGO").getDetalle();
String[] parts = imagenLogoBase64.split(",");
String imagen = parts[1];
hashMap.put("IMG_BASE64", imagen);
```

- **obtenerParametroSistema("LOGIN_LOGO")**: Obtiene el logo almacenado en la base de datos en base64.

- **split(",")**: Separa la imagen en base64, eliminando metadatos.
- Se almacena la imagen en hashMap para usarla en el reporte.

8. Generación del Reporte y Configuración de la Respuesta HTTP

```
reporte = reporteUtil.generarReporte(hashMap, nombreReporte, "PDF", context);
response.setContentType("application/pdf");
response.setHeader("Content-Disposition", "attachment; filename=\"ReporteEvaluacionCalificacion_"
    + proyectoEvaluacionSessController.getSelected().getProyectoId() + ".pdf\"");
response.setContentLength(reporte.length);
```

- **generarReporte()**: Se invoca la utilidad de **JasperReports** para generar el reporte en formato PDF.
- **Configuración de la respuesta HTTP**:
 - **Content-Type**: Define el tipo de contenido como PDF.
 - **Content-Disposition**: Configura el nombre del archivo para la descarga.
 - **Content-Length**: Establece el tamaño del archivo para una correcta transferencia.

9. Escritura y Envío del Reporte al Cliente

```
ServletOutputStream outputStream = response.getOutputStream();
outputStream.write(reporte, 0, reporte.length);
outputStream.flush();
outputStream.close();
```

- Se obtiene un flujo de salida (**ServletOutputStream**) para escribir el archivo en la respuesta.
- **write()**: Se escribe el contenido del reporte en el flujo de salida.
- **flush()**: Asegura que todos los datos sean enviados antes de cerrar el flujo.
- **close()**: Cierra el flujo para liberar los recursos.

10. Manejo de Excepciones

```
} catch (Exception ex) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
        resolveExpression(ex.getMessage()));
    LOG.error("Error al generar el reporte de evaluación calificación", ex);
    FacesContext.getCurrentInstance().addMessage(null, message);
}
```

- Captura cualquier error durante la generación del reporte y lo registra en los logs.
- Se muestra un mensaje de error en la interfaz si ocurre una excepción.

Reportes JasperReports

Reportes de Calificación por Evaluación

Estos reportes corresponden a los documentos generados para cada una de las evaluaciones individuales asignadas a un proyecto. La generación del reporte depende de la matriz de evaluación configurada en la convocatoria del proyecto, lo que determina la estructura y el contenido del informe. Cada reporte se encuentra disponible en la tabla de evaluaciones del proyecto, desde donde puede ser descargado de manera individual.

[+ Agregar](#)

Show entries

Search:

#	Tipo Evaluación	Tipo Evaluador	Nombres	Usuario Sigg	Opciones	Habilitar Evaluación
1	Técnica	Externo	LEONARDO SUAREZ BALDOVINO	EVA_LEOSUAREZB	  	
2	Técnica	Externo	John Fernando Cardona Tangarife	EVA_JFCARDONA	  	
3	Técnica	Externo	Carlos Antonio Pardo Adames	EVA_CAPARDO	  	
4	Panel		Panel prueba lasb		  	
5	Técnica	Externo	LUZ AMERICA CORDOBA CASTRO	EVA_LACORDOBA	  	
6	Técnica	Interno	Leonardo Andrés Suarez Baldovino	LEOSUAREZB	  	
7	Panel		PANEL PRUEBA		  	
8	Jurídica	Externo	Adriana María Martínez Arias	EVA_AMMARTINEZAR	  	

Showing 1 to 8 of 8 entries

Previous 1 Next

A continuación, cada reporte modificado:

ReporteEvaluaciónCalificacion.jrxml

Este reporte se genera cuando la matriz de evaluación asociada a la convocatoria del proyecto tiene configurado en el campo **COMENTARIO** el valor “P”, o no tiene configurado un valor para este campo, lo que indica que no se ha especificado una variante particular para su estructura. En estos casos, el sistema selecciona este formato de reporte como el predeterminado para presentar la evaluación.

A continuación, los principales cambios realizados:

1. Nuevo Parámetro NUMERO_EVALUADOR

El parámetro **NUMERO_EVALUADOR** permite identificar a cada evaluador sin exponer su nombre real, garantizando la anonimización en los reportes. Su valor es determinado en el controlador, donde se asigna el número correspondiente a cada evaluador dentro del proyecto.

PARÁMETRO	TIPO	DESCRIPCIÓN
NUMERO_EVALUADOR	Integer	Identificador del evaluador asignado a la evaluación

2. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA[
    !${NOMBRE_PANEL}.equals(null) ? ${NOMBRE_PANEL}
    : !${NOMBRE_PERSONA}.equals(null) ? ${NOMBRE_PERSONA}
    : !${FUNCIONARIO}.equals(null) ? ${FUNCIONARIO}
    : "No hay nombre asignado"
]]></textFieldExpression>
```

- **Expresión Nueva:**

```
<textFieldExpression><![CDATA["Evaluador " + ${NUMERO_EVALUADOR}]]></textFieldExpression>
```

Este cambio permite la anonimización total de los evaluadores en el reporte, eliminando la lógica condicional que verificaba los distintos campos (**NOMBRE_PANEL**, **NOMBRE_PERSONA**, **FUNCIONARIO**) para obtener el nombre del evaluador y asignando directamente el número del evaluador en el formato "**Evaluador X**" a través del parámetro **NUMERO_EVALUADOR**. Además, garantiza la coherencia con la numeración utilizada en la interfaz de usuario y facilita la estandarización en la referencia a los evaluadores en todos los reportes generados.

ReporteEvaluaciónCalificacionNueva.jrxml

Este reporte se genera cuando la matriz de evaluación asociada a la convocatoria del proyecto tiene configurado en el campo **COMENTARIO** el valor "**PRY**". En estos casos, el sistema selecciona este formato de reporte como el predeterminado para presentar la evaluación.

A continuación, los principales cambios realizados:

1. Nuevo Parámetro NUMERO_EVALUADOR

El parámetro **NUMERO_EVALUADOR** permite identificar a cada evaluador sin exponer su nombre real, garantizando la anonimización en los reportes. Su valor es determinado en el controlador, donde se asigna el número correspondiente a cada evaluador dentro del proyecto.

PARÁMETRO	TIPO	DESCRIPCIÓN
NUMERO_EVALUADOR	Integer	Identificador del evaluador asignado a la evaluación

2. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA[!${F{NOMBRE_PANEL}.equals(null)} ? ${F{NOMBRE_PANEL}}
: !${F{NOMBRE_PERSONA}.equals(null)} ? ${F{NOMBRE_PERSONA}}
: !${F{FUNCIONARIO}.equals(null)} ? ${F{FUNCIONARIO}}:"No hay nombre
asignado"]]></textFieldExpression>
```

- **Expresión Nueva:**

```
<textFieldExpression><![CDATA["Evaluador " + ${P{NUMERO_EVALUADOR}}]></textFieldExpression>
```

Este cambio permite la anonimización total de los evaluadores en el reporte, eliminando la lógica condicional que verificaba los distintos campos (**NOMBRE_PANEL**, **NOMBRE_PERSONA**, **FUNCIONARIO**) para obtener el nombre del evaluador y asignando directamente el número del evaluador en el formato "**Evaluador X**" a través del parámetro **NUMERO_EVALUADOR**. Además, garantiza la coherencia con la numeración utilizada en la interfaz de usuario y facilita la estandarización en la referencia a los evaluadores en todos los reportes generados.

ReporteEvaluaciónCalificacionRegalias.jrxml

Este reporte se genera cuando la matriz de evaluación asociada a la convocatoria del proyecto tiene configurado en el campo **COMENTARIO** el valor **"REG"**. En estos casos, el sistema selecciona este formato de reporte como el predeterminado para presentar la evaluación.

A continuación los principales cambios realizados:

1. Nuevo Parámetro NUMERO_EVALUADOR

El parámetro **NUMERO_EVALUADOR** permite identificar a cada evaluador sin exponer su nombre real, garantizando la anonimización en los reportes. Su valor es determinado en el controlador, donde se asigna el número correspondiente a cada evaluador dentro del proyecto.

PARÁMETRO	TIPO	DESCRIPCIÓN
NUMERO_EVALUADOR	Integer	Identificador del evaluador asignado a la evaluación

2. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA[!${F{NOMBRE_PANEL}.equals(null)} ? ${F{NOMBRE_PANEL}}
: !${F{NOMBRE_PERSONA}.equals(null)} ? ${F{NOMBRE_PERSONA}}
: !${F{FUNCIONARIO}.equals(null)} ? ${F{FUNCIONARIO}}:"No hay nombre
asignado"]]></textFieldExpression>
```

- **Expresión Nueva:**

```
<textFieldExpression><![CDATA["Evaluador " + ${F{NUMERO_EVALUADOR}}]></textFieldExpression>
```

Este cambio permite la anonimización total de los evaluadores en el reporte, eliminando la lógica condicional que verificaba los distintos campos (**NOMBRE_PANEL**, **NOMBRE_PERSONA**, **FUNCIONARIO**) para obtener el nombre del evaluador y asignando directamente el número del evaluador en el formato **"Evaluador X"** a través del parámetro **NUMERO_EVALUADOR**. Además,

garantiza la coherencia con la numeración utilizada en la interfaz de usuario y facilita la estandarización en la referencia a los evaluadores en todos los reportes generados.

ReporteEvaluaciónCalificacionProg.jrxml

Este reporte se genera cuando la matriz de evaluación asociada a la convocatoria del proyecto tiene configurado en el campo **COMENTARIO** diferente a los valores anteriores. En estos casos, el sistema selecciona este formato de reporte como el predeterminado para presentar la evaluación.

A continuación, los principales cambios realizados:

1. Nuevo Parámetro NUMERO_EVALUADOR

El parámetro **NUMERO_EVALUADOR** permite identificar a cada evaluador sin exponer su nombre real, garantizando la anonimización en los reportes. Su valor es determinado en el controlador, donde se asigna el número correspondiente a cada evaluador dentro del proyecto.

PARÁMETRO	TIPO	DESCRIPCIÓN
NUMERO_EVALUADOR	Integer	Identificador del evaluador asignado a la evaluación

2. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA[!${NOMBRE_PANEL}.equals(null) ? ${NOMBRE_PANEL}
: !${NOMBRE_PERSONA}.equals(null) ? ${NOMBRE_PERSONA}
: !${FUNCIONARIO}.equals(null) ? ${FUNCIONARIO}: "No hay nombre
asignado"]]></textFieldExpression>
```

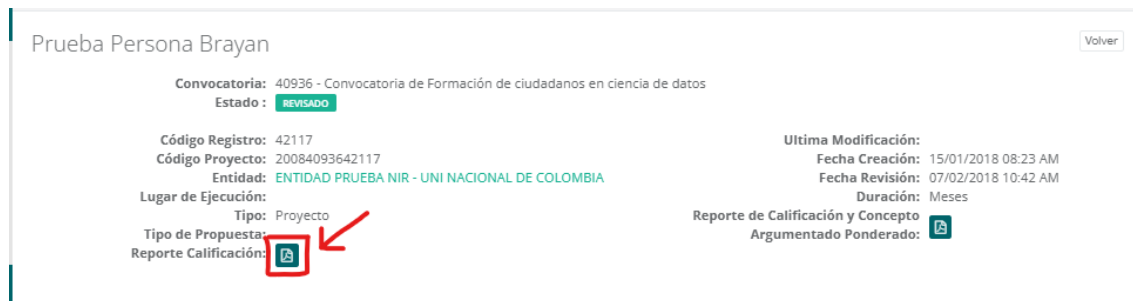
- **Expresión Nueva:**

```
<textFieldExpression><![CDATA["Evaluador " + ${NUMERO_EVALUADOR}]]></textFieldExpression>
```


Este cambio permite la anonimización total de los evaluadores en el reporte, eliminando la lógica condicional que verificaba los distintos campos (**NOMBRE_PANEL**, **NOMBRE_PERSONA**, **FUNCIONARIO**) para obtener el nombre del evaluador y asignando directamente el número del evaluador en el formato "**Evaluador X**" a través del parámetro **NUMERO_EVALUADOR**. Además, garantiza la coherencia con la numeración utilizada en la interfaz de usuario y facilita la estandarización en la referencia a los evaluadores en todos los reportes generados.

Reporte de Calificación

Este reporte corresponde al documento generado para cada evaluación del proyecto, detallando la calificación final y el concepto final otorgado por cada evaluador. Además, incluye el promedio final de las evaluaciones, permitiendo visualizar de manera consolidada el desempeño del proyecto según los criterios establecidos en la matriz de evaluación.



Prueba Persona Brayan Volver

Convocatoria: 40936 - Convocatoria de Formación de ciudadanos en ciencia de datos
Estado: **REVISADO**

Código Registro: 42117
Código Proyecto: 20084093642117
Entidad: **ENTIDAD PRUEBA NIR - UNI NACIONAL DE COLOMBIA**
Lugar de Ejecución:
Tipo: Proyecto
Tipo de Propuesta:
Reporte Calificación: 

Ultima Modificación:
Fecha Creación: 15/01/2018 08:23 AM
Fecha Revisión: 07/02/2018 10:42 AM
Duración: Meses
Reporte de Calificación y Concepto 
Argumentado Ponderado: 

ReporteEvaluaciónCalificación.jrxml

El reporte **ReporteEvaluaciónCalificación.jrxml** contiene el subreporte "**evaluadoresConcepto.jrxml**", el cual detalla la calificación final y el concepto final asignado por cada evaluador en la evaluación del proyecto. Para estructurar correctamente esta información, el reporte utiliza una consulta SQL dinámica que permite identificar y ordenar a los evaluadores según su tipo de evaluación y su posición dentro del proyecto.

1. SQL principal del Reporte

La consulta principal del reporte es utilizada para obtener los diferentes datos mostrados en el informe, incluyendo la lista de evaluadores asignados al proyecto, su calificación y su concepto final. Además, asigna un número secuencial (**POSICION**) a cada evaluación, lo que permite referenciar a los evaluadores en el reporte sin exponer su nombre real. También organiza la información según el tipo de evaluación, asegurando una estructura clara y ordenada. Adicionalmente, la consulta realiza cálculos como el puntaje promedio de las evaluaciones y agrega registros especiales cuando no hay evaluaciones de ciertos tipos, garantizando que el reporte siempre contenga información relevante.

```
SELECT * FROM (
SELECT UPPER(PRYEVALUACIONES.FUNCIONARIO) FUNCIONARIO,
(CASE
WHEN PRYEVALUACIONES.TIPO_EVALUACION = 'P' THEN
PRYEVALUACIONES.Nombre_Panel
WHEN PRYEVALUACIONES.Evaluador_Scienti_Id IS NOT NULL THEN
(CONCAT(CONCAT(SGPEVALUADORESSCIENTI.NOMBRE_EVALUADOR || ' ',
SGPEVALUADORESSCIENTI.PRIMER_APELLIDO || ' '),
SGPEVALUADORESSCIENTI.SEGUNDO_APELLIDO))
ELSE
(CONCAT(CONCAT(SGPPERSONAS.NOMBRES || ' ',
SGPPERSONAS.PRIMER_APELLIDO || ' '),
SGPPERSONAS.SEGUNDO_APELLIDO))
END ) NOMBRE,
NVL(dbms_lob.substr(PRYEVALUACIONES.CONCEPTO,4000,1), '') concepto,
NVL(dbms_lob.substr(PRYEVALUACIONES.CONCEPTO,8000,4001),'') concepto_parte2,
PRYEVALUACIONES.PUNTAJE,
PRYEVALUACIONES.USUARIO_CREADOR,
PRYEVALUACIONES.FECHA_CREACION,
PRYEVALUACIONES.USUARIO_MODIFICADOR,
PRYEVALUACIONES.FECHA_MODIFICACION,
UPPER(SGPPERSONAS.NOMBRES || ' ' || SGPPERSONAS.PRIMER_APELLIDO || ' ' ||
SGPPERSONAS.SEGUNDO_APELLIDO) AS NOMBRE_PERSONA,
(CASE
WHEN PRYEVALUACIONES.TIPO_EVALUACION = 'P' THEN
2
WHEN PRYEVALUACIONES.TIPO_EVALUACION = 'C' THEN
1
ELSE
0
END) ORDEN,
SUB.POSICION
FROM PRY_EVALUACIONES PRYEVALUACIONES,
SGP_PERSONAS SGPPERSONAS,
SGP_PROYECTOS SGPPROYECTOS,
SGP_EVALUADORES_SCIENTI SGPEVALUADORESSCIENTI,
(
SELECT p.evaluacion_proyecto_id,
ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC) AS POSICION
FROM PRY_EVALUACIONES p
WHERE p.proyecto_id = ${PROYECTO_ID}
) SUB
WHERE PRYEVALUACIONES.PERSONA_ID = SGPPERSONAS.PERSONA_ID(+)
AND (PRYEVALUACIONES.EVALUACION_PROYECTO_ID IN
```

```

        (SELECT EVALUACION_PROYECTO_ID FROM VW_FILTRO_EVALUACION))
    AND SGPPROYECTOS.PROYECTO_ID = PRYEVALUACIONES.PROYECTO_ID
    AND SGPEVALUADORESSCIENTI.EVALUADOR_SCIENTI_ID (+)= PRYEVALUACIONES.EVALUADOR_SCIENTI_ID
    AND SGPPROYECTOS.PROYECTO_ID = $P{PROYECTO_ID}
    AND SUB.EVALUACION_PROYECTO_ID (+)= PRYEVALUACIONES.EVALUACION_PROYECTO_ID
    AND SUB.EVALUACION_PROYECTO_ID IN (SELECT EP.EVALUACION_PROYECTO_ID
        FROM PRY_EVALUACIONES EP
        WHERE EP.PROYECTO_ID = $P{PROYECTO_ID})
    ORDER BY PRYEVALUACIONES.EVALUACION_PROYECTO_ID ASC
)
UNION ALL
SELECT '',
    CONCAT('PUNTAJE PROMEDIO', ''),
    '',
    '',
    (SELECT AVG(NVL(PUNTAJE,0))
        FROM PRY_EVALUACIONES E, SGP_PROYECTOS R
        WHERE E.PROYECTO_ID = R.PROYECTO_ID
        AND R.PROYECTO_ID = $P{PROYECTO_ID}
    AND (E.TIPO_EVALUACION <> 'P' OR E.TIPO_EVALUACION <> 'C')) PUNTAJE,
    '',
    TO_DATE('') FECHA_CREACION,
    '',
    TO_DATE(''),
    '',
    TO_NUMBER('1') ORDEN,
    TO_NUMBER('') POSICION
FROM DUAL
UNION ALL
SELECT '',
    CONCAT('PANEL', ''),
    '',
    '',
    TO_NUMBER('0') PUNTAJE,
    '',
    TO_DATE('') FECHA_CREACION,
    '',
    TO_DATE(''),
    '',
    TO_NUMBER('1') ORDEN,
    TO_NUMBER('') POSICION
FROM DUAL
WHERE (SELECT COUNT(*)
    FROM PRY_EVALUACIONES E, SGP_PROYECTOS R
    WHERE E.PROYECTO_ID = R.PROYECTO_ID
    AND R.PROYECTO_ID = $P{PROYECTO_ID}
    AND (E.TIPO_EVALUACION = 'P' OR E.TIPO_EVALUACION = 'C')) = 0
ORDER BY ORDEN ASC, POSICION ASC

```

A continuación una explicación detallada:

- La función **ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC)**: Asigna un número secuencial a cada evaluación dentro del proyecto, permitiendo identificar de manera única a cada evaluador.
- El operador **CASE**: Determina el nombre del evaluador utilizando una jerarquía entre los campos **NOMBRE_PANEL**, **NOMBRE_PERSONA** y **FUNCIONARIO**.
- El uso de **UNION ALL**: Permite incluir cálculos adicionales, como el promedio de las calificaciones (**AVG(PUNTAJE)**), y registros especiales para casos donde no existen evaluaciones de ciertos tipos.
- La condición **WHERE PRYEVALUACIONES.PROYECTO_ID = \$P{PROYECTO_ID}**: Filtra los datos asegurando que solo se obtengan las evaluaciones correspondientes al proyecto en curso.

Esta consulta nos permite obtener el **field POSICION**, el cual es utilizado en el contenido del reporte para referenciar a los evaluadores de manera anónima, reemplazando su nombre real por una numeración en el formato "Evaluador X".

2. Field POSICION

El campo POSICION se obtiene de la consulta SQL principal y es utilizado en el reporte para asignar un identificador numérico a cada evaluador dentro del proyecto. Este número permite hacer referencia a los evaluadores de manera anónima en lugar de mostrar su nombre real, asegurando la coherencia con la numeración utilizada en la interfaz del sistema.

FIELD	TIPO	DESCRIPCIÓN
POSICION	BigDecimal	Identificador numérico único asignado a cada evaluador dentro del proyecto.

3. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA[{$F{FUNCIONARIO} != null
? {$F{FUNCIONARIO}
: ($F{NOMBRE}.equals("PUNTAJE PROMEDIO")) ? "<style isBold='true'>" + {$F{NOMBRE}}+"</style>" :
($F{NOMBRE}.contains("PANEL")) ? "PANEL" : "Evaluador" } +
{$V{REPORT_COUNT}})]></textFieldExpression>
```

- **Expresión Nueva:**

```
< <textFieldExpression><![CDATA[{$F{NOMBRE}.equals("PUNTAJE PROMEDIO")
? "<style isBold='true'>" + {$F{NOMBRE}}+"</style>"
: ($F{NOMBRE}.equals("PANEL")) ? "PANEL" : "Evaluador" } +
{$F{POSICION}})]></textFieldExpression>
```

- Este cambio permite asignar correctamente el número del evaluador en base a la tabla de evaluaciones del proyecto, asegurando que la referencia en el reporte coincida con la numeración establecida en la interfaz de usuario. Se elimina la lógica que verificaba si el campo **FUNCIONARIO** era **null** para determinar el nombre del evaluador y, en su lugar, se usa directamente el campo **POSICION**, garantizando que la numeración corresponda a la asignada en la base de datos en lugar de calcularse dinámicamente con **REPORT_COUNT**.
- Se mantiene la verificación para resaltar en negrita el "**PUNTAJE PROMEDIO**", y se optimiza la condición para identificar los paneles de evaluación utilizando **equals("PANEL")** en lugar de **contains("PANEL")** para mayor precisión. Este ajuste mejora la coherencia en la estructura de los reportes generados, asegurando que la numeración de los evaluadores se mantenga consistente en todo el sistema y permitiendo una identificación clara de cada evaluación sin necesidad de exponer nombres reales, reforzando la anonimización de los datos.

Reporte de Calificación y Concepto Argumentado Ponderado

Este reporte corresponde al documento generado que presenta la evaluación cualitativa y cuantitativa de dos evaluadores específicos, desglosada por concepto y subconcepto de la matriz de evaluación. Además, incluye la ponderación de las calificaciones cuantitativas, permitiendo visualizar el impacto de cada criterio en la evaluación final del proyecto.

Brayan Silva Pruebaas Volver

Convocatoria: 40936 - Convocatoria de Formación de ciudadanos en ciencia de datos
Estado: **REVISADO**

Código Registro: 42143
Código Proyecto: 46704093642143
Entidad: PRUEBA PERSONA NATURAL BRAYAN
Lugar de Ejecución:
Tipo: Proyecto
Tipo de Propuesta:
Reporte Calificación: 

Ultima Modificación:
Fecha Creación: 07/02/2018 09:49 AM
Fecha Revisión: 24/02/2025 12:08 PM
Duración: Mese 
Reporte de Calificación y Concepto: 
Argumentado Ponderado:
Reporte Consolidado de Calificación: 

ReporteCalificacionConcepto.jrxml

El reporte **ReporteCalificacionConcepto.jrxml** incorpora el subreporte **conceptoEvaluadores.jrxml**, el cual proporciona el detalle de la evaluación de cada subcriterio de la matriz, mostrando tanto el concepto argumentado como el puntaje asignado por los dos evaluadores específicos. Este subreporte permite visualizar la evaluación de manera estructurada, asegurando que cada criterio y subcriterio refleje la calificación otorgada por cada evaluador.

1. SQL principal del Reporte

La consulta principal del subreporte es utilizada para recuperar los datos de cada evaluación individual, incluyendo la calificación numérica, el concepto argumentado y la información del evaluador asignado. Además, permite asignar un número de posición (**POSICION**) a cada evaluador dentro del proyecto, garantizando un orden secuencial que facilita su referencia en el reporte sin exponer su nombre real.

Esta consulta realiza la relación entre la tabla de evaluaciones (**PRY_EVALUACIONES**), los formatos de evaluación (**SGP_FORMATOS_EVALUACION**), y la información de los evaluadores (**SGP_PERSONAS** y **SGP_EVALUADORES_SCIENTI**). Asimismo, filtra los registros para incluir únicamente las evaluaciones asociadas al proyecto en curso (**\$P{proyecto_id}**) y a los identificadores de evaluación seleccionados.

```
SELECT fe.formato_id
      , e.evaluacion_proyecto_id
      , fe.aspecto_calificacion
      , e.calificacion
      , fe.entable
      , dbms_lob.substr(NVL(e.concepto,'Sin registro'),4000,1) concepto
      , dbms_lob.substr(NVL(e.concepto,''),8000,4001) concepto_parte2
      , ep.persona_id
      , ep.evaluador_scienti_id
      , (CASE
```

```

        WHEN ep.evaluador_scienti_id IS NOT NULL THEN
            (CONCAT(CONCAT(s.nombre_evaluador || ' ',
                s.primer_apellido || ' '),
                s.segundo_apellido))
        ELSE
            (CONCAT(CONCAT(pe.nombres || ' ',
                pe.primer_apellido || ' '),
                pe.segundo_apellido))
    END ) persona
    , ep.funcionario
    , ep.nombre_panel
    , sub.posicion
FROM pry_evaluaciones          ep
    , pry_evaluaciones_calificacion e
    , sgp_formatos_evaluacion     fe
    , sgp_personas                pe
    , sgp_evaluadores_scienti     s
    , (
        SELECT p.evaluacion_proyecto_id,
            ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC) AS posicion
        FROM PRY_EVALUACIONES p
        WHERE p.proyecto_id = $P{proyecto_id}
    ) sub
WHERE fe.formato_evaluacion_id      (+)= e.formato_evaluacion_id
    AND ep.evaluacion_proyecto_id    (+)= e.evaluacion_proyecto_id
    AND s.evaluador_scienti_id       (+)= ep.evaluador_scienti_id
    AND pe.persona_id                (+)= ep.persona_id
    AND e.proyecto_id                = $P{proyecto_id}
    AND e.formato_evaluacion_id      = $P{formato_evaluacion_id}
    AND (ep.evaluacion_proyecto_id    = $P{evaluacion_proyecto_id_1}
    OR ep.evaluacion_proyecto_id      = $P{evaluacion_proyecto_id_2})
    AND sub.evaluacion_proyecto_id    (+)= ep.evaluacion_proyecto_id
    AND sub.evaluacion_proyecto_id IN ($P{evaluacion_proyecto_id_1}, $P{evaluacion_proyecto_id_2})
ORDER BY ep.fecha_creacion ASC

```

A continuación una explicación detallada:

- **La función ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC):** Asigna un número secuencial (**POSICION**) a cada evaluación dentro del proyecto, permitiendo hacer referencia a los evaluadores sin exponer su identidad real. Se implementa en una subconsulta (sub) que es posteriormente unida a la consulta principal.
- **El uso de NVL() y dbms_lob.substr():** Garantiza que el campo CONCEPTO siempre tenga un valor, mostrando "Sin registro" cuando no hay información disponible. Además, permite manejar valores de tipo LOB, asegurando que los datos textuales sean accesibles en el reporte.
- **Filtros aplicados (WHERE):** Se asegura que solo se obtengan registros correspondientes al proyecto en curso mediante **e.proyecto_id = \$P{proyecto_id}**. Se filtran evaluaciones específicas usando **evaluacion_proyecto_id IN (\$P{evaluacion_proyecto_id_1}, \$P{evaluacion_proyecto_id_2})**. Además, se realiza un **JOIN** con la subconsulta sub para obtener la posición (**POSICION**) de cada evaluador.

Esta consulta nos permite obtener el field **POSICION**, el cual es utilizado en el contenido del reporte para referenciar a los evaluadores de manera anónima, reemplazando su nombre real por una numeración en el formato "**Evaluador X**" y asegurando que el orden de los evaluadores sea coherente en el reporte.

2. Field POSICION

El campo POSICION se obtiene de la consulta SQL principal y es utilizado en el reporte para asignar un identificador numérico a cada evaluador dentro del proyecto. Este número permite hacer referencia a los evaluadores de manera anónima en lugar de mostrar su nombre real, asegurando la coherencia con la numeración utilizada en la interfaz del sistema.

FIELD	TIPO	DESCRIPCIÓN
POSICION	BigDecimal	Identificador numérico único asignado a cada evaluador dentro del proyecto.

3. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA["Evaluador " + $V{REPORT_COUNT} + ": "]]></textFieldExpression>
```

- **Expresión Nueva:**

```
<textFieldExpression><![CDATA["Evaluador " + $F{POSICION} + ": "]]></textFieldExpression>
```

- La modificación en la lógica de visualización del **label** de referencia al evaluador reemplaza la numeración basada en el contador de filas del reporte (**\$V{REPORT_COUNT}**) por la posición calculada en la consulta SQL (**\$F{POSICION}**). Anteriormente, la identificación del evaluador dependía del orden en que los registros eran impresos en el reporte, lo que podía generar inconsistencias si los

datos no se ordenaban correctamente. Con esta actualización, la numeración de los evaluadores ahora se basa directamente en el número asignado en la base de datos mediante la función **ROW_NUMBER()**, garantizando una identificación estable e independiente del orden de visualización en JasperReports.

- Esta mejora permite mantener la coherencia entre la numeración en la base de datos y la visualización en el reporte, asegurando que cada evaluador conserve su número único dentro del proyecto sin importar el orden de evaluación. Además, evita posibles errores en la asignación de identificadores y mejora la precisión del reporte al alinear la numeración con la lógica establecida en el sistema, garantizando una presentación clara y estructurada de los datos.

El reporte **ReporteCalificacionConcepto.jrxml** incorpora además el subreporte **conceptoEvaluaciones.jrxml**, el cual proporciona el concepto final de la evaluación de cada evaluador.

1. SQL principal del Reporte

La consulta principal de este subreporte se utiliza para recuperar el concepto final de los evaluadores, asegurando que cada evaluación esté correctamente identificada mediante un número secuencial (**POSICION**). Este número permite referenciar a los evaluadores dentro del reporte sin exponer su identidad real.

La información se obtiene desde la tabla de evaluaciones (**PRY_EVALUACIONES**), relacionándola con la tabla de personas (**SGP_PERSONAS**) para extraer los datos del evaluador. Además, se emplea una subconsulta (**sub**), que asigna un número secuencial a cada evaluación dentro del proyecto mediante **ROW_NUMBER()**, garantizando que el orden de los evaluadores se mantenga en el reporte.

```
SELECT e.evaluacion_proyecto_id
      , e.proyecto_id
      , e.persona_id
      , (p.nombres || ' ' || p.primer_apellido || ' ' || p.segundo_apellido) persona
      , e.funcionario
      , e.nombre_panel
      , e.puntaje
      , dbms_lob.substr(NVL(e.concepto,'Sin registro'),4000,1) concepto
      , dbms_lob.substr(NVL(e.concepto,''),8000,4001) concepto_parte2
      , sub.posicion
FROM pry_evaluaciones e
     , sgp_personas      p
     , (
       SELECT p.evaluacion_proyecto_id,
              ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC) AS posicion
         FROM pry_evaluaciones p
        WHERE p.proyecto_id = ${proyecto_id}
       ) sub
WHERE p.persona_id      (+) = e.persona_id
   AND e.proyecto_id    = ${proyecto_id}
   AND (e.evaluacion_proyecto_id = ${evaluacion_proyecto_id_1}
        OR e.evaluacion_proyecto_id = ${evaluacion_proyecto_id_2})
   AND sub.evaluacion_proyecto_id      IN      (${evaluacion_proyecto_id_1},
${evaluacion_proyecto_id_2})
   AND sub.evaluacion_proyecto_id (+) = e.evaluacion_proyecto_id
ORDER BY e.fecha_creacion ASC
```

A continuación una explicación detallada:

- La función **ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC)**: Asigna un número secuencial (**POSICION**) a cada evaluación dentro del proyecto. Se implementa en una subconsulta (**sub**) que se une a la consulta principal para referenciar a los evaluadores de manera anónima.
- **Concatenación de datos personales (p.nombres || ' ' || p.primer_apellido || ' ' || p.segundo_apellido)**: Construye el nombre completo del evaluador, asegurando que se muestren todos sus apellidos y nombres en caso de que la anonimización no sea aplicada.
- **Uso de NVL() y dbms_lob.substr()**:

- **NVL(e.concepto, 'Sin registro')**: Garantiza que siempre haya un valor en el campo **CONCEPTO**, mostrando "Sin registro" cuando no se encuentra información.
- **dbms_lob.substr(...)**: Extrae el contenido de campos LOB, asegurando que los conceptos largos puedan ser accedidos correctamente en el reporte.
- **Filtros aplicados (WHERE)**:
 - **e.proyecto_id = \$P{proyecto_id}**: Filtra los registros para que solo se incluyan evaluaciones del proyecto en curso.
 - **e.evaluacion_proyecto_id IN (\$P{evaluacion_proyecto_id_1}, \$P{evaluacion_proyecto_id_2})**: Permite que solo se recuperen las evaluaciones específicas de los evaluadores seleccionados.
 - **sub.evaluacion_proyecto_id (+)= e.evaluacion_proyecto_id**: Realiza un **JOIN** con la subconsulta **sub**, asegurando que cada evaluación tenga su número de posición asignado.

Esta consulta nos permite obtener el field **POSICION**, utilizado en el reporte para referenciar a los evaluadores de manera anónima en el formato "**Evaluador X**", asegurando la coherencia en la numeración de los evaluadores en el informe.

2. Field POSICION

El campo **POSICION** se obtiene de la consulta SQL principal y es utilizado en el reporte para asignar un identificador numérico a cada evaluador dentro del proyecto. Este número permite hacer referencia a los evaluadores de manera anónima en lugar de mostrar su nombre real, asegurando la coherencia con la numeración utilizada en la interfaz del sistema.

FIELD	TIPO	DESCRIPCIÓN
POSICION	BigDecimal	Identificador numérico único asignado a cada evaluador dentro del proyecto.

3. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA["Evaluador " + ${REPORT_COUNT} + ": "]]></textFieldExpression>
```

- **Expresión Nueva:**

```
<textFieldExpression><![CDATA["Evaluador " + ${F{POSICION}} + ": "]]></textFieldExpression>
```

- La modificación en la lógica de visualización del **label** de referencia al evaluador reemplaza la numeración basada en el contador de filas del reporte (**\${REPORT_COUNT}**) por la posición calculada en la consulta SQL (**\${F{POSICION}}**). Anteriormente, la identificación del evaluador dependía del orden en que los registros eran impresos en el reporte, lo que podía generar inconsistencias si los datos no se ordenaban correctamente. Con esta actualización, la numeración de los evaluadores ahora se basa directamente en el número asignado en la base de datos mediante la función **ROW_NUMBER()**, garantizando una identificación estable e independiente del orden de visualización en JasperReports.
- Esta mejora permite mantener la coherencia entre la numeración en la base de datos y la visualización en el reporte, asegurando que cada evaluador conserve su número único dentro del proyecto sin importar el orden de evaluación. Además, evita posibles errores en la asignación de identificadores y mejora la precisión del reporte al alinear la numeración con la lógica establecida en el sistema, garantizando una presentación clara y estructurada de los datos.

Módulo de Beneficios Tributarios

Codificación

Componentes

Para la generación de los reportes de evaluación en el módulo de Beneficios Tributarios, se utilizan varios componentes dentro de la aplicación, los cuales trabajan en conjunto para permitir la consulta, procesamiento y descarga del reporte en formato PDF con la nueva estructura de anonimización de evaluadores.

Los principales componentes involucrados en esta modificación son los siguientes:

- **Vista (View):** Representada por el archivo **evaluacion.xhtml**, que contiene la interfaz de usuario en PrimeFaces.
- **Controlador (Controller):** Representado por la clase **ProyectoBeneficiosEvaluadoresConvController.java**, encargada de manejar la lógica de negocio relacionada con la evaluación de proyectos y la generación de reportes. Se ha ajustado la lógica de generación de reportes para incluir la nueva numeración de evaluadores y garantizar la referencia correcta a estos en los documentos generados.

- **Reportes (JasperReports Template):** Se han modificado los siguientes archivos JasperReports para implementar la anonimización y la referencia mediante numeración de evaluadores:
 - ReporteBeneficiosEvaluacionCalificacion
 - ReporteBeneficiosEvaluacion.jrxml
- **Base de Datos (Database):** Contiene la información de las evaluaciones de los proyectos, almacenadas en tablas específicas. Se accede a través de consultas JPA (**Java Persistence API**) y EJB (**Enterprise JavaBeans**) para recuperar los datos necesarios para los reportes. Se han ajustado las consultas para excluir los nombres de los evaluadores y, en su lugar, obtener su identificación basada en la nueva numeración asignada en la vista.

Cada uno de estos componentes interactúa de manera coordinada para garantizar la correcta generación de los reportes en Proyectos y Programas, asegurando la anonimización de los evaluadores y la correcta referencia a través de la numeración asignada en la interfaz.

Vista (evaluacion.hhtml)

El archivo encargado de la vista es **evaluacion.xhtml**. Al ingresar a la pantalla de **Evaluación de Proyectos/Programas** y seleccionar la opción **“Evaluar”** para un proyecto, se despliegan las distintas opciones para generar los reportes de evaluación, representadas mediante botones de descarga, los cuales permiten obtener los reportes en formato PDF.

Además, dentro de la interfaz se encuentra la tabla de evaluaciones, la cual muestra la información correspondiente a cada evaluación realizada dentro del proyecto. Como parte de la actualización, se ha incorporado una nueva columna denominada **“#”**, la cual asigna a cada evaluador un número único dentro del contexto del proyecto. Esta numeración se refleja en términos de **“Evaluador 1”**, **“Evaluador 2”**, **“Evaluador 3”**, ..., lo que permite identificar a los evaluadores de manera estandarizada en la interfaz y mantener la coherencia con la anonimización aplicada en los reportes generados.

Estructura de la columna **“#”** en la tabla de Evaluaciones

```
<h:column headerClass="column1">
  <!-- column header -->
  <f:facet name="header">#</f:facet>
  <!-- row record -->
```

```
#{{projectoBeneficiosEvaluadoresSessController.listaProyectoEvaluaciones.indexOf(evaluacion) +
1}}
</h:column>
```

A continuación, una explicación más detallada:

- **<h:column>**: Define una columna dentro de la tabla de evaluaciones.
- **headerClass="column1"**: Aplica una clase CSS al encabezado de la columna para su estilización.
- **<f:facet name="header">#</f:facet>**: Establece el encabezado de la columna, representado por el símbolo "#".
- **#{{projectoBeneficiosEvaluadoresSessController.listaProyectoEvaluaciones.indexOf(evaluacion) + 1}}**:
 - Obtiene el índice de la evaluación dentro de la lista **listaProyectoEvaluaciones**.
 - Le suma 1 para que la numeración inicie en 1 en lugar de 0.
 - Permite asignar un número único a cada evaluador dentro del contexto del proyecto.

Controlador (ProyectoBeneficiosEvaluadoresConvController.java)

El controlador **proyectoBeneficiosEvaluadoresConvController** es el bean de respaldo (Backing Bean) de la vista de evaluación de proyectos. Está anotado con **@ConversationScoped**, lo que significa que su ciclo de vida está ligado a una conversación de JSF, permitiendo mantener el estado entre múltiples solicitudes dentro de una misma sesión de usuario.

```
@Named(value = "proyectoBeneficiosEvaluadoresConvController")
@ConversationScoped
public class ProyectoBeneficiosEvaluadoresConvController extends
GenericConversationController<ProyectoDTO> implements Serializable {
```

A continuación una explicación más detallada de sus atributos:

- **@Named(value = "proyectoBeneficiosEvaluadoresConvController")**
 - Permite que el bean sea accesible desde las páginas XHTML mediante Expression Language (**#{{proyectoBeneficiosEvaluadoresConvController }}**).

- Se le asigna un nombre explícito (**proyectoBeneficiosEvaluadoresConvController**) para su uso en la vista.
- **@ConversationScoped**
 - Define que el bean tiene un ciclo de vida basado en conversaciones de JSF.
 - Esto significa que el estado del bean se mantiene mientras la conversación esté activa, evitando la pérdida de datos entre peticiones.
- **extends GenericConversationController<ProyectoDTO >**
 - **Herencia:** **evaluacionConvController** extiende de **GenericConversationController**, lo que sugiere que hereda métodos y funcionalidades genéricas para manejar conversaciones en JSF.
 - **Parámetro Genérico:** **ProyectoDTO**, indicando que este controlador maneja objetos de tipo **ProyectoDTO**.
- **implements Serializable**
 - Requisito para que el estado del bean pueda ser serializado y almacenado en la sesión si es necesario.

En conclusión el controlador **proyectoBeneficiosEvaluadoresConvController** es un bean de respaldo (**@ConversationScoped**) que gestiona la lógica relacionada con la evaluación de proyectos dentro de la aplicación. Hereda funcionalidades genéricas para la gestión de entidades y mantiene la conversación activa entre peticiones, asegurando que los datos no se pierdan durante la navegación del usuario.

Dentro de este controlador, el método **downloadReporteEvaluacionCalificacion()** es el encargado de generar el reporte individual de cada evaluación asignada al proyecto. Cada evaluación dentro de la tabla de evaluaciones tiene un botón de descarga, el cual invoca este método para generar y obtener el reporte correspondiente a esa evaluación en particular. A continuación, se presenta el código principal de este método:

```
public void downloadReporteEvaluacionCalificacion(Integer evaluacionProyectoId, Integer numeroEvaluador) {
```



```

        FacesContext context = FacesContext.getCurrentInstance();
        HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();
        String proyectoId =
String.valueOf(proyectoBeneficiosEvaluadoresSessController.getSelected().getProyectoId());
        String nombreReporte = "";
        try {
            ReporteUtil reporteUtil = new ReporteUtil();
            HashMap hashMap = new HashMap();
            hashMap.put("PROYECTO_ID", proyectoId);
            hashMap.put("evaluacion_proyecto_id", String.valueOf(evaluacionProyectoId));
            byte reporte[];

            //carpeta por defecto
            reporteUtil.setNomCarpeta("ReporteBeneficiosEvaluacionCalificacion");

            /**
             * Se verifica si la solicitud tiene la matriz configurada en la
             * convocatoria
             */
            ConvocatoriaMatrizDTO convocatoriaMatrizDTO =
obtenerConvocatoriaMatriz(proyectoBeneficiosEvaluadoresSessController.getSelected());
            if (convocatoriaMatrizDTO != null && convocatoriaMatrizDTO.getMatrizEvaluacionId() !=
null) {
                nombreReporte = "ReporteBeneficiosEvaluacionCalificacion";
            } else {
                throw new Exception("No se ha configurado matriz de evaluación. Por favor comuníquese
con soporte.");
            }

            //Se añade el número del evaluador
            hashMap.put("NUMERO_EVALUADOR", numeroEvaluador);

            //Se setea la imagen del logo parametrizada, guardada en la base de datos
            String imagenLogoBase64 =
beneficiosServiceFacade.obtenerParametroSistema("LOGIN_LOGO").getDetalle();
            String[] parts = imagenLogoBase64.split(",");
            String part1 = parts[0]; // data:image/jpeg;base64
            String imagen = parts[1]; // IMAGEN
            hashMap.put("IMG_BASE64", imagen);
            //
            reporte = reporteUtil.generarReporte(hashMap, nombreReporte, "PDF", context);
            response.setContentType("application/pdf");
            response.setHeader("Content-Disposition", "attachment;
filename=\"ReporteEvaluacionCalificacionProyecto_" + proyectoId + ".pdf\"");
            response.setContentLength(reporte.length);
            ServletOutputStream outputStream = response.getOutputStream();
            outputStream.write(reporte, 0, reporte.length);
            outputStream.flush();
            outputStream.close();
        } catch (Exception ex) {
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
            LOG.error("Error al generar el reporte de evaluacion calificación", ex);

            FacesContext.getCurrentInstance().addMessage(null, message);
        }
    }
}

```

A continuación, se describe en detalle su implementación, explicando cada una de sus partes y su impacto en el proceso de generación y descarga del informe:

1. Firma del método

```
public void downloadReporteEvaluacionCalificacion(Integer evaluacionProyectoId, Integer numeroEvaluador) { ... }
```

- **evaluacionProyectoId:** Identificador único de la evaluación seleccionada del proyecto.
- **numeroEvaluador:** Número asignado al evaluador (Ej: Evaluador 1, Evaluador 2, etc.).

2. Obtención del FacesContext y configuración de la respuesta HTTP

```
FacesContext context = FacesContext.getCurrentInstance();  
HttpServletResponse response = (HttpServletResponse) context.getExternalContext().getResponse();  
String proyectoId = String.valueOf(proyectoBeneficiosEvaluadoresSessController.getSelected().getProyectoId());  
String nombreReporte = "";
```

- **FacesContext.getCurrentInstance():** Obtiene la instancia actual de JSF.
- **context.getExternalContext().getResponse():** Permite acceder a la respuesta HTTP, donde se configurará el archivo PDF para descarga.
- **proyectoBeneficiosEvaluadoresSessController.getSelected().getProyectoId():** Recupera el ID del proyecto actualmente seleccionado en el módulo de Beneficios.
- **nombreReporte:** Variable que almacenará el nombre del reporte a generar según la configuración de la matriz de evaluación.

3. Creación del HashMap con los parámetros del reporte

```
ReporteUtil reporteUtil = new ReporteUtil();  
HashMap hashMap = new HashMap();  
hashMap.put("PROYECTO_ID", proyectoId);  
hashMap.put("evaluacion_proyecto_id", String.valueOf(evaluacionProyectoId));
```

- **ReporteUtil reporteUtil = new ReporteUtil();:** Instancia la utilidad para generar reportes.
- **HashMap hashMap = new HashMap();:** Se usa para almacenar los parámetros que se pasarán al reporte.
- **hashMap.put("PROYECTO_ID", proyectoId);:** Se añade el ID del proyecto al conjunto de parámetros.
- **hashMap.put("evaluacion_proyecto_id", String.valueOf(evaluacionProyectoId));:** Se añade el ID de la evaluación asociada al proyecto.

4. Verificación de la Matriz de Evaluación

Antes de generar el reporte, se verifica si la solicitud del proyecto tiene una matriz de evaluación configurada en la convocatoria.

```
ConvocatoriaMatrizDTO convocatoriaMatrizDTO =  
obtenerConvocatoriaMatriz(proyectoBeneficiosEvaluadoresSessController.getSelected());  
if (convocatoriaMatrizDTO != null && convocatoriaMatrizDTO.getMatrizEvaluacionId() != null) {  
    nombreReporte = "ReporteBeneficiosEvaluacionCalificacion";  
} else {  
    throw new Exception("No se ha configurado matriz de evaluación. Por favor comuníquese con  
soporte.");  
}
```

- **obtenerConvocatoriaMatriz(proyectoBeneficiosEvaluadoresSessController.getSelected())**: Obtiene la configuración de la matriz de evaluación asociada a la convocatoria del proyecto.
- **Condición if:**
 - Si la matriz de evaluación está configurada correctamente, se asigna el nombre del reporte correspondiente.
 - Si no, se lanza una excepción, evitando la generación de un reporte sin una estructura de evaluación definida.

5. Inclusión del Número del Evaluador en el Reporte

```
hashMap.put("NUMERO_EVALUADOR", numeroEvaluador);
```

- Se agrega el número del evaluador a los parámetros del reporte.
- Este valor se usará en los reportes para reemplazar el nombre del evaluador con su número asignado ("Evaluador 1", "Evaluador 2", etc.).

6. Configuración de la imagen del reporte

```
String imagenLogoBase64 =  
beneficiosServiceFacade.obtenerParametroSistema("LOGIN_LOGO").getDetalle();  
String[] parts = imagenLogoBase64.split(",");  
String imagen = parts[1];  
hashMap.put("IMG_BASE64", imagen);
```

- **obtenerParametroSistema("LOGIN_LOGO"):** Obtiene el logo almacenado en la base de datos en base64.
- **split(","):** Separa la imagen en base64, eliminando metadatos.
- Se almacena la imagen en hashMap para usarla en el reporte.

7. Generación del Reporte y Configuración de la Respuesta HTTP

```
reporte = reporteUtil.generarReporte(hashMap, nombreReporte, "PDF", context);
response.setContentType("application/pdf");
response.setHeader("Content-Disposition", "attachment;
filename=\"ReporteEvaluacionCalificacionProyecto_" + proyectoId + ".pdf\"");
response.setContentLength(reporte.length);
```

- **generarReporte():** Se invoca la utilidad de **JasperReports** para generar el reporte en formato PDF.
- **Configuración de la respuesta HTTP:**
 - **Content-Type:** Define el tipo de contenido como PDF.
 - **Content-Disposition:** Configura el nombre del archivo para la descarga.
 - **Content-Length:** Establece el tamaño del archivo para una correcta transferencia.

8. Escritura y Envío del Reporte al Cliente

```
ServletOutputStream outputStream = response.getOutputStream();
outputStream.write(reporte, 0, reporte.length);
outputStream.flush();
outputStream.close();
```

- Se obtiene un flujo de salida (**ServletOutputStream**) para escribir el archivo en la respuesta.
- **write():** Se escribe el contenido del reporte en el flujo de salida.
- **flush():** Asegura que todos los datos sean enviados antes de cerrar el flujo.
- **close():** Cierra el flujo para liberar los recursos.

9. Manejo de Excepciones

```
} catch (Exception ex) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
    resolveExpression(ex.getMessage()));
    LOG.error("Error al generar el reporte de evaluación calificación", ex);
    FacesContext.getCurrentInstance().addMessage(null, message);
}
```

- Captura cualquier error durante la generación del reporte y lo registra en los logs.

- Se muestra un mensaje de error en la interfaz si ocurre una excepción.

Reportes JasperReports

Reporte de Calificación por Evaluación

Estos reportes corresponden al documento generado para cada una de las evaluaciones individuales asignadas a un proyecto.

Show entries

Search:

#	Tipo Evaluación	Tipo Evaluador	Nombres	Usuario Sigu	Opciones	Habilitar Evaluación
1	Económica	Externo	CARLOS ALBERTO ROMERO	EVA_CAROMERO	  	

Showing 1 to 1 of 1 entries

Previous 1 Next

ReporteBeneficiosEvaluaciónCalificacion.jrxml

A continuación, los principales cambios realizados:

1. Nuevo Parámetro NUMERO_EVALUADOR

El parámetro **NUMERO_EVALUADOR** permite identificar a cada evaluador sin exponer su nombre real, garantizando la anonimización en los reportes. Su valor es determinado en el controlador, donde se asigna el número correspondiente a cada evaluador dentro del proyecto.

PARÁMETRO	TIPO	DESCRIPCIÓN
NUMERO_EVALUADOR	Integer	Identificador del evaluador asignado a la evaluación

2. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA[
```

```
!${NOMBRE_PANEL}.equals(null) ? ${NOMBRE_PANEL}
: !${NOMBRE_PERSONA}.equals(null) ? ${NOMBRE_PERSONA}
: !${FUNCIONARIO}.equals(null) ? ${FUNCIONARIO}
: "No hay nombre asignado"
]]></textFieldExpression>
```

- **Expresión Nueva:**

```
<textFieldExpression><![CDATA["Evaluador " + ${NUMERO_EVALUADOR}]]></textFieldExpression>
```

Este cambio permite la anonimización total de los evaluadores en el reporte, eliminando la lógica condicional que verificaba los distintos campos (**NOMBRE_PANEL**, **NOMBRE_PERSONA**, **FUNCIONARIO**) para obtener el nombre del evaluador y asignando directamente el número del evaluador en el formato "**Evaluador X**" a través del parámetro **NUMERO_EVALUADOR**. Además, garantiza la coherencia con la numeración utilizada en la interfaz de usuario y facilita la estandarización en la referencia a los evaluadores en todos los reportes generados.

Reporte de Calificación por Evaluación

Este reporte corresponde al documento generado para cada evaluación del proyecto, detallando la calificación final y el concepto final otorgado por cada evaluador. Además, incluye el promedio final de las evaluaciones, permitiendo visualizar de manera consolidada el desempeño del proyecto según los criterios establecidos en la matriz de evaluación.

Reporte Consolidado
Evaluaciones :



ReporteBeneficiosEvaluacion.jrxml

El reporte **ReporteBeneficiosEvaluacion.jrxml** contiene el subreporte "**evaluadoresConcepto.jrxml**", el cual detalla la calificación final y el concepto final asignado por cada evaluador en la evaluación del proyecto. Para estructurar correctamente esta información, el reporte utiliza una consulta SQL dinámica que permite identificar y ordenar a los evaluadores según su tipo de evaluación y su posición dentro del proyecto.

1. SQL principal del Reporte

La consulta principal del reporte es utilizada para obtener los diferentes datos mostrados en el informe, incluyendo la lista de evaluadores asignados al proyecto, su calificación y su concepto final. Además, asigna un número secuencial (**POSICION**) a cada evaluación, lo que permite referenciar a los evaluadores en el reporte sin exponer su nombre real. También organiza la información según el tipo de evaluación, asegurando una estructura clara y ordenada. Adicionalmente, la consulta realiza cálculos como el puntaje promedio de las evaluaciones y agrega registros especiales cuando no hay evaluaciones de ciertos tipos, garantizando que el reporte siempre contenga información relevante.

```
SELECT UPPER(PRYEVALUACIONES.FUNCIONARIO) FUNCIONARIO,
(CASE
  WHEN PRYEVALUACIONES.TIPO_EVALUACION = 'P' THEN
    PRYEVALUACIONES.Nombre_Panel
  WHEN PRYEVALUACIONES.Evaluador_Scienti_Id IS NOT NULL THEN
    (CONCAT(CONCAT(SGPEVALUADORESSCIENTI.NOMBRE_EVALUADOR || ' ',
      SGPEVALUADORESSCIENTI.PRIMER_APELLIDO || ' '),
      SGPEVALUADORESSCIENTI.SEGUNDO_APELLIDO))
  ELSE
    (CONCAT(CONCAT(SGPPERSONAS.NOMBRES || ' ',
      SGPPERSONAS.PRIMER_APELLIDO || ' '),
      SGPPERSONAS.SEGUNDO_APELLIDO))
END ) NOMBRE,
NVL(dbms_lob.substr(PRYEVALUACIONES.CONCEPTO,4000,1), '') concepto,
NVL(dbms_lob.substr(PRYEVALUACIONES.CONCEPTO,8000,4001), '') concepto_parte2,
PRYEVALUACIONES.PUNTAJE,
PRYEVALUACIONES.USUARIO_CREADOR,
PRYEVALUACIONES.FECHA_CREACION,
PRYEVALUACIONES.USUARIO_MODIFICADOR,
PRYEVALUACIONES.FECHA_MODIFICACION,
UPPER(SGPPERSONAS.NOMBRES || ' ' || SGPPERSONAS.PRIMER_APELLIDO || ' ' ||
SGPPERSONAS.SEGUNDO_APELLIDO) AS NOMBRE_PERSONA,
(CASE
  WHEN PRYEVALUACIONES.TIPO_EVALUACION = 'P' THEN
    2
  WHEN PRYEVALUACIONES.TIPO_EVALUACION = 'C' THEN
    1
  ELSE
    0
END) ORDEN,
SUB.POSICION
FROM PRY_EVALUACIONES      PRYEVALUACIONES,
SGP_PERSONAS              SGPPERSONAS,
SGP_PROYECTOS              SGPPROYECTOS,
SGP_EVALUADORES_SCIENTI   SGPEVALUADORESSCIENTI,
(
  SELECT p.evaluacion_proyecto_id,
    ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC) AS POSICION
  FROM PRY_EVALUACIONES p
  WHERE p.proyecto_id = ${PROYECTO_ID}
) SUB
WHERE PRYEVALUACIONES.PERSONA_ID = SGPPERSONAS.PERSONA_ID(+)
AND (PRYEVALUACIONES.EVALUACION_PROYECTO_ID IN
(SELECT EVALUACION_PROYECTO_ID FROM VW_FILTRO_EVALUACION))
AND SGPPROYECTOS.PROYECTO_ID = PRYEVALUACIONES.PROYECTO_ID
AND SGPEVALUADORESSCIENTI.EVALUADOR_SCIENTI_ID (+)= PRYEVALUACIONES.EVALUADOR_SCIENTI_ID
AND SGPPROYECTOS.PROYECTO_ID = ${PROYECTO_ID}
```

```

AND SUB.EVALUACION_PROYECTO_ID (+)= PRYEVALUACIONES.EVALUACION_PROYECTO_ID
AND SUB.EVALUACION_PROYECTO_ID IN (SELECT EP.EVALUACION_PROYECTO_ID
                                     FROM PRY_EVALUACIONES EP
                                     WHERE EP.PROYECTO_ID = $P{PROYECTO_ID})

UNION ALL
SELECT '',
       CONCAT('PUNTAJE PROMEDIO', ''),
       '',
       '',
       (SELECT AVG(NVL(PUNTAJE,0))
        FROM PRY_EVALUACIONES E, SGP_PROYECTOS R
        WHERE E.PROYECTO_ID = R.PROYECTO_ID
        AND R.PROYECTO_ID = $P{PROYECTO_ID}
        AND (E.TIPO_EVALUACION <> 'P' OR E.TIPO_EVALUACION <> 'C')) PUNTAJE,
       '',
       TO_DATE('') FECHA_CREACION,
       '',
       TO_DATE(''),
       '',
       TO_NUMBER('1') ORDEN,
       TO_NUMBER('') POSICION
FROM DUAL
ORDER BY ORDEN ASC, POSICION ASC

```

A continuación, una explicación detallada:

- **La función ROW_NUMBER() OVER (ORDER BY p.evaluacion_proyecto_id ASC):** Asigna un número secuencial a cada evaluación dentro del proyecto, permitiendo identificar de manera única a cada evaluador y garantizando un orden lógico en el reporte.
- **El operador CASE:** Determina el nombre del evaluador aplicando una jerarquía entre los campos **NOMBRE_PANEL**, **NOMBRE_PERSONA** y **FUNCIONARIO**, asegurando que se muestre el nombre correcto según la información disponible.
- **El uso de UNION ALL:**
 - Permite incluir cálculos adicionales, como el promedio de las calificaciones (**AVG(PUNTAJE)**), asegurando que el reporte contenga una visión consolidada de las evaluaciones.
 - Agrega registros especiales para casos en los que no existen evaluaciones de ciertos tipos, evitando que el reporte quede vacío o incompleto.
- **La condición WHERE PRYEVALUACIONES.PROYECTO_ID = \$P{PROYECTO_ID}:** Filtra los datos, asegurando que solo se obtengan las evaluaciones correspondientes al proyecto en curso y excluyendo registros no relacionados con la evaluación actual.

Esta consulta nos permite obtener el **field POSICION**, el cual es utilizado en el contenido del reporte para referenciar a los evaluadores de manera anónima, reemplazando su nombre real por una numeración en el formato "**Evaluador X**". Esto garantiza la confidencialidad de los evaluadores en el reporte y mantiene una estructura clara y ordenada en la visualización de los resultados.

2. Field POSICION

El campo POSICION se obtiene de la consulta SQL principal y es utilizado en el reporte para asignar un identificador numérico a cada evaluador dentro del proyecto. Este número permite hacer referencia a los evaluadores de manera anónima en lugar de mostrar su nombre real, asegurando la coherencia con la numeración utilizada en la interfaz del sistema.

FIELD	TIPO	DESCRIPCIÓN
POSICION	BigDecimal	Identificador numérico único asignado a cada evaluador dentro del proyecto.

3. Cambio en la Identificación del Evaluador

Modificación en la lógica de visualización del label de referencia al evaluador:

- **Expresión Anterior:**

```
<textFieldExpression><![CDATA[ ${FUNCIONARIO} != null
? ${FUNCIONARIO}
: (${NOMBRE}.equals("PUNTAJE PROMEDIO")) ? "<style isBold='true'>" + ${NOMBRE}+"</style>"
:${NOMBRE}]]></textFieldExpression>
```

- **Expresión Nueva:**

```
<textFieldExpression><![CDATA[ ${NOMBRE}.equals("PUNTAJE PROMEDIO") ? "<style
isBold='true'>" + ${NOMBRE}+"</style>" : "Evaluador " +
(${POSICION}]]></textFieldExpression>
```

- La modificación en la lógica de visualización del **label** de referencia al evaluador permite asignar correctamente el número del evaluador en base a la tabla de evaluaciones del proyecto, asegurando que la referencia en el reporte coincida con la numeración establecida en la interfaz de usuario. Se elimina la lógica anterior que verificaba si el campo **FUNCIONARIO** era **null** para determinar el nombre del evaluador y, en su lugar, se utiliza directamente el campo **POSICION**, garantizando

que la numeración mostrada en el reporte sea consistente con la asignada en la base de datos. Esta modificación también reemplaza la cálculo dinámico basado en **REPORT_COUNT**, lo que evita posibles discrepancias en la numeración de los evaluadores y mejora la precisión en la identificación de cada evaluación dentro del proyecto.

- Además, se mantiene la verificación para resaltar en negrita el **"PUNTAJE PROMEDIO"**, asegurando una mejor visualización en el reporte. Se optimiza la condición utilizada para identificar los paneles de evaluación, reemplazando el uso de **contains("PANEL")** por **equals("PANEL")**, lo que aumenta la precisión al diferenciar estos elementos. Estos cambios refuerzan la anonimización de los evaluadores al eliminar cualquier referencia a nombres reales y estandarizar la numeración en todo el sistema, permitiendo una identificación clara y consistente en los reportes generados.

b. Desarrollo Nuevo Informe de Evaluación Tercer Evaluador

Solicitud de requerimiento No. 154299

Descripción y alcance del Requerimiento

Desarrollo nuevo Informe Evaluación. El informe debe ser de carácter cualitativo, presentando de manera estructurada los comentarios y observaciones de cada evaluador organizados según los criterios y subcriterios establecidos en los términos de referencia de la convocatoria. Finalmente, el informe debe contener una anotación que remita expresamente a los términos de referencia de la convocatoria respectiva, para que los proponentes puedan revisar las condiciones aplicables en los casos en que haya un segundo evaluador. Con este nuevo informe se busca garantizar mayor transparencia y claridad en la retroalimentación a los proponentes, asegurando que puedan comprender la evaluación de su propuesta de manera integral.

Desarrollo e Implementación

Este desarrollo tiene como objetivo la implementación del Reporte Consolidado de Calificación en la aplicación SIGP WEB. Este reporte tiene como objetivo consolidar y presentar de manera detallada las evaluaciones realizadas a un proyecto durante su etapa de evaluación, garantizando transparencia y acceso a la información sin comprometer la confidencialidad de los evaluadores.

El reporte reúne únicamente las tres primeras evaluaciones registradas para cada proyecto, mostrando tanto calificaciones cuantitativas como cualitativas sin realizar promedios ni

ponderaciones. Además, para mantener la confidencialidad, la información de los evaluadores no es visible, reflejando únicamente los datos de la evaluación realizada.

Los cambios implementados incluyen:

1. Generación del Reporte Consolidado de Calificación:

- Consolidación de las primeras tres evaluaciones registradas de cada proyecto.
- Presentación de calificaciones cuantitativas (valores numéricos) y cualitativas (argumentación del evaluador).
- Exclusión de los datos personales de los evaluadores, manteniendo únicamente la información de evaluación.

2. Condiciones para la Generación del Reporte:

- El reporte solo está disponible si el proyecto cuenta con al menos tres evaluaciones registradas.
- En caso de que alguna evaluación no esté completa, se muestra la leyenda "Sin registro" en los campos correspondientes.

3. Integración con la Interfaz de Usuario:

- Implementación de un botón en la pantalla de evaluación de proyectos para la generación del reporte.
- Uso de PrimeFaces para la visualización y descarga en formato PDF.

4. Implementación Técnica:

- Desarrollo de la funcionalidad utilizando Java EE, JSF y PrimeFaces.
- Uso de JasperReports para la generación del reporte en PDF.
- Consultas optimizadas con JPA (Criteria API) para recuperar las evaluaciones del sistema.

5. Tecnologías Utilizadas

- El desarrollo de esta funcionalidad se realizó con:
- Java EE y EJB para la lógica de negocio.
- JSF (JavaServer Faces) y PrimeFaces para la interfaz de usuario.
- JasperReports para la generación del reporte en PDF.
- JPA con Criteria API para la manipulación de datos desde la base de datos.

Codificación

Componentes

Para la generación del Reporte Consolidado de Calificación, se utilizan varios componentes dentro de la aplicación, los cuales trabajan en conjunto para permitir la consulta, generación y descarga del reporte en formato PDF.

Los principales componentes involucrados son los siguientes:

- **Vista (View):** Representada por el archivo **evaluacion.xhtml**, que contiene la interfaz de usuario en PrimeFaces y define el botón de generación del reporte.
- **Controlador (Controller):** Representado por la clase **EvaluacionConvController.java**, encargada de manejar la lógica de negocio relacionada con la evaluación de proyectos y la generación del reporte.
- **Reporte (JasperReports Template):** Representado por el archivo **ReporteConsolidadoCalificacion.jrxml**, el cual define la estructura y el diseño del reporte en JasperReports. Este archivo incluye tanto la plantilla principal como sus respectivos subreportes, que permiten desglosar y organizar la información detallada de las evaluaciones. En tiempo de ejecución, el reporte es compilado y procesado dinámicamente para generar el archivo PDF final con los datos consolidados de las tres primeras evaluaciones del proyecto.
- **Base de Datos (Database):** Contiene la información de las evaluaciones de los proyectos, almacenadas en tablas específicas. Se accede a través de consultas JPA (Java Persistence API) y EJB (Enterprise JavaBeans) para obtener los datos necesarios para el reporte.

Cada uno de estos componentes interactúa de manera coordinada para garantizar la correcta generación y descarga del reporte consolidado.

Vista (evaluacion.xhtml)

El archivo encargado de la vista es **evaluacion.xhtml**. Al ingresar a la pantalla de Evaluación del módulo de Proyectos/Programas y seleccionar la opción “Evaluar” para un proyecto, se encuentra disponible la opción para generar el Reporte Consolidado de Calificación. Esta funcionalidad está representada mediante un botón, el cual permite descargar el reporte en formato PDF.

Reporte Consolidado de Calificación:

Estructura del botón en XHTML

El siguiente fragmento de código define el botón en PrimeFaces, asegurando que solo se renderice cuando el proyecto tenga al menos tres evaluaciones registradas:

```
<ui:fragment rendered="#{proyectoEvaluacionSessController.listaProyectoEvaluaciones.size() > 2}">
    <!-- Reporte Consolidado de Calificación. -->

    <dt>#{msg['gestion.proyectos.evaluacion.evaluadores.generarReporte.consolidadoCalificacion']}</dt>
    <dt>
        <dd>
            <p:commandLink
                title="#{msg['gestion.proyectos.evaluacion.evaluadores.generarReporte.consolidadoCalificacion']}"
                styleClass=" btn btn-xs btn-primary sbTooltip"

                actionListener="#{evaluacionConvController.downloadReporteConsolidadCalificacion()}"
                immediate="true"
                ajax="false"
                style="margin-top: 10px"
            >
                <i class="fa fa-file-pdf-o"></i>
            </p:commandLink>
        </dd>
    </dt>
</ui:fragment>
```

A continuación, una explicación más detallada:

1. Renderización condicional del botón

```
<ui:fragment rendered="#{proyectoEvaluacionSessController.listaProyectoEvaluaciones.size() > 2}">
```

- El componente **<ui:fragment>** es un contenedor de contenido que no genera una etiqueta HTML en la salida final. Se utiliza para agrupar elementos y controlar su visibilidad de manera dinámica sin afectar la estructura del DOM. La propiedad **rendered** permite condicionar la renderización del fragmento según una expresión EL (Expression Language).
- El controlador **proyectoEvaluacionSessController** es un managed bean de tipo **ViewScoped** que maneja la sesión de evaluación del proyecto.
- El elemento **listaProyectoEvaluaciones** es un atributo de tipo **List<ProyectoEvaluacionDTO>**, que contiene todas las evaluaciones asociadas al proyecto en evaluación.
- La lista se obtiene de la base de datos a través de un DTO (Data Transfer Object) que consulta la entidad **PRY_EVALUACIONES**.
- La condición **size() > 2** verifica que existan al menos tres evaluaciones para permitir la generación del reporte.

2. Botón de Descarga

```
<p:commandLink
title="#{msg['gestion.proyectos.evaluacion.evaluadores.generarReporte.consolidadoCalificacio
n']}"
styleClass=" btn btn-xs btn-primary sbTooltip"

actionListener="#{evaluacionConvController.downloadReporteConsolidadCalificacion()}"
immediate="true"
ajax="false"
style="margin-top: 10px"
>
<i class="fa fa-file-pdf-o"></i>
</p:commandLink>
```

- El botón de tipo **p:commandLink** es un componente de PrimeFaces que se usa para ejecutar una acción en el servidor al hacer clic.
- En este caso, se utiliza para invocar el método **downloadReporteConsolidadCalificacion()** en el controlador **evaluacionConvController**, que genera el reporte consolidado en formato PDF.

Atributos Botón:

- **title:** Establece un tooltip con el texto del archivo de propiedades (msg).
- **styleClass:** Aplica clases CSS para definir el tamaño (btn-xs), el color (btn-primary) y el estilo de tooltip (sbTooltip).

- **actionListener:** Define la acción que ejecutará el controlador (**downloadReporteConsolidadCalificacion()**).
- **Immediate="true":** Evita validaciones de JSF y ejecuta la acción de inmediato.
- **ajax="false":** Desactiva AJAX para que la respuesta sea una descarga de archivo.
- **style:** Aplica un margen superior de 10px para separar el botón visualmente.

Controlador (EvaluacionConvController.java)

El controlador **evaluacionConvController** es el bean de respaldo (Backing Bean) de la vista de evaluación de proyectos. Está anotado con **@ConversationScoped**, lo que significa que su ciclo de vida está ligado a una conversación de JSF, permitiendo mantener el estado entre múltiples solicitudes dentro de una misma sesión de usuario.

```
@Named(value = "evaluacionConvController")
@ConversationScoped
public class EvaluacionConvController extends
GenericConversationController<ProyectoEvaluacionDTO> implements Serializable {
```

A continuación, una explicación más detallada de sus atributos:

- **@Named(value = "evaluacionConvController")**
 - Permite que el bean sea accesible desde las páginas XHTML mediante Expression Language (**#{evaluacionConvController}**).
 - Se le asigna un nombre explícito (**evaluacionConvController**) para su uso en la vista.
- **@ConversationScoped**
 - Define que el bean tiene un ciclo de vida basado en conversaciones de JSF.
 - Esto significa que el estado del bean se mantiene mientras la conversación esté activa, evitando la pérdida de datos entre peticiones.
- **extends GenericConversationController<ProyectoEvaluacionDTO>**
 - **Herencia:** **evaluacionConvController** extiende de **GenericConversationController**, lo que sugiere que hereda métodos y funcionalidades genéricas para manejar conversaciones en JSF.

- **Parámetro Genérico: ProyectoEvaluacionDTO**, indicando que este controlador maneja objetos de tipo **ProyectoEvaluacionDTO**.
- **implements Serializable**
 - Requisito para que el estado del bean pueda ser serializado y almacenado en la sesión si es necesario.

En conclusión, el controlador **EvaluacionConvController** es un bean de respaldo (@ConversationScoped) que gestiona la lógica relacionada con la evaluación de proyectos dentro de la aplicación. Hereda funcionalidades genéricas para la gestión de entidades y mantiene la conversación activa entre peticiones, asegurando que los datos no se pierdan durante la navegación del usuario.

Dentro de este controlador, el método **downloadReporteConsolidadCalificacion()** juega un papel clave en la generación de reportes consolidados de calificación, a continuación el código principal:

```
public void downloadReporteConsolidadCalificacion() {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));

    try {
        FacesContext context = FacesContext.getCurrentInstance();
        HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();
        Long proyectoId = proyectoEvaluacionSessController.getSelected().getProyectoId();

        Map<String, Object> filters = new HashMap<>();
        filters.put("proyectoId", proyectoId);
        int countEvaluaciones =
proyectosServiceFacade.proyectoEvaluacionesListCount(filters);
        if (countEvaluaciones > 0) {

            ReporteUtil reporteUtil = new ReporteUtil();
            HashMap hashMap = new HashMap();
            hashMap.put("PROYECTO_ID", proyectoId);

            //Se setea la imagen del logo parametrizada, guardada en la base de datos
            String imagenLogoBase64 =
proyectosServiceFacade.obtenerParametroSistema("LOGIN_LOGO").getDetalle();
            String[] parts = imagenLogoBase64.split(",");
            //String part1 = parts[0]; // data:image/jpeg;base64
            String imagen = parts[1]; // IMAGEN
            hashMap.put("IMG_BASE64", imagen);
```



```
//Se setea la imagen secundaria del logo parametrizada, guardada en la base de
datos
String imagenLogoSecundarioBase64 =
projectosServiceFacade.obtenerParametroSistema("LOGIN_LOGO2").getDetalle();
String imagenSecundaria = null;
if (imagenLogoSecundarioBase64 != null) {
    String[] parts2 = imagenLogoSecundarioBase64.split(",");
    imagenSecundaria = parts2[1]; // IMAGEN
}
hashMap.put("IMG_SECUNDARIA_BASE64", imagenSecundaria);

//Obtener Matriz ID
ValorDefinicionDTO matrizEvaluacionId =
obtenerMatrizEvaluacionId(projectoEvaluacionSessController.getSelected());
String tipoMatriz = "";
if (matrizEvaluacionId.getComentario() != null &&
matrizEvaluacionId.getComentario().equals("REG")) {
    tipoMatriz = matrizEvaluacionId.getComentario(); //Matriz de Regalias
} else {
    tipoMatriz = "P"; // Matriz de Proyectos
}
hashMap.put("matrizEvaluacionId", matrizEvaluacionId.getValorDefinicionId());
hashMap.put("tipo_matriz", tipoMatriz);

List<ProyectoEvaluacionDTO> evaluacionesPryList =
projectosServiceFacade.proyectoEvaluacionesList(-1, -1, "fechaCreacion", "A", filters);

//Ordenar por id ASC
evaluacionesPryList = evaluacionesPryList.stream()
.sorted(Comparator.comparingLong(ProyectoEvaluacionDTO::getEvaluacionProyectoId))
.collect(Collectors.toList());

Long evaluacionId_1 = evaluacionesPryList.get(0).getEvaluacionProyectoId();
Long evaluacionId_2 = evaluacionesPryList.get(1).getEvaluacionProyectoId();
Long evaluacionId_3 = evaluacionesPryList.get(2).getEvaluacionProyectoId();

hashMap.put("evaluacion_proyecto_id_1", evaluacionId_1);
hashMap.put("evaluacion_proyecto_id_2", evaluacionId_2);
hashMap.put("evaluacion_proyecto_id_3", evaluacionId_3);

byte reporte[];
reporteUtil.setNomCarpeta("ReporteConsolidadoCalificacion");
reporte = reporteUtil.generarReporte(hashMap, "ReporteConsolidadoCalificacion",
"PDF", context);
response.setContentType("application/pdf");
response.setHeader("Content-Disposition", "attachment;
filename=\"ReporteConsolidadoCalificacion\" + proyectoId + ".pdf\"");
response.setContentLength(reporte.length);
ServletOutputStream ouputStream = response.getOutputStream();
ouputStream.write(reporte, 0, reporte.length);
ouputStream.flush();
ouputStream.close();
} else {
    throw new
SystemException("#{msg['gestion.proyectos.evaluacion.evaluadores.generarReporte.calificacion
Concepto.error.empty']}");
}
} catch (Exception ex) {
    LOG.error("Error al generar el Reporte Consolidad de Calificacion", ex);
}
```

```
message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
} finally {
    FacesContext.getCurrentInstance().addMessage(null, message);
}
}
```

Este método es el encargado de generar y descargar el **Reporte Consolidado de Calificación** en formato PDF. Se encarga de obtener los datos necesarios, establecer parámetros y generar el archivo para que el usuario lo descargue. A continuación, se describe en detalle su implementación, explicando cada una de sus partes y su impacto en el proceso de generación y descarga del informe:

1. Firma del método

```
public void downloadReporteConsolidadCalificacion() { ... }
```

- **Modificador de acceso: public** → Puede ser invocado desde otras clases y la vista.
- **Tipo de retorno: void** → No retorna ningún valor, su propósito es generar y enviar el archivo.
- **Parámetros:** Ninguno.

2. Creación del mensaje de éxito inicial

```
FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
```

- **FacesMessage:** Se usa para manejar mensajes en la vista JSF.
- **SEVERITY_INFO:** Indica que el mensaje inicial es de tipo informativo.
- **resolveExpression():** Obtiene valores desde el archivo de propiedades, permitiendo mensajes dinámicos en la interfaz.

3. Obtención del FacesContext y HttpServletResponse

```
FacesContext context = FacesContext.getCurrentInstance();
HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();
```

- **FacesContext.getCurrentInstance():** Obtiene la instancia actual de JSF.
- **context.getExternalContext().getResponse():** Permite acceder al objeto **HttpServletResponse**, que se usa para enviar el archivo PDF como respuesta HTTP.

4. Obtener el ID del proyecto

```
Long proyectoId = proyectoEvaluacionSessController.getSelected().getProyectoId();
```

- **proyectoEvaluacionSessController.getSelected():** Obtiene una instancia de **ProyectoDTO**, representando el proyecto actualmente seleccionado en la sesión. Esta entidad contiene la información del proyecto en evaluación, incluyendo su identificador (**proyectoId**), el cual se utiliza para filtrar las evaluaciones asociadas y generar el reporte consolidado.
- **getProyectoId():** Recupera el identificador único del proyecto.

5. Verificar si hay evaluaciones disponibles

```
Map<String, Object> filters = new HashMap<>();  
filters.put("proyectoId", proyectoId);  
int countEvaluaciones = proyectosServiceFacade.proyectoEvaluacionesListCount(filters);  
if (countEvaluaciones > 0) {
```

- **filters.put("proyectoId", proyectoId);** Crea un filtro para buscar evaluaciones asociadas al **proyectoId**.
- **proyectosServiceFacade.proyectoEvaluacionesListCount(filters);** Obtiene la cantidad total de evaluaciones de tipo **ProyectoEvaluacionDTO** filtradas por el id del proyecto.
- **Condición if (countEvaluaciones > 0):** Verifica si hay al menos una evaluación antes de proceder con la generación del reporte.

6. Creación del mapa de parámetros (hashMap)

```
ReporteUtil reporteUtil = new ReporteUtil();  
HashMap hashMap = new HashMap();  
hashMap.put("PROYECTO_ID", proyectoId);
```

- **ReporteUtil reporteUtil = new ReporteUtil();**: Instancia la utilidad para generar reportes.
- **HashMap hashMap = new HashMap();**: Se usa para almacenar los parámetros que se pasarán al reporte.

7. Obtención y configuración de imágenes desde la base de datos

7.1. Imagen Principal

```
String imagenLogoBase64 =  
proyectosServiceFacade.obtenerParametroSistema("LOGIN_LOGO").getDetalle();  
String[] parts = imagenLogoBase64.split(",");  
String imagen = parts[1];  
hashMap.put("IMG_BASE64", imagen);
```

- **obtenerParametroSistema("LOGIN_LOGO")**: Obtiene el logo almacenado en la base de datos en base64.
- **split(",")**: Separa la imagen en base64, eliminando metadatos.
- Se almacena la imagen en hashMap para usarla en el reporte.

7.2. Imagen Secundaria

```
String imagenLogoSecundarioBase64 =  
proyectosServiceFacade.obtenerParametroSistema("LOGIN_LOGO2").getDetalle();  
String imagenSecundaria = null;  
if (imagenLogoSecundarioBase64 != null) {  
    String[] parts2 = imagenLogoSecundarioBase64.split(",");  
    imagenSecundaria = parts2[1];  
}  
hashMap.put("IMG_SECUNDARIA_BASE64", imagenSecundaria);
```

- **obtenerParametroSistema("LOGIN_LOGO2")**: Obtiene el logo almacenado en la base de datos.
- **split(",")**: Separa la imagen en base64, eliminando metadatos.
- Se almacena la imagen en hashMap para usarla en el reporte.
- Verifica que **LOGIN_LOGO2** exista antes de procesarlo.

8. Obtener la matriz de evaluación

```
ValorDefinicionDTO matrizEvaluacionId =
obtenerMatrizEvaluacionId(proyectoEvaluacionSessController.getSelected());
String tipoMatriz = "";
if (matrizEvaluacionId.getComentario() != null &&
matrizEvaluacionId.getComentario().equals("REG")) {
    tipoMatriz = matrizEvaluacionId.getComentario();
} else {
    tipoMatriz = "P";
}
hashMap.put("matrizEvaluacionId", matrizEvaluacionId.getValorDefinicionId());
hashMap.put("tipo_matriz", tipoMatriz);
```

- **obtenerMatrizEvaluacionId():** Método que obtiene el identificador de la matriz de evaluación, retornando un objeto de tipo **ValorDefinicionDTO**.
- **tipoMatriz:** Determina si la matriz es de Regalías ("REG") o de Proyectos ("P") según el valor del comentario en **matrizEvaluacionId**.
- Se almacenan estos valores en hashMap para ser utilizados en la generación del reporte.

9. Obtener y ordenar evaluaciones del proyecto

```
List<ProyectoEvaluacionDTO> evaluacionesPryList =
proyectosServiceFacade.proyectoEvaluacionesList(-1, -1, "fechaCreacion", "A", filters);
evaluacionesPryList = evaluacionesPryList.stream()
    .sorted(Comparator.comparingLong(ProyectoEvaluacionDTO::getEvaluacionProyectoId))
    .collect(Collectors.toList());
```

Objetivo: Obtener y ordenar las evaluaciones del proyecto por su identificador de manera ascendente, asegurando que se procesen en el orden en que fueron creadas.

- **proyectosServiceFacade.proyectoEvaluacionesList(-1, -1, "fechaCreacion", "A", filters);:**
 - Obtiene una lista de evaluaciones de tipo **ProyectoEvaluacionDTO** asociadas al proyecto, aplicando los filtros definidos.
 - Los parámetros **-1, -1** indican que no hay paginación.
 - **"fechaCreacion"** define el criterio de ordenación por defecto.
 - **"A"** indica el orden ascendente.
- **evaluacionesPryList.stream().sorted(Comparator.comparingLong(ProyectoEvaluacionDTO::getEvaluacionProyectoId)).collect(Collectors.toList());:**

- Convierte la lista en un flujo (Stream).
- Aplica **sorted(Comparator.comparingLong(ProyectoEvaluacionDTO::getEvaluacionProyectold))** para ordenar la lista por el identificador único de la evaluación (**evaluacionProyectold**) en orden ascendente.
- **collect(Collectors.toList())** recopila los elementos ordenados en una nueva lista.

10. Extraer IDs de las tres primeras evaluaciones

```
Long evaluacionId_1 = evaluacionesPryList.get(0).getEvaluacionProyectoId();
Long evaluacionId_2 = evaluacionesPryList.get(1).getEvaluacionProyectoId();
Long evaluacionId_3 = evaluacionesPryList.get(2).getEvaluacionProyectoId();
hashMap.put("evaluacion_proyecto_id_1", evaluacionId_1);
hashMap.put("evaluacion_proyecto_id_2", evaluacionId_2);
hashMap.put("evaluacion_proyecto_id_3", evaluacionId_3);
```

- Extrae los IDs de las tres primeras evaluaciones y los guarda en **hashMap**.

11. Generación y descarga del reporte

```
byte reporte[];
reporteUtil.setNomCarpeta("ReporteConsolidadoCalificacion");
reporte = reporteUtil.generarReporte(hashMap, "ReporteConsolidadoCalificacion", "PDF",
context);
response.setContentType("application/pdf");
response.setHeader("Content-Disposition", "attachment;
filename=\"ReporteConsolidadoCalificacion\" + proyectoId + ".pdf\"");
response.setContentLength(reporte.length);
ServletOutputStream ouputStream = response.getOutputStream();
ouputStream.write(reporte, 0, reporte.length);
ouputStream.flush();
ouputStream.close();
```

11.1. Generación del reporte

- **reporteUtil.setNomCarpeta("ReporteConsolidadoCalificacion");**: Define el nombre de la carpeta donde se encuentra el reporte y sub-reportes en el sistema de archivos.
- **reporte = reporteUtil.generarReporte(hashMap, "ReporteConsolidadoCalificacion", "PDF", context);**
 - Llama al método generarReporte, que utiliza hashMap para pasar los parámetros necesarios al reporte JasperReports.
 - El reporte se genera en formato PDF y se almacena en el array de bytes reporte[].

11.2. Configuración de la respuesta HTTP para la descarga

- **response.setContentType("application/pdf");** Especifica que la respuesta HTTP contendrá un archivo en formato PDF.
- **response.setHeader("Content-Disposition", "attachment; filename=\"ReporteConsolidadoCalificacion\" + proyectId + ".pdf\");** Define el encabezado HTTP para forzar la descarga del archivo con un nombre personalizado que incluye el proyectId.
- **response.setContentLength(reporte.length);** Establece la longitud del contenido en bytes para que el navegador pueda manejar la descarga correctamente.

11.3. Escritura del archivo en la salida de la respuesta

- **ServletOutputStream outputStream = response.getOutputStream();** Obtiene el flujo de salida para escribir la respuesta en el navegador.
- **outputStream.write(reporte, 0, reporte.length);** Escribe el contenido del archivo PDF en el flujo de salida.
- **outputStream.flush();** Asegura que todos los datos se envíen al cliente antes de cerrar el flujo.
- **outputStream.close();** Cierra el flujo de salida para liberar los recursos.

12. Manejo de excepciones

```
} catch (Exception ex) {  
    LOG.error("Error al generar el Reporte Consolidado de Calificacion", ex);  
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",  
    resolveExpression(ex.getMessage()));  
} finally {  
    FacesContext.getCurrentInstance().addMessage(null, message);  
}
```

- Captura y registra cualquier error durante la generación del reporte.
- Envía un mensaje de error a la interfaz si algo falla.

Reporte (ReporteConsolidadoCalificacion.jrxml)

El reporte principal **ReporteConsolidadoCalificacion.jrxml** define la estructura general del informe consolidado de calificación. Este archivo establece el formato y la organización de los elementos visuales que componen el reporte.

Dentro de este archivo se encuentran los siguientes elementos:

- **Encabezado:** Contiene el título del reporte y las imágenes correspondientes (logotipos).
- **Datos del Proyecto:** Se muestran datos clave como el código, título, entidad responsable y convocatoria del proyecto evaluado.
- **Subreportes:** Este reporte incluye tres subreportes encargados de mostrar información detallada de las evaluaciones.
- **Pie de Página:** Muestra información sobre la generación del reporte y la numeración de páginas.

Los reportes en JasperReports tienen tres componentes principales que estructuran su funcionamiento y permiten la generación de documentos dinámicos:

1. SQL (Consulta Principal)

- Es la base de la interacción del reporte con la base de datos.
- Se utiliza para recuperar los datos necesarios para la visualización del reporte.
- La consulta puede recibir parámetros para filtrar o personalizar los datos mostrados.

2. Parámetros

- Son valores dinámicos que se utilizan en el reporte para diferentes propósitos.
- Generalmente, se usan para filtrar datos en la consulta SQL, personalizar encabezados o configurar el comportamiento del reporte.
- Los parámetros pueden ser de diferentes tipos, como String, Long, Date, entre otros.

3. Subreportes

- Son reportes anidados dentro del reporte principal.
- Permiten dividir la información en secciones más organizadas y reutilizables.
- Cada subreporte puede recibir parámetros desde el reporte principal para garantizar la coherencia de los datos.
- Se definen en el reporte principal mediante expresiones de ruta, utilizando el directorio especificado en SUBREPORT_DIR.

Este esquema modular permite que JasperReports genere reportes complejos de forma organizada y escalable, evitando la sobrecarga de información en un solo archivo y mejorando la legibilidad y el mantenimiento del código. A continuación la descripción de estos ítems para el reporte principal:

3. Consulta Principal del Reporte

```
SELECT p.CODIGO_PROYECTO,
       p.TITULO,
       p.proyecto_id,
       e.nombre_entidad,
       c.descripcion_convocatoria
FROM ENT_ENTIDADES e,
     SGP_PROYECTOS p,
     SB_DEPENDENCIAS d,
     SGP_CONVOCATORIAS c,
     ADM_VALORES_DEFINICIONES v
WHERE c.CONVOCATORIA_ID(+) = p.CONVOCATORIA_ID
      AND v.VALOR_DEFINICION_ID(+) = p.ESTADO_PROYECTO_ID
      AND e.ENTIDAD_ID (+) = p.ENTIDAD_N_ID
      AND d.DEPENDENCIA_ID (+) = p.DEPENDENCIA_ID
      AND p.PROYECTO_ID = $P{PROYECTO_ID}
```

4. Parámetros del Reporte

PARÁMETRO	TIPO	DESCRIPCIÓN
-----------	------	-------------

PROYECTO_ID	Long	Identificador del proyecto evaluado.
IMG	String	Ruta de la imagen del reporte.
SUBREPORT_DIR	String	Directorio donde se encuentran los subreportes.
IMG_BASE64	String	Imagen en formato Base64 utilizada en el encabezado.
IMG_SECUNDARIA_BASE64	String	Imagen secundaria en formato Base64 utilizada en el encabezado.
matrizEvaluacionId	Long	Identificador de la matriz de evaluación.
tipo_matriz	String	Tipo de matriz (Proyectos o Regalías).
evaluacion_proyecto_id_1	Long	Identificador de la primera evaluación del proyecto.
evaluacion_proyecto_id_2	Long	Identificador de la segunda evaluación del proyecto.
evaluacion_proyecto_id_3	Long	Identificador de la tercera evaluación del proyecto.

5. Sub Reportes

Este reporte principal incorpora tres subreportes encargados de representar la información detallada de las evaluaciones.

SUBREPORTE	UBICACIÓN	DESCRIPCIÓN
formatoEvaluaciones.jasper	\$P{SUBREPORT_DIR} + "formatoEvaluaciones.jasper"	Presenta información relacionada con los formatos de evaluación.
conceptoEvaluaciones.jasper	\$P{SUBREPORT_DIR} + "conceptoEvaluaciones.jasper"	Muestra los conceptos y criterios evaluados en el proyecto.
puntajeFinalPromedio.jasper	\$P{SUBREPORT_DIR} + "puntajeFinalPromedio.jasper"	Contiene los cálculos de puntajes finales y promedios de la evaluación.

Subreporte (formatoEvaluaciones.jrxml)

El subreporte **formatoEvaluaciones.jrxml** es una sección auxiliar dentro del Reporte Consolidado de Calificación. Su función principal es estructurar y mostrar la información relacionada con los

formatos de evaluación utilizados en la matriz de evaluación del proyecto. Este subreporte contiene:

- Textos descriptivos relacionados con los aspectos de evaluación.
- Clasificación de criterios (criterio o subcriterio).
- Dos subreportes adicionales, encargados de mostrar información más detallada sobre los evaluadores y los puntajes promedios.

1. Consulta Principal del Reporte

```
SELECT fe.formato_evaluacion_id,
       fe.orden,
       fe.aspecto_calificacion,
       fe.entable,
       fe.formato_padre_id,
       TO_NUMBER(AVG(NVL(e.calificacion, 0))) AS suma_calificaciones
FROM sgp_formatos_evaluacion fe,
     pry_evaluaciones_calificacion e
WHERE fe.formato_id = $P{matrizEvaluacionId}
     AND UPPER(fe.aspecto_calificacion) != UPPER('CRITERIOS DE EVALUACIÓN')
     AND UPPER(fe.aspecto_calificacion) != UPPER('TOTAL')
     AND e.proyecto_id = $P{proyecto_id}
     AND fe.formato_evaluacion_id = e.formato_evaluacion_id(+)
GROUP BY fe.formato_evaluacion_id,
         fe.orden,
         fe.aspecto_calificacion,
         fe.entable,
         fe.formato_padre_id
ORDER BY fe.orden ASC;
```

La consulta utilizada en este subreporte obtiene los aspectos de calificación de la matriz de evaluación y sus respectivas calificaciones.

2. Parámetros del Reporte

PARÁMETRO	TIPO	DESCRIPCIÓN
proyecto_id	Long	Identificador del proyecto evaluado.
SUBREPORT_DIR	String	Ruta donde se encuentran los subreportes.
matrizEvaluacionId	Long	Identificador de la matriz de evaluación.
evaluacion_proyecto_id_1	Long	Identificador de la primera evaluación.
evaluacion_proyecto_id_2	Long	Identificador de la segunda evaluación.
evaluacion_proyecto_id_3	Long	Identificador de la tercera evaluación.

proyecto_id	Long	Identificador del proyecto evaluado.
-------------	------	--------------------------------------

3. Sub Reportes

Este reporte principal incorpora dos subreportes adicionales que complementan la información de los aspectos evaluados.

SUBREPORTES	UBICACIÓN	DESCRIPCIÓN
conceptoEvaluadores.jasper	\$P{SUBREPORT_DIR} + "conceptoEvaluadores.jasper"	Muestra la evaluación realizada por cada evaluador.
puntajePromedio.jasper	\$P{SUBREPORT_DIR} + "puntajePromedio.jasper"	Presenta el puntaje promedio obtenido en cada criterio.

Este subreporte actúa como una estructura intermedia que organiza y presenta los datos de evaluación antes de delegar la visualización detallada a sus subreportes.

Subreporte (conceptoEvaluadores.jrxml)

El subreporte **conceptoEvaluaciones.jrxml** es una sección clave dentro del Reporte Consolidado de Calificación. Su propósito principal es mostrar la calificación cuantitativa y cualitativa otorgada por cada evaluador para los conceptos definidos en la matriz de evaluación.

Este subreporte contiene:

- El título "**Concepto de los evaluadores**", que indica la sección donde se muestran las calificaciones.
- La calificación numérica otorgada por cada evaluador para un criterio específico.
- El concepto argumentado (cualitativo) que justifica la calificación asignada.
- La identificación de cada evaluador, presentada de manera anónima como "Evaluador 1", "Evaluador 2", etc.

El subreporte es dinámico y muestra la información en función de las evaluaciones realizadas para cada concepto en la matriz.

1. Consulta Principal del Reporte

```
SELECT fe.formato_id,
       e.evaluacion_proyecto_id,
       fe.aspecto_calificacion,
       e.calificacion,
       fe.entrabable,
       dbms_lob.substr(NVL(e.concepto,'Sin registro'),4000,1) concepto,
       dbms_lob.substr(NVL(e.concepto,''),8000,4001) concepto_parte2,
       ep.persona_id,
       ep.evaluador_scienti_id,
       (CASE
        WHEN ep.evaluador_scienti_id IS NOT NULL THEN
          (CONCAT(CONCAT(s.nombre_evaluador || ' ',
                        s.primer_apellido || ' '),
                   s.segundo_apellido))
        ELSE
          (CONCAT(CONCAT(pe.nombres || ' ',
                        pe.primer_apellido || ' '),
                   pe.segundo_apellido))
        END) persona,
       ep.funcionario,
       ep.nombre_panel
FROM pry_evaluaciones          ep,
     pry_evaluaciones_calificacion e,
     sgp_formatos_evaluacion    fe,
     sgp_personas               pe,
     sgp_evaluadores_scienti    s
WHERE fe.formato_evaluacion_id      (+)= e.formato_evaluacion_id
     AND ep.evaluacion_proyecto_id  (+)= e.evaluacion_proyecto_id
     AND s.evaluador_scienti_id      (+)= ep.evaluador_scienti_id
     AND pe.persona_id               (+)= ep.persona_id
     AND e.proyecto_id               = ${proyecto_id}
     AND e.formato_evaluacion_id      = ${formato_evaluacion_id}
     AND (ep.evaluacion_proyecto_id  = ${evaluacion_proyecto_id_1}
          OR ep.evaluacion_proyecto_id = ${evaluacion_proyecto_id_2}
          OR ep.evaluacion_proyecto_id = ${evaluacion_proyecto_id_3})
ORDER BY ep.fecha_creacion ASC;
```

La consulta SQL utilizada en este subreporte obtiene las calificaciones y conceptos argumentados por los evaluadores para un concepto específico dentro de la matriz de evaluación.

2. Parámetros del Reporte

PARÁMETRO	TIPO	DESCRIPCIÓN
proyecto_id	Long	Identificador del proyecto evaluado.
SUBREPORT_DIR	String	Ruta donde se encuentran los subreportes.
evaluacion_proyecto_id_1	Long	Identificador de la primera evaluación.
evaluacion_proyecto_id_2	Long	Identificador de la segunda evaluación.
evaluacion_proyecto_id_3	Long	Identificador de la tercera evaluación.

formato_evaluacion_id	Long	Identificador del concepto dentro de la matriz de evaluación.
-----------------------	------	---

Subreporte (puntajePromedio.jrxml)

El subreporte **puntajePromedio.jrxml** es una sección auxiliar dentro del Reporte Consolidado de Calificación que se encarga de mostrar los puntajes otorgados por cada evaluador para cada concepto de la matriz de evaluación, sin realizar ningún cálculo de promedios o ponderaciones.

Este subreporte contiene:

- Una estructura en la que se listan los puntajes individuales de los tres evaluadores.
- Campos que muestran la calificación numérica de cada evaluador, indicando "Sin Registro" si no se ha registrado una calificación.
- Identificadores anónimos de los evaluadores, mostrando "Evaluador 1", "Evaluador 2" y "Evaluador 3".
- Separadores visuales para mejorar la legibilidad de la información.
- Este subreporte es utilizado dentro del formatoEvaluaciones.jrxml, proporcionando una visualización clara de los puntajes asignados en cada evaluación.

1. Consulta Principal del Reporte

```
SELECT MAX(CASE
    WHEN ep.evaluacion_proyecto_id = ${evaluacion_proyecto_id_1} THEN
        e.calificacion
    END) AS Calificacion_1,
    MAX(CASE
    WHEN ep.evaluacion_proyecto_id = ${evaluacion_proyecto_id_2} THEN
        e.calificacion
    END) AS Calificacion_2,
    MAX(CASE
    WHEN ep.evaluacion_proyecto_id = ${evaluacion_proyecto_id_3} THEN
        e.calificacion
    END) AS Calificacion_3
FROM pry_evaluaciones ep,
     pry_evaluaciones_calificacion e,
     sgp_formatos_evaluacion fe
WHERE fe.formato_evaluacion_id = e.formato_evaluacion_id
AND ep.evaluacion_proyecto_id = e.evaluacion_proyecto_id
AND e.proyecto_id = ${proyecto_id}
AND e.formato_evaluacion_id = ${formato_evaluacion_id}
```

```
AND ep.evaluacion_proyecto_id IN ($P{evaluacion_proyecto_id_1},
$P{evaluacion_proyecto_id_2}, $P{evaluacion_proyecto_id_3});
```

La consulta SQL utilizada en este subreporte obtiene las calificaciones individuales de cada evaluador para el concepto en evaluación.

2. Parámetros del Reporte

PARÁMETRO	TIPO	DESCRIPCIÓN
proyecto_id	Long	Identificador del proyecto evaluado.
SUBREPORT_DIR	String	Ruta donde se encuentran los subreportes.
formato_evaluacion_id	Long	Identificador del concepto dentro de la matriz de evaluación.
evaluacion_proyecto_id_1	Long	Identificador de la primera evaluación.
evaluacion_proyecto_id_2	Long	Identificador de la segunda evaluación.
evaluacion_proyecto_id_3	Long	Identificador de la tercera evaluación.

Subreporte (conceptoEvaluaciones.jxrm)

El subreporte **conceptoEvaluaciones.jxrm** es una sección dentro del Reporte Consolidado de Calificación que presenta la evaluación cualitativa final de cada evaluador sobre el proyecto.

Este subreporte contiene:

- El concepto argumentado final, que representa la opinión general del evaluador sobre el proyecto y su desempeño en la matriz de evaluación.
- La identificación anónima de los evaluadores, presentada como "Evaluador 1", "Evaluador 2" y "Evaluador 3".
- Un formato estructurado, donde se concatena la evaluación en caso de que esté almacenada en múltiples partes.
- Esta sección del reporte permite visualizar la percepción global de los evaluadores sin revelar su identidad, manteniendo la confidencialidad.

1. Consulta Principal del Reporte

```
SELECT e.evaluacion_proyecto_id,
       e.proyecto_id,
       e.persona_id,
       (p.nombres || ' ' || p.primer_apellido || ' ' || p.segundo_apellido) persona,
       e.funcionario,
       e.nombre_panel,
       e.puntaje,
       dbms_lob.substr(NVL(e.concepto,'Sin registro'),4000,1) concepto,
       dbms_lob.substr(NVL(e.concepto,''),8000,4001) concepto_parte2
FROM   pry_evaluaciones e,
       sgp_personas p
WHERE  p.persona_id (+) = e.persona_id
AND    e.proyecto_id = $P{proyecto_id}
AND    (e.evaluacion_proyecto_id = $P{evaluacion_proyecto_id_1}
OR      e.evaluacion_proyecto_id = $P{evaluacion_proyecto_id_2}
OR      e.evaluacion_proyecto_id = $P{evaluacion_proyecto_id_3})
ORDER BY e.fecha_creacion ASC;
```

La consulta SQL utilizada en este subreporte obtiene los conceptos argumentativos finales dados por los evaluadores sobre el proyecto.

2. Parámetros del Reporte

PARÁMETRO	TIPO	DESCRIPCIÓN
proyecto_id	Long	Identificador del proyecto evaluado.
SUBREPORT_DIR	String	Ruta donde se encuentran los subreportes.
evaluacion_proyecto_id_1	Long	Identificador de la primera evaluación.
evaluacion_proyecto_id_2	Long	Identificador de la segunda evaluación.
evaluacion_proyecto_id_3	Long	Identificador de la tercera evaluación.

Subreporte (puntajeFinalPromedio.jxml)

El subreporte **puntajeFinalPromedio.jxml** forma parte del Reporte Consolidado de Calificación y tiene como propósito mostrar la evaluación cuantitativa final otorgada por cada evaluador sobre el proyecto, sin realizar promedios ni ponderaciones.

Este subreporte contiene:

- El puntaje final asignado por cada evaluador, representando su calificación general del proyecto.

- La identificación anónima de los evaluadores, listados como "Evaluador 1", "Evaluador 2" y "Evaluador 3".
- Manejo de valores nulos, donde si un evaluador no ha asignado un puntaje, se muestra "Sin Registro".
- Separadores visuales, que ayudan a estructurar la presentación de la información.
- Este subreporte permite visualizar la calificación cuantitativa de cada evaluador de manera individual, asegurando transparencia sin consolidar los valores en un solo puntaje.

1. Consulta Principal del Reporte

```
SELECT
    MAX(CASE WHEN e.evaluacion_proyecto_id = ${evaluacion_proyecto_id_1} THEN e.puntaje END)
AS puntaje_evaluador_1,
    MAX(CASE WHEN e.evaluacion_proyecto_id = ${evaluacion_proyecto_id_2} THEN e.puntaje END)
AS puntaje_evaluador_2,
    MAX(CASE WHEN e.evaluacion_proyecto_id = ${evaluacion_proyecto_id_3} THEN e.puntaje END)
AS puntaje_evaluador_3
FROM pry_evaluaciones e
WHERE e.proyecto_id = ${proyecto_id}
AND e.evaluacion_proyecto_id IN (${evaluacion_proyecto_id_1},
${evaluacion_proyecto_id_2}, ${evaluacion_proyecto_id_3});
```

La consulta SQL utilizada en este subreporte obtiene los puntajes finales asignados por cada evaluador para el proyecto en evaluación.

2. Parámetros del Reporte

PARÁMETRO	TIPO	DESCRIPCIÓN
proyecto_id	Long	Identificador del proyecto evaluado.
SUBREPORT_DIR	String	Ruta donde se encuentran los subreportes.
evaluacion_proyecto_id_1	Long	Identificador de la primera evaluación.
evaluacion_proyecto_id_2	Long	Identificador de la segunda evaluación.
evaluacion_proyecto_id_3	Long	Identificador de la tercera evaluación.

c. Implementar Registro de Múltiples Líneas Temáticas y Múltiples Sublíneas Temáticas

Solicitud de requerimiento No. 157596

Descripción y alcance del Requerimiento

El sistema debe permitir seleccionar mínimo una línea temática y asociar más de una sublínea a cada línea.

La presente solicitud de basa en las modificaciones objeto de la Adenda No. 1 de la convocatoria 37 publicada el 30 de abril de 2025.

Desarrollo e Implementación

Este requerimiento tiene como objetivo la implementación de una nueva sección dentro del formulario de proyectos para la gestión de líneas temáticas y sublíneas temáticas asociadas. Esta funcionalidad fue desarrollada para permitir una mayor flexibilidad en el registro del enfoque temático de los proyectos, cumpliendo con los lineamientos técnicos establecidos en las convocatorias y mejorando la trazabilidad y calidad de la información registrada en el sistema.

Los requerimientos cubiertos en este desarrollo incluyen:

1. Visualización de líneas temáticas registradas:

- Mostrar un listado de todas las líneas temáticas asociadas al proyecto actual.
- Validar la existencia de datos para condicionar la visualización del bloque; si no existen líneas asociadas, ocultar la sección.

2. Gestión de sublíneas temáticas por línea:

- Para cada línea temática listada, mostrar las sublíneas temáticas asociadas.
- Incluir en la tabla la opción de eliminar una sublínea específica.

3. Asociación de múltiples líneas y sublíneas temáticas:

- Permitir la selección de una línea temática y la asociación simultánea de múltiples sublíneas temáticas mediante controles tipo selectOne y selectMany.

4. Modificación del reporte de ficha técnica del proyecto:

- Ajustar el reporte generado en PDF mediante iText para incluir una nueva sección titulada “Líneas temáticas y sublíneas asociadas”.

5. Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- Java EE con Enterprise JavaBeans (EJB) para la lógica de negocio.
- JSF (JavaServer Faces) y PrimeFaces para la interfaz de usuario.
- JPA (Criteria API) para la manipulación de datos en la base de datos.
- Generación de reportes en PDF utilizando componentes personalizados.
- Procedimientos almacenados para la actualización de tablas y sincronización de datos.

Formulario de Líneas y Sublíneas Temáticas

Componentes

Para generar el formulario que permite adicionar y/o borrar las diferentes Líneas y Sublíneas Temáticas que existen en la aplicación, se utilizan los siguientes componentes en la aplicación:

- **Vista (View):** Representada por el archivo líneasSublíneas.xhtml, que es una página XHTML asociada a la interfaz de usuario.
- **Controlador (Controller de Vista y Conversación):** Representado por las clases LineaSublineaSessController y LineaSublineaConvController, que son los controladores de la capa de presentación.
- **Controlador Reporte:** Representado por la Clase **ItxtReporteProyectosSessController** que es el controlador que gestiona la generación de la ficha técnica del proyecto.
- **Servicio EJB (EJB Service):** Representado por la clase ProyectoServiceBean, que es el EJB (Enterprise JavaBeans) para gestionar las líneas y sublíneas temáticas.
- **Base de Datos (Database):** Representada por la entidad LineaSublineaDTO, un data transfer object que hace referencia a la tabla PRY_LINEAS_SUBLINEAS indicando que hay alguna interacción con la base de datos para manejar estos tipos.

- **JPA (Java Persistence API):** Representado por la clase **CriteriaBuilder**, que es una interfaz de JPA utilizada para construir consultas dinámicamente.
- **PrimeFaces:** Representado por la clase **LazyDataModel**, que es la implementación específica de PrimeFaces para el manejo de datos paginados de manera eficiente.

Codificación

Interfaz de usuario (lineasSublineas.xhtml)

Campo "Línea Temática":

Este componente permite al usuario seleccionar una línea temática dentro del formulario. Al realizar una selección, se dispara una actualización dinámica (mediante AJAX) que carga automáticamente las sublíneas temáticas asociadas a la línea seleccionada.

```
<p:outputLabel value="#{msg['formulario.proyectos.registro.lineaSublinea.linea.label']}"
styleClass="col-lg-2 control-label"
for="lineaTematica-lineasSublineas"
/>
<div class="col-lg-9">
<p:selectOneMenu id="lineaTematica-lineasSublineas"
converter="lineaTematicaConverter" value="#{lineasSublineasSessController.lineaSelected}"
required="true"
requiredMessage="#{msg['formulario.proyectos.registro.lineaSublinea.linea.required']}"
style="width:100%"
autoWidth="false"
>
<p:ajax event="change"
update="sublineaTematica-lineasSublineas"
listener="#{lineasSublineasSessController.sublineasTematicasxLineaTematica}"
oncomplete="initializeChosenClean();"
/>
<f:selectItem itemLabel="#{msg['sb.selectOneMenu.registro']}" />
<f:selectItems value="#{lineasSublineasSessController.listaLineasTematicasConv}"
var="lineaTematica" itemLabel="#{lineaTematica.descripcionValorDefinicion}"
itemValue="#{lineaTematica}"/>
</p:selectOneMenu>
</div>
```

A continuación, una explicación más detallada:

1. Propiedades del componente:

- **id="lineaTematica-lineasSublineas":** Identificador único del componente en la vista.
- **converter="lineaTematicaConverter":** Convierte entre el objeto **Java LineaTematica** y

su representación en cadena para el **frontend**. Este conversor es necesario para que **JSF** sepa cómo interpretar el objeto seleccionado.

- **value="#{lineasSublineasSessController.lineaSelected}"**: Propiedad del managed bean donde se almacena la línea temática seleccionada.
- **required="true"**: Hace que este campo sea obligatorio.
- **requiredMessage="..."**: Mensaje de validación que se mostrará si no se selecciona ninguna opción.
- **style="width:100%"**: Aplica estilo CSS para que el **select** ocupe el 100% del ancho de su contenedor.
- **autoWidth="false"**: Previene que **PrimeFaces** ajuste automáticamente el ancho del menú.

2. Ajax dinámico al cambiar la selección:

- **event="change"**: Se activa cuando el usuario cambia la selección.
- **update="sublineaTematica-lineasSublineas"**: Componente que se actualizará (por ejemplo, un **select** de sublíneas temáticas dependientes).
- **listener="#{lineasSublineasSessController.sublineasTematicasxLineaTematica}"**: Método del bean que se ejecutará para actualizar las sublíneas correspondientes a la línea temática seleccionada.
- **oncomplete="initializeChosenClean();" : JavaScript** que se ejecuta una vez completada la llamada Ajax (por ejemplo, para reinicializar estilos o plugins).

Campo "Sublínea Temática":

Una vez seleccionada la línea temática, este componente permite al usuario seleccionar múltiples sublíneas temáticas correspondientes a esa línea. Está optimizado con el plugin Chosen, lo que mejora la experiencia de selección múltiple y permite buscar dentro de las opciones.

```
<p:outputLabel value="#{msg['formulario.proyectos.registro.lineaSublinea.subLinea.label']}"
styleClass="col-lg-2 control-label"
for="sublineaTematica-lineasSublineas"
/>
<div class="col-lg-9">
```

```
<h:selectManyMenu id="sublineaTematica-lineasSublineas"
value="#{lineasSublineasSessController.sublineasSelected}" required="true"
styleClass="form-control chosenSelectSublineasxLinea"
requiredMessage="#{msg['formulario.proyectos.registro.lineaSublinea.subLinea.required']}"
converter="sublineaTematicaConverter"
>
<f:selectItems value="#{lineasSublineasSessController.sublineasTematicasLinea}"
var="subLineaTemat" itemLabel="#{subLineaTemat.descripcionValorDefinicion}"
itemValue="#{subLineaTemat}"
/>
</h:selectManyMenu>
</div>
<script>
function initializeChosenClean() { try {
const selector = '#lineasublineadtoForm\\:sublineaTematica-lineasSublineas'; const $select =
$(selector);
if ($select.length) {
// Elimina cualquier contenedor huérfano anterior
$('#lineasublineadtoForm_sublineaTematica_lineasSublineas_chosen').remove();
// Reinicia el plugin chosen si ya estaba if ($select.data('chosen')) {
$select.chosen('destroy');
}
// Vuelve a aplicar Chosen
$select.chosen({ width: "100%",
placeholder_text_multiple:
"#{msg['formulario.proyectos.registro.lineaSublinea.subLinea.placeholder']}"
});
} else {
console.warn("No se encontró el select para inicializar chosen.");
}
} catch (e) {
console.error('Error al inicializar Chosen:', e);
}
}
$(document).ready(function () { initializeChosenClean();
});
</script>
```

A continuación, una explicación más detallada:

1. Propiedades del componente:

```
h:selectManyMenu id="sublineaTematica-lineasSublineas"
value="#{lineasSublineasSessController.sublineasSelected}" required="true"
styleClass="form-control chosenSelectSublineasxLinea"
requiredMessage="#{msg['formulario.proyectos.registro.lineaSublinea.subLinea.required']}"
converter="sublineaTematicaConverter"
>
```

- **id="sublineaTematica-lineasSublineas"**: Identificador del menú de selección

múltiple.

- **value="#{lineasSublineasSessController.sublineasSelected}"**: Lista de objetos seleccionados. Debe ser un **List<ValorDefinicionDTO>** en el bean.
- **required="true"**: Hace que sea obligatorio seleccionar al menos una opción.
- **requiredMessage="..."**: Mensaje que se muestra si no se cumple la validación.
- **styleClass="form-control chosenSelectSublineasxLinea"**: Clase CSS usada tanto para estilo como para enganchar el plugin Chosen (un mejorador de selects).
- **converter="sublineaTematicaConverter"**: Conversor que transforma entre el objeto **ValorDefinicionDTO** y su valor **string**. Necesario para que JSF pueda manejar correctamente los valores seleccionados.

```
<f:selectItems value="#{lineasSublineasSessController.sublineasTematicasLinea}"
var="subLineaTemat" itemLabel="#{subLineaTemat.descripcionValorDefinicion}"
itemValue="#{subLineaTemat}"
/>
```

2. Carga de opciones (<f:selectItems>):

- **value**: Lista de sublíneas temáticas disponibles, filtradas por la línea temática seleccionada.
- **var**: Variable local para iterar sobre cada objeto de la lista.
- **itemLabel**: Texto que se mostrará al usuario.
- **itemValue**: Objeto real que será enviado al **backend**. Requiere el uso del **converter**.

3. Script JavaScript: initializeChosenClean():

Este script tiene como objetivo aplicar correctamente el plugin **JavaScript** llamado **Chosen** a un componente JSF (<h:selectManyMenu>) después de que la página cargue o después de una actualización parcial (**AJAX**).

```
function initializeChosenClean() {
  try {

    const selector = '#lineasublineadtoForm\\:sublineaTematica-lineasSublineas';
    const $select = $(selector);

    if ($select.length) {

      // Limpia el contenedor anterior si ya existía

      $('#lineasublineadtoForm_sublineaTematica_lineasSublineas_chosen').remove();

      // Si Chosen ya está aplicado, lo destruye
      if ($select.data('chosen')) {

        $select.chosen('destroy');

      }

      // Reaplica Chosen con configuración deseada

      $select.chosen({
        width: "100%",

        placeholder_text_multiple:
        "#{msg['formulario.proyectos.registro.lineaSublinea.subLinea.placeholder']}"

      });

    } else {

      console.warn("No se encontró el select para inicializar chosen.");

    }

  } catch (e) {

    console.error('Error al inicializar Chosen:', e);

  }

}
```

A continuación, una explicación más detallada:

- **Busca el elemento <select> en el DOM:**

Define el selector con `const selector = '#lineasublineadtoForm\\:sublineaTematica- lineasSublineas';` y luego lo obtiene con jQuery: `const $select = $(selector);`.

- **Verifica si el elemento existe**

Comprueba que el select realmente esté en la página con `if ($select.length)` antes de seguir con la inicialización.

- **Elimina cualquier rastro visual anterior de Chosen**

Borra el contenedor generado por el plugin anterior con

```
$('#lineasublineadtoForm_sublineaTematica_lineasSublineas_chosen').remove();
```

Esto previene duplicados si el componente fue actualizado por AJAX.

- **Destruye la instancia previa del plugin Chosen**

Si el plugin ya estaba aplicado, lo desmonta con `if ($select.data('chosen')) {`

`$select.chosen('destroy'); }`.

- **Inicializa Chosen desde cero con nueva configuración**

Aplica el plugin de nuevo con:

```
$select.chosen({ width: "100%", placeholder_text_multiple:
"#{msg['formulario.proyectos.registro.lineaSublinea.subLinea.placeholder']}" });
```

Esto asegura que el select tenga estilo moderno, permita búsquedas y muestre un placeholder.

- **Controla posibles errores**

Si algo falla en el proceso, el `catch` captura la excepción y la muestra en consola con `console.error('Error al inicializar Chosen:', e);`.

- **Se ejecuta al cargar la página**

Gracias a `$(document).ready(function () { initializeChosenClean(); });`, el plugin se aplica automáticamente cuando el DOM está listo.

Botón “Adicionar”:

Este botón permite al usuario agregar las sublíneas temáticas seleccionadas en el menú múltiple, asociándolas a la línea temática previamente elegida. Es el paso final para registrar la relación entre línea y sublíneas en el sistema.

```
<p:commandLink id="agregarBtn"

    value="#{msg['sb.component.boton.adicionar']}"
    update="#{lineasSublineasSessController.modulo}Form, listaLineasSublineas"
    actionListener="#{lineasSublineasConvController.guardar(ae)}"
    styleClass="btn btn btn-primary"

title="#{msg['formulario.proyectos.registro.lineaSublinea.addLinea.btn.aria.label']}"
ariaLabel="#{msg['formulario.proyectos.registro.lineaSublinea.addLinea.btn.aria.label']}"
```

A continuación, una explicación más detallada:

1. Propiedades del componente:

- **id="agregarBtn"**: Identificador único del botón dentro de la vista, útil para referencias desde el frontend o AJAX.
- **value="#{msg['sb.component.boton.adicionar']}"**: Define el texto visible del botón, internacionalizado desde un archivo de propiedades (por ejemplo, "Adicionar").
- **update="#{lineasSublineasSessController.modulo}Form, listaLineasSublineas"**: Indica qué componentes JSF se deben actualizar tras ejecutar la acción; en este caso, se actualiza el formulario principal y la lista de líneas y sublíneas registradas.
- **actionListener="#{lineasSublineasConvController.guardar(ae)}"**: Llama al método del backend encargado de guardar la asociación entre la línea temática seleccionada y las sublíneas marcadas.
- **styleClass="btn btn btn-primary"**: Aplica estilos visuales tipo Bootstrap para que el botón tenga apariencia destacada como botón principal.
- **title y ariaLabel**: Proveen accesibilidad y ayuda contextual, mostrando un tooltip al pasar el cursor y permitiendo que lectores de pantalla describan correctamente el botón.
- **<i class="fa fa-plus"></i>**: Añade un icono de suma (ícono de Font Awesome) para reforzar visualmente la acción de "agregar" o "adicionar".

Lista de Sublíneas Temáticas por Líneas Temáticas:

Esta tabla permite visualizar las sublíneas temáticas agrupadas por sus respectivas líneas temáticas. Muestra los datos previamente registrados por el usuario, y permite identificar fácilmente la relación entre ambos niveles de categorización.

```
<p:dataTable value="#{lineasSublineasSessController.listLineasSublineas}" var="item"
              sortBy="#{item.lineaId.descripcionValorDefinicion}"
              styleClass="table table-striped table-bordered table-hover
dataTables-entidades" rowKey="#{item.lineaSublineaId}"
              style="margin-top: 1rem;" paginator="true" rows="10"
              rowsPerPageTemplate="5,10,15,20" lazy="true"
paginatorPosition="bottom">
    <!-- Headers -->
    <p:columnGroup type="header">
        <p:row>
            <p:column
headerText="#{msg['formulario.proyectos.registro.lineaSublinea.table.linea']}" style="width:
45%"/>
            <p:column
headerText="#{msg['formulario.proyectos.registro.lineaSublinea.table.subLinea']}"
style="width: 45%"/>
            <p:column
headerText="#{msg['formulario.proyectos.registro.lineaSublinea.table.acciones']}"
style="width: 10%; text-align: center;" />
        </p:row>
    </p:columnGroup>
    <!-- Línea -->
    <p:column                                groupRow="true"
sortBy="#{item.lineaId.descripcionValorDefinicion}">
        <h:outputText value="#{item.lineaId.descripcionValorDefinicion}"
style="font-weight: bold; background-color: #f0f0f0;
padding: 8px; display:
block;" />
    </p:column>
    <!-- Sublínea -->
    <p:column>
        #{item.sublineaId.descripcionValorDefinicion}
    </p:column>
</p:dataTable>
```

A continuación, una explicación más detallada:

1. Configuración general de la tabla (<p:dataTable>):

- **value="#{lineasSublineasSessController.listLineasSublineas}":** Fuente de datos que contiene la lista de objetos con relaciones entre línea y sublínea temática.
- **var="item":** Variable que representa cada fila de la tabla durante la iteración.
- **sortBy="#{item.lineaId.descripcionValorDefinicion}":** Ordena las filas por el nombre de la línea temática.
- **styleClass="table table-striped table-bordered table-hover dataTables- entidades":** Aplica estilos Bootstrap para formato de tabla, bordes, sombreado y efectos hover.
- **rowKey="#{item.lineaSublineaId}":** Define una clave única para cada fila,

necesaria para operaciones como selección o edición.

- **style="margin-top: 1rem;"**: Agrega espacio superior para separar visualmente la tabla del contenido anterior.
- **paginator="true"**: Activa la paginación para mejorar la navegación cuando hay muchos registros.
- **rows="10"**: Número de filas mostradas por página por defecto.
- **rowsPerPageTemplate="5,10,15,20"**: Permite al usuario cambiar el número de filas por página.
- **lazy="true"**: Indica que los datos pueden cargarse de forma perezosa (ideal para listas grandes y rendimiento).
- **paginatorPosition="bottom"**: Coloca los controles de paginación en la parte inferior de la tabla.

2. Encabezados de la tabla (<p:columnGroup type="header">):

- **<p:column headerText="..." />**: Define las etiquetas visibles en el encabezado:
 - **Línea temática**: Ocupa el 45% del ancho
 - **Sublínea temática**: También 45%.
 - **Acciones**: Ocupa el 10% y se centra (para botones como eliminar o editar, si se agregan después).

3. Encabezados de la tabla (<p:columnGroup type="header">):

- **Línea Temática (<p:column> con groupRow="true")**:
 - Agrupa visualmente las sublíneas bajo cada línea temática.
 - Muestra el valor con estilo diferenciado (**<h:outputText>** con fondo gris claro y texto en negrita).
- **Sublínea Temática (<p:column>)**:
 - Muestra directamente el texto de la sublínea relacionada con la línea de la fila.

Botón "Eliminar"

Este botón permite al usuario eliminar sublíneas temáticas específicas de la lista visible. Al hacer clic, se ejecuta la acción de eliminación en el backend y se actualiza la interfaz para reflejar los cambios realizados.

- **title="#{msg['sb.component.boton.eliminar']}":** Texto que aparece como tooltip al pasar el cursor sobre el botón, indicando su función.
- **ariaLabel="#{msg['formulario.proyectos.registro.lineaSublinea.deleteLinea.btn.aria.label']}":** Etiqueta accesible para lectores de pantalla, mejora la accesibilidad.
- **styleClass="btn btn-sm btn-primary":** Aplica estilos Bootstrap para que el botón sea pequeño y con apariencia primaria (color azul).
- **actionListener="#{lineasSublineasConvController.eliminar(item)}":** Llama al método que elimina la línea/sublínea específica representada por la variable item.
- **process="@this":** Indica que solo el propio botón se procesa en la petición AJAX, optimizando rendimiento.
- **update="@form, #{lineasSublineasSessController.modulo}Form":** Actualiza el formulario actual y el formulario principal del módulo después de la eliminación para reflejar los cambios.

Controlador (LineaSublineaSessController)

El usuario ingresa a la interfaz de usuario de la pantalla de gestión de líneas y sublíneas temáticas (XHTML), la cual es procesada por el controlador. Este controlador al acceder a la pantalla invoca al método **cargarListas()**, que consiste en diversos métodos que realizan varias operaciones al momento de cargar la pantalla. La anotación **@PostConstruct** indica que el método **cargarListas()** se ejecutará después de que la clase sea construida y todas las líneas y sublíneas temáticas hayan sido inyectadas. En este caso, se utiliza para inicializar algunas variables y cargar listas al iniciar la vista o el bean de respaldo.

Veamos una explicación de las partes clave del código:

```
@PostConstruct
public void cargarListas() {
    this.sublineasSelected = new ArrayList<>();
    this.lineaSelected = new ValorDefinicionDTO();
    try {
        Map<String, Object> filters = new HashMap<>();
        //Filtro por proyecto ID
        filters.put("proyectoId",
            appSessController.getProyectoSelected().getProyectoId());
        int count = proyectoServiceFacade.lineaSublineaListCount(filters);
        if (count > 0) {
            listLineasSublineas = proyectoServiceFacade.lineaSublineaList(-1, -1,
                "lineaId.descripcionValorDefinicion", "A", filters);
            //Ordenar sublineas por lineas listLineasSublineas.sort(Comparator
                .comparing((LineaSublineaDTO item) ->
                    item.getLineaId().getDescripcionValorDefinicion(),
                    Comparator.nullsLast(String::compareToIgnoreCase))
                .thenComparing(item ->
                    item.getSublineaId().getDescripcionValorDefinicion(),
                    Comparator.nullsLast(String::compareToIgnoreCase))

            );
        } else {
            listLineasSublineas = new ArrayList<>();
        }
        this.cargarLineasTematicasConvocatoria();
    } catch (Exception ex) {
        LOG.error("Error al cargar listas ", ex);
    }
}
```

@PostConstruct:

Este método se ejecuta después de que la instancia de la clase ha sido creada y todas las líneas y sublíneas temáticas (si las hay) han sido inyectadas. Aquí se realiza la inicialización de la clase, llamando al método **cargarLineasTematicasConvocatoria()** y configurando un modelo de datos para manejar la carga de datos de manera eficiente en una tabla o componente similar en una interfaz de usuario.

cargarLineasTematicasConvocatoria():

```
public void cargarLineasTematicasConvocatoria() {
    try {
        listaLineasTematicasConv = new ArrayList<>();
        Map<String, Object> filters = new HashMap<>();
    }
}
```

```

        filters.put("convocatoriaId",
appSessController.getProyectoSelected().getConvocatoriaId().getConvocatoriaId());
        List<ConvocatoriaLineaDTO> listaLineasConvocatoria
            = convocatoriaServiceFacade.convocatoriaLineasList(-1, -1,
"lineaId.descripcionValorDefinicion", "A", filters);
        if (listaLineasConvocatoria != null && !listaLineasConvocatoria.isEmpty()) {
            for (ConvocatoriaLineaDTO convocatoriaLineaDTO : listaLineasConvocatoria) {
                listaLineasTematicasConv.add(convocatoriaLineaDTO.getLineaId());
            }
        }
    } catch (Exception ex) {
        LOG.error("Error la consultar las lineas tematicas de la convocatoria ", ex);
    }
}

```

El método **cargarLineasTematicasConvocatoria()** se encarga de inicializar y cargar la lista **listaLineasTematicasConv**, que contiene objetos de tipo línea temática para ser usados en los componentes de selección del formulario.

Para ello, crea un mapa de filtros donde incluye el identificador de la convocatoria actualmente seleccionada, obtenido desde el controlador de sesión de la aplicación (**appSessController**). Luego invoca el método **convocatoriaLineasList** del **convocatoriaServiceFacade**, el cual realiza una consulta paginada (sin límites, con parámetros -1) ordenada por la descripción de la línea temática, y con estado activo ("A"). Si la consulta devuelve resultados, itera sobre ellos extrayendo la propiedad **lineaId** de cada objeto **ConvocatoriaLineaDTO** y la agrega a la lista interna. Todo el proceso está envuelto en un bloque try-catch para capturar y registrar cualquier excepción que pueda ocurrir durante la llamada al servicio o la manipulación de datos.

sublineasTematicasxLineaTematica():

```

public void sublineasTematicasxLineaTematica() {
    try {
        List<ValorDefinicionDTO> listaSublineas = new ArrayList<>();
        if (lineaSelected.getTipoValidacion() != null) {
            listaSublineas
projectoServiceFacade.obtenerValoresDefinicion(lineaSelected.getTipoValidacion().getTipoDefi
ni cion()
        );
    }
    this.setSublineasTematicasLinea(listaSublineas);
} catch (Exception ex) {
    LOG.error("Error al cargar las sublineas tematicas por linea tematica ", ex);
}
}

```

El método **sublineasTematicasxLineaTematica()** sirve para cargar las sublíneas temáticas asociadas a la línea temática seleccionada (**lineaSelected**). Para ello, inicializa una lista vacía de objetos **ValorDefinicionDTO** que representarán las sublíneas. Luego, verifica si la línea seleccionada tiene definido un tipo de validación (**tipoValidacion**), y en ese caso invoca el método **obtenerValoresDefinicion** del servicio **proyectoServiceFacade**, pasando como parámetro el tipo de definición relacionado a ese tipo de validación. El resultado, que contiene las sublíneas temáticas correspondientes, se asigna a la propiedad **sublineasTematicasLinea** para ser utilizada en la vista. El proceso está protegido con un bloque try-catch que captura y registra cualquier excepción que pueda ocurrir durante la obtención de datos.

Controlador (LineaSublineaConvController)

El usuario accede a la interfaz de usuario de la pantalla de gestión de líneas y sublíneas temáticas (XHTML), la cual es gestionada por el controlador al ejecutar acciones en la pantalla.

guardar()

Uno de los métodos clave gestionados por el controlador es **guardar()**, responsable de persistir la asociación entre una línea temática seleccionada y una o varias sublíneas temáticas.

```
@Override
public void guardar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ValorDefinicionDTO linea = lineasSublineasSessController.getLineaSelected();
        List<ValorDefinicionDTO> sublineas
            = lineasSublineasSessController.getSublineasSelected();
        //Auditoria
        AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
        appSessionVO.setProgram(program);
        for (ValorDefinicionDTO sublinea : sublineas) {
            LineaSublineaDTO lineaSublineaDTO = new LineaSublineaDTO();
            lineaSublineaDTO.setProyectoId(appSessController.getProyectoSelected());
            lineaSublineaDTO.setLineaId(linea);
            lineaSublineaDTO.setSublineaId(sublinea);

            lineaSublineaDTO.setUsuarioCreador(appSessController.getProyectoSelected().getProyectoId().toString());
        }
        lineaSublineaDTO.setFechaCreacion(new Date());
        lineaSublineaDTO.setAppSessionVO(appSessionVO);
        proyectoServiceFacade.guardarLineaSublinea(lineaSublineaDTO);
    }
    lineasSublineasSessController.setSelected(new LineaSublineaDTO());
}
```



```
        lineasSublineasSessController.cargarListas();
    } catch (Exception ex) {
        message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
        LOG.error("Error al guardar la region de proyecto ", ex);
        FacesContext.getCurrentInstance().addMessage(null, message);
    } finally {
        FacesContext.getCurrentInstance().addMessage(null, message);
    }
}
```

Este método procesa los datos ingresados por el usuario en el formulario, registra la operación para efectos de auditoría y actualiza el estado de la vista para reflejar correctamente las relaciones creadas.

A continuación, una explicación más detallada:

- Se crea un **FacesMessage** de tipo **INFO** con mensajes i18n (**resolveExpression("#{msg[...]}")**) para mostrar un mensaje de éxito tras la operación.
- Se obtiene la línea temática seleccionada con **lineasSublineasSessController.getLineaSelected()** (tipo **ValorDefinicionDTO**).
- Se obtiene la lista de sublíneas seleccionadas con **lineasSublineasSessController.getSublineasSelected()** (lista de **ValorDefinicionDTO**).
- Se recupera la sesión de seguridad **AppSessionVO** desde **SeguridadSessController** y se le asigna el nombre del programa (**appSessionVO.setProgram(program)**) para efectos de auditoría.
- Por cada sublínea seleccionada:
 - Se instancia un **LineaSublineaDTO**.

- Se le asigna el proyecto seleccionado (**appSessController.getProyectoSelected()**), la línea (linea) y la sublínea (sublinea).
 - Se registra el usuario creador con el **ID** del proyecto como string (**getProyectoSelected().getProjectoid().toString()**), y la fecha de creación (**new Date()**).
 - Se adjunta el **AppSessionVO** para efectos de trazabilidad/auditoría.
 - Se guarda la entidad usando **proyectoServiceFacade.guardarLineaSublinea(...)**.
- Después del guardado, se reinicia el objeto seleccionado (**setSelected(new LineaSublineaDTO())**) y se recargan las listas (**cargarListas()**).
 - En caso de excepción, se reemplaza el mensaje por uno de tipo **ERROR**, se imprime el error en el log y se muestra el mensaje de error en pantalla.
 - Finalmente, siempre se agrega el **FacesMessage** (éxito o error) al **FacesContext** para su visualización en la interfaz.

eliminar(LineaSublineaDTO lineaSublineaDTO)

Uno de los métodos clave gestionados por el controlador es **eliminar(LineaSublineaDTO lineaSublineaDTO)**, el cual se encarga de procesar la eliminación de una relación existente entre una línea temática y una sublínea temática.

Este método se ejecuta cuando el usuario selecciona una combinación específica en la interfaz y solicita su eliminación, garantizando que la operación se registre correctamente en auditoría y que la vista se actualice para reflejar los cambios.

```
public void eliminar(LineaSublineaDTO lineaSublineaDTO) {  
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,  
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),  
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));  
    try {  
        //Auditoria  
        AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();  
        appSessionVO.setProgram(program);  
        lineaSublineaDTO.setAppSessionVO(appSessionVO);  
        proyectoServiceFacade.eliminarLineaSublinea(lineaSublineaDTO);  
        lineasSublineasSessController.setSelected(new LineaSublineaDTO());  
    }  
}
```

```
        lineasSublineasSessController.cargarListas();
        this.obtenerPestana = 0;
    } catch (Exception ex) {
        message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
        resolveExpression("#{msg['sb.mensajes.error.general.conversation']}"));
        LOG.error("Error al eliminar la region priorizada del proyecto ", ex);
    } finally {
        FacesContext.getCurrentInstance().addMessage(null, message);
    }
}
```

- Se crea un **FacesMessage** de tipo **INFO** con mensajes internacionalizados (**#{msg[...]}**) para mostrar al usuario una notificación de éxito.
- Se obtiene la sesión de seguridad (**AppSessionVO**) desde **seguridadSessController** y se le asigna el programa actual (**setProgram(program)**) para fines de auditoría.
- Se adjunta el **AppSessionVO** al DTO recibido como parámetro (**lineaSublineaDTO.setAppSessionVO(...)**).
- Se llama a **proyectoServiceFacade.eliminarLineaSublinea(...)** para eliminar la relación línea–sublínea del proyecto en base de datos.
- Se limpia la selección actual (**setSelected(new LineaSublineaDTO())**) y se recargan las listas en sesión (**cargarListas()**), asegurando que la vista refleje el cambio.
- Se resetea la variable **obtenerPestana** a 0 (posiblemente para mostrar una pestaña específica del formulario después de la eliminación).
- Si ocurre una excepción, se genera un **FacesMessage** de tipo **ERROR** con mensaje genérico de error de conversación, y se registra el error en el log.
- Finalmente, siempre se añade el **FacesMessage** resultante al **FacesContext** para mostrarlo en el frontend.

Servicio EJB (ProyectoServiceBean)

El controlador se comunica con el Servicio EJB de la pantalla (**ProyectoServiceBean**) llamando al método **lineaSublineaList()**, el cual nos proporciona la información necesaria desde la base de datos para ser visualizada en pantalla.

Es fundamental instanciar dos objetos clave utilizados para realizar consultas con la Java Persistence API (JPA): **EntityManager** y **CriteriaBuilder**. A continuación, la función de cada uno de ellos:

```
private EntityManager em;
private CriteriaBuilder cb;
private CriteriaQuery<LineaSublinea> queryLineaSublinea;
private Root<LineaSublinea> LineaSublineaRoot;

public void setEntityManager(EntityManager em) {
    this.em = em;
}
```

EntityManager

El **EntityManager** es una interfaz en JPA que proporciona métodos para interactuar con la base de datos. Su función principal es gestionar las entidades en el contexto de persistencia, permitiendo realizar operaciones de lectura y escritura en la base de datos. Algunas de las funciones claves del **EntityManager** incluyen:

- **Persistencia de Entidades:** **EntityManager** se utiliza para persistir (almacenar) nuevas entidades en la base de datos.
- **Actualización de Entidades:** Se utiliza para actualizar entidades existentes en el contexto de persistencia.
- **Eliminación de Entidades:** Permite eliminar entidades de la base de datos.
- **Consultas JPA:** Facilita la ejecución de consultas JPA para recuperar datos de la base de datos.

En el código, **EntityManager** se utiliza para crear instancias de consultas (**queryLineaSublinea**) y para configurar el contexto de persistencia necesario para realizar operaciones en la base de datos.

CriteriaBuilder

CriteriaBuilder es una interfaz en JPA que se utiliza para construir consultas de manera programática y sin depender de lenguajes de consulta específicos como SQL. Permite construir consultas de manera más dinámica y tipada que las consultas basadas en cadenas.

Algunas de las funciones clave de **CriteriaBuilder** incluyen:

- **Creación de Consultas:** Permite construir consultas JPA utilizando una API de construcción de consultas.
- **Construcción de Predicados:** Facilita la creación de predicados (condiciones) que se

pueden combinar para formar consultas más complejas.

- **Operaciones Agregadas:** Proporciona métodos para realizar operaciones agregadas como contar, sumar, promediar, etc.

En el código, **CriteriaBuilder** se utiliza para construir las consultas (**queryLineaSublinea**) y para crear predicados que representan condiciones específicas que se aplican a la consulta.

CriteriaQuery y Root

- **CriteriaQuery<LineaSublinea>:** **CriteriaQuery** es una interfaz que define la estructura básica de una consulta JPA. **CriteriaQuery<LineaSublinea>** específicamente indica que la consulta está dirigida a la entidad **LineaSublinea**. Se utiliza para construir la parte estática y común de la consulta.
- **Root<LineaSublinea>:** **Root** representa la raíz de la entidad en la consulta. En este contexto, **lineaSublineaRoot** se utiliza para definir la tabla principal de la consulta y acceder a los atributos de la entidad **LineaSublinea**.

En resumen, **EntityManager** se encarga de la gestión de entidades y de la ejecución de consultas, mientras que **CriteriaBuilder** proporciona una interfaz para construir consultas de manera programática y **CriteriaQuery** junto con **Root** se utilizan para definir la estructura de la consulta. Estos componentes trabajan juntos para construir y ejecutar consultas JPA de manera eficiente.

Método **LineaSublineaList(int first, int pageSize, String sortField, String sortOrder, Map<String, Object> filters)**

Este método **LineaSublineaList** en el EJB (Enterprise JavaBean) realiza la consulta y la carga de datos relacionados con las líneas y sublíneas temáticas.

```
@Override
public List<LineaSublinea> LineaSublineaList(int first, int pageSize, String sortField,
String sortOrder, Map<String, Object> filters) {
    cb = em.getCriteriaBuilder();
    queryLineaSublinea = cb.createQuery(LineaSublinea.class);
    lineaSublineaRoot = queryLineaSublinea.from(LineaSublinea.class);
    queryLineaSublinea.select(lineaSublineaRoot);
    //Where
    queryLineaSublinea.where(cb.and(filtrosLineaSublinea(filters).toArray(new
    Predicate[]{})));
    //Order
```

```

        if (sortOrder.equals("A")) {
            queryLineaSublinea.orderBy(cb.asc(getLineaSublineaPath(sortField)));
        } else {
            queryLineaSublinea.orderBy(cb.desc(getLineaSublineaPath(sortField)));
        }
    }
    //pagination
    TypedQuery<LineaSublinea> tq = em.createQuery(queryLineaSublinea);
    if (pageSize >= 0) {
        tq.setMaxResults(pageSize);
    }
    if (first >= 0) {
        tq.setFirstResult(first);
    }
    return tq.getResultList();
}

```

Aquí está una explicación detallada de lo que hace:

1. Construcción de la Consulta:

```

cb = em.getCriteriaBuilder();
queryLineaSublinea = cb.createQuery(LineaSublinea.class); lineaSublineaRoot =
queryLineaSublinea.from(LineaSublinea.class); queryLineaSublinea.select(LineaSublineaRoot);

```

- Se obtiene un **CriteriaBuilder** para construir la consulta.
- Se crea una consulta (**queryLineaSublinea**) para la entidad **LineaSublinea**.
- Se crea una raíz (**lineaSublineaRoot**) que representa la tabla de la entidad.
- La consulta selecciona todos los campos de la entidad.

2. Filtros:

```

//Where
queryLineaSublinea.where(cb.and(filtrosLineaSublinea(filters).toArray(new Predicate[]{})));

```

- Se llama al método **filtrosLineaSublinea (filters)** para obtener un conjunto de predicados que representan los filtros a aplicar.
- Los predicados se combinan con una conjunción lógica "AND" y se agregan a la cláusula "WHERE" de la consulta.

3. Ordenamiento:

```
if (sortOrder.equals("A")) {
    queryLineaSublinea.orderBy(cb.asc(getLineaSublineaPath(sortField)));
} else {
    queryLineaSublinea.orderBy(cb.desc(getLineaSublineaPath(sortField)));
}
```

- Se determina la dirección del ordenamiento (ascendente o descendente) y se aplica al campo especificado en la consulta.

4. Paginación:

```
//pagination
TypedQuery<LineaSublinea> tq = em.createQuery(queryLineaSublinea); if (pageSize >= 0) {
    tq.setMaxResults(pageSize);
}
if (first >= 0) { tq.setFirstResult(first);
}
```

- Se crea un objeto **TypedQuery** a partir de la consulta.
- Se configura la paginación utilizando **setMaxResults** y **setFirstResult** según los parámetros **pageSize** y **first**.

Finalmente, la lista de **LineaSublinea** resultante se devuelve al llamador del método (**return tq**). Este método realiza una consulta dinámica basada en filtros, ordenamiento y paginación, lo que es común en aplicaciones web para mostrar datos de manera eficiente en una interfaz de usuario paginada y filtrada.

Método **GlobalFilter()** y **GlobalPath()**

El método **globalFilter()** se utiliza para aplicar un filtro global a una consulta, y el método **globalPath()** complementa esta funcionalidad proporcionando el camino (path) adecuado para un campo específico en una entidad particular. Estos métodos permiten la búsqueda global en cualquier pantalla de la aplicación, facilitando la construcción de predicados dinámicos en consultas JPA.

Método `globalFilter(Map<String, Object> filters, List<String> camposFiltro, String pathFilterMethod, List<Predicate> lista)`

```
/**  
 * Metodo para la busqueda global de cualquier pantalla  
 */  
private void globalFilter(Map<String, Object> filters, List<String> camposFiltro, String  
pathFilterMethod, List<Predicate> lista) {... 101 lines }
```

Este método realiza lo siguiente:

- **Inicialización:**

```
Path pathFilter = null;  
  
// Evitar duplicación de filtros para las mismas columnas  
Set<String> camposFiltrados = new HashSet<>();  
  
if (filters != null && !filters.isEmpty()) {  
    Predicate filterCondition = cb.conjunction();
```

- Se inicializa la variable **pathFilter** a **null**.
- Se define una estructura de datos llamada **Set** (en este caso, una implementación de **HashSet**), cuyo propósito es almacenar un conjunto de valores únicos. En este contexto, la variable **camposFiltrados** se utiliza para evitar la duplicación de filtros aplicados a las mismas columnas en una consulta dinámica.
- Se verifica si hay filtros en el mapa **filters**. Si hay, se inicia la construcción del predicado global.
- El método es genérico y funcionable para cualquier Entidad del proyecto

- **Procesamiento de Filtros:**

```
for (Map.Entry<String, Object> filter : filters.entrySet()) {
```

- Se itera sobre cada entrada del **Map<String, Object> filters**, donde:
 - **filter.getKey()** es el nombre del campo.
 - **filter.getValue()** es el valor de búsqueda.
- Se ignoran los valores **null** o vacíos.

- **Identificación de "globalFilter":**

```
if (filter.getKey().equals("globalFilter")) {
```

- Se verifica si la clave es "globalFilter".
- Se itera sobre la lista **camposFiltro** para aplicar filtros en cada campo permitido.

- **Procesamiento de filtros personalizados:**

```
String tipo = "";
String campo = camposFiltro.get(i);

// custom
Map<String, String> items = null;
String[] splitCustom = campo.split("\\|");
if (splitCustom.length

    > 1) {
        tipo = "CUSTOM";
        String[] splitCustomComa = splitCustom[1].split(",");
        if (splitCustomComa.length > 0) {
            items = new TreeMap<>();
            for (String itemValues : splitCustomComa) {
                String[] splitCustomDot = itemValues.split(":");
                items.put(splitCustomDot[0], splitCustomDot[1]);
            }
        }
        pathFilter = globalPath(pathFilterMethod, splitCustom[0]);
    }
}
```

- **TreeMap<String, String>:** Se usa para almacenar equivalencias entre valores legibles por humanos y los valores reales en la base de datos.
 - Ejemplo: { "1": "Activo", "0": "Inactivo" }
- **split("\\|"):** Divide la cadena en dos partes (**campo**|**mapa_de_valores**), donde la segunda parte contiene los valores personalizados.
- **globalPath(pathFilterMethod, splitCustom[0]):** Método que obtiene la referencia (**Path<T>**) a un atributo específico de la entidad.

- **Identificación de tipos BOOLEAN y DATE:**

```
if (tipo.isEmpty ()
    ) {
```

```
String[] split = campo.split("\\.");
if (split.length > 1) {
    campo = split[0];
    tipo = split[1];
}
if (tipo.contains("BOOLEAN") || tipo.contains("DATE")) {
    pathFilter = globalPath(pathFilterMethod, campo);
} else {
    pathFilter = globalPath(pathFilterMethod, camposFiltro.get(i));
}
}
```

- Se verifica si el campo contiene un sufijo (**campo.tipo**).
- **BOOLEAN** y **DATE** requieren un manejo especial en la consulta, por lo que se extrae el **Path<T>** correspondiente.
- **globalPath(pathFilterMethod, campo)**: Obtiene la ruta del campo dentro de la entidad JPA.
- **Construcción de la condición de filtro:**

a) BOOLEAN

```
case "BOOLEAN":
String booleanSearch = filter.getValue().toString().toLowerCase(); if (i == 0) {
filterCondition = cb.equal(pathFilter, (booleanSearch.contains("si") ? Boolean.TRUE :
booleanSearch.contains("no") ? Boolean.FALSE : null));
} else {
filterCondition = cb.or(filterCondition, cb.equal(pathFilter, (booleanSearch.contains("si") ?
Boolean.TRUE : booleanSearch.contains("no") ? Boolean.FALSE : null)));
}
break;
```

- **cb.equal(pathFilter, valor)**: Genera una condición **WHERE campo = valor**.
- Se convierte **"Si"** y **"No"** en **true** o **false** antes de aplicar la condición.

b) DATE

```
case "DATE":
String dateSearch = filter.getValue().toString().toLowerCase(); if (i == 0) {
filterCondition = cb.like(cb.function("TO_CHAR",
String.class, pathFilter, cb.literal("dd/MM/yyyy")), "%" + dateSearch + "%");
} else {
filterCondition = cb.or(filterCondition, cb.like(cb.function("TO_CHAR", String.class,
pathFilter, cb.literal("dd/MM/yyyy")), "%" + dateSearch + "%"));
}
break;
```

- **cb.function("TO_CHAR", String.class, pathFilter, cb.literal("dd/MM/yyyy")):** Convierte la fecha en una cadena con formato **dd/MM/yyyy**.
- **cb.like(...):** Se usa **LIKE** para permitir búsquedas parciales en fechas ("15/03" buscará 15/03/2025).

c) CUSTOM

```
case "CUSTOM":
String customSearch = filter.getValue().toString().toLowerCase(); String value = null;
if (items != null && !items.isEmpty()) {
for (Map.Entry<String, String> entry : items.entrySet()) {
if (customSearch.contains(entry.getValue().toLowerCase())) { value = entry.getKey();
}
}
}
if (value != null) { if (i == 0) {
filterCondition = cb.equal(pathFilter, value);
} else {
filterCondition = cb.or(filterCondition, cb.equal(pathFilter, value));
}
```

- Convierte los valores personalizados ("**Activo**" -> "**1**") antes de filtrar.
- **cb.equal(pathFilter, value):** Filtra por la clave correspondiente.

d) LIKE

```
default:

if (i == 0) {

filterCondition = cb.like(cb.lower(pathFilter), "%" +
filter.getValue().toString().toLowerCase() + "%");

} else {

filterCondition = cb.or(filterCondition, cb.like(cb.lower(pathFilter), "%" +
filter.getValue().toString().toLowerCase() + "%"));

}
```

- **cb.lower(pathFilter):** Convierte el campo a minúsculas para hacer la búsqueda insensible a mayúsculas.
- **"%" + filtro + "%"** :
 - Se agrega % antes y después del valor para permitir búsquedas parciales.

- **Agregar la condición final:**

```
lista.add(filterCondition);
```

- **lista.add(filterCondition):** Agrega el Predicate a la lista de condiciones que se usarán en la consulta CriteriaQuery.

El método **globalFilter** implementa una estrategia flexible y dinámica para la construcción de consultas en **JPA** utilizando **CriteriaBuilder**, lo que permite aplicar múltiples filtros sin necesidad de definir consultas estáticas. Su diseño modular soporta diferentes tipos de filtros (**BOOLEAN**, **DATE**, **CUSTOM** y búsquedas de texto con **LIKE**), proporcionando versatilidad en la manipulación de datos. Además, utiliza **cb.conjunction()** para generar condiciones dinámicas, asegurando que cada filtro se agregue correctamente según su tipo y formato esperado.

Uno de los aspectos clave del método es la capacidad de interpretar filtros personalizados mediante **CUSTOM**, lo que permite mapear valores de entrada en etiquetas más intuitivas para el usuario. Asimismo, el uso de **cb.function("TO_CHAR")** en el caso de fechas facilita la búsqueda de fechas en un formato legible sin necesidad de convertirlas manualmente en la aplicación. Esto optimiza las consultas sin comprometer la eficiencia, ya que se ejecutan directamente en la base de datos.

Finalmente, el método sigue un enfoque seguro y escalable, asegurando que los filtros nulos o vacíos no afecten la consulta final. El uso de **Path<T>** permite acceder dinámicamente a los atributos de la entidad sin necesidad de **hardcodear** nombres de columnas, mejorando la mantenibilidad del código. Con su diseño actual, **globalFilter** es altamente adaptable para cualquier necesidad de filtrado en aplicaciones empresariales, permitiendo una experiencia de búsqueda eficiente y sin pérdida de rendimiento.

Método globalPath(String pathMethod, String field)

```
private Path<?> globalPath(String pathMethod, String field) {  
    switch (pathMethod) {  
  
        case "getLineaSublineaPath":  
  
            return getLineaSublineaPath(field);  
        default:
```

Este método realiza lo siguiente:

1. Selección del Método de Path:

```
switch (pathMethod) {  
  
    case "getLineaSublineaPath":  
  
        return getLineaSublineaPath(field)  
    default:
```

- Se selecciona el método de path basado en el parámetro **pathMethod**.
- Se pueden agregar tantos métodos Path como sea necesarios. Lo ideal es que sea uno por Entidad.

2. Retorno del Path:

- Si el **pathMethod** es conocido (en este caso, "getLineaSublineaPath"), se llama al método correspondiente (**getLineaSublineaPath**) con el campo proporcionado y se retorna el path resultante.
- Si el **pathMethod** no es conocido, se retorna **null**.

En resumen, estos métodos proporcionan una forma modular y dinámica de aplicar filtros globales en consultas JPA, facilitando la construcción de consultas dinámicas y adaptativas en base a los filtros proporcionado.

Método filtrosLineaSublinea (Map<String, Object> filters)

El método **filtrosLineaSublinea()** se encarga de construir una lista de predicados (**Predicate**) que representan los filtros aplicados a la consulta de líneas y sublíneas temáticas.

```
private List<Predicate> filtrosLineaSublinea(Map<String, Object> filters) {  
    List<Predicate> lista = new ArrayList<>();  
  
    List<String> camposFiltro = new ArrayList<>();  
    Path pathFilter;  
    camposFiltro.add("proyectoId.proyectoId");  
    camposFiltro.add("proyectoId");  
    camposFiltro.add("lineaSublineaId");  
  
    camposFiltro.add("lineaId.descripcionValorDefinicion");  
    camposFiltro.add("sublineaId.descripcionValorDefinicion");  
  
    //Filtro por Proyecto ID  
  
    Object proyectoId = filters.get("proyectoId");  
    if (proyectoId != null) {
```

Ahora, veamos en detalle este método:

1. Inicialización:

```
List<Predicate> lista = new ArrayList<>();  
List<String> camposFiltro = new ArrayList<>();
```

- Se inicializa una lista de predicados (**lista**) que se utilizará para almacenar los filtros.
- Se inicializa una lista de nombres de campos de filtro (**camposFiltro**).

2. Definición de Campos de Filtro global:

```
camposFiltro.add("proyectoId.proyectoId");  
camposFiltro.add("proyectoId");  
camposFiltro.add("lineaSublineaId");  
camposFiltro.add("lineaId.descripcionValorDefinicion");  
camposFiltro.add("sublineaId.descripcionValorDefinicion");
```

3. Manejo de Filtros Avanzados:

```
//Filtro por Proyecto ID  
  
Object proyectoId = filters.get("proyectoId");  
if (proyectoId != null) {
```

- Verificación y aplicación de filtros avanzados
- Fundamental que cada filtro que se agregue se encuentre en el Path indicado, para que de esta manera el método sepa dónde y contra que campo buscar

4. Filtro Global:

```
//Filtro Global  
globalFilter(filters, camposFiltro, "getLineaSublineaPath", lista);
```

- Se llama a un método llamado **globalFilter** que aplica un filtro global en base a una lista de campos y utiliza el método **getLineaSublineaPath** para obtener los paths correspondientes.

5. Retorno de la Lista de Predicados:

```
return lista;
```

Método getLineaSublineaPath(String field)

El método **getLineaSublineaPath(String field)** devuelve un objeto **Path** que representa el camino (path) a un atributo específico de la entidad **LineaSublinea** que se utilizará en la construcción de predicados.

```
private Path<?> getLineaSublineaPath(String field) {  
  
    //Join  
  
    Join<LineaSublinea, Proyecto> proyectoId = lineaSublineaRoot.join(LineaSublinea_.proyectoId,  
JoinType.LEFT);  
  
    Join<LineaSublinea, ValorDefinicion> lineaId = lineaSublineaRoot.join(LineaSublinea_.lineaId,  
JoinType.LEFT);  
  
    Join<LineaSublinea, ValorDefinicion> sublineaId =  
lineaSublineaRoot.join(LineaSublinea_.sublineaId, JoinType.LEFT);  
  
    Path<?> path = null;  
    if (null == field) {  
  
        path = lineaSublineaRoot.get(LineaSublinea_.lineaSublineaId);  
  
    } else {  
  
        switch (field) {  
  
            case "lineaSublineaId":  
  
                path = lineaSublineaRoot.get(LineaSublinea_.lineaSublineaId);  
                break;  
  
            case "proyectoId.proyectoId":
```

Ahora, veamos en detalle este método:

1. Asignación de Path en Base al Campo Especificado:

```
Path<?> path = null;
switch (field) {

    case "lineaSublineaId":

        path = lineaSublineaRoot.get(LineaSublinea_.lineaSublineaId);
        break;
```

- Cada case debe corresponder con el nombre del campo y/o búsqueda que se va a realizar
- Al utilizar jpa, se debe identificar cuando se hace referencia a un campo o a un objeto de la Entidad

2. Retorno del Path:

```
return path;
```

En resumen, estos métodos están diseñados para facilitar la construcción de predicados y paths necesarios para aplicar filtros en una consulta JPA (Java Persistence API).

El método **filtrosLineaSublinea()** maneja tanto filtros simples como filtros avanzados, mientras que **getLineaSublineaPath()** devuelve el **Path** correspondiente a un campo específico de la entidad **LineaSublinea**. Estos métodos son utilizados en la construcción de consultas dinámicas en base a los filtros proporcionados.

Controlador (ItextReporteProyectosSessController)

Este controlador gestiona la generación del reporte PDF correspondiente a la ficha técnica del proyecto. Desde esta clase se construye el documento, incorporando los distintos bloques de información que describen el proyecto, incluyendo la nueva sección de líneas temáticas y sublíneas temáticas asociadas. Esta información se obtiene mediante servicios y se inserta dinámicamente en el diseño del reporte para garantizar una visualización clara y estructurada de los datos temáticos registrados.

Dentro de este controlador el método **obtenerLineasSublineasTematicas()** juega un papel clave proporcionando la información necesaria para visualizar las diferentes líneas y sublíneas temáticas.


```

    public String obtenerLineasSublineasTematicas() throws Exception {
        lineasSublineasTematicasProyecto
        =
        proyectosServiceFacadeLocal.obtenerLineasSublineasTematicas(proyectoActual);
        String html = "";
        if (lineasSublineasTematicasProyecto != null
            && !lineasSublineasTematicasProyecto.isEmpty()) {
            reporteHtmlUtil = new ReporteHtmlUtil();
            String titulo = reporteHtmlUtil.crearHtmlHeader("Líneas y Sublíneas Temáticas");
            StringBuilder contenido = new StringBuilder();
            String lineaAnterior = null;
            contenido.append("<ul>"); // Lista principal
            for (LineaSublinea lineaSublinea : lineasSublineasTematicasProyecto) {
                String lineaActual
                =
                lineaSublinea.getLineaId().getDescripcionValorDefinicion();
                if (!lineaActual.equals(lineaAnterior)) {
                    if (lineaAnterior != null) {
                        contenido.append("</ul></li>"); // Cierra sublista de sublíneas
                    }
                    contenido.append("<br /> ");
                    contenido.append("<li><strong>")
                        .append(reporteHtmlUtil.codificar(lineaActual))
                        .append("</strong><ul>")
                        .append("<br /> "); // Nueva línea con sublista lineaAnterior =
                    lineaActual;
                }
                contenido.append("<li>")

                .append(reporteHtmlUtil.codificar(lineaSublinea.getSublineaId().getDescripcionValorDefinicion()))
                .append("</li>");
            }
            if (lineaAnterior != null) {
                contenido.append("</ul></li>"); // Cierra la última sublista
            }
            contenido.append("</ul>"); // Cierra la lista principal html = titulo +
            contenido.toString();
        }
        return html;
    }

```

A continuación, una explicación más detallada:

Consulta de datos:

- Se obtiene la lista **lineasSublineasTematicasProyecto** invocando el método **obtenerLineasSublineasTematicas(proyectoActual)** del **proyectosServiceFacadeLocal**.
- Se valida que la lista no sea nula ni esté vacía antes de continuar.

Inicialización de utilidades:

- Se instancia **ReporteHtmlUtil**, una clase de utilidad para generación segura de HTML y codificación de contenido.

Generación del encabezado:

- Se crea un título HTML mediante **crearHtmlHeader("Líneas y Sublíneas Temáticas")**.

Construcción del contenido HTML:

- Se inicia una lista HTML principal (****).
- Se recorre la lista de **LineaSublinea**:
 - Se compara la línea temática actual con la anterior.
 - Si es una nueva línea:
 - Se cierra la sublista anterior (si aplica).
 - Se inicia un nuevo **** con la descripción de la línea en negrita, seguido de una sublista **** para sus sublíneas.
 - Se agrega cada sublínea como un elemento **** dentro de la sublista correspondiente.
- Al finalizar el recorrido:
 - Se cierran las etiquetas de la última sublista y de la lista principal.

Retorno del contenido:

- Se concatena el título con el cuerpo HTML construido y se retorna como resultado.

d. Soporte de Asignación de Evaluadores de Proyectos en la Convocatoria 963-2025 Orquídeas

Solicitud de requerimiento No. 158979

Descripción y alcance del Requerimiento

Brindar soporte operativo y técnico para el proceso de asignación de evaluadores a los proyectos postulados en la Convocatoria 963-2025 Conv "Orquídeas: Mujeres en Inteligencia Artificial, Ciencias y Tecnologías Cuánticas 2025".

Desarrollo e Implementación

El requerimiento consistió en **brindar soporte operativo y técnico para el proceso de asignación de evaluadores** a los proyectos postulados en la **Convocatoria 963-2025 “Orquídeas: Mujeres en Inteligencia Artificial, Ciencias y Tecnologías Cuánticas 2025”** dentro del sistema SIGP.

Como parte de la solución, se realizó el **cargue manual de evaluadores** en el sistema, garantizando que los proyectos postulados en la convocatoria contaran con los evaluadores asignados para dar continuidad al proceso de revisión y evaluación.

Este proceso incluyó las siguientes actividades principales:

- Verificación de la información de los evaluadores suministrados para la convocatoria.
- Cargue manual de los evaluadores en el módulo correspondiente del sistema SIGP.
- Validación de la correcta asociación de evaluadores a los proyectos postulados.
- Pruebas de verificación para confirmar que la asignación quedara registrada y operativa en el sistema.

Con esta implementación se aseguró la continuidad del proceso de evaluación de la convocatoria, garantizando la trazabilidad de los evaluadores asignados y permitiendo que los usuarios funcionales iniciaran las actividades de revisión y calificación.

Soporte para Asignar Manual Proyectos a Convocatoria 963-2025 Orquideas

 **De** Leonardo Suarez <lasb@softwarebuilders.co>
Destinatario Harold Alexander Méndez Guevara <hamendez@minciencias.gov.co>
Cc César Riaño <cesar_riano@softwarebuilders.co>, José Ignacio Español Perez <jiespanol@minciencias.gov.co>, Jemeneses <jemeneses@minciencias.gov.co>
Fecha 2025-05-20 16:16

Señores
Oficina OTSI

Cordial saludo,

Me permito informar que ya se realizó la asignación de los evaluadores a los proyectos correspondientes a la convocatoria No. 963 de 2025 "Orquideas Mujeres en Inteligencia Artificial, Ciencia y Tecnologías Cuánticas" en el sistema SIGP.

Quedamos atentos a cualquier comentario adicional que permita continuar con el desarrollo del proceso de evaluación.

--
Cordialmente,

Ing. Leonardo Suárez Baldovino
Software Builders Ltda.
 lasb@softwarebuilders.co |  www.softwarebuilders.co
 Carrera 700 # 51 - 57

El 2025-05-19 18:09, Harold Alexander Méndez Guevara escribió:

e. Implementar Separación Técnica de Funciones en el Módulo de Administración del SIGP

Solicitud de requerimiento No. 157816

Descripción y alcance del Requerimiento

Se solicita la implementación de una separación técnica de funciones dentro del módulo de administración del SIGP, creando dos roles distintos con permisos claramente diferenciados:

- Administrador de gestión de usuarios: responsable de crear usuarios, asignar permisos, activar o desactivar cuentas, realizar traslados de dependencia, y gestionar la asignación de usuarios a convocatorias y proyectos.
- Administrador de configuración: encargado de la parametrización de convocatorias, la generación de reportes del sistema, así como del soporte funcional de la aplicación. Además, deberá realizar la verificación y seguimiento de las funcionalidades específicas asociadas a los formularios del SIGP, incluyendo:
 - Formulario de Entidades
 - Formulario de Beneficios Tributarios

- Formulario de Reconocimiento de Actores
- Formulario de Programas y Proyectos

Este ajuste debe aplicarse como una división técnica dentro del módulo actual de administración, con permisos separados por tipo de función.

El crecimiento funcional del SIGP ha evidenciado la necesidad de mayor especialización y segmentación de responsabilidades dentro del módulo de administración. Y es que actualmente, un único rol con amplios privilegios representa una fuente de riesgo y dificulta el control de acceso.

Separar los roles permite distribuir tareas con mayor claridad. Por un lado, la administración de usuarios requiere un enfoque organizativo y de control de accesos; por otro, la configuración y verificación de componentes funcionales necesita perfiles con conocimiento técnico y del comportamiento del sistema. Además, el soporte funcional a los formularios y la validación de sus flujos son tareas que deben mantenerse bajo responsabilidad de un rol especializado.

Esto mejora la trazabilidad de acciones, mitiga errores operativos, y permite una gestión más alineada con buenas prácticas en entornos institucionales que manejan información crítica y procesos estratégicos.

Desarrollo e Implementación

Se presenta un informe técnico sobre el desarrollo realizado en el módulo de Gestión de Usuarios de la aplicación SIGP WEB, orientado a implementar una funcionalidad que permita la asignación de funcionarios a convocatorias (y, de manera opcional, a proyectos específicos) directamente desde el menú “Usuarios”, sin necesidad de acceder al menú de convocatorias.

El objetivo principal de esta mejora es optimizar el flujo de trabajo y delegar esta tarea a usuarios con permisos limitados, evitando exponer la configuración completa de las convocatorias y manteniendo la seguridad del sistema.

Alcance del Desarrollo

Los requerimientos cubiertos en este desarrollo incluyen:

- 1. Incorporación de un botón “Asignar a convocatoria” en la pantalla de detalle del funcionario:**

- Desde la vista de detalle de un funcionario, el nuevo botón permite redirigir a la pantalla funcionarios Evaluación.
- Esta pantalla permite asignar al funcionario a una convocatoria para su evaluación y, opcionalmente, a uno o varios proyectos específicos.

2. Nueva funcionalidad “Funcionarios Evaluación”:

- Visualización de datos del usuario: muestra la información relevante del funcionario seleccionado.
- Tabla de registros asociados: tabla en modo lazy loading para consultar las convocatorias y proyectos asociados al funcionario.

3. Formulario de asignación:

- Campo Convocatoria con búsqueda tipo autocomplete.
- Campo Proyectos (opcional) con selección múltiple mediante autocomplete múltiple.
- Gestión de proyectos asignados: botón dentro de la tabla que abre un diálogo mostrando los proyectos asociados a la convocatoria seleccionada.

4. Tecnologías Utilizadas

El desarrollo de estas funcionalidades se basó en las siguientes tecnologías:

- **Java EE con Enterprise JavaBeans (EJB)** para la lógica de negocio.
- **JSF (JavaServer Faces) y PrimeFaces** para la interfaz de usuario.
- **JPA (Criteria API)** para la consulta y manipulación de datos en la base de datos.
- **Carga diferida (lazy loading)** para optimizar la visualización de grandes volúmenes de datos en tablas.

Gestión de Usuarios

Componentes

Para implementar el botón que redirige a la funcionalidad Funcionarios Evaluación, se emplearon los siguientes componentes:

- **Vista (View):** Representada por el archivo usuarios.xhtml, el cual define la interfaz de usuario en formato XHTML. Este archivo contiene los elementos gráficos y enlaces necesarios para interactuar con el usuario final.
- **Controlador (Conversation Controller):** Implementado en la clase UsuarioConvController, responsable de manejar la lógica de la capa de presentación. Este controlador administra el flujo de la conversación, gestiona los eventos generados en la vista y coordina la comunicación con la capa de negocio.

Codificación

Interfaz de usuario (usuarios.xhtml)

Botón "Asignar a Convocatoria":

Este componente permite al usuario acceder a la funcionalidad Funcionarios Evaluación directamente desde la pantalla de gestión de usuarios.

```
<p:commandLink styleClass="btn btn-xs btn-primary sbTooltip" id="asignarConvocatoria"
title="#{msg['sb.menu.gestion.administracion.usuario.asignarConvocatoria.button']}"
immediate="true"
process="@this" ajax="false"
rendered="#{SeguridadSessController.mapaMenu.containsKey('GEST6131')}"
actionListener="#{usuarioConvController.asignarFuncionarioConvocatoria(usuario)}"
>
    <i class="fa fa-user-plus"></i>
</p:commandLink>
```

A continuación, una explicación más detallada:

- **styleClass="btn btn-xs btn-primary sbTooltip"**: Aplica estilos **CSS** para mostrar el botón con formato pequeño, color primario y soporte para mostrar un **tooltip**.
- **id="asignarConvocatoria"**: Identificador único del componente dentro de la vista, utilizado para referencia en el manejo de eventos o en pruebas.

- **title="#{msg['sb.menu.gestion.administracion.usuario.asignarConvocatoria.button']}":** Texto que se muestra como **tooltip**, obtenido del archivo de internacionalización de mensajes.
- **immediate="true":** Indica que el evento de acción se procesa en la fase de **Apply Request Values** del ciclo **JSF**, omitiendo validaciones y conversión de datos previas.
- **process="@this":** Define que únicamente este componente se procesa en el ciclo de JSF, ignorando otros elementos de la vista.
- **ajax="false":** Especifica que la acción no se ejecutará mediante **AJAX**, sino que provocará una recarga completa de la página o navegación a otra vista.
- **rendered="#{SeguridadSessController.mapaMenu.containsKey('GEST613 1')}":** Condición que controla la visibilidad del botón; solo se renderiza si el usuario tiene permisos asociados a la clave 'GEST6131'.
- **actionListener="#{usuarioConvController.asignarFuncionarioConvocatoria(usuario)}":** Método del **controller** que se ejecuta al hacer clic, encargado de iniciar el proceso de asignación del funcionario a una convocatoria.

Controlador (UsuarioConvController.java)

Método “asignarFuncionarioConvocatoria”:

El método **asignarFuncionarioConvocatoria** se ejecuta cuando el usuario selecciona la opción Asignar a Convocatoria en la interfaz gráfica.

```
public void asignarFuncionarioConvocatoria(UsuarioDTO usuarioDTO) {
    setUtilSessionUser(usuarioDTO);
    this.redirectPage("app/administracion/funcionariosConvocatorias.xhtml");
}

private void setUtilSessionUser(UsuarioDTO usuarioDTO) {
    Map<String, Object> mapReferencias = new HashMap<>();
    mapReferencias.put("usuarioConvocatoriaSelected", usuarioDTO);
    usuarioViewController.setSelected(usuarioDTO);
    colcienciasSessController.setUtilObject(mapReferencias);
}
```

El método **asignarFuncionarioConvocatoria** se ejecuta cuando el usuario selecciona la opción Asignar a Convocatoria en la interfaz gráfica. Este método recibe como parámetro un objeto **UsuarioDTO**, el cual representa al usuario que será asignado.

Internamente, invoca a **setUtilSessionUser**, encargado de preparar el contexto de sesión para la nueva vista. En este proceso, se construye un **Map<String, Object>** que asocia la clave

"**usuarioConvocatoriaSelected**" con el **UsuarioDTO** recibido, se establece este usuario como el seleccionado en **usuarioViewController** y se registra dicho mapa en **colcienciasSessController** mediante **setUtilObject**.

Una vez configurada la información necesaria en la sesión, el método principal realiza una redirección hacia la página **funcionariosConvocatorias.xhtml**, ubicada en el módulo de administración, permitiendo así que la nueva funcionalidad acceda al usuario previamente seleccionado y a los datos preparados en la sesión para su procesamiento.

Funcionarios Evaluación

Componentes

Para generar el formulario que permite adicionar y/o borrar las convocatorias para evaluar de un funcionario, se utilizan los siguientes componentes en la aplicación:

- **Vista (View):** Representada por el archivo **funcionariosConvocatorias.xhtml**, que define la interfaz de usuario en formato XHTML, incluyendo la estructura del formulario, la tabla de registros y los controles de asignación.
- **Controladores (View Controller y Conversation Controller):** Implementados en las clases **FuncionarioConvocatoriaViewController** y **FuncionarioConvocatoriaConvController**, responsables de gestionar la lógica de la capa de presentación, manejar eventos de usuario y coordinar la comunicación con la capa de negocio.
- **Servicio EJB (EJB Service):** Implementado en la clase **ConvocatoriaFuncionarioServiceBean**, un componente Enterprise JavaBean encargado de la lógica de negocio relacionada con la gestión de convocatorias asignadas a funcionarios para su evaluación.
- **Base de Datos (Database):** Representada por la entidad **ConvocatoriaFuncionarioDTO**, un Data Transfer Object que mapea la tabla **SGP_CONVOCATORIAS_FUNCIONARIOS**, utilizada para almacenar y manipular las asignaciones entre funcionarios y convocatorias.
- **JPA (Java Persistence API):** Uso de la interfaz **CriteriaBuilder** para construir consultas

dinámicas y parametrizadas hacia la base de datos de forma eficiente y segura.

- **PrimeFaces:** Utilización de la clase LazyDataModel para la carga diferida (lazy loading) de datos en tablas, optimizando el rendimiento en la visualización de grandes volúmenes de información.

Codificación

Interfaz de usuario (funcionariosConvocatorias.xhtml)

Tabla de Datos

Esta tabla permite al usuario visualizar las convocatorias asignadas al funcionario seleccionado en la gestión de usuarios. Incluye información relevante de cada convocatoria y una funcionalidad para consultar los proyectos asociados a la misma, en caso de existir.

```
<p:dataTable id="#{funcionarioConvocatoriaViewController.modulo}Table"
              var = "convFuncionario"
              value="#{funcionarioConvocatoriaViewController.modelo}"
              rowKey="#{convFuncionario.convocatoriaFuncionarioId}"
              selection="#{funcionarioConvocatoriaConvController.selected}"
              selectionMode="single"
              paginator="true" rows="10" lazy="true" reflow="true"
              sortBy="#{convFuncionario.usuarioId.descripcionUsuario}"
              sortOrder="ASCENDING"
              currentPageReportTemplate="Total de registros: {totalRecords}"
              paginatorTemplate="{CurrentPageReport} {FirstPageLink}
{PreviousPageLink} {PageLinks} {NextPageLink} {LastPageLink} {RowsPerPageDropdown}"
              paginatorPosition="bottom"
              emptyMessage="#{msg['sb.datatable.registro.empty']}"
              widgetVar="items#{funcionarioConvocatoriaViewController.modulo}">
```

A continuación, una explicación más detallada:

- **id="#{funcionarioConvocatoriaViewController.modulo}Table"**: Identificador único de la tabla, construido dinámicamente a partir del módulo actual.
- **var="convFuncionario"**: Variable de iteración que representa cada fila de la tabla, permitiendo acceder a sus atributos.
- **value="#{funcionarioConvocatoriaViewController.modelo}"**: Fuente de datos para la tabla, enlazada al modelo gestionado por el **View Controller**.
- **rowKey="#{convFuncionario.convocatoriaFuncionarioId}"**: Llave única que identifica cada registro, necesaria para la selección y actualización de datos.

- **selection="#{funcionarioConvocatoriaConvController.selected}"**: Objeto que almacena el registro actualmente seleccionado en la tabla.
- **selectionMode="single"**: Configura la tabla para permitir la selección de una sola fila a la vez.
- **paginator="true"**: Habilita la paginación en la tabla.
- **rows="10"**: Define el número de filas que se muestran por página.
- **lazy="true"**: Activa la carga diferida (**lazy loading**) para optimizar el rendimiento con grandes volúmenes de datos.
- **reflow="true"**: Ajusta automáticamente el diseño de la tabla para adaptarse a pantallas pequeñas o dispositivos móviles.
- **sortBy="#{convFuncionario.usuarioId.descripcionUsuario}"**: Campo por el cual se ordena inicialmente la tabla.
- **sortOrder="ASCENDING"**: Orden ascendente predeterminado en la tabla.
- **currentPageReportTemplate="Total de registros: {totalRecords}"**: Texto que se muestra para informar el número total de registros.
- **paginatorTemplate="{CurrentPageReport} {FirstPageLink} {PreviousPageLink} {PageLinks} {NextPageLink} {LastPageLink} {RowsPerPageDropdown}"**: Plantilla que define el diseño y elementos visibles en la paginación.
- **paginatorPosition="bottom"**: Ubica la barra de paginación en la parte inferior de la tabla.
- **emptyMessage="#{msg['sb.datatable.registro.empty']}"**: Mensaje mostrado cuando no hay registros que mostrar.
- **widgetVar="items#{funcionarioConvocatoriaViewController.modulo}"**: Nombre de la variable de widget de **PrimeFaces**, utilizada para manipular la tabla mediante **JavaScript**.

Botón "Proyectos Asociados"

Este botón permite visualizar los proyectos asociados a la convocatoria en la que está asignado un usuario. Se encuentra dentro de la tabla principal y, al hacer clic, abre un diálogo modal que muestra dicha información.

```
<p:commandLink id="btnProyectosAsignados"
```

```
oncomplete="PF('projectsTable').clearFilters(); PF('projectsTable').unselectAllRows();
$('#dlgProyectosAsignados').modal();"
styleClass="btn btn-primary" process="@this"
actionListener="#{funcionarioConvocatoriaConvController.cargarProyectosAsignados(convFuncionario io)}"
update="formProyectosAsignados, formProyectosAsignados:proyectosAsignadosTable">
<i class="fa fa-tasks"></i>
#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.label']}
</p:commandLink>
```

A continuación, una explicación más detallada:

- **id="btnProyectosAsignados"**: Identificador único del botón dentro de la vista, usado para referencia y manipulación.
- **oncomplete="PF('projectsTable').clearFilters(); PF('projectsTable').unselectAllRows(); \$('#dlgProyectosAsignados').modal();"**
 - **PF('projectsTable').clearFilters();** Limpia los filtros aplicados a la tabla de proyectos.
 - **PF('projectsTable').unselectAllRows();** Deselecciona cualquier fila previamente seleccionada en la tabla de proyectos.
 - **\$('#dlgProyectosAsignados').modal();**: Muestra el diálogo modal con la información de los proyectos asociados.
- **styleClass="btn btn-primary"**: Aplica el estilo CSS de botón principal para resaltar la acción.
- **process="@this"**: Procesa únicamente este componente durante el ciclo JSF, ignorando otros elementos de la vista.
- **actionListener="#{funcionarioConvocatoriaConvController.cargarProyectosAsignados(convFuncionario)}"**: Llama al método del **controller** encargado de cargar la lista de proyectos asociados a la convocatoria seleccionada.
- **update="formProyectosAsignados, formProyectosAsignados:proyectosAsignadosTable"**: Actualiza el formulario y la tabla de proyectos dentro del diálogo, asegurando que la información esté actualizada antes de mostrarse.
- **Contenido interno:**
 - **<i class="fa fa-tasks"></i>**: Ícono representativo de tareas/proyectos.
 - **#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.asignados.label']}**: Texto del botón, obtenido desde el archivo de internacionalización.

Tabla de Proyectos Asociados

Esta tabla permite al usuario visualizar los proyectos seleccionados para una convocatoria asignada al funcionario que ha sido seleccionado en la gestión de usuarios. Muestra información relevante de cada proyecto asociado.

```
<p:dataTable id="proyectosAsignadosTable" var="convFuncionarioPry"
  value="#{funcionarioConvocatoriaViewController.listaConvocatoriaFuncionarioProyectos}"
  rowKey="#{convFuncionarioPry.convocatoriaFuncionarioProyectoId}" selectionMode="single"
  paginator="true" rows="5"
  lazy="false" reflow="true" sortBy="#{convFuncionarioPry.proyectoId.proyectoId}"
  sortOrder="ASCENDING"
  currentPageReportTemplate="Total de registros: {totalRecords}"
  paginatorTemplate="{CurrentPageReport} {FirstPageLink} {PreviousPageLink}
  {PageLinks} {NextPageLink} {LastPageLink} {RowsPerPageDropdown}" paginatorPosition="bottom"
  emptyMessage="#{msg['sb.datatable.registro.empty']}" widgetVar="projectsTable"
  filteredValue="#{funcionarioConvocatoriaViewController.listaConvocatoriaFuncionarioProye
ctosFiltro}">
```

A continuación, una explicación más detallada:

- **id="proyectosAsignadosTable"**: Identificador único de la tabla que permite su referencia en la vista o desde **JavaScript**.
- **var="convFuncionarioPry"**: Variable de iteración que representa cada fila de la tabla, usada para acceder a los atributos del proyecto.
- **value="#{funcionarioConvocatoriaViewController.listaConvocatoriaFuncionarioProyectos}"**: Lista de proyectos asociados a la convocatoria, obtenida desde el **View Controller**.
- **rowKey="#{convFuncionarioPry.convocatoriaFuncionarioProyectoId}"**: Llave única para identificar cada proyecto en la tabla, necesaria para la selección de registros.
- **selectionMode="single"**: Permite la selección de una sola fila a la vez.
- **paginator="true"**: Habilita la paginación en la tabla.
- **rows="5"**: Número de filas visibles por página.
- **lazy="false"**: La carga de datos no es diferida (**lazy loading**); todos los datos están disponibles desde el inicio.
- **reflow="true"**: Ajusta el diseño de la tabla para adaptarse a pantallas pequeñas o dispositivos móviles.
- **sortBy="#{convFuncionarioPry.proyectoId.proyectoId}"**: Campo por el que se ordena inicialmente la tabla.
- **sortOrder="ASCENDING"**: Orden ascendente predeterminado.
- **currentPageReportTemplate="Total de registros: {totalRecords}"**: Muestra el total

de registros visibles en la tabla.

- **paginatorTemplate="{CurrentPageReport} {FirstPageLink} {PreviousPageLink} {PageLinks} {NextPageLink} {LastPageLink} {RowsPerPageDropdown}"**: Define el formato y elementos visibles de la paginación.
- **paginatorPosition="bottom"**: Ubica el paginador en la parte inferior de la tabla.
- **emptyMessage="#{msg['sb.datatable.registro.empty']}"**: Mensaje mostrado cuando no hay registros disponibles.
- **widgetVar="projectsTable"**: Nombre de la variable de widget de **PrimeFaces** para manipular la tabla mediante JavaScript.
- **filteredValue="#{funcionarioConvocatoriaViewController.listaConvocatoriaFuncionarioProyectosFilter}"**: Lista que almacena los resultados filtrados de la tabla, útil para búsquedas y filtros dinámicos.

Formulario de Ingreso/Edición

Este formulario permite seleccionar la convocatoria y, de forma opcional, asociar proyectos a un usuario. Se compone de dos campos principales:

1. Campo autocomplete para seleccionar la Convocatoria.
2. Campo autocomplete múltiple para seleccionar los Proyectos (dependientes de la convocatoria).

A continuación, su documentación:

Selección de Convocatoria

```
<p:autoComplete id="conv-FC"
value="#{funcionarioConvocatoriaViewController.selected.convocatoriaId}"
converter="convocatoriaConverter"
completeMethod="#{funcionarioConvocatoriaConvController.completeConvocatorias}"
placeholder="#{msg['gestion.configuracion.convocatoria.funcionarios.convocatoria.placeholder']}"
forceSelection="true" queryDelay="1000" minQueryLength="3"
itemLabel="#{convocatoria.convocatoriaId}#{empty convocatoria.descripcionConvocatoria ? '' : '- '}"
itemValue="#{convocatoria}" scrollHeight="200" style="width: 100%" inputStyleClass="form-control" required="true"
requiredMessage="#{msg['gestion.configuracion.convocatoria.funcionarios.convocatoria.required']}"
>
<p:ajax event="itemSelect" update="autoCompleteProyectos, msg"
listener="#{funcionarioConvocatoriaViewController.cleanProyectoAfterConvSelect}" />
```

</p:autoComplete>

A continuación, una explicación más detallada:

- **id="conv-FC"**: Identificador único del campo de autocompletar de convocatoria.
- **value="#{funcionarioConvocatoriaViewController.selected.convocatoriald} "**: Propiedad enlazada al objeto seleccionado, que almacena la convocatoria asignada al funcionario.
- **converter="convocatoriaConverter"**: Conversor encargado de transformar entre el objeto Convocatoria y su representación de texto para la vista.
- **completeMethod="#{funcionarioConvocatoriaConvController.completeConvocatorias}"**: Método que obtiene las opciones de convocatorias coincidentes con el texto ingresado por el usuario.
- **var="convocatoria"**: Variable utilizada para iterar sobre los resultados y referirse a cada convocatoria individualmente.
- **placeholder="#{msg['gestion.configuracion.convocatoria.funcionarios.convocatoria.placeholder']}"**: Texto de ayuda mostrado cuando el campo está vacío, obtenido del archivo de internacionalización.
- **forceSelection="true"**: Obliga a que el valor ingresado corresponda a una opción existente, evitando entradas manuales inválidas.
- **queryDelay="1000"**: Tiempo (en milisegundos) que espera antes de ejecutar la búsqueda tras escribir.
- **minQueryLength="3"**: Número mínimo de caracteres que el usuario debe escribir antes de iniciar la búsqueda.
- **itemLabel="#{convocatoria.convocatoriald}#{empty convocatoria.descripcionConvocatoria ? " : " : " - "}"**: Texto mostrado en la lista de resultados, compuesto por el ID de la convocatoria y su descripción.
- **itemValue="#{convocatoria}"**: Objeto que se asignará al campo al seleccionar una opción.
- **scrollHeight="200"**: Altura máxima de la lista desplegable en píxeles.
- **style="width: 100%"**: Ancho del componente al 100% del contenedor.
- **inputStyleClass="form-control"**: Clase CSS aplicada al campo de texto para integrar el estilo con el formulario.
- **required="true"**: Indica que este campo es obligatorio.
- **requiredMessage="#{msg['gestion.configuracion.convocatoria.funcionarios.convocatoria.required']}"**: Mensaje que se muestra si el campo queda vacío.
- **Etiqueta interna <p:ajax>**:
 - **event="itemSelect"**: Evento que se dispara al seleccionar una opción.

- **update="autoCompleteProyectos, msg"**: Componentes que se actualizan tras la selección (el campo de proyectos y mensajes).
- **listener="#{funcionarioConvocatoriaViewController.cleanProjectAfterConvSelect}"**: Método que limpia los proyectos seleccionados si se cambia la convocatoria.

Selección de Proyectos

```
<p:autoComplete id="autoCompleteProyectos"
value="#{funcionarioConvocatoriaViewController.selectedProyectos}"
completeMethod="#{funcionarioConvocatoriaViewController.buscarProyectos}" var="proyecto"
itemLabel="#{proyecto.proyectoId} - #{proyecto.titulo}"
itemValue="#{proyecto}"
multiple="true" forceSelection="true" size="50"
dropdown="true" scrollHeight="300"
inputStyleClass="form-control" style="width: 100%"
minQueryLength="3" required="false"
requiredMessage="#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder']}"
placeholder="#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder']}"
converter="proyectoConverter"
>
    <p:ajax event="itemSelect" update="msg" />
</p:autoComplete>
```

A continuación, una explicación más detallada:

- **id="autoCompleteProyectos"**: Identificador único del campo de autocompletar de proyectos.
- **value="#{funcionarioConvocatoriaViewController.selectedProyectos}"**: Lista enlazada al **backing bean** que almacena los proyectos seleccionados por el usuario.
- **completeMethod="#{funcionarioConvocatoriaViewController.buscarProyectos}"**: Método que busca proyectos que coincidan con el texto ingresado, filtrando por la convocatoria previamente seleccionada.
- **var="proyecto"**: Variable de iteración utilizada para referirse a cada proyecto en la lista de resultados.
- **itemLabel="#{proyecto.proyectoId} - #{proyecto.titulo}"**: Texto que se mostrará en el desplegable, compuesto por el ID y el título del proyecto.
- **itemValue="#{proyecto}"**: Objeto Proyecto que se asignará a la lista de seleccionados cuando el usuario lo elija.

- **multiple="true"**: Permite la selección de múltiples proyectos.
- **forceSelection="true"**: Obliga a que el valor ingresado sea uno de los elementos listados, evitando entradas no válidas.
- **size="50"**: Ancho visual del campo en caracteres (referencia, aunque el **style** sobrescribe este valor).
- **dropdown="true"**: Muestra un ícono que permite desplegar manualmente la lista de opciones.
- **scrollHeight="300"**: Altura máxima del panel de opciones (en píxeles) antes de que aparezca el **scroll**.
- **inputStyleClass="form-control"**: Clase **CSS** aplicada al campo para integrarlo visualmente con el resto del formulario.
- **style="width: 100%"**: Ancho total del componente ajustado al contenedor.
- **minQueryLength="3"**: Número mínimo de caracteres que se deben escribir antes de realizar la búsqueda.
- **required="false"**: Campo opcional, por lo que no es obligatorio seleccionar proyectos.
- **requiredMessage="#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder']}"**: Mensaje que se mostraría si el campo fuera obligatorio y estuviese vacío (en este caso no aplica porque **required=false**).
- **placeholder="#{msg['gestion.configuracion.convocatoria.funcionarios.proyecto.placeholder']}"**: Texto guía que aparece cuando el campo está vacío.
- **converter="proyectoConverter"**: Conversor encargado de transformar entre el objeto Proyecto y su representación de texto para la vista.
- **Etiqueta interna <p:ajax>**:
- **event="itemSelect"**: Evento que se dispara cuando el usuario selecciona un proyecto.
- **update="msg"**: Actualiza el componente de mensajes para mostrar cualquier notificación relacionada con la selección.

View Controller (FuncionarioConvocatoriaViewController.java)

Método "CargarListas"

El método **cargarListas()**, anotado con **@PostConstruct**, se ejecuta automáticamente después de que el **bean** ha sido construido y todas sus dependencias inyectadas. Su propósito principal es inicializar variables y preparar los datos que se mostrarán en la vista al cargar la pantalla de gestión de funcionarios por convocatoria. Primero, obtiene del **colcienciasSessController** el

objeto de utilidades que contiene referencias previas, recuperando de ahí el usuario seleccionado (**usuarioSelected**). Luego, carga la lista de estados (**listaEstados**) desde la configuración de la aplicación.

A continuación, se inicializa un **LazyDataModel<ConvocatoriaFuncionarioDTO>** para manejar la carga paginada de datos en la tabla principal. Dentro de su método **load()**, se configuran los filtros, incluyendo uno específico por **usuarioid**, y se consulta al servicio **convocatoriaServiceFacade** para obtener tanto el número total de registros como la lista correspondiente a la página solicitada. También se **sobreescriben** los métodos **getRowKey()** y **getRowData()** para identificar de forma única cada registro y recuperar un objeto individual desde la base de datos, asegurando que los datos se mantengan consistentes y no se actualicen por referencia directa a la lista principal. Con este enfoque, la tabla puede mostrar datos de forma eficiente, cargando únicamente los registros necesarios en cada interacción del usuario.

@PostConstruct:

```
@PostConstruct
public void cargarListas() {

    Map<String, Object> mapReferencias = (Map<String, Object>)
colcienciasSessController.getUtilObject();
    usuarioSelected = (UsuarioDTO) mapReferencias.get("usuarioConvocatoriaSelected");
    listaEstados = configApplController.getMapValorDefinicion().get("ESTADO");
    this.model = new LazyDataModel<ConvocatoriaFuncionarioDTO>() {
        @Override
        public List<ConvocatoriaFuncionarioDTO> load(int first, int pageSize, String
sortField, SortOrder sortOrder, Map<String, FilterMeta> filterBy) {
            Map<String, Object> filters = new TreeMap<>();
            filterBy.forEach((k, v) -> {
                if (k.equals("globalFilter")) {
                    if (v.getFilterValue() != null
                        && !v.getFilterValue().toString().isEmpty()) {
                        filters.put(k, v.getFilterValue());
                    }
                }
            });
            try {
                //Filtro por usuario id
                filters.put("usuarioId", usuarioSelected.getUsuarioId());

                int count =
convocatoriaServiceFacade.convocatoriaFuncionariosListCount(filters);
                if (count > 0) {
                    list = convocatoriaServiceFacade.convocatoriaFuncionariosList(first,
pageSize, sortField, sortOrder.equals(SortOrder.ASCENDING) ? "A" : "D", filters);
                }
                model.setRowCount(count);
            } catch (Exception ex) {
                LOG.error("Error al cargar el listado de funcionarios por convocatoria ",
ex);
            }
        }
    };
}
```

```

    }
    return list;
}

@Override
public Object getRowKey(ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO) {
    return convocatoriaFuncionarioDTO.getConvocatoriaFuncionarioId();
}

@Override
public ConvocatoriaFuncionarioDTO getRowData(String rowKey) {
    try {
        //Se consulta en la base de datos para mantener independiente el objeto que trae en la lista
        //principal y no lo actualice por referencia
        ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
        convocatoriaServiceFacade.queryConvocatoriaFuncionario(Long.parseLong(rowKey));
        return convocatoriaFuncionarioDTO;
    } catch (Exception ex) {
        LOG.error("no se pudo cargar el funcionario convocatoria ( id => " +
        rowKey
        + ")", ex);
    }
    return null;
}
};
}

```

Método “buscarProyectos”:

El método **buscarProyectos()** es el encargado de respaldar la funcionalidad del campo autocomplete de proyectos en la interfaz de usuario, permitiendo recuperar de forma dinámica los proyectos asociados a la convocatoria seleccionada conforme el usuario va escribiendo en el campo de búsqueda.

```

public List<ProyectoDTO> buscarProyectos(String query) {
    try {
        ConvocatoriaDTO convocatoriaSelected = this.selected.getConvocatoriaId();
        if (convocatoriaSelected != null) {
            if (query != null && !query.isEmpty()) {
                System.out.println("entra a buscarProyectos");
                Boolean isConvEstimulos = convocatoriaSelected.getConvEstimulos() != null
                && convocatoriaSelected.getConvEstimulos().equals("S");
                // Implementa la lógica para buscar proyectos basados en la consulta List<Long> estadosFiltro
                = Arrays.asList(
                    listaEstados.get("REVISADO").getValorDefinicionId()
                    , listaEstados.get("EVALUADO").getValorDefinicionId()
                    , listaEstados.get("SEGUIMIENTO").getValorDefinicionId()
                    , listaEstados.get("NEGADO").getValorDefinicionId()
                    , listaEstados.get("APROBADO").getValorDefinicionId()
                );

                ProyectoDTO proyectoFilter = new ProyectoDTO();
                proyectoFilter.setConvocatoriaId(convocatoriaSelected);
            }
        }
    }
}

```

```

        Map<String, Object> filters = new HashMap();
        filters.put("query", query);
        int count = 0;
        if (isConvEstimulos) {
// Proyectos de Estimulos Tributarios
            count = beneficiosServiceFacade.countProyectos(filters,
estadosFiltro, proyectoFilter, true);
            System.out.println("fullProyectos ESTIMULOS COUNT: " + count);
            if (count > 0) {
                return beneficiosServiceFacade.obtenerTodosProyectos(-1, -1,
"proyectoId", "A", filters, estadosFiltro, proyectoFilter);
            }
        } else {
// Proyectos y Programas
            count = proyectosServiceFacade.countProyectos(filters, estadosFiltro,
proyectoFilter, false);
            System.out.println("fullProyectos PROYECTOS COUNT: " + count);
            if (count > 0) {
                return proyectosServiceFacade.obtenerTodosProyectos(-1, -1,
"proyectoId", "A", filters, estadosFiltro, proyectoFilter);
            }
        }
    }
}
} catch (Exception ex) {
    LOG.error("Error al buscar los proyectos por autocomplete ", ex);
}
return null;
}

```

Su ejecución comienza validando que exista una convocatoria seleccionada (**convocatoriaSelected**) y que el texto de búsqueda (**query**) no esté vacío. A partir de ahí, determina si la convocatoria corresponde a Estímulos Tributarios verificando el valor de la propiedad **convEstimulos**.

Luego, define una lista de estados de proyecto válidos para la búsqueda (**REVISADO, EVALUADO, SEGUIMIENTO, NEGADO, APROBADO**) y prepara un filtro de búsqueda (**ProyectoDTO** y un mapa de parámetros). Dependiendo del tipo de convocatoria, consulta el servicio correspondiente (**beneficiosServiceFacade** para estímulos tributarios o **proyectosServiceFacade** para proyectos/programas) para obtener el conteo y, si existen resultados, devuelve la lista completa de proyectos que coinciden con el criterio. En caso de error o de no cumplirse las condiciones, retorna **null**. De esta manera, el método actúa como puente entre el autocomplete de la vista y la capa de servicios, entregando al usuario solo los proyectos filtrados y relevantes.

Conversation Controller (FuncionarioConvocatoriaConvController)

Método "guardar":

El método **guardar()** del **FuncionarioConvocatoriaConvController** se encarga de persistir en base de datos la información de un funcionario asignado a una convocatoria, junto con los proyectos asociados.

```
@Override
public void guardar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
        resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
        resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
            funcionarioConvocatoriaViewController.getSelected();

        convocatoriaFuncionarioDTO.setUsuarioCreador(seguridadSessController.getSelected().getUsuario());
        convocatoriaFuncionarioDTO.setFechaCreacion(new Date());

        convocatoriaFuncionarioDTO.setUsuarioId(funcionarioConvocatoriaViewController.getUsuarioSeleccionado());
    };
    //Auditoria
    AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
    appSessionVO.setPrograma(programa);
    convocatoriaFuncionarioDTO.setAppSessionVO(appSessionVO);
    convocatoriaFuncionarioDTO
        =
        convocatoriaServiceFacade.guardarConvocatoriaFuncionario(convocatoriaFuncionarioDTO);
    guardarProyectosAsignados(convocatoriaFuncionarioDTO);

    funcionarioConvocatoriaViewController.setSelected(new
        ConvocatoriaFuncionarioDTO());
    funcionarioConvocatoriaViewController.cargarListas();
    //se deshabilita el boton de editar / eliminar ya que no habra nada seleccionado despues de la
    operacion
    this.onRowUnselect(null);
    this.obtenerPestana = 0;
} catch (SystemException ex) {
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
        resolveExpression(ex.getMessage()));
    LOG.error("Error al guardar el funcionario por convocatoria ", ex);
} catch (Exception ex) {
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
        resolveExpression(ex.getMessage()));
    LOG.error("Error al guardar el funcionario por convocatoria ", ex);
} finally {
    FacesContext.getCurrentInstance().addMessage(null, message);
}
}
```

Primero, construye un mensaje de éxito que será mostrado al usuario si la operación se realiza correctamente. Luego, obtiene el objeto **ConvocatoriaFuncionarioDTO** actualmente

seleccionado en la vista y le asigna datos de auditoría, como el usuario que realiza la operación, la fecha de creación y el usuario asociado a la convocatoria.

También se registra información de sesión (**AppSessionVO**) para trazabilidad.

Luego, se invoca a **convocatoriaServiceFacade.guardarConvocatoriaFuncionario()** para almacenar la información y luego llama a **guardarProyectosAsignados()** para registrar los proyectos vinculados. Tras guardar, limpia el objeto seleccionado, recarga las listas visibles en pantalla (**cargarListas()**), deshabilita los botones de edición/eliminación (**onRowUnselect(null)**) y restablece la pestaña activa a la inicial. En caso de producirse una excepción, captura el error, registra el incidente en el log y muestra un mensaje de error al usuario. Finalmente, sea éxito o fallo, envía el mensaje correspondiente al **FacesContext** para que sea **renderizado** en la interfaz.

Método "guardarProyectosAsignados":

El método **guardarProyectosAsignados()** es el encargado de registrar en base de datos los proyectos asociados a un funcionario dentro de una convocatoria específica.

```
public void guardarProyectosAsignados(ConvocatoriaFuncionarioDTO
convocatoriaFuncionarioDTO) throws Exception {
    System.out.println("entra a guardarProyectosAsignados");
    //Eliminamos los proyectos existentes eliminarProyectosAsignados();
    //Registramos los proyectos asignados List<ProyectoDTO> nuevosProyectos =
        funcionarioConvocatoriaViewController.getSelectedProyectos();
    if (nuevosProyectos != null && !nuevosProyectos.isEmpty()) {
        for (ProyectoDTO proyectoDTO : nuevosProyectos) {
            //Auditoria
            AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
            appSessionVO.setProgram(programa);
            ConvocatoriaFuncionarioProyectoDTO funcionarioProyectoAdd =
convocatoriaServiceFacade.obtenerConvocatoriaFuncionarioProyecto(convocatoriaFuncionarioDTO.
ge tConvocatoriaFuncionarioId()
, proyectoDTO.getProyectoId()
);
            //Se edita el registro
            if (funcionarioProyectoAdd != null &&
funcionarioProyectoAdd.getConvocatoriaFuncionarioProyectoId() != null) {
                // System.out.println("proyectoDTO Editar " + proyectoDTO);
                funcionarioProyectoAdd.setAppSessionVO(appSessionVO);

                funcionarioProyectoAdd.setUsuarioModificador(seguridadSessController.getSelected().getUsuari
o()); //Usuario
                funcionarioProyectoAdd.setFechaModificacion(new Date());

                convocatoriaServiceFacade.editarConvocatoriaFuncionarioProyecto(funcionarioProyectoAdd);
            } else { //Se guarda el registro
```

```
//      System.out.println("proyectoDTO      Guardar      "      +      proyectoDTO);
ConvocatoriaFuncionarioProyectoDTO funcionarioProyectoDTO = new
    ConvocatoriaFuncionarioProyectoDTO();
    funcionarioProyectoDTO.setProyectoId(proyectoDTO);

funcionarioProyectoDTO.setConvocatoriaFuncionarioId(convocatoriaFuncionarioDTO);
funcionarioProyectoDTO.setAppSessionVO(appSessionVO);

funcionarioProyectoDTO.setUsuarioCreador(seguridadSessController.getSelected().getUsuario())
; /
        / Usuario
        funcionarioProyectoDTO.setFechaCreacion(new Date());

convocatoriaServiceFacade.guardarConvocatoriaFuncionarioProyecto(funcionarioProyectoDTO);
    }
}
}
```

Primero, invoca a **eliminarProyectosAsignados()** para limpiar cualquier asociación previa y evitar duplicados. Luego, obtiene desde el **ViewController** la lista de proyectos actualmente seleccionados por el usuario en la interfaz.

Si existen proyectos seleccionados, recorre cada uno de ellos y verifica si ya existe en la base de datos una relación entre el funcionario y el proyecto usando **convocatoriaServiceFacade.obtenerConvocatoriaFuncionarioProyecto()**. Si la asociación ya existe, la actualiza con datos de auditoría, usuario modificador y fecha de modificación. Si no existe, crea una nueva instancia de **ConvocatoriaFuncionarioProyectoDTO**, asigna la información necesaria (proyecto, funcionario, auditoría, usuario creador y fecha de creación) y la guarda mediante **convocatoriaServiceFacade.guardarConvocatoriaFuncionarioProyecto()**.

Método “guardarProyectosAsignados”:

El método **eliminarProyectosAsignados()** se encarga de borrar todas las asociaciones existentes entre un funcionario y los proyectos vinculados a una convocatoria determinada.

```
public void eliminarProyectosAsignados() {
    try {
        Map<String, Object> filters = new HashMap<>();
        filters.put("usuarioId",

funcionarioConvocatoriaViewController.getUsuarioSelected().getUsuarioId());
        filters.put("convocatoriaId",

funcionarioConvocatoriaViewController.getSelected().getConvocatoriaId().getConvocatoriaId())
;
        int count
            =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosListCount(filters);
```

```

        if (count > 0) {
            List<ConvocatoriaFuncionarioProyectoDTO> proyectosFuncionarioList =
convocatoriaServiceFacade.convocatoriaFuncionarioProyectosList(-1,
"proyectoId.proyectoId", "A", filters);
            if (proyectosFuncionarioList != null && !proyectosFuncionarioList.isEmpty())
{
                for (ConvocatoriaFuncionarioProyectoDTO funcionarioProyectoDTO
: proyectosFuncionarioList) {
//Auditoria
                    AppSessionVO appSessionVO =
seguridadSessController.getAppSessionVO();
                    appSessionVO.setProgram(programa);
                    funcionarioProyectoDTO.setAppSessionVO(appSessionVO);

convocatoriaServiceFacade.eliminarConvocatoriaFuncionarioProyecto(funcionarioProyectoDTO);
                }
            }
        } catch (Exception ex) {
            LOG.error("Error al eliminar los proyectos asignados del funcionario ", ex);
        }
    }
}

```

Para ello, construye un mapa de filtros que contiene el **usuarioid** y el **convocatoriaId** obtenidos desde el **ViewController**. Con estos filtros, consulta cuántos registros existen mediante **convocatoriaServiceFacade.convocatoriaFuncionarioProyectosListCount()**.

Si hay registros, obtiene la lista completa de asociaciones actuales usando **convocatoriaServiceFacade.convocatoriaFuncionarioProyectosList()**. Luego, recorre cada elemento y le asigna los datos de auditoría correspondientes (incluyendo el programa desde **AppSessionVO**), para finalmente eliminarlos de la base de datos llamando a **convocatoriaServiceFacade.eliminarConvocatoriaFuncionarioProyecto()**.

Método "editar":

El método **editar()** actualiza la información de un funcionario asociado a una convocatoria, junto con los proyectos que tenga asignados.

```

@Override
public void editar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
    try {
        ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
funcionarioConvocatoriaViewController.getSelected();

convocatoriaFuncionarioDTO.setUsuarioModificador(seguridadSessController.getSelected().getUs
uario());
convocatoriaFuncionarioDTO.setFechaModificacion(new Date());
    }
}

```



```
convocatoriaFuncionarioDTO.setUsuarioId(funcionarioConvocatoriaViewController.getUsuarioSelec
ted()
    );
//Auditoria
AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
appSessionVO.setProgram(programa);
convocatoriaFuncionarioDTO.setAppSessionVO(appSessionVO);

convocatoriaServiceFacade.editarConvocatoriaFuncionario(convocatoriaFuncionarioDTO);
guardarProyectosAsignados(convocatoriaFuncionarioDTO);
funcionarioConvocatoriaViewController.setSelected(new
ConvocatoriaFuncionarioDTO());
funcionarioConvocatoriaViewController.cargarListas();
//se deshabilita el boton de editar / eliminar ya que no habra nada seleccionado despues de la
operación
    this.onRowUnselect(null);
    this.obtenerPestana = 0;
} catch (SystemException ex) {
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
    LOG.error("Error al editar el funcionario por convocatoria ", ex);
} catch (Exception ex) {
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
    LOG.error("Error al editar el funcionario de la Convocatoria ", ex);
} finally {
    FacesContext.getCurrentInstance().addMessage(null, message);
}
}
```

Primero, obtiene el objeto seleccionado desde el **ViewController** y le asigna datos de auditoría como el usuario modificador, la fecha de modificación y el programa activo. Luego, llama a **convocatoriaServiceFacade.editarConvocatoriaFuncionario()** para guardar los cambios y ejecuta **guardarProyectosAsignados()** para actualizar la relación con los proyectos correspondientes.

Tras completar la edición, reinicia el objeto seleccionado, recarga las listas de la vista y desactiva los botones de edición/eliminación, asegurando que la interfaz quede limpia y lista para una nueva operación. En caso de error, genera mensajes de retroalimentación en la interfaz y registra el incidente en el log.

Método "eliminar":

El método **eliminar()** se encarga de remover una convocatoria asociada a un funcionario.

```
@Override
public void eliminar(ActionEvent ae) {
    FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
resolveExpression("#{msg['sb.mensajes.titulo.exito']}"),
resolveExpression("#{msg['sb.mensajes.transaccion.satisfactorio']}"));
}
```

```
try {
    ConvocatoriaFuncionarioDTO convocatoriaFuncionarioDTO =
funcionarioConvocatoriaViewController.getSelected();
//Auditoria
    AppSessionVO appSessionVO = seguridadSessController.getAppSessionVO();
    appSessionVO.setProgram(programa);
    convocatoriaFuncionarioDTO.setAppSessionVO(appSessionVO);
    convocatoriaServiceFacade.eliminarConvocatoriaFuncionario(convocatoriaFuncionarioDTO
);
    funcionarioConvocatoriaViewController.setSelected(new ConvocatoriaFuncionarioDTO());
    funcionarioConvocatoriaViewController.cargarListas();
    this.onRowUnselect(null);
} catch (Exception ex) {
    message = new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERROR",
resolveExpression(ex.getMessage()));
    LOG.error("Error al eliminar el funcionario de la Convocatoria ", ex);
} finally {
    FacesContext.getCurrentInstance().addMessage(null, message);
}
}
```

Para ello, obtiene el registro seleccionado desde el **ViewController**, le añade la información de auditoría correspondiente (como el programa en uso) y luego invoca a **convocatoriaServiceFacade.eliminarConvocatoriaFuncionario()** para ejecutar la eliminación en la capa de servicio.

Una vez completada la operación, reinicia el objeto seleccionado, recarga las listas de datos en pantalla y limpia la selección de filas en la interfaz. Si ocurre algún error durante el proceso, genera un mensaje de error para el usuario y registra el detalle en el **log**.

Método “completeConvocatorias”:

El método **completeConvocatorias()** implementa la funcionalidad de autocompletado para mostrar convocatorias activas en el sistema a partir del texto que el usuario introduce.

```
public List<ConvocatoriaDTO> completeConvocatorias(String query) {
    List<ConvocatoriaDTO> convocatorias = new ArrayList<>();
    Map<String, Object> filters = new HashMap<>();
    filters.put("query2", query);
    filters.put("activo", "S");
    try {
        convocatorias = convocatoriaServiceFacade.convocatoriasList(-1, -1,
"descripcionConvocatoria", "A", filters);
        funcionarioConvocatoriaViewController.setConvocatoriasList(convocatorias);
    } catch (Exception ex) {
        LOG.error("Error al consultar el autocompletar de convocatorias ", ex);
    }
    return convocatorias;
}
```

Recibe el parámetro **query**, lo incluye junto con el filtro de estado **activo ("S")** en un mapa de filtros y llama a **convocatoriaServiceFacade.convocatoriasList()** para obtener el listado filtrado, ordenado por la descripción de la convocatoria.

El resultado se guarda en el **ViewController** para mantenerlo disponible en la vista, y finalmente se retorna la lista de convocatorias. Si ocurre algún error durante la consulta, este se registra en el log sin interrumpir la ejecución.

Método "cargarProyectosAsignados":

El método **cargarProyectosAsignados()** se ejecuta al presionar el botón de Proyectos Asignados en la tabla, permitiendo visualizar los proyectos vinculados a un funcionario en una convocatoria específica.

```
public List<ConvocatoriaDTO> completeConvocatorias(String query) {  
    List<ConvocatoriaDTO> convocatorias = new ArrayList<>();  
    Map<String, Object> filters = new HashMap<>();  
    filters.put("query2", query);  
    filters.put("activo", "S");  
    try {  
        convocatorias = convocatoriaServiceFacade.convocatoriasList(-1, -1,  
"descripcionConvocatoria", "A", filters);  
        funcionarioConvocatoriaViewController.setConvocatoriasList(convocatorias);  
    } catch (Exception ex) {  
        LOG.error("Error al consultar el autocompletar de convocatorias ", ex);  
    }  
    return convocatorias;  
}
```

Recibe el **ConvocatoriaFuncionarioDTO** seleccionado, lo guarda en el **ViewController** y construye un filtro usando su identificador (**convocatoriaFuncionarioId**).

Luego consulta la cantidad de proyectos asociados y, si existen, obtiene el listado ordenado por **projectId**. Esta lista se asigna tanto a la variable principal como a la variable de filtro del **ViewController**, para que pueda mostrarse y filtrarse en la interfaz. Si ocurre algún error, se registra en el log sin interrumpir el flujo.

Servicio EJB (ConvocatoriaFuncionarioServiceBean)

El controlador se comunica con el	Servicio EJB	de la pantalla
(ConvocatoriaFuncionarioServiceBean)	llamando	al
	método	

convocatoriaFuncionariosList(), el cual nos proporciona la información necesaria desde la base de datos para ser visualizada en pantalla.

Es fundamental instanciar dos objetos clave utilizados para realizar consultas con la Java Persistence API (JPA): **EntityManager** y **CriteriaBuilder**. A continuación, la función de cada uno de ellos:

```
private EntityManager em; private CriteriaBuilder cb;
//ConvocatoriaFuncionario
private CriteriaQuery<ConvocatoriaFuncionario> queryConvocatoriaFuncionario; private
Root<ConvocatoriaFuncionario> ConvocatoriaFuncionarioRoot;
public void setEntityManager(EntityManager em) { this.em = em;
}
```

EntityManager

Es una interfaz en JPA que proporciona métodos para interactuar con la base de datos. Su función principal es gestionar las entidades en el contexto de persistencia, permitiendo realizar operaciones de lectura y escritura en la base de datos. Algunas de las funciones claves del **EntityManager** incluyen:

- **Persistencia de Entidades:** **EntityManager** se utiliza para persistir (almacenar) nuevas entidades en la base de datos.
- **Actualización de Entidades:** Se utiliza para actualizar entidades existentes en el contexto de persistencia.
- **Eliminación de Entidades:** Permite eliminar entidades de la base de datos.
- **Consultas JPA:** Facilita la ejecución de consultas JPA para recuperar datos de la base de datos.

En el código, **EntityManager** se utiliza para crear instancias de consultas (**queryConvocatoriaFuncionario**) y para configurar el contexto de persistencia necesario para realizar operaciones en la base de datos.

CriteriaBuilder

Es una interfaz en JPA que se utiliza para construir consultas de manera programática y sin depender de lenguajes de consulta específicos como SQL. Permite construir consultas de manera más dinámica y tipada que las consultas basadas en cadenas.

Algunas de las funciones clave de **CriteriaBuilder** incluyen:

- **Creación de Consultas:** Permite construir consultas JPA utilizando una API de construcción de consultas.
- **Construcción de Predicados:** Facilita la creación de predicados (condiciones) que se pueden combinar para formar consultas más complejas.
- **Operaciones Agregadas:** Proporciona métodos para realizar operaciones agregadas como contar, sumar, promediar, etc.

En el código, **CriteriaBuilder** se utiliza para construir las consultas (**queryConvocatoriaFuncionario**) y para crear predicados que representan condiciones específicas que se aplican a la consulta.

CriteriaQuery y Root

- **CriteriaQuery<ConvocatoriaFuncionario>:** **CriteriaQuery** es una interfaz que define la estructura básica de una consulta JPA. **CriteriaQuery<ConvocatoriaFuncionario>** específicamente indica que la consulta está dirigida a la entidad **ConvocatoriaFuncionario**. Se utiliza para construir la parte estática y común de la consulta.
- **Root<ConvocatoriaFuncionario>:** **Root** representa la raíz de la entidad en la consulta. En este contexto, **convocatoriaFuncionarioRoot** se utiliza para definir la tabla principal de la consulta y acceder a los atributos de la entidad **ConvocatoriaFuncionario**.

En resumen, **EntityManager** se encarga de la gestión de entidades y de la ejecución de consultas, mientras que **CriteriaBuilder** proporciona una interfaz para construir consultas de manera programática y **CriteriaQuery** junto con **Root** se utilizan para definir la estructura de la consulta. Estos componentes trabajan juntos para construir y ejecutar consultas JPA de manera eficiente.

Método ConvocatoriaFuncionarioList(int first, int pageSize, String sortField, String sortOrder, Map<String, Object> filters)

Este método **ConvocatoriaFuncionarioList** en el EJB (Enterprise JavaBean) realiza la consulta y la carga de datos relacionados con las convocatorias.

```
@Override
public List<ConvocatoriaFuncionario> ConvocatoriaFuncionarioList(int first, int pageSize,
String sortField, String sortOrder, Map<String, Object> filters) {
```

```
cb = em.getCriteriaBuilder();
queryConvocatoriaFuncionario = cb.createQuery(ConvocatoriaFuncionario.class);
convocatoriaFuncionarioRoot = queryConvocatoriaFuncionario.from(ConvocatoriaFuncionario.class);
queryConvocatoriaFuncionario.select(convocatoriaFuncionarioRoot);
//Where
queryConvocatoriaFuncionario.where(cb.and(filtrosConvocatoriaFuncionario(filters).toArray(new Predicate[] {
    {
    }
})));
//Order
if (sortOrder.equals("A")) {
    queryConvocatoriaFuncionario.orderBy(cb.asc(getConvocatoriaFuncionarioPath(sortField)));
} else {
    queryConvocatoriaFuncionario.orderBy(cb.desc(getConvocatoriaFuncionarioPath(sortField)));
}
//pagination
TypedQuery<ConvocatoriaFuncionario> tq = em.createQuery(queryConvocatoriaFuncionario);
if (pageSize >= 0) {
    tq.setMaxResults(pageSize);
}
if (first >= 0) {
    tq.setFirstResult(first);
}
return tq.getResultList();
}
```

Aquí está una explicación detallada de lo que hace:

1. Construcción de la Consulta:

```
cb = em.getCriteriaBuilder();
queryConvocatoriaFuncionario = cb.createQuery(ConvocatoriaFuncionario.class);
convocatoriaFuncionarioRoot = queryConvocatoriaFuncionario.from(ConvocatoriaFuncionario.class);
queryConvocatoriaFuncionario.select(convocatoriaFuncionarioRoot);
```

- Se obtiene un **CriteriaBuilder** para construir la consulta.
- Se crea una consulta (**queryConvocatoriaFuncionario**) para la entidad **ConvocatoriaFuncionario**.
- Se crea una raíz (**convocatoriaFuncionarioRoot**) que representa la tabla de la entidad.
- La consulta selecciona todos los campos de la entidad.

2. Filtros:

```
//Where  
queryConvocatoriaFuncionario.where(cb.and(filtrosConvocatoriaFuncionario(filters).toArray(new  
Predicate[]{})));
```

- Se llama al método **filtrosConvocatoriaFuncionario (filters)** para obtener un conjunto de predicados que representan los filtros a aplicar.
- Los predicados se combinan con una conjunción lógica "AND" y se agregan a la cláusula "WHERE" de la consulta.

3. Ordenamiento:

```
if (sortOrder.equals("A")) {  
    queryConvocatoriaFuncionario.orderBy(cb.asc(getConvocatoriaFuncionarioPath(sortField)));  
} else {  
    queryConvocatoriaFuncionario.orderBy(cb.desc(getConvocatoriaFuncionarioPath(sortField)));  
}
```

- Se determina la dirección del ordenamiento (ascendente o descendente) y se aplica al campo especificado en la consulta.

4. Paginación:

```
//pagination  
TypedQuery<ConvocatoriaFuncionario> tq = em.createQuery(queryConvocatoriaFuncionario);  
if (pageSize >= 0) {  
    tq.setMaxResults(pageSize);  
}  
if (first >= 0) { tq.setFirstResult(first);  
}
```

- Se crea un objeto **TypedQuery** a partir de la consulta.
- Se configura la paginación utilizando **setMaxResults** y **setFirstResult** según los parámetros **pageSize** y **first**.

Finalmente, la lista de **ConvocatoriaFuncionario** resultante se devuelve al llamador del método (**return tq**). Este método realiza una consulta dinámica basada en filtros, ordenamiento y paginación, lo que es común en aplicaciones web para mostrar datos de manera eficiente en una interfaz de usuario paginada y filtrada.

Método GlobalFilter() y GlobalPath()

El método **globalFilter()** se utiliza para aplicar un filtro global a una consulta, y el método **globalPath()** complementa esta funcionalidad proporcionando el camino (path) adecuado para un campo específico en una entidad particular. Estos métodos permiten la búsqueda global en cualquier pantalla de la aplicación, facilitando la construcción de predicados dinámicos en consultas JPA.

Método `globalFilter(Map<String, Object> filters, List<String> camposFiltro, String pathFilterMethod, List<Predicate> lista)`

```
/**
 * Metodo para la busqueda global de cualquier pantalla
 */
private void globalFilter(Map<String, Object> filters, List<String> camposFiltro, String
pathFilterMethod, List<Predicate> lista) {... 101 lines }
```

Este método realiza lo siguiente:

- **Inicialización:**

```
Path pathFilter = null;
// Evitar duplicación de filtros para las mismas columnas
Set<String> camposFiltrados = new HashSet<>();
if (filters != null && !filters.isEmpty()) {
    Predicate filterCondition = cb.conjunction();
    // ...
}
```

- Se inicializa la variable **pathFilter** a **null**.
- Se define una estructura de datos llamada **Set** (en este caso, una implementación de **HashSet**), cuyo propósito es almacenar un conjunto de valores únicos. En este contexto, la variable **camposFiltrados** se utiliza para evitar la duplicación de filtros aplicados a las mismas columnas en una consulta dinámica.
- Se verifica si hay filtros en el mapa **filters**. Si hay, se inicia la construcción del predicado global.
- El método es genérico y funcionable para cualquier Entidad del **convocatoriaFuncionario**

- **Procesamiento de Filtros:**

```
for (Map.Entry<String, Object> filter : filters.entrySet()) {
```



```
if (filter.getValue() != null && !filter.getValue().equals("")) {
```

- Se itera sobre cada entrada del **Map<String, Object> filters**, donde:
 - **filter.getKey()** es el nombre del campo.
 - **filter.getValue()** es el valor de búsqueda.
- Se ignoran los valores **null** o vacíos.
- **Identificación de "globalFilter":**

```
if (filter.getKey().equals("globalFilter")) {  
    for (int i = 0; i < camposFiltro.size(); i++) {
```

- Se verifica si la clave es **"globalFilter"**.
- Se itera sobre la lista **camposFiltro** para aplicar filtros en cada campo permitido.
- **Procesamiento de filtros personalizados:**

```
String tipo = "";  
String campo = camposFiltro.get(i);  
  
// custom  
Map<String, String> items = null;  
String[] splitCustom = campo.split("\\|");  
if (splitCustom.length  
  
    > 1) {  
    tipo = "CUSTOM";  
    String[] splitCustomComa = splitCustom[1].split(",");  
    if (splitCustomComa.length > 0) {  
        items = new TreeMap<>();  
        for (String itemValues : splitCustomComa) {  
            String[] splitCustomDot = itemValues.split(":");  
            ol items  
                .put(splitCustomDot[0], splitCustomDot[1]);  
        }  
    }  
    pathFilter = globalPath(pathFilterMethod, splitCustom[0]);  
}
```

- **TreeMap<String, String>:** Se usa para almacenar equivalencias entre valores legibles por humanos y los valores reales en la base de datos.
 - Ejemplo: { "1": "Activo", "0": "Inactivo" }
- **split("\\|"):** Divide la cadena en dos partes (**campo|mapa_de_valores**),

donde la segunda parte contiene los valores personalizados.

- **globalPath(pathFilterMethod, splitCustom[0]):** Método que obtiene la referencia (**Path<T>**) a un atributo específico de la entidad.
- **Identificación de tipos BOOLEAN y DATE:**

```
if (tipo.isEmpty ()
) {
String[] split = campo.split("\\.");
if (split.length > 1) {
campo = split[0];
tipo = split[1];
}
if (tipo.contains("BOOLEAN") || tipo.contains("DATE")) {
pathFilter = globalPath(pathFilterMethod, campo);
} else {
pathFilter = globalPath(pathFilterMethod, camposFiltro.get(i));
}
}
```

- Se verifica si el campo contiene un sufijo (**campo.tipo**).
- **BOOLEAN** y **DATE** requieren un manejo especial en la consulta, por lo que se extrae el **Path<T>** correspondiente.
- **globalPath(pathFilterMethod, campo):** Obtiene la ruta del campo dentro de la entidad JPA.
- **Construcción de la condición de filtro:**

a) BOOLEAN

```
case "BOOLEAN":
String booleanSearch = filter.getValue().toString().toLowerCase(); if (i == 0) {
filterCondition = cb.equal(pathFilter, (booleanSearch.contains("si") ?
Boolean.TRUE : booleanSearch.contains("no") ? Boolean.FALSE : null));
} else {
```

- **cb.equal(pathFilter, valor):** Genera una condición **WHERE campo = valor**.
- Se convierte "Si" y "No" en **true** o **false** antes de aplicar la condición.

b) DATE

```
case "DATE":
String dateSearch = filter.getValue().toString().toLowerCase(); if (i == 0) {
filterCondition = cb.like(cb.function("TO_CHAR",
String.class, pathFilter, cb.literal("dd/MM/yyyy")), "%" + dateSearch + "%");
```

```

    } else {
        filterCondition = cb.or(filterCondition, cb.like(cb.function("TO_CHAR",
String.class, pathFilter, cb.literal("dd/MM/yyyy")), "%" + dateSearch + "%"));
    }
    break;

```

- **cb.function("TO_CHAR", String.class, pathFilter, cb.literal("dd/MM/yyyy")):** Convierte la fecha en una cadena con formato **dd/MM/yyyy**.
- **cb.like(...):** Se usa **LIKE** para permitir búsquedas parciales en fechas ("15/03" buscará 15/03/2025).

c) CUSTOM

```

case "CUSTOM":
    String customSearch = filter.getValue().toString().toLowerCase(); String value = null;
    if (items != null && !items.isEmpty()) {
        for (Map.Entry<String, String> entry : items.entrySet()) {
            if (customSearch.contains(entry.getValue().toLowerCase())) {
                value = entry.getKey();
            }
        }
    }
    if (value != null) { if (i == 0) {
        filterCondition = cb.equal(pathFilter, value);
    } else {
        filterCondition = cb.or(filterCondition, cb.equal(pathFilter, value));
    }
    }
    break;

```

- Convierte los valores personalizados ("**Activo**" -> "**1**") antes de filtrar.
- **cb.equal(pathFilter, value):** Filtra por la clave correspondiente.

d) LIKE

```

default:
    if (i == 0) {
        filterCondition = cb.like(cb.lower(pathFilter), "%" +
filter.getValue().toString().toLowerCase() + "%");
    } else {
        filterCondition = cb.or(filterCondition, cb.like(cb.lower(pathFilter), "%" +
filter.getValue().toString().toLowerCase() + "%"));
    }

```

- **cb.lower(pathFilter):** Convierte el campo a minúsculas para hacer la búsqueda insensible a mayúsculas.
- **"%" + filtro + "%"** :

- Se agrega % antes y después del valor para permitir búsquedas parciales.
- **Agregar la condición final:**

```
lista.add(filterCondition);
```

- lista.add(filterCondition): Agrega el Predicate a la lista de condiciones que se usarán en la consulta CriteriaQuery.

El método **globalFilter** implementa una estrategia flexible y dinámica para la construcción de consultas en **JPA** utilizando **CriteriaBuilder**, lo que permite aplicar múltiples filtros sin necesidad de definir consultas estáticas. Su diseño modular soporta diferentes tipos de filtros (**BOOLEAN**, **DATE**, **CUSTOM** y búsquedas de texto con **LIKE**), proporcionando versatilidad en la manipulación de datos. Además, utiliza **cb.conjunction()** para generar condiciones dinámicas, asegurando que cada filtro se agregue correctamente según su tipo y formato esperado.

Uno de los aspectos clave del método es la capacidad de interpretar filtros personalizados mediante **CUSTOM**, lo que permite mapear valores de entrada en etiquetas más intuitivas para el usuario. Asimismo, el uso de **cb.function("TO_CHAR")** en el caso de fechas facilita la búsqueda de fechas en un formato legible sin necesidad de convertirlas manualmente en la aplicación. Esto optimiza las consultas sin comprometer la eficiencia, ya que se ejecutan directamente en la base de datos.

Finalmente, el método sigue un enfoque seguro y escalable, asegurando que los filtros nulos o vacíos no afecten la consulta final. El uso de **Path<T>** permite acceder dinámicamente a los atributos de la entidad sin necesidad de **hardcodear** nombres de columnas, mejorando la mantenibilidad del código. Con su diseño actual, **globalFilter** es altamente adaptable para cualquier necesidad de filtrado en aplicaciones empresariales, permitiendo una experiencia de búsqueda eficiente y sin pérdida de rendimiento.

Método globalPath(String pathMethod, String field)

```
private Path<?> globalPath(String pathMethod, String field) {  
    switch (pathMethod) {  
        case "getConvocatoriaFuncionarioPath":  
            return getConvocatoriaFuncionarioPath(field);  
        default:  
            return null;  
    }  
}
```

Este método realiza lo siguiente:

1. Selección del Método de Path:

```
switch (pathMethod) {  
    case "getConvocatoriaFuncionarioPath":  
        return getConvocatoriaFuncionarioPath(field)  
    default:  
        return null;  
}
```

- Se selecciona el método de path basado en el parámetro **pathMethod**.
- Se pueden agregar tantos métodos Path como sea necesarios. Lo ideal es que sea uno por Entidad.

2. Retorno del Path:

- Si el **pathMethod** es conocido (en este caso, "getConvocatoriaFuncionarioPath"), se llama al método correspondiente (**getConvocatoriaFuncionarioPath**) con el campo proporcionado y se retorna el path resultante.
- Si el **pathMethod** no es conocido, se retorna **null**.

En resumen, estos métodos proporcionan una forma modular y dinámica de aplicar filtros globales en consultas JPA, facilitando la construcción de consultas dinámicas y adaptativas en base a los filtros proporcionado.

Método **filtrosConvocatoriaFuncionario (Map<String, Object> filters)**

El método **filtrosConvocatoriaFuncionario()** se encarga de construir una lista de predicados (**Predicate**) que representan los filtros aplicados a la consulta de funcionarios convocatorias.

```
private List<Predicate> filtrosConvocatoriaFuncionario(Map<String, Object> filters) {  
    List<Predicate> lista = new ArrayList<>();  
    List<String> camposFiltro = new ArrayList<>();  
    Path pathFilter;  
    camposFiltro.add("usuarioId.usuario");  
    camposFiltro.add("usuarioId.descripcionUsuario");  
    camposFiltro.add("usuarioId.dependenciaId.nombreDependencia");  
    //Filtro por convocatoria ID  
    Object convocatoriaId = filters.get("convocatoriaId");  
    if (convocatoriaId != null) {  
  
        lista.add(cb.equal(getConvocatoriaFuncionarioPath("convocatoriaId.convocatoriaId"), (Long) convocatoriaId));  
    }  
}
```

```
//Filtro por usuario ID
Object usuarioId = filters.get("usuarioId");
if (usuarioId != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuarioId"), (Long)
usuarioId));
}
//Filtro por usuario
Object usuario = filters.get("usuario");
if (usuario != null) {
    lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuario"), (String)
usuario));
}

//Filtro Global
globalFilter(filters, camposFiltro, "getConvocatoriaFuncionarioPath", lista);
return lista;
}
```

Ahora, veamos en detalle este método:

1. Inicialización:

```
List<Predicate> lista = new ArrayList<>();
List<String> camposFiltro = new ArrayList<>();
```

- Se inicializa una lista de predicados (**lista**) que se utilizará para almacenar los filtros.
- Se inicializa una lista de nombres de campos de filtro (**camposFiltro**).

2. Definición de Campos de Filtro global:

```
camposFiltro.add("usuarioId.usuario");
camposFiltro.add("usuarioId.descripcionUsuario");
camposFiltro.add("usuarioId.dependenciaId.nombreDependencia");
```

3. Manejo de Filtros Avanzados:

```
//Filtro por convocatoria ID
Object convocatoriaId = filters.get("convocatoriaId"); if (convocatoriaId != null) {
lista.add(cb.equal(getConvocatoriaFuncionarioPath("convocatoriaId.convocatoriaId"), (Long)
convocatoriaId));
}
//Filtro por usuario ID
Object usuarioId = filters.get("usuarioId"); if (usuarioId != null) {
lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuarioId"), (Long) usuarioId));
}
//Filtro por usuario
Object usuario = filters.get("usuario"); if (usuario != null) {
```

```
lista.add(cb.equal(getConvocatoriaFuncionarioPath("usuarioId.usuario"), (String) usuario));
}
```

- Verificación y aplicación de filtros avanzados
- Fundamental que cada filtro que se agregue se encuentre en el Path indicado, para que de esta manera el método sepa dónde y contra que campo buscar

4. Filtro Global:

```
//Filtro Global
globalFilter(filters, camposFiltro, "getConvocatoriaFuncionarioPath", lista);
```

- Se llama a un método llamado **globalFilter** que aplica un filtro global en base a una lista de campos y utiliza el método **getConvocatoriaFuncionarioPath** para obtener los paths correspondientes.

5. Retorno de la Lista de Predicados:

```
return lista;
```

Método getConvocatoriaFuncionarioPath(String field)

El método **getConvocatoriaFuncionarioPath(String field)** devuelve un objeto **Path** que representa el camino (path) a un atributo específico de la entidad **ConvocatoriaFuncionario** que se utilizará en la construcción de predicados.

```
private Path<?> getConvocatoriaFuncionarioPath(String field) {
//Joins
    Join<ConvocatoriaFuncionario, Convocatoria> convocatoriaId =
convocatoriaFuncionarioRoot.join(ConvocatoriaFuncionario_.convocatoriaId, JoinType.LEFT);
    Join<ConvocatoriaFuncionario, Usuario> usuarioId =
convocatoriaFuncionarioRoot.join(ConvocatoriaFuncionario_.usuarioId, JoinType.LEFT);
    Join<Usuario, Dependencia> dependenciaId = usuarioId.join(Usuario_.dependenciaId,
JoinType.LEFT);

    Path<?> path = null;
    if (null == field) {
        path
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.convocatoriaFuncionarioId);
    } else {
        switch (field) {
            case "convocatoriaFuncionarioId":
                path
=
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.convocatoriaFuncionarioId);
        }
    }
}
```

```

        break;
        case "usuarioCreador":
            path
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.usuarioCreador);
            break;
        case "fechaCreacion":
            path
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.fechaCreacion);
            break;
        case "convocatoriaId.convocatoriaId":
            path = convocatoriaId.get(Convocatoria_.convocatoriaId);
            break;
        case "convocatoriaId.descripcionConvocatoria":
            path = convocatoriaId.get(Convocatoria_.descripcionConvocatoria);
            break;
        case "usuarioId.usuarioId":
            path = usuarioId.get(Usuario_.usuarioId);
            break;
        case "usuarioId.usuario":
            path = usuarioId.get(Usuario_.usuario);
            break;
        case "usuarioId.descripcionUsuario":
            path = usuarioId.get(Usuario_.descripcionUsuario);
            break;
        case "usuarioId.estadoUsuario":
            path = usuarioId.get(Usuario_.estadoUsuario);
            break;
        case "usuarioId.dependenciaId.dependenciaId":
            path = dependenciaId.get(Dependencia_.dependenciaId);
            break;
        case "usuarioId.dependenciaId.nombreDependencia":
            path = dependenciaId.get(Dependencia_.nombreDependencia);
            break;
        default:
            break;
    }
}
return path;
}

```

Ahora, veamos en detalle este método:

1. Asignación de Path en Base al Campo Especificado:

```

Path<?> path = null; switch (field) {
case "convocatoriaFuncionarioId": path =
convocatoriaFuncionarioRoot.get(ConvocatoriaFuncionario_.convocatoriaFuncionarioId); break;
// ... otros casos para diferentes campos
}

```

- Cada case debe corresponder con el nombre del campo y/o búsqueda que se va a realizar

- Al utilizar jpa, se debe identificar cuando se hace referencia a un campo o a un objeto de la Entidad

2. Retorno del Path:

```
return path;
```

En resumen, estos métodos están diseñados para facilitar la construcción de predicados y paths necesarios para aplicar filtros en una consulta JPA (Java Persistence API).

El método **filtrosConvocatoriaFuncionario()** maneja tanto filtros simples como filtros avanzados, mientras que **getConvocatoriaFuncionarioPath()** devuelve el **Path** correspondiente a un campo específico de la entidad **ConvocatoriaFuncionario**. Estos métodos son utilizados en la construcción de consultas dinámicas en base a los filtros proporcionados.

f. Inconsistencias en el reporte de proyectos – Convocatoria 968

Solicitud de requerimiento No. 159867

Descripción y alcance del Requerimiento

Desde la dirección de la OTSI Se me encarga como líder funcional de SIGP

1. ¿Indicar que es lo que está sucediendo con la plataforma SIGP??
2. ¿Qué informe entrega el Proveedor de Software Builders explicando:
 - 2.1 Qué está ocurriendo
 - 2.2 ¿Y el procedimiento y el proceso para la corrección de estos problemas?

Desde la Dirección de Gestión de Recursos, y conforme a lo establecido en el Procedimiento M801PR04 – Verificación de Requisitos de Mecanismos de Operación de CTel, específicamente en la actividad 2:

"Solicitar a la Oficina de Tecnología y Sistemas de Información, a través de la Mesa de Servicios, el reporte de propuestas recibidas en el mecanismo de operación de CTel, estableciendo un tiempo máximo de 2 días hábiles para recibir dicho reporte."

En este sentido, en el marco de la Convocatoria 968 – Calificación de proyectos que aplican a los ingresos no constitutivos de renta y/o ganancia ocasional (Art. 57-2 del Estatuto Tributario), se elevó la solicitud a través del Ticket No. 159410 ante la Mesa de Servicios el día viernes 6 de junio

de 2025, solicitando el reporte general de las propuestas radicadas en el Sistema de Gestión de Proyectos – SIGP.

Una vez recibido el reporte, se identificaron inconsistencias entre la cantidad de proyectos registrados en el SIGP (670) y los reportados en el informe entregado (677). Esta diferencia impide el inicio del proceso de verificación de requisitos, y considerando los plazos establecidos en el cronograma, genera un impacto negativo en la ejecución del proceso, afectando no solo la verificación, sino también las etapas subsiguientes.

Por lo anterior, se solicita de manera prioritaria la subsanación de la inconsistencia y el envío del reporte corregido, con el fin de dar inicio al proceso de verificación y cumplir con la entrega del informe a la Dirección

Técnica a más tardar el 18 de junio de 2025, sin afectar la etapa de evaluación. Desde la administración del sistema de información considero que se trata de fallas recurrentes en el sistema de información, y que si bien hemos tenido el apoyo de soporte estas fallas no pueden ser persistentes ni tener arreglos temporales, se debe dar una falla definitiva ya que está impactando la operación del sistema de información misional del ministerio

Desarrollo e Implementación

En atención a la solicitud recibida desde la Dirección de la OTSI respecto a la situación presentada con el reporte de propuestas radicadas en la plataforma SIGP para la Convocatoria 968 – Calificación de proyectos que aplican a los ingresos no constitutivos de renta y/o ganancia ocasional, me permito informar lo siguiente:

1. Sobre la situación en la plataforma SIGP:

No se ha identificado una falla en el funcionamiento de la plataforma SIGP. El sistema está operativo y los datos correspondientes a la convocatoria han sido almacenados correctamente. La inconsistencia detectada entre el número de proyectos registrados (670) y los reportados en el archivo generado (677) no obedece a una falla técnica del sistema, sino a una diferencia en la interpretación y validación del reporte generado mediante consulta SQL.

2. Informe del proveedor Software Builders:

○ 2.1. Qué está ocurriendo:

El reporte no fue generado directamente por Software Builders, sino por personal de la entidad, extrayendo los datos mediante un script SQL. El reporte contiene 7 registros adicionales, que corresponden a casos en los cuales algunos proyectos registraron personal con el rol de **Investigador Principal**, y estos investigadores están asociados a múltiples registros en la base de datos externa de **SCIENTI** (específicamente en la vista VW_SCIENTI_PERSONA).

El SQL utilizado incluye relaciones con esta vista externa, y aunque los campos de CvLAC no se muestran en el reporte final (porque están comentados en el SELECT), sí se ejecutan las relaciones a nivel de consulta. Esto ocasiona que, si una persona tiene más de un registro en SCIENTI con el mismo número de documento, el reporte duplique las filas correspondientes a ese proyecto.

Casos identificados:

- Proyecto 116299
 - INVESTIGADOR PRINCIPAL: Johanna Alexandra Patiño López
 - Numero Documento registrado en SIGP: 11111111
 - Consultado en CvLAC: 3 registros en SCIENTI.
- Proyecto 116345
 - INVESTIGADOR PRINCIPAL: Marcela Tabares Tabares
 - Numero Documento registrado en SIGP: 11111111
 - Consultado en CvLAC: 3 registros en SCIENTI.
- Proyecto 116461
 - INVESTIGADOR PRINCIPAL: Roberto Gálvez Valencia
 - Numero Documento registrado en SIGP: 1127228220
 - Consultado en CvLAC: 2 registros en SCIENTI.
- Proyecto 116500
 - INVESTIGADOR PRINCIPAL: Iris Andrés Moya Varela Martínez
 - Numero Documento registrado en SIGP: 11111111
 - Consultado en CvLAC: 3 registros en SCIENTI.

En todos estos casos, el campo de identificación se repite en SCIENTI en las personas registradas, lo que causa la duplicación.

○ **2.2. Procedimiento y proceso de corrección:**

Dado que el origen del problema radica en la base de datos de SCIENTI (fuente de datos externa al SIGP), sugerimos dos caminos de solución:

- Que los administradores de la base de datos de SCIENTI realicen una depuración y consolidación de los registros duplicados, mejorando la calidad y unicidad de los datos allí almacenados.
- O, como se ha hecho en ejercicios anteriores, que el reporte generado sea revisado manualmente por el personal responsable, para filtrar los registros duplicados y entregar un informe depurado.

Es importante aclarar que **SIGP simplemente consulta la información disponible en las vistas proporcionadas por SCIENTI**. Software Builders se responsabiliza únicamente del funcionamiento del sistema SIGP. La información externa consultada a través de vistas o enlaces de datos es responsabilidad de las fuentes proveedoras de esa información.

Se adjuntan dos scripts SQL como soporte técnico: el primero muestra los registros repetidos del Investigador Principal al hacer la relación con la vista VW_SCIENTI_PERSONA, evidenciando los datos duplicados existentes en la base de datos de SCIENTI; el segundo script genera el mismo reporte **sin realizar el JOIN con dicha vista**, lo que permite observar los datos sin duplicidad, facilitando la validación del número real de proyectos radicados.

Nota: La elaboración de este reporte tomó 5 horas, las cuales serán registradas dentro de la bolsa de horas de desarrollo acordada.

g. Reporte de Evaluación de la Convocatoria 963-2025 Orquídeas

Solicitud de requerimiento No. 159965

Descripción y alcance del Requerimiento

Reporte Convocatoria 963. Dentro del proceso de seguimiento de evaluación de la convocatoria No 963 de 2025, se requiere un reporte # 7 del SIGP de la evaluación con corte al día 16 de junio 2025 a las 9:00 a.m. Dado que la entrega del proceso de evaluación desde la Dirección de Vocaciones y Formación a la Dirección de Gestión de Recursos de la CTel se debe hacer el martes 17 de junio, requerimos el reporte para las 11:00 a.m. de hoy lunes 16 de junio en caso de tener que tomar acciones frente al proceso con los evaluadores.

Desarrollo e Implementación

En atención a la solicitud, me permito informar que se generó y adjuntó el **reporte #7 del SIGP**, correspondiente al proceso de evaluación de la convocatoria No. 963 de 2025, con corte al 16 de junio de 2025 a las 9:00 a.m.

Para la elaboración de este informe, se desarrolló una consulta específica en la base de datos del SIGP que permitió extraer la información consolidada por cada evaluador y por criterios de calificación asociados a los proyectos asignados. Esta información fue validada y estructurada en el formato solicitado, con el fin de facilitar su análisis previo a la entrega formal del proceso de evaluación a la Dirección de Gestión de Recursos de la CTel.

Cabe aclarar que, si en el reporte no se visualizan los 22 criterios por cada evaluación, corresponde a evaluaciones que aún no han sido iniciadas por el evaluador respectivo. Los criterios de evaluación son generados automáticamente una vez el evaluador ingresa al sistema y accede a la pantalla de evaluación del proyecto asignado.

Nota: La elaboración de este reporte tomó 1,5 horas, las cuales serán registradas dentro de la bolsa de horas de desarrollo acordada.

h. Apoyo Renovación Certificado SSL del SIGP

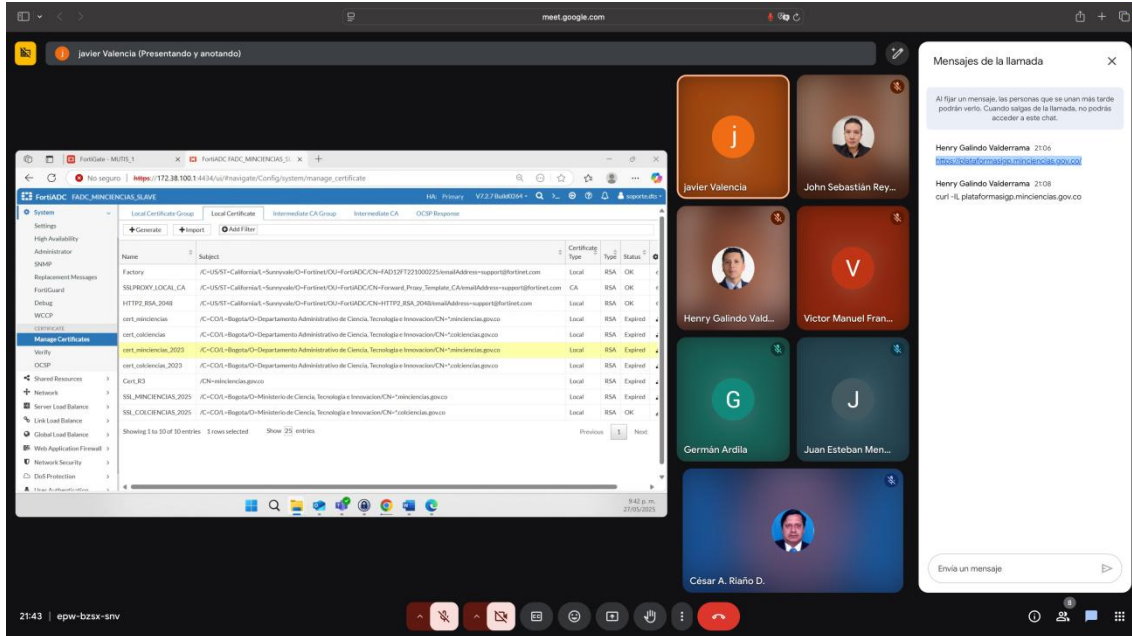
Solicitud de requerimiento No. 163314

Descripción y alcance del Requerimiento

El certificado SSL para el ingreso al SIGP venció el 27/05/2025 a las 6 p.m., por lo que es necesario su actualización para poder ingresar al sistema en modo seguro por la dirección <https://plataformasigp.minciencias.gov.co:7003>. Se solicita a Software Builders dar apoyo y acompañamiento al equipo de trabajo del Ministerio en las tareas de renovación y configuración del nuevo certificado.

Desarrollo e Implementación

Se realiza acompañamiento para la instalación del certificado SSL en el balanceador, a través de la reunión virtual:

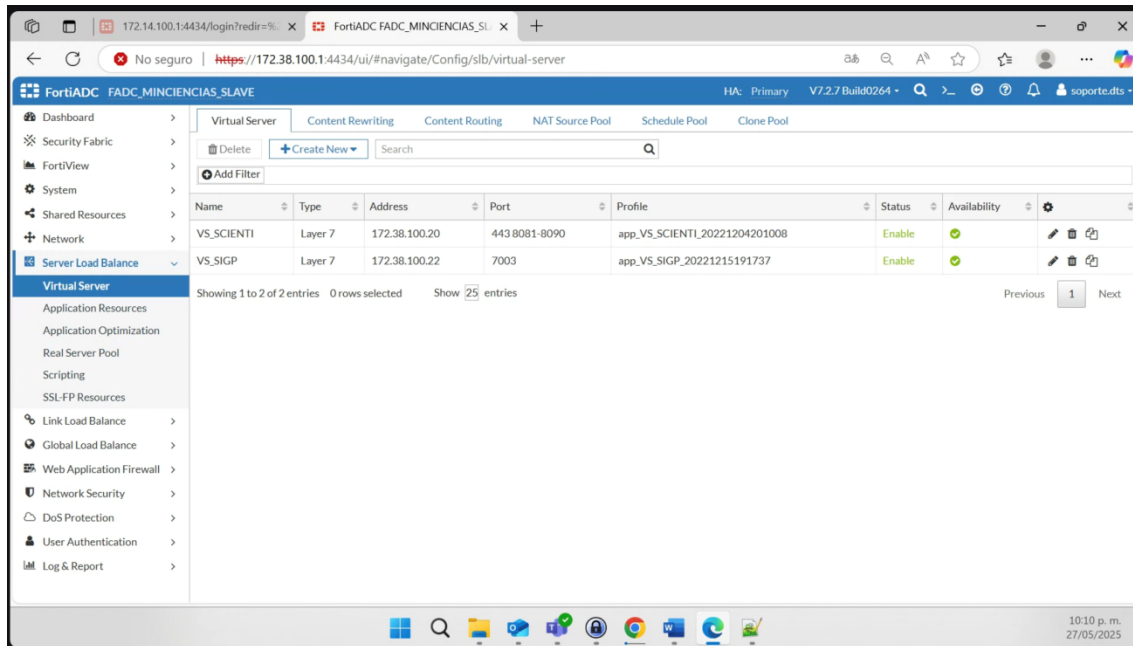


The screenshot shows a Google Meet interface with a presentation window displaying the FortiADC configuration page. The presentation is titled "FortiADC - FAD1C_MINCENCIAS_SLAVE" and shows the "Manage Certificates" section. The table below represents the data visible in the presentation:

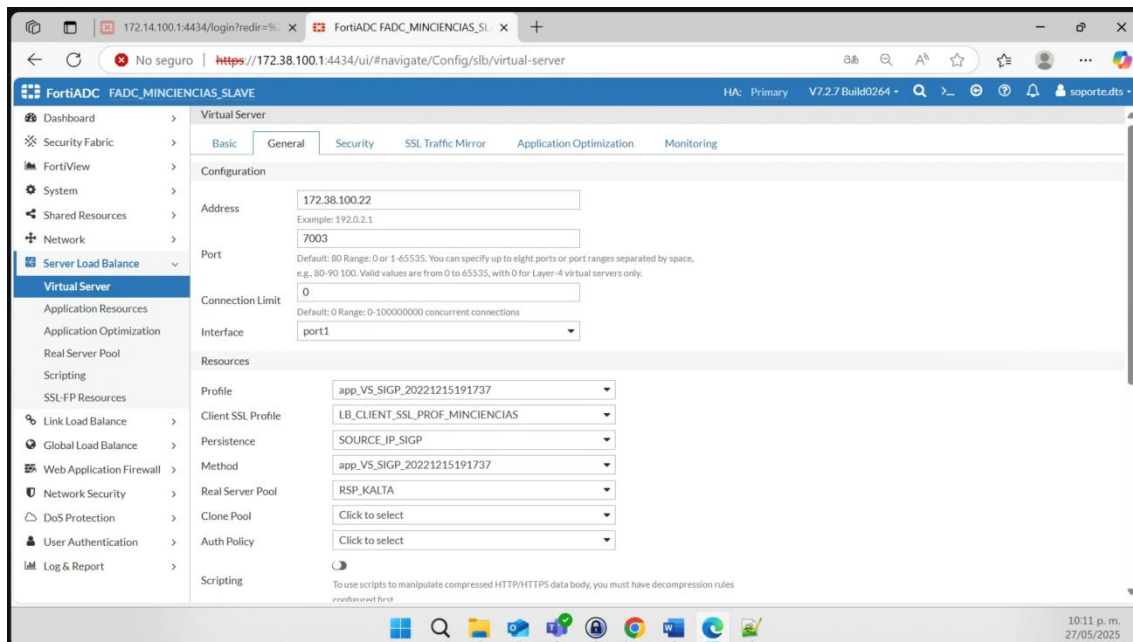
Name	Subject	Certificate Type	Type	Status
Factory	/C=US/ST=California/L=San Jose/O=Fortinet/OU=FortiADC/CN=Factory	Local	RSA	OK
SSLPROXY.LOCAL.CA	/C=US/ST=California/L=San Jose/O=Fortinet/OU=FortiADC/CN=Forward_Proxy_Template_CA/serialAddress=serial@fortinet.com	CA	RSA	OK
HTTP2_RSA_2048	/C=US/ST=California/L=San Jose/O=Fortinet/OU=FortiADC/CN=HTTP2_RSA_2048/serialAddress=serial@fortinet.com	Local	RSA	OK
WCCP	/C=US/ST=California/L=San Jose/O=Fortinet/OU=FortiADC/CN=HTTP2_RSA_2048/serialAddress=serial@fortinet.com	Local	RSA	OK
cert_casidencias	/C=CO/ST=Región de Murcia/O=Departamento Administrativo de Ciencia, Tecnología e Innovación/CN=cert_casidencias	Local	RSA	Expired
cert_casidencias_2025	/C=CO/ST=Región de Murcia/O=Departamento Administrativo de Ciencia, Tecnología e Innovación/CN=cert_casidencias_2025	Local	RSA	Expired
cert_j33	/C=CO/ST=Región de Murcia/O=Departamento Administrativo de Ciencia, Tecnología e Innovación/CN=cert_j33	Local	RSA	Expired
SSL_MINCENCIAS_2025	/C=CO/ST=Región de Murcia/O=Departamento Administrativo de Ciencia, Tecnología e Innovación/CN=SSL_MINCENCIAS_2025	Local	RSA	Expired
SSL_CONCENCIAS_2025	/C=CO/ST=Región de Murcia/O=Departamento Administrativo de Ciencia, Tecnología e Innovación/CN=SSL_CONCENCIAS_2025	Local	RSA	Expired

The meeting interface shows several participants: Javier Valencia (Presenting and annotating), John Sebastián Rey..., Henry Galindo Vald..., Victor Manuel Fran..., Germán Ardila, Juan Esteban Men..., and César A. Riaño D. The chat window on the right shows messages from Henry Galindo Valderrama at 21:05 and 21:08, both mentioning the URL <https://plataformasig.minciencias.gov.co>.

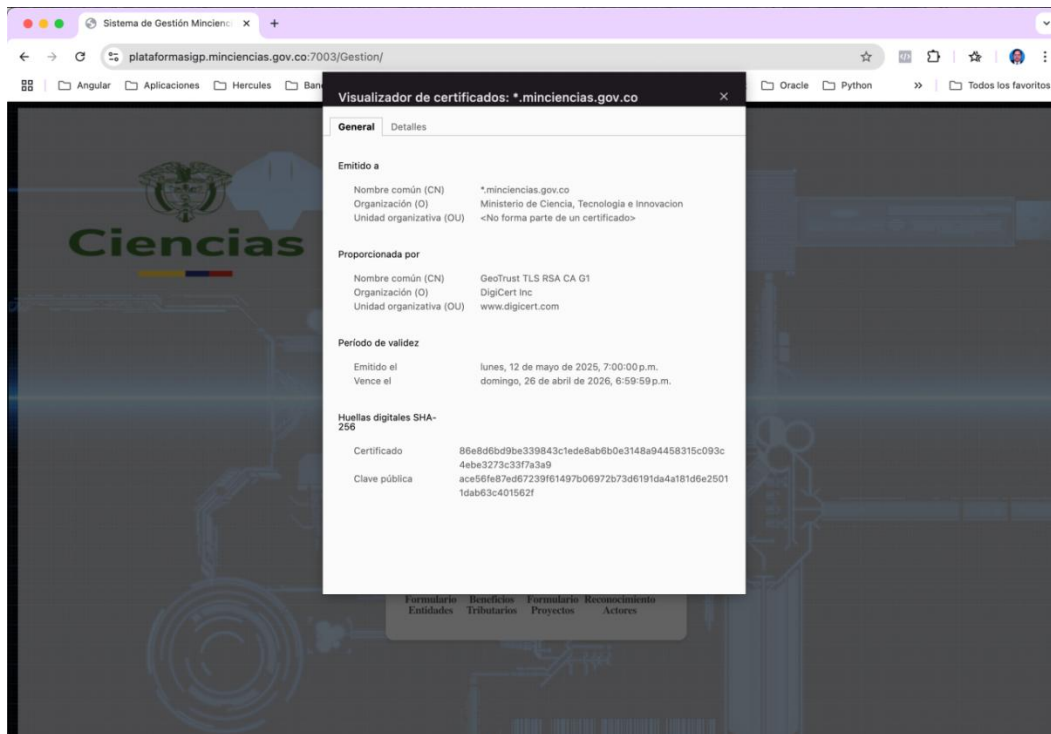
Se crea el virtual server VS_SIGP para configurar allí el certificado:



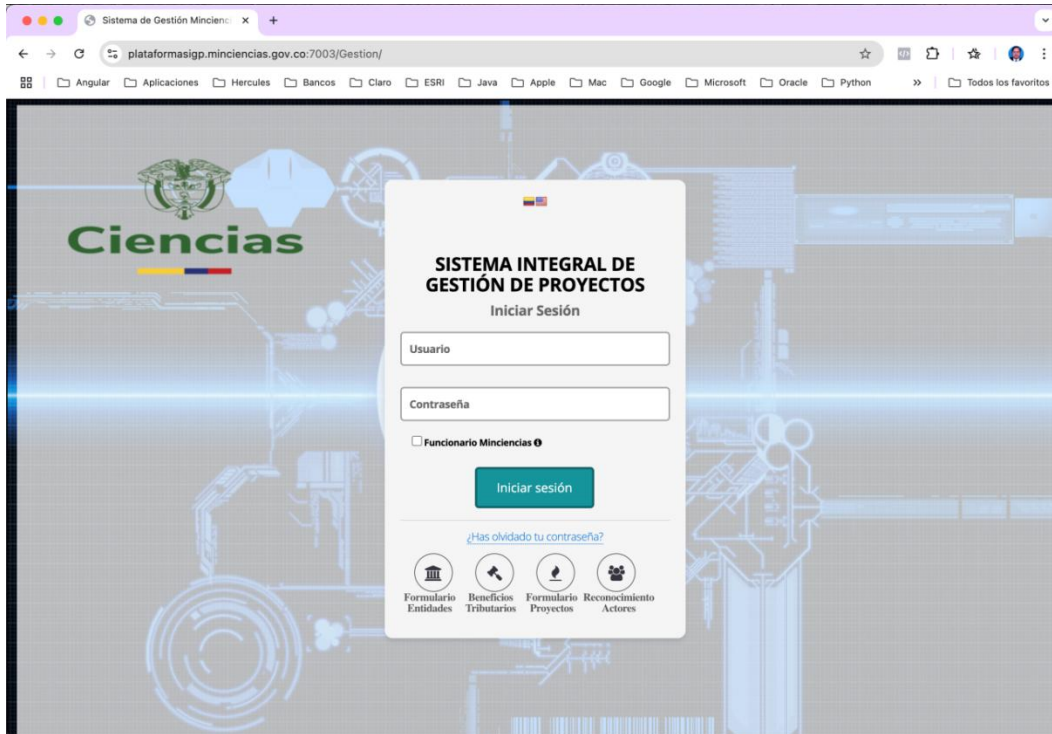
Se configura el Client SSL Profile LB_CLIENT_SSL_PROF_MINCIENCIAS:



Se revisa al ingreso el certificado instalado en el módulo de Gestión del SIGP:



Se revisa el ingreso por https a los módulos del SIGP:



Sistema de Gestión Minciencias

plataformasigp.minciencias.gov.co:7003/Gestion/

Angular Aplicaciones Hercules Bancos Claro ESRI Java Apple Mac Google Microsoft Oracle Python Todos los favoritos

Ciencias

SISTEMA INTEGRAL DE GESTIÓN DE PROYECTOS
Iniciar Sesión

Usuario

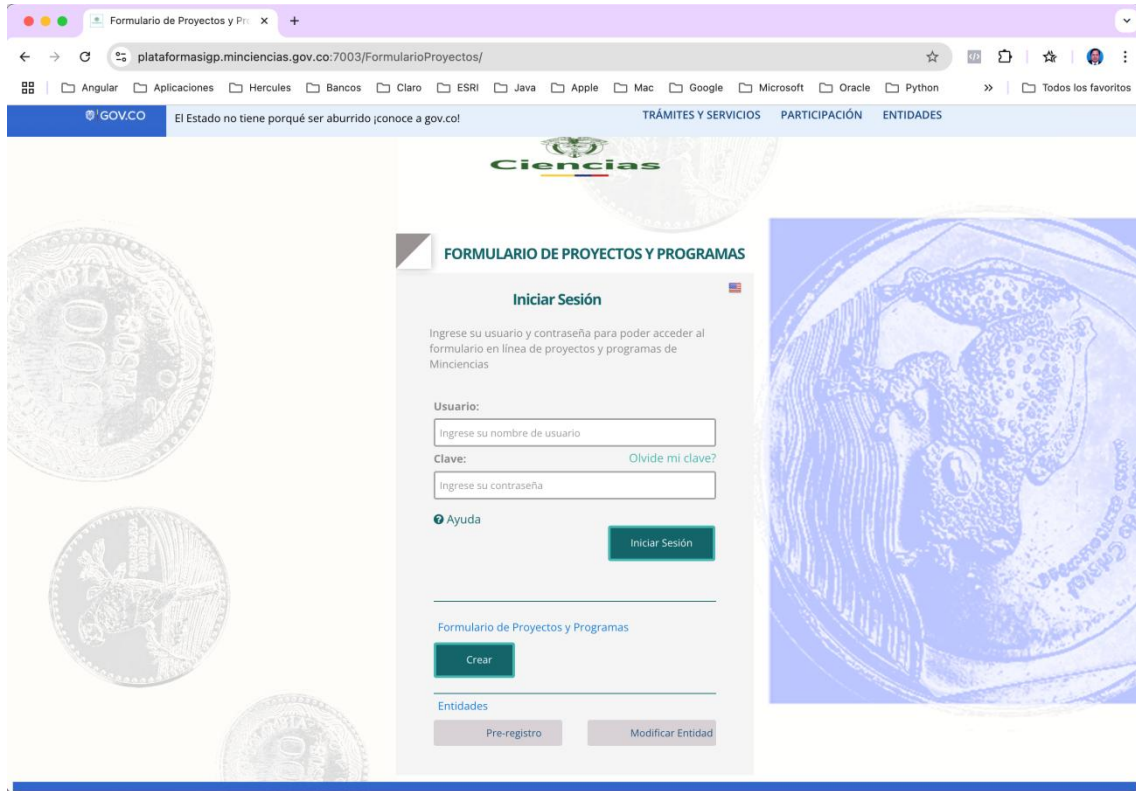
Contraseña

☐ Funcionario Minciencias

Iniciar sesión

[¿Has olvidado tu contraseña?](#)

Formulario Entidades Beneficios Tributarios Formulario Proyectos Reconocimiento Actores



Formulario de Proyectos y Programas

plataformasigp.minciencias.gov.co:7003/FormularioProyectos/

GOV.CO El Estado no tiene porqué ser aburrido ¡conoce a gov.co!

TRÁMITES Y SERVICIOS PARTICIPACIÓN ENTIDADES

FORMULARIO DE PROYECTOS Y PROGRAMAS

Iniciar Sesión

Ingrese su usuario y contraseña para poder acceder al formulario en línea de proyectos y programas de Minciencias

Usuario:
Ingrese su nombre de usuario

Clave:
Ingrese su contraseña

[Olvide mi clave?](#)

[Ayuda](#)

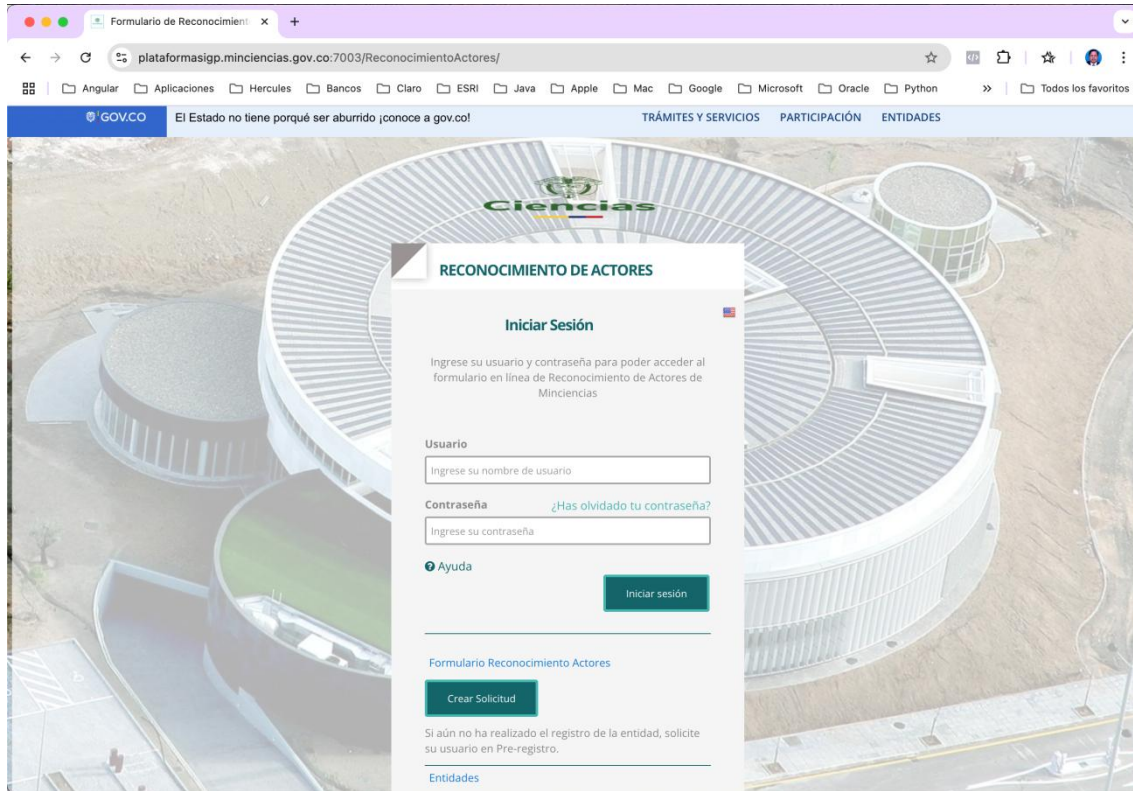
Iniciar Sesión

Formulario de Proyectos y Programas

Crear

Entidades

[Pre-registro](#) [Modificar Entidad](#)



Formulario de Reconocimiento: x +

plataformasigp.minciencias.gov.co:7003/ReconocimientoActores/

GOV.CO El Estado no tiene porqué ser aburrido ¡conoce a gov.co!

TRÁMITES Y SERVICIOS PARTICIPACIÓN ENTIDADES

RECONOCIMIENTO DE ACTORES

Iniciar Sesión

Ingrese su usuario y contraseña para poder acceder al formulario en línea de Reconocimiento de Actores de Minciencias

Usuario
Ingrese su nombre de usuario

Contraseña [¿Has olvidado tu contraseña?](#)
Ingrese su contraseña

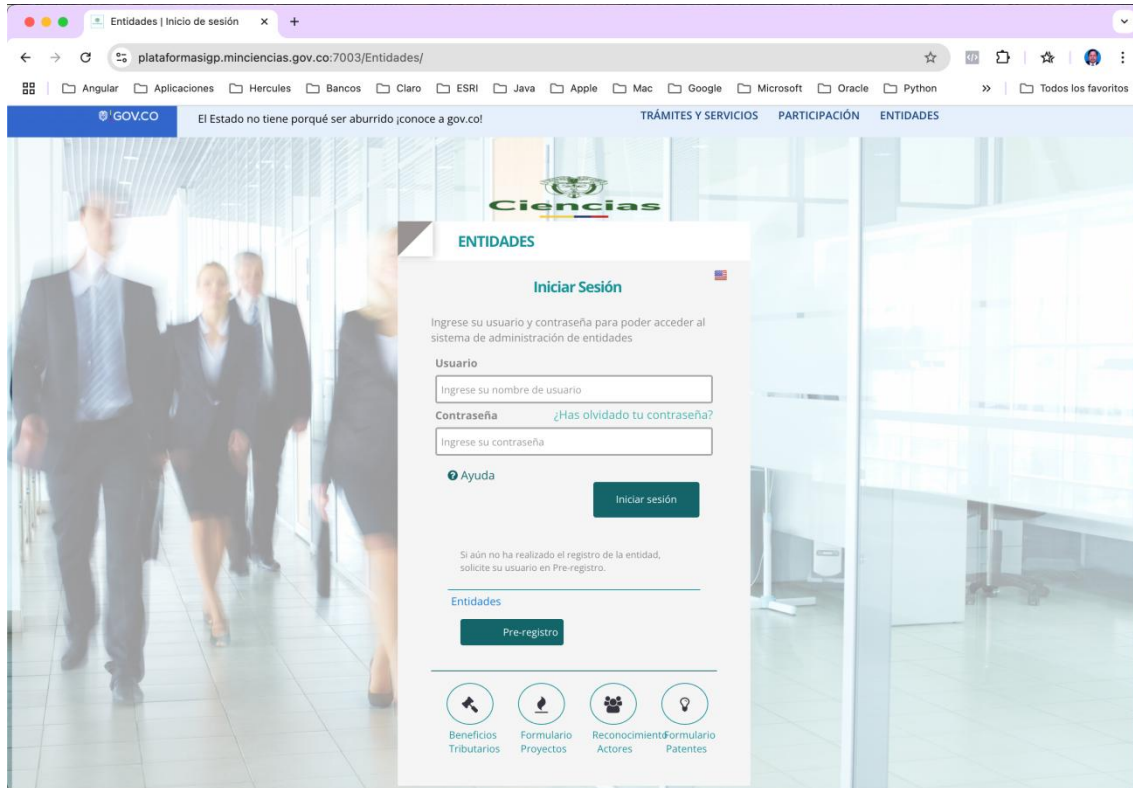
[Ayuda](#) [Iniciar sesión](#)

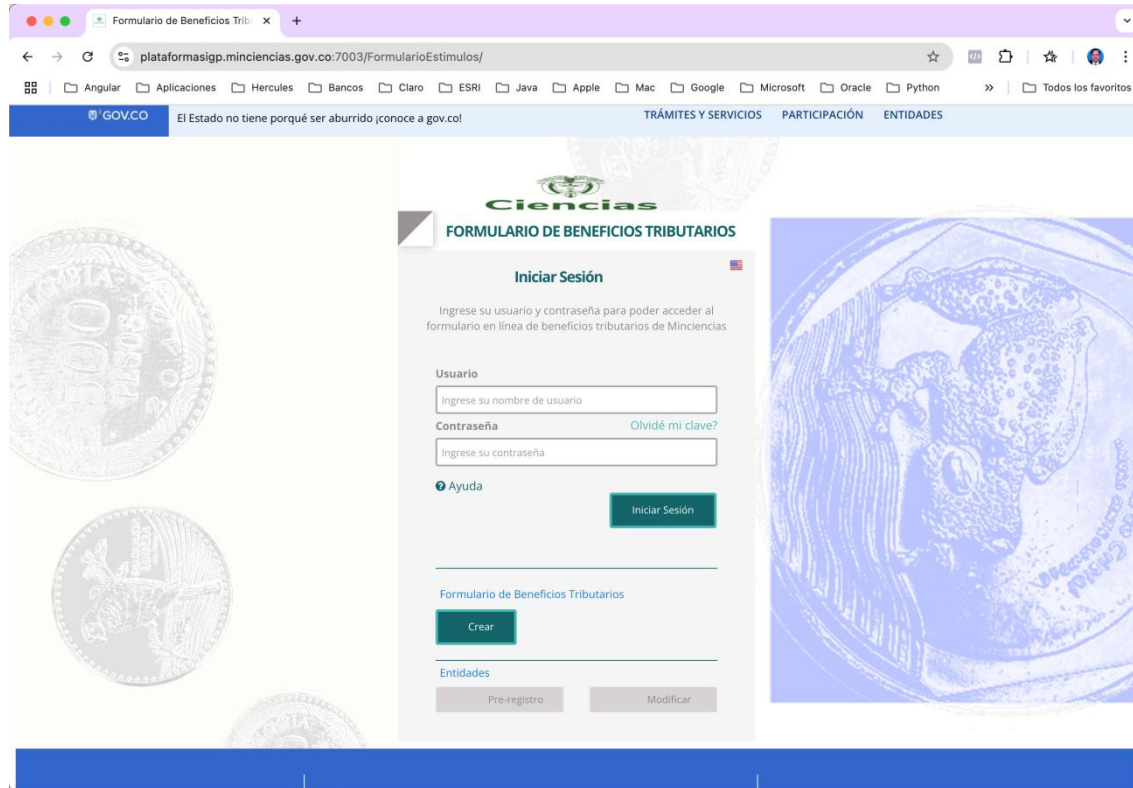
Formulario Reconocimiento Actores

[Crear Solicitud](#)

Si aún no ha realizado el registro de la entidad, solicite su usuario en Pre-registro.

[Entidades](#)





i. Migración del Módulo HÉRCULES a Nuevo Servidor

Solicitud de requerimiento No. 163310

Descripción y alcance del Requerimiento

El módulo HÉRCULES en ambiente de producción, encargado de la seguridad y auditoria del SIGP, se encuentra actualmente alojado en el servidor Kogui (172.16.100.47). El servidor Kogui aloja también al SIGP viejo utilizando al servidor de aplicaciones Oracle Application Server (OAS). La funcionalidad requerida del SIGP viejo será movida al SIGP actual en junio de este año y se requiere que el sistema HÉRCULES del ambiente productivo sea movido a otro servidor, con el fin de dar de baja al servidor Kogui, ya que no se tiene soporte del fabricante sobre OAS. Luego se requiere mover a Hércules del ambiente de producción al servidor Kaltapruebas (172.21.100.65).

Desarrollo e Implementación

Configurar DADs WebTier

Se configuran conexiones de mod_plsql en archivo /u01/oracle/product/fmw/P_OHS/12.1.2/user_projects/domains/base_domain/config/fmwconfig/components/OHS/instances/ohs1/mod_plsql/dads.conf:

```
# =====
#                               mod_plsql DAD Configuration File
# =====
<Location /hercules>
  SetHandler pls_handler
  Order allow,deny
  Allow from All
  AllowOverride None
  PlsqlDatabaseUsername hercules
  PlsqlDatabasePassword Hercules2025
  PlsqlDatabaseConnectionString scan-scienti:1534:sign ServiceNameFormat
  PlsqlNLSLanguage SPANISH_SPAIN.UTF8
  PlsqlAuthenticationMode CustomOwa
  PlsqlDocumentTablename hercules.sb_documentos
  PlsqlDocumentPath docs
  PlsqlDocumentProcedure sbweb.download
  PlsqlErrorStyle ApacheStyle
  PlsqlAlwaysDescribeProcedure Off
</Location>
<Location /hercules_sylk>
  SetHandler pls_handler
  Order allow,deny
  Allow from All
  AllowOverride None
  PlsqlDatabaseUsername hercules
  PlsqlDatabasePassword Hercules2025
  PlsqlDatabaseConnectionString scan-scienti:1534:sign ServiceNameFormat
  PlsqlNLSLanguage SPANISH_SPAIN.WE8MSWIN1252
  PlsqlAuthenticationMode Basic
  PlsqlRequestValidationFunction hercules.owa_custom_sylk
  PlsqlDocumentTablename hercules.sb_documentos
  PlsqlDocumentPath docs
  PlsqlDocumentProcedure sbweb.download
  PlsqlErrorStyle ApacheStyle
  PlsqlAlwaysDescribeProcedure Off
</Location>
```


Actualizar objetos y permisos

Se dan permisos en Producción sobre objetos de Hércules y se crean sinónimos:

```
GRANT EXECUTE ON sb_auditoria TO sigp_web WITH GRANT OPTION;
GRANT EXECUTE ON sb_trabajos TO sigp_web WITH GRANT OPTION;
GRANT EXECUTE ON sb_utilidad TO sigp_web WITH GRANT OPTION;
GRANT EXECUTE ON sql_auditoria TO sigp_web WITH GRANT OPTION;
GRANT EXECUTE ON f_descripcion_auditoria TO sigp_web WITH GRANT OPTION;
GRANT SELECT ON vw_auditorias TO sigp_web WITH GRANT OPTION;
CREATE PUBLIC SYNONYM sb_auditoria FOR hercules.sb_auditoria;
CREATE PUBLIC SYNONYM sb_trabajos FOR hercules.sb_trabajos;
CREATE PUBLIC SYNONYM sb_utilidad FOR hercules.sb_utilidad;
CREATE PUBLIC SYNONYM sql_auditoria FOR hercules.sql_auditoria;
CREATE PUBLIC SYNONYM f_descripcion_auditoria FOR hercules.f_descripcion_auditoria;
CREATE PUBLIC SYNONYM vw_auditorias FOR hercules.vw_auditorias;
```

Se actualizan tablas de Hércules en Producción:

```
-- Add/modify columns
alter table SB_TABLAS add aplicacion_id NUMBER(15);
-- Add comments to the columns
comment on column SB_TABLAS.aplicacion_id
is 'Referencia a la Aplicación a la que pertenece la tabla';

-- Add/modify columns
alter table SB_CAMPOS add tabla_id NUMBER(15);
alter table SB_CAMPOS add usuario_fk VARCHAR2(30);
-- Add comments to the columns
comment on column SB_CAMPOS.tabla_id
is 'Referencia a la Tabla';
comment on column SB_CAMPOS.usuario_fk
is 'Usuario propietario de la Tabla Referenciada';

-- Add/modify columns
alter table SB_MENSAJES_ERRORES add aplicacion_id NUMBER(15);
alter table SB_MENSAJES_ERRORES add tabla_id NUMBER(15);
-- Add comments to the columns
comment on column SB_MENSAJES_ERRORES.aplicacion_id
is 'Referencia a la Aplicación a la que pertenece la tabla';
comment on column SB_MENSAJES_ERRORES.tabla_id
is 'Referencia a la Tabla';
```

Se actualizan vistas sobre el diccionario de datos:

```
CREATE OR REPLACE VIEW seg_col_comentarios AS
SELECT owner      usuario
      , table_name tabla
      , column_name columna
      , comments   comentario
FROM dba_col_comments;

CREATE OR REPLACE VIEW seg_tab_comentarios AS
SELECT owner      usuario
      , table_name tabla
      , table_type tipo
      , comments   comentario
FROM dba_tab_comments;
```

```
CREATE OR REPLACE VIEW seg_jobs_database AS
SELECT j.owner          propietario
      , j.job_name       nombre_job
      , j.comments       comentario
      , j.job_type       tipo_job
      , j.job_action      accion
      , j.start_date      fecha_inicial
      , j.repeat_interval intervalo_repeticion
      , j.job_class       clase_job
      , j.enabled         activo
      , j.auto_drop       auto_borrado
      , j.state           estado -- SCHEDULED - RUNNING - DISABLED - SUCCEEDED
      , j.last_start_date ultima_corrida
      , j.last_run_duration ultima_duracion
      , j.next_run_date    siguiente_corrida
      , j.program_owner   programa_propietario
      , j.program_name    programa
FROM dba_scheduler_jobs j
ORDER BY j.owner, j.job_name
;
```

Se crea la función f_descripcion_auditoria:

```
CREATE OR REPLACE FUNCTION f_descripcion_auditoria(p_aplicacion IN VARCHAR2,
                                                    p_tabla      IN VARCHAR2,
                                                    p_campo      IN VARCHAR2,
                                                    p_valor       IN VARCHAR2)
RETURN VARCHAR2 IS
  v_usuario_fk   sb_campos.usuario_fk%TYPE;
  v_tabla_fk     sb_campos.tabla_fk%TYPE;
  v_campo_fk     sb_campos.campo_fk%TYPE;
  v_descripcion_fk sb_tablas.campos_descripcion%TYPE;
  v_descripcion  VARCHAR2(4000);
  v_query        VARCHAR2(4000);
  v_dinamico     INTEGER;
  v_res          INTEGER;

  CURSOR c_campo IS
  SELECT c.usuario_fk
        , c.tabla_fk
        , c.campo_fk
        , t.campos_descripcion descripcion_fk
  FROM sb_campos c
        , sb_tablas t
        , sb_aplicaciones a
  WHERE c.aplicacion = p_aplicacion
        AND c.tabla = p_tabla
        AND c.campo = p_campo
        AND t.tabla = c.tabla_fk
        AND a.aplicacion = t.aplicacion
        AND a.usuario = c.usuario_fk
  ORDER BY c.posicion;

  CURSOR c_tabla IS
  SELECT a.usuario
        , t.campos_descripcion
  FROM sb_campos c
        , sb_tablas t
```



```

, sb_aplicaciones a
WHERE c.aplicacion = p_aplicacion
AND c.tabla = p_tabla
AND c.campo = p_campo
AND c.llave_primaria = 'S'
AND t.aplicacion = c.aplicacion
AND t.tabla = c.tabla
AND a.aplicacion = c.aplicacion;
BEGIN
OPEN c_campo;
FETCH c_campo INTO v_usuario_fk, v_tabla_fk, v_campo_fk, v_descripcion_fk;
IF c_campo%FOUND THEN
-- Armar Consulta
IF v_usuario_fk IS NOT NULL AND
v_tabla_fk IS NOT NULL AND
v_campo_fk IS NOT NULL AND
v_descripcion_fk IS NOT NULL THEN
v_query := 'SELECT '||v_descripcion_fk||
' FROM '||v_usuario_fk||'.'||v_tabla_fk||
' WHERE '||v_campo_fk||' = :valor';
--dbms_output.put_line(v_query);
-- Preparar y Ejecutar Consulta
v_dinamico := DBMS_SQL.open_cursor;
DBMS_SQL.PARSE(v_dinamico, v_query, DBMS_SQL.NATIVE);
DBMS_SQL.BIND_VARIABLE(v_dinamico,':valor',p_valor);
DBMS_SQL.DEFINE_COLUMN(v_dinamico,1,v_descripcion,4000);
v_res := DBMS_SQL.EXECUTE(v_dinamico);
IF v_res != 0 THEN
DBMS_OUTPUT.put_line(v_res);
END IF;
IF DBMS_SQL.FETCH_ROWS(v_dinamico) > 0 THEN
DBMS_SQL.COLUMN_VALUE(v_dinamico,1,v_descripcion);
END IF;
DBMS_SQL.close_cursor(v_dinamico);
ELSE
v_usuario_fk := NULL;
v_descripcion_fk := NULL;
OPEN c_tabla;
FETCH c_tabla INTO v_usuario_fk, v_descripcion_fk;
CLOSE c_tabla;
IF v_usuario_fk IS NOT NULL AND
v_descripcion_fk IS NOT NULL THEN
v_query := 'SELECT '||v_descripcion_fk||
' FROM '||v_usuario_fk||'.'||p_tabla||
' WHERE '||p_campo||' = :valor';
--dbms_output.put_line(v_query);
-- Preparar y Ejecutar Consulta
v_dinamico := DBMS_SQL.open_cursor;
DBMS_SQL.PARSE(v_dinamico, v_query, DBMS_SQL.NATIVE);
DBMS_SQL.BIND_VARIABLE(v_dinamico,':valor',p_valor);
DBMS_SQL.DEFINE_COLUMN(v_dinamico,1,v_descripcion,4000);
v_res := DBMS_SQL.EXECUTE(v_dinamico);
IF v_res != 0 THEN
DBMS_OUTPUT.put_line(v_res);
END IF;
IF DBMS_SQL.FETCH_ROWS(v_dinamico) > 0 THEN
DBMS_SQL.COLUMN_VALUE(v_dinamico,1,v_descripcion);
END IF;
DBMS_SQL.close_cursor(v_dinamico);
END IF;
END IF;

```

```
END IF;
CLOSE c_campo;
IF v_descripcion IS NULL THEN
    v_descripcion := p_valor;
END IF;
RETURN(v_descripcion);
END f_descripcion_auditoria;
/
```

Se crea la función sql_auditoria:

```
CREATE OR REPLACE FUNCTION sql_auditoria(p_aplicacion IN VARCHAR2
                                         ,p_tabla      IN VARCHAR2
                                         ,p_campo_llave IN VARCHAR2
                                         ,p_valor_llave IN VARCHAR2
                                         ,p_fecha_inicial IN VARCHAR2
                                         ,p_fecha_final  IN VARCHAR2
                                         )
RETURN VARCHAR2 IS
    v_comando      VARCHAR2(32767);
    v_campo_llave  VARCHAR2(4000);
    v_antes        VARCHAR2(4000);
    v_despues      VARCHAR2(4000);
    v_campo        VARCHAR2(30);
    v_usuario      VARCHAR2(30);
    v_tipo_dato    VARCHAR2(20);

    -- Leer usuario de aplicación
    CURSOR c_aplicacion IS
    SELECT a.usuario
    FROM sb_aplicaciones a
    WHERE a.aplicacion = p_aplicacion;

    -- Leer tipo de dato campo llave
    CURSOR c_tabla IS
    SELECT c.tipo_dato
    FROM sb_campos c
    WHERE c.aplicacion = p_aplicacion
    AND c.tabla = p_tabla
    AND c.campo = p_campo_llave;
BEGIN
    v_campo := f_llave_primaria(p_aplicacion,p_tabla);

    OPEN c_aplicacion;
    FETCH c_aplicacion INTO v_usuario;
    CLOSE c_aplicacion;

    OPEN c_tabla;
    FETCH c_tabla INTO v_tipo_dato;
    CLOSE c_tabla;

    v_campo_llave := 'AND d.campo      = ''' || [[CAMPO]] || '''
                    AND d. [[VALOR]] IN (SELECT [[CAMPO]] FROM [[USUARIO]]. [[TABLA]] WHERE [[CAMPO_LLAVE]]
= [[VALOR_LLAVE]])';
    v_comando := 'SELECT au.auditoria_id
                , au.detalle_auditoria_id
                , au.fecha_auditoria
                , au.operacion
                , au.usuario
```

```

, au.terminal
, au.campo
, au.valor_antes
, au.valor_despues
, au.codigo_valor_antes
, au.codigo_valor_despues
FROM vw_auditorias au
, (SELECT a.auditoria_id
    FROM sb_auditorias a
    , sb_detalle_auditorias d
    WHERE TRUNC(a.fecha_hora) BETWEEN TO_DATE('[[FECHA_INICIAL]]','DD/MM/YYYY') AND
TO_DATE('[[FECHA_FINAL]]','DD/MM/YYYY')
    AND a.aplicacion = '[[APLICACION]]'
    AND a.tabla = '[[TABLA]]'
    AND d.auditoria_id = a.auditoria_id
    [[ANTES]]
UNION
SELECT a.auditoria_id
    FROM sb_auditorias a
    , sb_detalle_auditorias d
    WHERE TRUNC(a.fecha_hora) BETWEEN TO_DATE('[[FECHA_INICIAL]]','DD/MM/YYYY') AND
TO_DATE('[[FECHA_FINAL]]','DD/MM/YYYY')
    AND a.aplicacion = '[[APLICACION]]'
    AND a.tabla = '[[TABLA]]'
    AND d.auditoria_id = a.auditoria_id
    [[DESPUES]]
) da
WHERE au.auditoria_id = da.auditoria_id
ORDER BY au.auditoria_id, au.llave_primaria DESC, au.detalle_auditoria_id
';
v_comando := REPLACE(v_comando,'[[FECHA_INICIAL]]',p_fecha_inicial);
v_comando := REPLACE(v_comando,'[[FECHA_FINAL]]',p_fecha_final);
v_comando := REPLACE(v_comando,'[[APLICACION]]',p_aplicacion);
v_comando := REPLACE(v_comando,'[[TABLA]]',p_tabla);
IF p_campo_llave IS NOT NULL AND p_valor_llave IS NOT NULL THEN
    v_campo_llave := REPLACE(v_campo_llave,'[[CAMPO_LLAVE]]',p_campo_llave);
    v_campo_llave := REPLACE(v_campo_llave,'[[CAMPO]]',v_campo);
    v_campo_llave := REPLACE(v_campo_llave,'[[USUARIO]]',v_usuario);
    v_campo_llave := REPLACE(v_campo_llave,'[[TABLA]]',p_tabla);
    IF v_tipo_dato IN ('INTEGER','NUMBER') THEN
        v_campo_llave := REPLACE(v_campo_llave,'[[VALOR_LLAVE]]',p_valor_llave);
    ELSE
        v_campo_llave := REPLACE(v_campo_llave,'[[VALOR_LLAVE]]',''||p_valor_llave||'');
    END IF;
    v_antes := REPLACE(v_campo_llave,'[[VALOR]]','valor_antes');
    v_comando := REPLACE(v_comando,'[[ANTES]]',v_antes);
    v_despues := REPLACE(v_campo_llave,'[[VALOR]]','valor_despues');
    v_comando := REPLACE(v_comando,'[[DESPUES]]',v_despues);
ELSE
    v_comando := REPLACE(v_comando,'[[ANTES]]','');
    v_comando := REPLACE(v_comando,'[[DESPUES]]','');
END IF;

RETURN(v_comando);
END sql_auditoria;
/

```

Se actualiza la vista vw_auditorias:

```
CREATE OR REPLACE VIEW VW_AUDITORIAS AS
SELECT /*+ index(a audito_fecha_hora_i)
        index(d detaud_uk)
        */
    a.auditoria_id
  , TO_CHAR(a.fecha_hora,'DD/MM/YYYY HH24:MI:SS') fecha_auditoria
  , a.aplicacion
  , a.tabla
  , DECODE(a.operacion,
    'A','ACTUALIZAR',
    'B','BORRAR',
    'C','CREAR',
    'D','DESCARGAR',
    'E',DECODE(a.tabla,'ACCESO AL SISTEMA','INGRESAR','CONSULTAR'),
    'S','SALIR') operacion
  , a.usuario
  , a.terminal
  , d.detalle_auditoria_id
  , d.campo
  , f_descripcion_auditoria(a.aplicacion,a.tabla,d.campo,d.valor_antes) valor_antes
  , f_descripcion_auditoria(a.aplicacion,a.tabla,d.campo,d.valor_despues) valor_despues
  , d.valor_antes codigo_valor_antes
  , d.valor_despues codigo_valor_despues
  , NVL(d.llave_primaria,DECODE(d.campo,c.campo,1,0)) llave_primaria
  , d.verificacion
FROM sb_auditorias a
  , sb_detalles_auditorias d
  , (SELECT ca.aplicacion
    , ca.tabla
    , ca.campo
    , ca.posicion
    FROM sb_campos ca
    WHERE ca.llave_primaria = 'S'
  ) c
WHERE d.auditoria_id = a.auditoria_id
AND c.aplicacion (+)= a.aplicacion
AND c.tabla (+)= a.tabla
AND c.posicion (+)= 1;
```

Reiniciar servicios de WebTier

Se reinician los servicios de Webtier:

```
# service httpora stop
# service httpora start
```

j. Renderizar Campos Formulario Beneficios Tributarios

Solicitud de requerimiento No. 160870

Descripción y alcance del Requerimiento

Se solicita la eliminación de los siguientes botones dentro del aplicativo del Sistema de Información de Gestión de Proyectos (SIGP):

- ¿Vincula Proyecto Financiado por Minciencias?
- ¿Registra Código BPIN?
- ¿Registra Número Contrato?

Estos elementos se encuentran actualmente habilitados en el formulario de registro de proyectos. El ajuste consiste en poder deshabilitar la visualización de estos botones, garantizando que no generen alertas, errores o validaciones en el flujo del proceso de inscripción de proyectos en convocatorias vigentes o futuras.

La eliminación de estos botones responde a la necesidad de asegurar la alineación del aplicativo SIGP con los lineamientos actuales establecidos en el documento de Tipologías de Proyectos versión 8. En esta versión, se establece que todos los proyectos deben ser evaluados nuevamente en el marco de cada convocatoria. Por tanto, los contenidos, características y estructuras de los proyectos antiguos no son válidos ni comparables con los criterios actualizados, lo que hace innecesaria y confusa la solicitud de información relacionada con códigos BPIN, números de contrato anteriores o financiación previa por parte de Minciencias.

Eliminar estos campos redundantes optimiza el proceso de inscripción, evita errores de interpretación por parte de los usuarios y mejora la trazabilidad de los proyectos en función de los criterios actuales.

Desarrollo e Implementación

Para el desarrollo de este requerimiento, se busco parametrizar los campos indicados, permitiendo así su flexibilidad al momento de ser visibles o no por convocatoria.

Se realizo lo siguiente:

- Crear registros de campos para ser parametrizados por convocatoria

--Campos

```
insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
values (null, 'CAMPOSPRY', '480', 'VINCULA PROYECTO FINANCIADO (BT)', 'S', 'CARACTER', null,
null, null, null, null, 'SIGP_WEB', to_date('01-07-2025', 'dd-mm-yyyy'), null, null, null,
null, null, null, null, null);
```

```
insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
```

```
values (null, 'CAMPOSPRY', '490', 'CÓDIGO BPIN (BT)', 'S', 'CARACTER', null, null, null, null,
null, 'SIGP_WEB', to_date('01-07-2025', 'dd-mm-yyyy'), null, null, null, null, null, null,
null, null);
```

```
insert into adm_valores_definiciones (VALOR_DEFINICION_ID, TIPO_DEFINICION, VALOR_DEFINICION,
DESCRIPCION_VALOR_DEFINICION, ACTIVO, TIPO_DATO, TAMANO, DECIMALES, COMENTARIO, OBLIGATORIO,
TIPO_VALIDACION, USUARIO_CREADOR, FECHA_CREACION, USUARIO_MODIFICADOR, FECHA_MODIFICACION,
ORDEN, VALOR_POND_I, VALOR_POND_D, VALOR_POND_A, VALOR_POND_B, VALOR_POND_S)
values (null, 'CAMPOSPRY', '500', 'NÚMERO CONTRATO (BT)', 'S', 'CARACTER', null, null, null,
null, null, 'SIGP_WEB', to_date('01-07-2025', 'dd-mm-yyyy'), null, null, null, null, null,
null, null, null);
```

- Agregar en el formulario de beneficios tributarios lógica para mostrar el campo de Proyecto Financiado por Minciencias

```
<!-- VINCULA PROYECTO FINANCIADO POR MINCIENCIAS -->
<ui:fragment rendered="#{proyectoSessController.renderCampo('480', 'CAMPOSPRY')}">
    <div class="row">
        <div class="col-lg-6">
            <div class="form-group">
                <p:outputLabel for="proyectoFinanciado"
                    value="#{appSessController.proyectoSelected.convocatoriaId.t
ipoTramiteId.valorDefinicion ne '040'
                    ?
msg['formulario.estimulos.proyectos.proyectoFinanBoolean.label']
                    :
msg['formulario.estimulos.proyectos.proyectoVincDoctoresBoolean.label']}"
                    styleClass="col-lg-6 control-label"
                    />
                <div class="col-lg-6">
                    <p:toggleSwitch id="proyectoFinanciado"
                        value="#{appSessController.proyectoSelected.proyecto
FinanciadoSINO}"
                        disabled="#{!proyectoConvController.renderRegistroPr
oyecto() or proyectoSessController.listProyectoFinanciado.size() > 0}"
                        required="false"
                        >
                        <f:passThroughAttribute
                            name="aria-label"
                            value="#{msg['formulario.estimulos.proyectos.proyectoFinanBoolea
n.label.placeholder']}"
                        />
                    </p:toggleSwitch>
                </div>
            </div>
        </div>
        <div class="col-lg-6" jsf:id="messageProFin">
            <ui:fragment rendered="#{appSessController.proyectoSelected.proyectoFinanciadoSINO
and proyectoSessController.listProyectoFinanciado != null}">
                <div class="form-group">
                    <div tabindex="0" class="alert alert-info alert-dismissable">
                        <button aria-hidden="true" data-dismiss="alert" class="close"
type="button"></button>
                        #{msg['formulario.estimulos.proyectos.proyectoFinanBoolean.info']}
                    </div>
                </div>
            </ui:fragment>
        </div>
    </div>
</div>
```

```
</ui:fragment>
```

- Agregar en el formulario de beneficios tributarios lógica para mostrar el campo de Código BPIN

```
<!-- CÓDIGO BPIN -->
<ui:fragment rendered="#{proyectoSessController.renderCampo('490', 'CAMPOSPRY')}">
  <div class="row">
    <div class="col-lg-6">
      <div class="form-group">
        <p:outputLabel for="codigoBPINBool"
          value="#{msg['formulario.estimulos.proyectos.codigoBPINBoole
an.label']}"
          styleClass="col-lg-6 control-label"
        />
        <div class="col-lg-6">
          <p:toggleSwitch id="codigoBPINBool"
            value="#{appSessController.proyectoSelected.registraCodi
goBPINSINO}"
            required="false"
          >
            <f:passThroughAttribute
              name="aria-label"
              value="#{msg['formulario.estimulos.proyectos.codigoBPINBoolean.l
abel.placeholder']}"
            />
            <p:ajax event="change"
              update="panelCodigoBPIN"
              oncomplete="$('##{proyectoSessController.modulo}Form\\\\\\\\codi
goBPINBool').focus();"
            />
          </p:toggleSwitch>
        </div>
      </div>
    </div>
    <h:panelGroup layout="block" id="panelCodigoBPIN">
      <ui:fragment
        rendered="#{appSessController.proyectoSelected.registraCodigoBPINSINO}">
        <div class="col-lg-6">
          <div class="form-group">
            <p:outputLabel
              value="#{msg['formulario.estimulos.proyectos.codigoBPIN.label']}"
              for="codigoBPIN"
              styleClass="col-lg-4 control-label"/>
            <div class="col-lg-8">
              <p:inputTextarea
                id="codigoBPIN"
                required="true"
                value="#{appSessController.proyectoSelected.
codigoBPIN}"
                requiredMessage="#{msg['formulario.estimulos
.proyectos.codigoBPIN.required']}"
                styleClass="form-control inputArea"
                placeholder="#{msg['formulario.estimulos.pro
yectos.codigoBPIN.placeholder']}"
                counter="displayCodigoBPIN" maxLength="30"
                counterTemplate="{0} #{msg['sb.mensajes.caracteres.label']}"
                rows="1"
              />
            </div>
          </div>
        </div>
      </ui:fragment>
    </h:panelGroup>
  </div>
</ui:fragment>
```

```

        />
        <h:outputText id="displayCodigoBPIN" style="font-style: italic"
    />
    </div>
  </div>
</div>
</ui:fragment>
</h:panelGroup>
</div>
</ui:fragment>

```

- Agregar en el formulario de beneficios tributarios lógica para mostrar el campo de Número Contrato

```

<!-- NÚMERO CONTRATO -->
<ui:fragment rendered="#{proyectoSessController.renderCampo('500', 'CAMPOSPRY')}">
  <div class="row">
    <div class="col-lg-6">
      <div class="form-group">
        <p:outputLabel for="numeroContratoBool"
          value="#{msg['formulario.estimulos.proyectos.numeroContratoBo
oolean.label']}"
          styleClass="col-lg-6 control-label"
        />
        <div class="col-lg-6">
          <p:toggleSwitch id="numeroContratoBool"
            value="#{appSessController.proyectoSelected.registraNume
roContratoSINO}"
            required="false"
          >
            <f:passThroughAttribute
              name="aria-label"
              value="#{msg['formulario.estimulos.proyectos.numeroContratoBoole
an.label.placeholder']}"
            />
            <p:ajax event="change"
              update="panelNumeroContrato"
              oncomplete="$('##{proyectoSessController.modulo}Form\\\\\\\\nume
roContratoBool').focus();"
            />
          </p:toggleSwitch>
        </div>
      </div>
    </div>
    <h:panelGroup layout="block" id="panelNumeroContrato">
      <ui:fragment
        rendered="#{appSessController.proyectoSelected.registraNumeroContratoSINO}">
        <div class="col-lg-6">
          <div class="form-group">
            <p:outputLabel
              value="#{msg['formulario.estimulos.proyectos.numeroContrato.label']}"
              for="numeroContrato"
              styleClass="col-lg-4 control-label"/>
            <div class="col-lg-8">
              <p:inputTextarea
                id="numeroContrato"
                required="true"

```



```

numeroContrato}"
value="#{appSessController.proyectoSelected.
requiredMessage="#{msg['formulario.estimulos
.styleClass="form-control inputArea"
placeholder="#{msg['formulario.estimulos.pro
yectos.numeroContrato.placeholder']}"
counter="displayNumeroContrato" maxLength="30"
counterTemplate="{0} #{msg['sb.mensajes.caracteres.label']}"
rows="1"
/>
<h:outputText id="displayNumeroContrato" style="font-style: italic"
/>
</div>
</div>
</div>
</ui:fragment>
</h:panelGroup>
</div>
</ui:fragment>

```

- Se ajusta lógica de cargue de campos parametrizados por convocatoria

```

public boolean renderCampo(String campo, String pantalla) {
    HashMap<String, Boolean> camposValor;
    boolean render = true;
    HashMap<String, Boolean> mapCampos =
appSessController.getMapCampos();
    if (mapCampos != null) {
        if (mapCampos.containsKey(pantalla)) {
            camposValor = mapCampos.get(pantalla);
            if (camposValor.containsKey(campo)) {
                render = camposValor.get(campo);
            }
        }
    }
    return render;
}

```

- Para que estos campos no se visualicen en el formulario de beneficios tributarios, en la convocatoria deben ser parametrizados en el menú de Campos y ponerlos como Visible = 'NO'.

k. Apoyo Parametrización Rubrica Conv 965-2025 Movilidades Académicas con Europa

Solicitud de requerimiento No. 161086

Descripción y alcance del Requerimiento

Se solicita el apoyo de la OTSI en la parametrización en SIGP, de la rúbrica de evaluación de la Convocatoria No. 965 de 2025 "Movilidades Académicas con Europa 2025".

Dado el cronograma de la convocatoria, se requiere contar con la rúbrica parametrizada para el miércoles 9 de julio, día en el cual se hacen las pruebas de parametrización para aprobación el 10 de julio.

La rúbrica fue aprobada por la directora de Vocaciones y Formación - Diana Rúa

Desarrollo e Implementación

El requerimiento consistió en brindar **soporte y apoyo a la Dirección de Vocaciones y Formación** para la **parametrización de la rúbrica de evaluación** correspondiente a la **Convocatoria No. 965 de 2025 “Movilidades Académicas con Europa 2025”** en el sistema SIGP.

Se realizó la configuración de los criterios, ponderaciones y aspectos definidos en la rúbrica aprobada por la Dirección, asegurando su correcta incorporación en el módulo de convocatorias del sistema. Posteriormente, se efectuaron pruebas de verificación funcional que permitieron validar la coherencia y el correcto funcionamiento de la configuración implementada.

Como parte de este proceso, se recibió retroalimentación por parte de la usuaria **Carolina Munera**, quien realizó pruebas en el SIGP y confirmó que:

- La rúbrica de evaluación parametrizada se encontraba acorde con lo solicitado, autorizando su apertura para el inicio del proceso de evaluación.
- El proceso de evaluación se llevaría a cabo del **18 al 28 de julio de 2025**.
- Las propuestas de I+D+i con puntaje igual o superior a **70 puntos** conformarían el banco de propuestas elegibles.
- Se debía eliminar la prueba de evaluación del proyecto **ID 115206 “Justicia, ethos para la paz, comprensión...”** utilizada durante las pruebas iniciales.

Con esto se dio cumplimiento al requerimiento, asegurando la disponibilidad y correcta configuración de la rúbrica en el tiempo establecido, habilitando así el proceso de evaluación de la convocatoria.

INFORME DE ACTIVIDADES DE SOPORTE EN EL USO DEL SIGP EN LA CREACION DE CONVOCATORIAS Y TRANSFERIR EL CONOCIMIENTO DEL SIGP A LOS FUNCIONARIOS DE MINCIENCIAS, (No. Obligación Contrato: 2.2.3, 2.2.4)

Fecha: 17/Julio/2025 – 15/Agosto/2025
Elaborado por: Leonardo Andres Suarez Baldovino
Entidad: Ministerio de Ciencias, Tecnología e Innovación - Minciencias

Introducción

El presente informe detalla las actividades de soporte realizadas en el uso del Sistema de Información para la Gestión de Proyectos (SIGP), abarcando la parametrización de convocatorias, la generación de reportes, la solución de requerimientos y la resolución de errores.

Durante este período, se atendieron diversas solicitudes reportadas por los usuarios en la plataforma. A continuación, se presenta el resumen de las actividades realizadas, agrupadas por tipo de solicitud.

Resumen de Actividades

Tipo de Solicitud	Cantidad de Casos
Solución de Errores	1
Total	1

Detalle de Casos Atendidos

A continuación, se detallan las solicitudes gestionadas, incluyendo la descripción del caso, su estado y la acción realizada:

Solución de Errores

Se solucionaron errores reportados por los usuarios en la plataforma:

N° Solicitud	Descripción	Estado	Observaciones	Acción Realizada
162266	- Error Falta Criterio Evaluación Proyecto 116129	Solucionado	-	[Ver detalle]

Solicitud 162266

Asunto		Tiempo total de actividad	
Convocatoria No. 965 de 2025		00:06:24	
Descripción			
<p>Reciban un cordial saludo.</p> <p>Al revisar el reporte de evaluación en el SIGP del 29 de julio de 2025 de la Convocatoria No. 965 de 2025 "Movilidades Académicas con Europa", se evidencia que el proyecto ID 116129 de la movilidad de Francia no registra el subcriterio 4.1 para el evaluador 1 (EVA 1), por lo cual se solicita revisar en SIGP la novedad dado que el Evaluador 2 si tiene registro del subcriterio 4.1 lo cual hace parte de la rúbrica de evaluación.</p> <p>Se adjuntan los reportes de evaluación de los dos evaluadores del proyecto ID 116129.</p> <p>Quedamos atentos a sus comentarios.</p>			
Fecha/hora de apertura	Última modificación	Fecha/hora de resolución	Fecha/hora de cierre
31/07/2025 12:01:30	04/08/2025 16:00:00	01/08/2025 17:31:15	04/08/2025 16:00:00

Acción Realizada 162266

Se realizó la revisión correspondiente frente a la novedad reportada del proyecto ID 116129, relacionado con la ausencia del subcriterio 4.1 en la evaluación del Evaluador 1.

Al verificar los criterios de evaluación asociados al evaluador en el sistema, se identificó que contaba con únicamente 27 criterios asignados, mientras que la rúbrica de la convocatoria contempla un total de 28 criterios. Se evidenció que el criterio faltante era precisamente el 4.1. Al profundizar en la revisión, se encontró que dicho criterio se encontraba erróneamente asociado a

otra evaluación del mismo evaluador, correspondiente al proyecto ID 116121, el cual tenía 30 criterios asignados, excediendo el número definido por la rúbrica (29 criterios).

Aunque no fue posible identificar con certeza el origen del error, se trató de un caso atípico que afectó puntualmente las evaluaciones de este evaluador.

Como medida correctiva, se procedió por base de datos a reasignar el criterio sobrante de la evaluación del proyecto 116121 al proyecto 116129, al cual le correspondía. Adicionalmente, se realizó el recalcu de los puntajes de ambas evaluaciones para garantizar la integridad de los resultados.

- Soporte proyecto con 30 registros

	CALIFICACION_EVALUACION_ID	PROYECTO_ID	EVALUACION_PROYECTO_ID	FORMATO_EVALUACION_ID	ORDEN	ENTRABLE	CALIFICACION	USUARIO_CREADOR	FECHA_CREACION	USUARIO
1	932405	116121	182520	15886	41	S	10,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MG
2	932620	116121	182520	15841	10	N	0,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
3	932621	116121	182520	15846	11	S	4,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
4	932622	116121	182520	15847	12	S	4,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
5	932623	116121	182520	15848	13	S	3,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
6	932624	116121	182520	15849	14	S	3,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
7	932625	116121	182520	15850	15	S	4,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
8	932626	116121	182520	15836	16	N	18,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
9	932627	116121	182520	15842	20	N	0,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
10	932628	116121	182520	15851	21	S	8,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
11	932629	116121	182520	15837	22	N	8,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
12	932630	116121	182520	15843	30	N	0,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
13	932631	116121	182520	15852	31	S	10,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
14	932632	116121	182520	15853	32	S	3,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
15	932633	116121	182520	15854	33	S	4,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
16	932634	116121	182520	15838	34	N	17,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
17	932635	116121	182520	15844	40	N	0,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
18	932636	116121	182520	15855	41	S	4,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
19	932637	116121	182520	15856	42	S	3,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
20	932638	116121	182520	15857	43	S	10,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
21	932639	116121	182520	15858	44	S	5,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
22	932640	116121	182520	15839	45	N	22,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
23	932641	116121	182520	15845	50	N	0,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
24	932642	116121	182520	15859	51	S	5,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
25	932643	116121	182520	15863	52	S	5,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
26	932644	116121	182520	15840	53	N	10,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
27	932645	116121	182520	15860	60	S	0,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG
28	932646	116121	182520	15861	70	S	3,00	SIGP_WEB	25/07/2025 3:38:50 p. m.	EVA_MG

- Soporte proyecto con 27 registros

	CALIFICACION_EVALUACION_ID	PROYECTO_ID	EVALUACION_PROYECTO_ID	FORMATO_EVALUACION_ID	ORDEN	ENTRABLE	CALIFICACION	USUARIO_CREADOR	FECHA_CREACION	USUARIO_N
1	942397	116129	181745	15870	10	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
2	942398	116129	181745	15881	11	S	13,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
3	942399	116129	181745	15882	12	S	15,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
4	942400	116129	181745	15865	13	N	28,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
5	942401	116129	181745	15871	20	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
6	942402	116129	181745	15883	21	S	5,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
7	942403	116129	181745	15884	22	S	5,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
8	942404	116129	181745	15866	23	N	10,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
9	942405	116129	181745	15872	30	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
10	942406	116129	181745	15885	31	S	10,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
11	942407	116129	181745	15867	33	N	10,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
12	942408	116129	181745	15873	40	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
13	942410	116129	181745	15868	43	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
14	942411	116129	181745	15874	50	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
15	942412	116129	181745	15887	51	S	20,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
16	942413	116129	181745	15869	52	N	20,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
17	942414	116129	181745	15875	60	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
18	942415	116129	181745	15888	61	S	8,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
19	942416	116129	181745	15876	62	N	8,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
20	942417	116129	181745	15877	70	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
21	942418	116129	181745	15889	71	S	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
22	942419	116129	181745	15878	72	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
23	942420	116129	181745	15879	80	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
24	942421	116129	181745	15890	81	S	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
25	942422	116129	181745	15891	82	S	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	EVA_MGNA
26	942423	116129	181745	15880	83	N	0,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	
27	942424	116129	181745	15864	90	N	76,00	SIGP_WEB	28/07/2025 1:37:58 p. m.	

Conclusiones

Durante este período se brindó soporte técnico en la plataforma SIGP, permitiendo la parametrización de convocatorias, parametrización de rubricas de evaluación, la generación de reportes, la solución de errores y la gestión de requerimientos.

Se continúa con el seguimiento de los casos que permanecen en estado **Pendiente Usuario** y **Pendiente Proveedor**, a la espera de la validación o documentación requerida.

Para garantizar una mejor eficiencia en la resolución de solicitudes, se recomienda:

- Mantener una comunicación constante con los usuarios para agilizar las aclaraciones necesarias.
- Optimizar los tiempos de respuesta en los casos que requieren intervención de terceros.
- Seguir documentando los cambios realizados para futuras referencias.

Firma:

César A. Riaño D.

Ing. César Augusto Riaño Delgado
Gerente Proyecto