

Unificación del lenguaje de programación e integración de plataformas

Informe técnico de cierre contractual

Plataforma Geográfica Institucional del Distrito de Medellín
Geografía Satelital GeoSAT S.A.S.

1. Presentación	2
2. Objetivo	2
4. Alcance	2
5. Base tecnológica común identificada	3
5.1 Relación con la arquitectura	3
6. Desarrollos que sustentan la unificación	4
6.1 Monitoreo de servicios y dashboard	4
6.2 Adecuación de entornos y scripts geográficos	4
7. Integración con los procesos de las dependencias	5
7.1 Catastro y generación de mapas SERAG	5
7.2 Movilidad y procesamiento de rutas	5
7.3 DAGRD y estructura común de consulta de riesgos	5
8. Automatización del mantenimiento y jobs	5
8.1 Lenguajes utilizados en la automatización	6
9. Resultado y alcance de la acreditación	6
Continuidad de la base tecnológica común	6
10. Conclusión	6

1. Presentación

Durante la ejecución contractual se desarrollaron y ajustaron componentes de software para conectar servicios geográficos, bases de datos, aplicaciones institucionales y procesos automáticos de administración. Este informe presenta las evidencias de ese trabajo y explica su contribución a la unificación tecnológica de los componentes intervenidos de la Plataforma Geográfica.

La mayoría de los desarrollos realizados durante el contrato se implementaron utilizando Python 3.x como lenguaje principal para aplicaciones de monitoreo, scripts de geoprocésamiento y tareas de mantenimiento. En menor medida, se utilizó SQL para programar jobs de bases de datos.

2. Objetivo

Acreditar la unificación del lenguaje principal de los desarrollos intervenidos durante el contrato en torno a Python 3.x, utilizado mayoritariamente en aplicaciones de monitoreo, scripts de geoprocésamiento y tareas de mantenimiento, y describir su integración con jobs de bases de datos programados en SQL y servicios que apoyan los procesos institucionales.

4. Alcance

Comprende el sistema de monitoreo y su dashboard, Mapas SERAG y su interacción con GICAT, la aplicación de rutas de transporte público, el diccionario del servicio de riesgos del DAGRD y las rutinas de mantenimiento y tareas programadas descritas en los soportes. La arquitectura se utiliza para ubicar sus entornos de ejecución y relaciones tecnológicas.

La unificación descrita corresponde al uso predominante de Python 3.x en los desarrollos del contrato y al empleo complementario de SQL en los jobs de bases de datos. Comprende la articulación de estos componentes con las plataformas existentes mediante servicios, conexiones y estructuras de intercambio de información.

5. Base tecnológica común identificada

La unificación se manifiesta en la reutilización de un lenguaje para distintas funciones: procesamiento geográfico, automatización, exposición de funcionalidades y seguimiento operativo. A su vez, las aplicaciones se articulan mediante servicios y estructuras de datos que permiten conservar tecnologías especializadas en cada componente.

Componente	Lenguaje o tecnología documentada	Papel en la integración
Monitoreo y dashboard	Python 3.x; Dash, Flask, Plotly, Pandas y Waitress.	Consulta de servicios y consolidación de reportes Excel, CSV y JSON en una interfaz común.
Geoprocesos y entornos	Python y ArcPy; entornos Python 3.x en la arquitectura.	Procesamiento cartográfico y ejecución de automatizaciones vinculadas a ArcGIS.
Rutas de transporte público	FastAPI, tecnología del ecosistema Python; PostGIS y procedimientos almacenados.	La aplicación permite ejecutar procesos de rutas desde una interfaz web.
Operaciones en las bases de datos	SQL, utilizado en menor medida para programar jobs de bases de datos y ajustar consultas y funciones.	Consulta, mantenimiento y corrección de datos usados por los servicios.
Servicio de riesgos DAGRD	Respuesta JSON y diccionario de campos; lenguaje interno no especificado.	Contrato de datos compartido para los equipos que consumen el servicio.
Ejecución automática	Python 3.x como lenguaje predominante de mantenimiento; Task Scheduler para su programación.	Ejecución periódica de rutinas de mantenimiento y sincronización de insumos.

5.1 Relación con la arquitectura

El diagrama ubica Python 3.x y /propy3 en Argos01, Argos03 y Argos04; muestra Task Scheduler en Argos01 y Argos03, servidores PostgreSQL, el conjunto ArcGIS Enterprise y componentes GeoServer y GeoNetwork. Apache, IIS y Tomcat apoyan la publicación y el enrutamiento.

ArcPy, Dash, Flask y FastAPI son bibliotecas o marcos de desarrollo; JSON, XML y CSV son formatos de intercambio; REST es un estilo de interfaz. Su utilización permite integrar los sistemas, pero no equivale por sí misma a usar un único lenguaje de programación.

6. Desarrollos que sustentan la unificación

6.1 Monitoreo de servicios y dashboard

Se documentó la evolución del dashboard hacia Dash y Flask, con visualización mediante Plotly y procesamiento de información con Pandas. El despliegue utiliza Waitress y configuración de rutas para operar detrás del proxy Apache. Con ello, la presentación de resultados y las funciones de seguimiento se articulan en el ecosistema Python.

La herramienta integra el reporte de auditoría en Excel, los resultados del monitoreo HTTP en CSV y los históricos de indicadores en JSON. Un proceso en segundo plano consulta las URL configuradas cada 60 segundos, registra respuesta y latencia, y permite visualizar su comportamiento. La documentación también describe actualización de la interfaz y recarga cada cinco minutos; estos intervalos corresponden a funciones distintas.

La lista de optimizaciones registra refactorización del monitoreo REST, soporte WMS, rotación de logs, reportes con fecha y hora, detección de servicios nuevos, alertas a Slack y estadísticas mensuales. También reporta la reducción de la ejecución de 40–50 minutos a 1–2 minutos; esta cifra se presenta como resultado consignado en el soporte, sin una nueva medición de desempeño.

Aporte a la integración. Los estados técnicos de distintos servicios se reúnen en una misma herramienta para el equipo administrador. Los reportes e históricos facilitan relacionar incidentes, cambios en el inventario y comportamiento de los servicios sin consultar cada aplicación por separado.

6.2 Adecuación de entornos y scripts geográficos

Para IPAT se registró la corrección de la importación de ArcPy después del cambio de ArcGIS Pro 3.3 a 3.5, mediante la actualización de referencias al entorno Python clonado. Se ajustó además la generación de nombres temporales mediante UUID para evitar colisiones.

En Argos01, después de la actualización a ArcGIS Server 11.5, se reportó la clonación del entorno Python y la incorporación de psycopg2 y geopandas requeridas por los geoprosesos. La adecuación del entorno de ejecución permite mantener las dependencias que necesitan los desarrollos sobre una base identificable para su administración.

Estos ajustes sustentan la continuidad de los desarrollos sobre Python 3.x y la administración de sus dependencias. La utilización de un lenguaje principal compartido facilita el mantenimiento de los scripts y su adaptación a los entornos de ejecución de la Plataforma Geográfica.

7. Integración con los procesos de las dependencias

7.1 Catastro y generación de mapas SERAG

El script `mapa_SERAG.py` utiliza ArcPy y se publica como servicio de geoprocetamiento consumible por GICAT. Recibe parámetros de consulta y presentación, utiliza información catastral, genera un PDF y devuelve su URL. Este mecanismo concentra la lógica cartográfica en el servicio para que la aplicación consumidora solicite el resultado mediante una interfaz definida.

7.2 Movilidad y procesamiento de rutas

El manual de la aplicación identifica FastAPI y PostGIS. Describe una interfaz para procesar todas las rutas o una selección, consultar el estado de conexión y presentar resultados con cantidad de rutas y tiempo de ejecución. El flujo invoca `sp_process_all_pending_routes()`, `sp_refinar_rutas()` y `sp_sincronizar_rutas_oddo()`; el manual asocia este último a la sincronización con Odoo.

La lista registra correcciones de la vista SQL para evitar identificadores y geometrías nulos, ajustes del tipo de dato de las paradas y operaciones `ST_Split` y `ST_Snap` para mejorar la topología. También informa la entrega de la capa resultante a Movilidad para revisión. Así, la interfaz web y las operaciones sobre la base de datos apoyan un mismo proceso de actualización de rutas; no se presume aceptación final de la dependencia.

7.3 DAGRD y estructura común de consulta de riesgos

El diccionario de `/riesgo/consultarRiesgo` define una respuesta JSON con riesgos, CBML, red hídrica, zonas de protección, barrio, comuna y amenazas. Especifica nombres, tipos de datos y tratamiento de valores null. Esta estandarización facilita que otros sistemas interpreten la respuesta de forma consistente.

8. Automatización del mantenimiento y jobs

Se tienen 66 optimizaciones, entre scripts, tareas programadas, bases de datos, servicios, aplicaciones y Enterprise. Cinco son “Tarea Programada”; uno de ellos corresponde a actualización de credenciales y otro agrupa dos tareas.

Rutina documentada	Programación reportada	Función y evidencia
Limpieza de temporales IPAT	Diaria; hora no indicada.	Elimina imágenes de más de ocho días y proyectos APRX temporales.
leer_directorio y Cobertura_banco_imagen	Cada seis horas.	Sincronizan archivos del banco de imágenes desde la NAS.
Mantenimiento de BD Catastro	Diaria a las 4 a. m.	Compress, reconstrucción de índices, análisis y desconexión de sesiones colgadas.
Copia del Banco de Imágenes	Diaria a las 8 a. m.	Copia archivos entre carpetas de la NAS.
Mantenimiento de BD Atlas	Diario según el nombre del registro; sin hora.	Rutina para nueve propietarios de datos; compress, índices y análisis.
Jobs de vistas y funciones de BD	Horario no indicado.	Correcciones de <code>sync_proceso_unico()</code> y <code>trfn_update_shape()</code> .

8.1 Lenguajes utilizados en la automatización

Las tareas de mantenimiento desarrolladas durante el contrato utilizaron mayoritariamente Python 3.x, en coherencia con los entornos que muestra la arquitectura. En menor medida se empleó SQL para programar jobs de bases de datos. Ambos lenguajes cumplen funciones complementarias: Python automatiza y coordina rutinas, mientras SQL ejecuta operaciones directamente sobre los datos. El mantenimiento de Catastro incluye SQL nativo mediante `pg_terminate_backend` y herramientas de mantenimiento de geodatabases. Contribución a la operación integrada

Estas automatizaciones mantienen disponibles los insumos usados por aplicaciones y servicios, depuran archivos temporales y ejecutan mantenimiento recurrente. La actualización de credenciales de las tareas de Argos03, registrada en el soporte, permitió corregir fallos asociados al cambio de contraseña. El beneficio documentado está en la continuidad de procesos compartidos y en la reducción de intervenciones manuales.

9. Resultado y alcance de la acreditación

La ejecución contractual consolidó Python 3.x como lenguaje principal de la mayoría de los desarrollos de monitoreo, geoprocetamiento y mantenimiento. La unificación se refleja en el dashboard, la adecuación de entornos y scripts y el uso de FastAPI en la aplicación de rutas. SQL se utilizó en menor medida para programar jobs de bases de datos, complementando las automatizaciones y los mecanismos de integración de la Plataforma Geográfica.

Para Catastro, el servicio SERAG enlaza la generación de mapas y su consumo desde GICAT; para Movilidad, la aplicación articula el procesamiento de rutas con procedimientos de base de datos; para DAGRD, el diccionario establece la estructura de respuesta que deben interpretar los consumidores. El monitoreo y las rutinas programadas apoyan transversalmente la operación de estas soluciones.

Continuidad de la base tecnológica común

La base tecnológica resultante permite mantener los desarrollos principales dentro del ecosistema Python 3.x y conservar en SQL las operaciones propias de los jobs de bases de datos. Para facilitar su continuidad, la documentación técnica debe mantener identificados los scripts, las dependencias, los entornos y los horarios de ejecución. En el caso de SERAG, corresponde armonizar el manual heredado con la migración reportada en la lista de optimizaciones.

10. Conclusión

Durante el contrato se utilizó Python 3.x como lenguaje principal para la mayoría de las aplicaciones de monitoreo, los scripts de geoprocetamiento y las tareas de mantenimiento desarrolladas. SQL tuvo un uso complementario y de menor proporción en la programación de jobs de bases de datos. Esta base común, articulada mediante servicios e interfaces de intercambio, sustenta la unificación de los desarrollos intervenidos y su integración con las plataformas que apoyan las actividades y proyectos de la Alcaldía de Medellín.