Uso de Mapbox Vector Tiles en la gestión de datos geoespaciales: Servidor MVT-Rust-Agil-Simple

Autores: Fernando Jose Jachuf [1*] y Carlos Alberto Salinas[1]

[1] IDECOR - Infraestructura de Datos Espaciales de Córdoba

Contacto: jjachuf@gmail.com

Resumen: El presente documento explora el uso de las Mapbox Vector Tiles (MVT) como una herramienta eficiente para la gestión y análisis de datos geoespaciales. Se analiza su formato binario, ampliamente reconocido por su compatibilidad e interoperabilidad con diversas aplicaciones SIG y web. Además, se presentan estándares relacionados con el manejo de teselas vectoriales, como WMTS, TMS y OGC API Tiles, destacando sus beneficios frente a los formatos ráster tradicionales. Finalmente, se introduce MVT Server como una solución práctica para la implementación y administración de datos geoespaciales en este formato, ofreciendo una perspectiva integral de las metodologías, casos de uso y desafíos asociados.

Palabras clave: Mapbox Vector Tiles, Teselas Vectoriales, Gestión de datos geoespaciales, Interoperabilidad, Calidad de datos, PostGIS.

Introducción

La gestión eficiente de la información geoespacial (IG) constituye un componente esencial en diversos ámbitos, tales como la planificación, la toma de decisiones estratégicas, la optimización de recursos y la administración territorial. En este marco, la adopción del formato de archivo Mapbox Vector Tiles (MVT) emerge como un estándard clave para el manejo y análisis de dichos datos, facilitando su transferencia desde múltiples fuentes hacia sistemas de destino donde pueden ser almacenados y utilizados de manera efectiva.

El presente documento tiene como objetivo analizar el papel fundamental del formato Mapbox Vector Tiles (MVT) en la gestión de datos geoespaciales, centrándose en su aplicación práctica, las mejores prácticas asociadas y los desafíos específicos que emergen en este ámbito. Además, se presenta MVT Server como una alternativa para la implementación y manejo de datos geoespaciales utilizando este formato. A través de un análisis detallado de herramientas, metodologías y casos de uso pertinentes, se pretende ofrecer una perspectiva integral y actualizada sobre la implementación de MVT en el campo de la información geoespacial.

Mapbox Vector Tiles

El estándar Mapbox Vector Tiles (MVT) es un formato de archivo binario eficiente para almacenar y codificar datos geoespaciales en teselas vectoriales. Este formato es ampliamente adoptado y soportado por diversas aplicaciones cliente, lo que facilita la interoperabilidad y la visualización dinámica de datos geoespaciales en aplicaciones SIG y web.

Formato y Estándares

El identificador “application/x-protobuf;type=mapbox-vector-tile” es del tipo Multipurpose Internet Mail Extensions (MIME) que se utiliza para especificar el formato de los archivos de teselas vectoriales Mapbox Vector Tiles (MVT) cuando se transfieren a través de la web.

El segmento application/x-protobuf indica que el contenido está codificado usando Protocol Buffers (Protobuf), un formato binario desarrollado por Google para la serialización eficiente de datos. La extensión “type=mapbox-vector-tile” aclara que esos datos serializados corresponden específicamente al estándar de teselas vectoriales definido por Mapbox, diferenciándolos de otros posibles usos de Protobuf.

Este tipo de identificación surge de la necesidad de los navegadores, servidores y aplicaciones de GIS de reconocer rápidamente el formato y la finalidad de los archivos intercambiados, facilitando su correcto procesamiento y visualización. Así, cuando un servidor envía teselas MVT a un cliente web o una aplicación, declara este tipo MIME para garantizar la interoperabilidad y el manejo adecuado de los recursos geoespaciales.

Este formato es ampliamente recomendado debido a su eficiencia y su compatibilidad con una variedad de herramientas y aplicaciones geoespaciales. En los servicios de mapas tradicionales, los estándares utilizados incluyen Web Map Service (WMS) para la provisión de imágenes y Web Feature Service (WFS) para el acceso a datos geográficos primarios. En lo que respecta a las teselas, se emplean estándares como Web Map Tile Service (WMTS) y Tile Map Service (TMS). Además, la nueva generación de estándares OGC API, con énfasis en OGC API Tiles, ofrece capacidades avanzadas para proporcionar datos en formato ráster y vectorial como teselas.

Evolución de las Teselas Vectoriales

La evolución de las tecnologías de teselas vectoriales ha marcado un hito en el mundo de la información geoespacial, permitiendo la gestión y visualización dinámica de grandes volúmenes de datos con una eficiencia sin precedentes. A continuación, se presenta una cronología de los avances más relevantes en la adopción y el desarrollo de este formato, que han transformado la manera en que se almacenan, procesan y distribuyen los datos geográficos

- **Antes de 2014:** Los servicios de mapas web se basaban principalmente en formatos ráster (imágenes), utilizando estándares como WMS (Web Map Service). Estos servicios entregan representaciones visuales de datos geográficos sin permitir el acceso directo a los atributos o geometrías subyacentes.
- **2014:** PostGIS 2.4 introduce nuevas funciones espaciales, incluyendo la función ST_AsMVT, que permite convertir directamente registros almacenados en una base de datos PostGIS al formato Mapbox Vector Tile (MVT), incluyendo tanto la geometría como los atributos alfanuméricos. Asimismo, se incorporaron otras funciones como ST_TileEnvelope para especificar la porción de datos a convertir de una tesela y ST_AsMVTGeom para transformar geometrías individuales al espacio de coordenadas de la tesela.
- **2015:** Se revisa y confirma el estándar ISO 19128:2005, que define la interfaz de los servidores de mapas web (WMS).
- **2018:** Se revisa y confirma el estándar ISO 19142:2010, que define la interfaz de los servicios de entidades web (WFS), los cuales permiten el acceso a los datos geográficos originales.
- **A finales de 2020**, Mapbox anunció un cambio en la licencia de Mapbox GL JS, pasando de un modelo open source a una licencia propietaria, lo que marcó un giro significativo en el acceso y uso de esta popular biblioteca para visualización de mapas. Esto conduce a que la comunidad cree un fork (copia modificada y desarrollada de un proyecto de software original) de código abierto llamado MapLibre GL JS en diciembre de 2020, manteniendo la funcionalidad y permitiendo la continuación del desarrollo bajo una licencia más colaborativa.
- **2020:** en GIS StackExchange se plantearon discusiones sobre las mejores prácticas para servir (proveer o entregar) teselas vectoriales, destacando opciones como la generación, el uso de cachés y servidores especializados como Koop de Esri o pg_tileserv de Crunchy Data.

- **2023:** se continuaron explorando soluciones relacionadas, subrayando la eficiencia de herramientas como pg_tileserv y la extensión mvt_postgis para servir teselas directamente desde bases de datos PostGIS, junto con la implementación de configuraciones de caché.
- **Mayo de 2024:** Fabian Rechsteiner publica su tesis de maestría y el repositorio de GitHub "vector-tiles-benchmark". La tesis compara el rendimiento de seis servidores de teselas vectoriales de código abierto (BBOX, Ldproxy, Martin, pg_tileserv, Tegola y TiPg) al servir datos desde una base de datos PostGIS utilizando Docker[1] y pruebas de rendimiento con Apache JMeter[2]. También se crea un sitio web con Maplibre GL JS para visualizar los resultados.
- **2025:** MapLibre publica la versión 0.16.0 de Martin, un servidor de teselas vectoriales y herramientas escrito en lenguaje de programación Rust, capaz de servir teselas desde PostGIS, MBTiles y PMTiles, con énfasis en la velocidad y el alto tráfico. En este mismo año mvt-proj publica la versión 0.11.2 de MVT Server, un servidor de teselas vectoriales sencillo y de alta velocidad desarrollado en Rust, utilizando el framework[3] web Salvo y requiriendo PostgreSQL con PostGIS 3.0+. Otro hito de este año es go-spatial publica la versión v0.21.2 de Tegola, un servidor de teselas vectoriales escrito en Go, con soporte para PostGIS, GeoPackage y SAP HANA Spatial, además de funcionalidades de caching y procesamiento geométrico.

De manera continua e desarrollan y utilizan diversas herramientas y librerías (como mvt de DougLau, librerías para OpenLayers, Leaflet y Deck.gl) para la codificación, decodificación y visualización de teselas vectoriales. Se exploran diferentes métodos de despliegue para servidores en Rust, incluidos contenedores Docker y VPS, como se menciona en Shuttle.dev. Organizaciones como la Infraestructura de Datos Espaciales de España (IDEE) promueven y explican el uso de servicios de mapas de teselas vectoriales, incluyendo los formatos y métodos de generación (como con PostGIS, GeoServer/GeoWebCache y Tippecanoe), así como los clientes que los consumen (OpenLayers, Leaflet, Deck.gl, MapLibre). MapTiler Server implementa la funcionalidad de conectarse con bases de datos PostgreSQL para servir teselas vectoriales y ráster en tiempo real a medida que los datos son editados.

Generación y Servicio de Teselas Vectoriales

La generación de teselas vectoriales puede realizarse de varias maneras. PostgreSQL/PostGIS incluye la función ST_AsMVT que permite convertir registros almacenados en la base de datos al formato MVT, incluyendo geometría y atributos. También existen herramientas como GeoServer/GeoWebCache y Tippecanoe para la generación de teselas vectoriales.

Servidores de Teselas Vectoriales

Un servidor de teselas vectoriales es una aplicación que sirve teselas bajo demanda a los clientes. Diversas soluciones open-source han sido desarrolladas para este fin.

---

[1] Docker: Es una plataforma de código abierto que permite desarrollar, empaquetar y ejecutar aplicaciones dentro de contenedores, que es una unidad ligera y portátil que incluye todo lo necesario para ejecutar una aplicación.

[2] Apache JMeter: Es una herramienta de código abierto diseñada para realizar pruebas de carga y rendimiento sobre aplicaciones y servicios.

[3] Framework: conjunto estructurado de herramientas, librerías y convenciones que facilita el desarrollo y mantenimiento de aplicaciones o sistemas de software.

Martin

Martin es un servidor de teselas rápido y ligero escrito en Rust, capaz de servir teselas directamente desde bases de datos PostGIS, archivos MBTiles y PMTiles. Optimiza para velocidad y tráfico pesado, y permite combinar múltiples fuentes de teselas en una sola.

pg_tileserv

Desarrollado por CrunchyData, el servidor pg_tileserv está diseñado para crear OGC API Tiles desde una base de datos PostGIS "on the fly". Es eficiente en el manejo de datos de puntos y ofrece rendimiento optimizado para datos dinámicos.

Tegola

Tegola es un servidor de teselas vectoriales escrito en Go, con soporte para PostGIS, GeoPackage y SAP HANA Spatial. Ofrece procesamiento nativo de geometrías, soporte para múltiples proyecciones y capacidades de caché.

MVT Server

MVT Server es un servidor de teselas vectoriales desarrollado en Rust, que permite publicar cualquier tabla o vista con un campo de geometría como teselas vectoriales. Incluye administración web, almacenamiento en caché, y soporte para diferentes variantes de fuente.

Consumo y Estilización de Teselas Vectoriales

Las teselas vectoriales no contienen información de estilo, por lo que los clientes deben aplicar una simbología para visualizar los datos. En QGIS, la estilización de teselas vectoriales se basa en reglas aplicadas a las características, definidas en archivos QML o MapBox GL Json.

MapLibre GL JS

MapLibre GL JS es una biblioteca JavaScript de código abierto para mostrar mapas en sitios web, capaz de consumir teselas MVT generadas por servidores como Martin y aplicarles estilos utilizando WebGL.

Deck.gl

Deck.gl es otra biblioteca JavaScript para la visualización geoespacial, compatible con teselas MVT.

QGIS

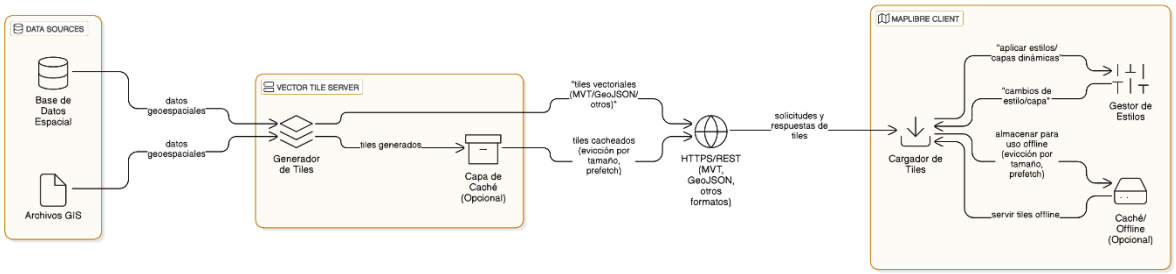
QGIS soporta la carga y estilización de teselas vectoriales, permitiendo la aplicación de simbología y etiquetas basadas en reglas.

Mapea

Mapea es una API de desarrollo de visualizadores de mapas de la Junta de Andalucía, con soporte para capas de tipo MVT.

Figura 1.

Esquema de funcionamiento Servidor de Vector Tiles



MVT Server

Desarrollado principalmente por Jose Jachuf, integrante del equipo de Infraestructura de Datos Espaciales de la Provincia de Córdoba (IDECOR), el servidor MVT Server es una alternativa robusta, ágil y sencilla de servir datos geográficos. Implementado en RUST un lenguaje de programación moderno y seguro, sus principales características son:

- Seguridad de memoria
- Rendimiento
- Concurrencia
- Expresividad
- Interoperabilidad

Gracias a estos puntos podemos resolver las siguientes ventajas de desarrollar el producto en este lenguaje

1. Alta Velocidad y Rendimiento: Varios servidores de teselas vectoriales de código abierto están desarrollados en Rust, incluyendo Martin, BBOX y MVT Server. El servidor Martin está explícitamente optimizado para la velocidad y el tráfico intenso, y MVT Server también se describe como un servidor de "alta velocidad".
2. Resultados favorables en Benchmarking: En una comparación de rendimiento de seis soluciones de servidores de teselas vectoriales de código abierto para datos de PostGIS, el servidor Martin (desarrollado en Rust) demostró el rendimiento más rápido en todos los escenarios de prueba; entregó las teselas "dos o tres veces más rápido que el segundo servidor más rápido", el segundo servidor en rendimiento fue Tegola escrito en GO. El servidor BBOX, también escrito en Rust, fue el tercer servidor más rápido.
3. Eficiencia en el uso Recursos: Las aplicaciones web escritas en Rust suelen tener una huella de memoria baja, típicamente alrededor de 50-150 MB, dependiendo de la funcionalidad de la aplicación. Esta eficiencia es una característica general de Rust que beneficia a las aplicaciones de servidor, ya que pueden operar de manera eficiente con menos recursos, lo cual es relevante para entornos como VPS. Los servidores Rust como Martin son reconocidos como "ligeros".
4. Naturaleza Compilada y Despliegue: Cuando se compila Rust, se genera un ejecutable binario. Esto hace que los programas Rust sean adecuados para ejecutarse en contenedores o, alternativamente, en un VPS.

Requisitos y base de datos

El servidor MVT Server requiere para funcionar un servidor PostgreSQL con la extensión PostGIS versión 3.0.0 o superior instalada, ya sea local o remota. Para su funcionamiento, también necesita tener disponible el puerto 5800 por defecto, aunque es configurable.

Instalación y ejecución

El código debe descargarse y compilarse utilizando el entorno de programación Rust. Para ello, es necesario ejecutar el comando `cargo build --release`, el cual permite generar una versión optimizada y apta para entornos de producción.

Figura 2.
Instalación de MVT Server

```
# Clone the repository
git clone https://github.com/mvt-proj/mvt-rs.git
# Navigate to the project directory
cd mvt-rs

# Compile for production
cargo build --release
```

El servidor requiere la configuración de variables de entorno clave para garantizar su correcto funcionamiento. Entre estas, se destaca la necesidad de establecer un archivo .env en la raíz del proyecto, donde se definan las variables esenciales para su operación.

- conexión a la base de datos (DBCONN)
- tamaño del pool de conexiones (POOLSIZEMIN, POOLSIZEMAX)
- dirección y puerto del servidor (IPHOST, PORT)
- conexión a Redis (REDISCONN)
- secretos para JWT y sesiones (JWTSECRET, SESSIONSECRET)

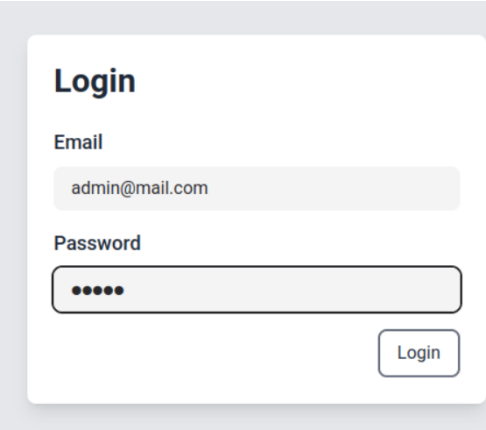
Figura 3.
Configuración de Variables de Entorno

```
# Database connection URL (PostgreSQL)
DBCONN=postgres://user:pass@host:port/db

# Connection pool size
POOLSIZEMIN=3   # Minimum size of the connection pool
POOLSIZEMAX=5   # Maximum size of the connection pool

# Server settings
IPHOST=0.0.0.0  # The IP address where the server will listen
PORT=5800       # The port on which the server will run

# Redis connection (optional, overrides disk cache if provided)
REDISCONN=redis://127.0.0.1:6379

# Security settings
JWTSECRET=supersecretjwt # Used to create and validate JWT tokens
SESSIONSECRET=supersecretsession # Secret key for session management

# Directories
CONFIG=/path_to/config_dir             # Directory path for configuration files
CACHE=/path_to/cache_dir               # Directory path for cache storage
MAPASSETS=/path_to/map_assets_dir      # Directory path for map_assets storage
```

Panel de administración

En el contexto de la configuración inicial del servidor, se establece un usuario por defecto identificado como ‘admin’, cuya dirección de correo electrónico es admin@mail.com y cuya contraseña predeterminada es ‘admin’. Es fundamental modificar dicha contraseña tras el primer acceso al sistema con el fin de garantizar la seguridad de las credenciales. El acceso al panel de administración se realiza a través de http://localhost:5800 o mediante la dirección previamente configurada para este propósito.

Figura 4.
Acceso al panel de administración



1. El panel de administración constituye una herramienta integral para la gestión de múltiples aspectos del servidor, tales como usuarios, grupos, categorías, catálogo y estilos. Dentro de este entorno, es posible llevar a cabo la publicación de capas accediendo al apartado denominado "Catalog" (Catálogo).
2. Nueva capa: Dentro del Catálogo, se seleccionará la opción "Add Layer" (Añadir Capa).
3. Formulario de configuración de la capa: En el formulario (Figura 5) es posible completar la información referente a las principales propiedades de la capa a publicar.

Nombre: Representa el identificador único de la capa dentro del sistema. Es recomendable usar una palabra en minúsculas, sin espacios, para facilitar la gestión y referencia.

Alias: Es una etiqueta más descriptiva que el nombre, pensada para proporcionar un contexto más claro sobre la capa. Puede incluir espacios y caracteres que hagan más comprensible su propósito.

Fuente de datos: Corresponde al esquema y la tabla seleccionados de la base de datos PostgreSQL. Estos determinan el contenido geográfico y los atributos de la capa que se publicará.

Fields: Incluyen las propiedades que se desean exponer de la tabla seleccionada. La recomendación es limitarse solo a los campos necesarios, para optimizar el rendimiento y evitar incluir información innecesaria.

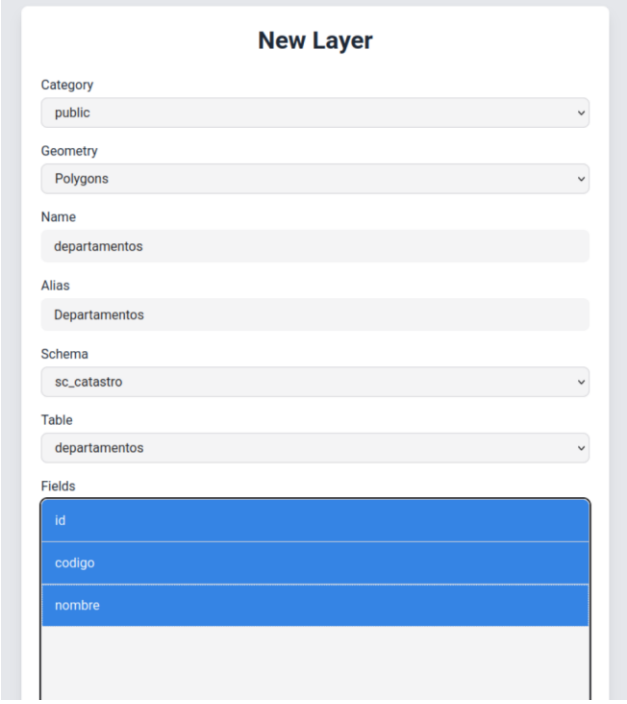
ZMin y ZMax: Definen los niveles de zoom mínimo y máximo en los que la capa será visible. Esto ayuda a optimizar la representación de datos y a mejorar la experiencia del usuario. Los

valores deben ser configurados con precisión, utilizando herramientas como el mapa integrado para visualizar los resultados.

Caché: Permite definir la duración de los datos almacenados en la memoria caché. Para capas que no cambian con frecuencia, se recomienda establecer una duración infinita (valor de caché igual a 0), mientras que para datos dinámicos podría ajustarse a una duración más corta. La caché se puede purgar en cualquier momento. Cada capa gestiona la expiración de su caché de forma independiente. La caché se guarda localmente en el directorio "cache" por defecto, pero se puede configurar otra ubicación.

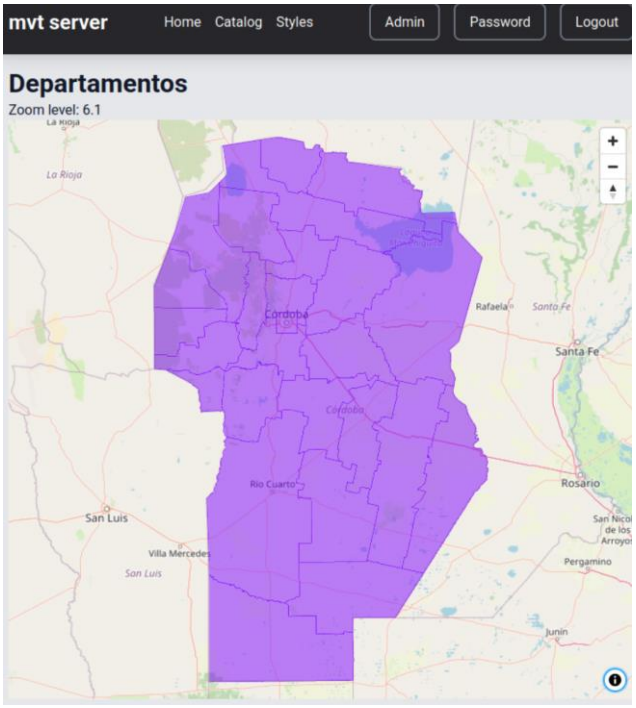
El resto de los campos (variables) se pueden dejar con sus valores por defecto.

Figura 5.
Formulario de alta de una capa



4. Al finalizar la carga de datos en el formulario y configuración de la capa se pueden guardar los cambios.
5. Luego, se puede verificar o validar la información almacenada para cada capa publicada (Figura 6).

Figura 6.
Previsualización de la capa



Después de seguir estos pasos, la capa de datos estará configurada en el servidor MVT Server y disponible para ser solicitada por clientes (como QGIS, MapLibre GL JS, OpenLayers, Leaflet, etc.) que consumirán las teselas vectoriales generadas dinámicamente por el servidor a partir de la base de datos PostgreSQL

Configuración de Estilos

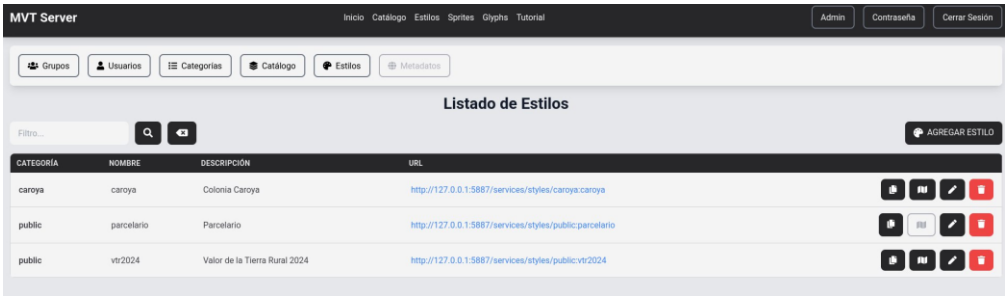
Además de servir teselas, MVT Server puede servir estilos, que pueden ser consumidos en QGIS o en MapLibre. El servidor también permite servir sprites[4] y glifos[5] personalizados.

1. ***Gestión de Estilos:*** Dentro de la consola de administración, en la sección dedicada a los estilos. Se pueden definir y gestionar los estilos de renderizado para cada una de las capas. Estos estilos determinan cómo aparecerán las capas en el mapa, incluyendo aspectos como colores, símbolos y etiquetas (Figura 7).

Figura 7.
Pantalla de Gestión de estilos

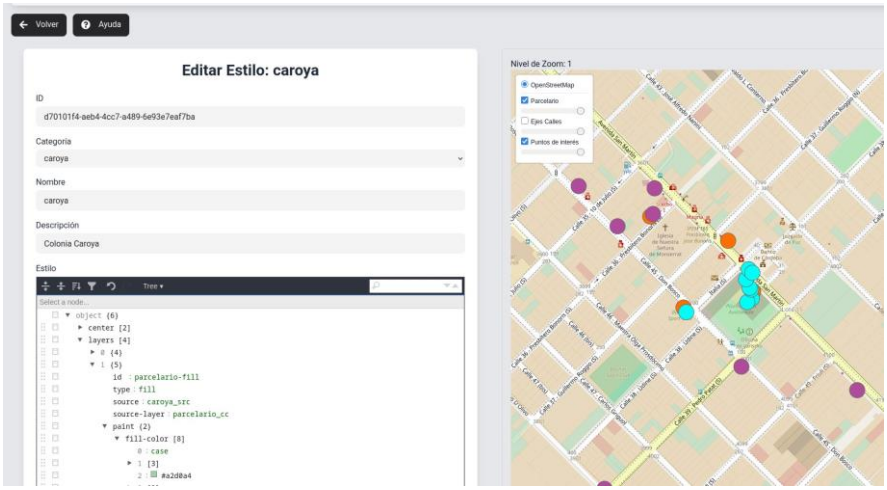
[4] Sprite es una imagen que contiene varios íconos o gráficos individuales en una sola hoja. En un pie de página, los sprites se usan para representar visualmente acciones o accesos, como íconos de redes sociales, botones de contacto, o logos.

[5] Glifo es una representación gráfica de un carácter tipográfico. Cuando se usan fuentes de íconos (como Font Awesome, Material Icons, etc.), los glifos permiten mostrar íconos como si fueran texto.



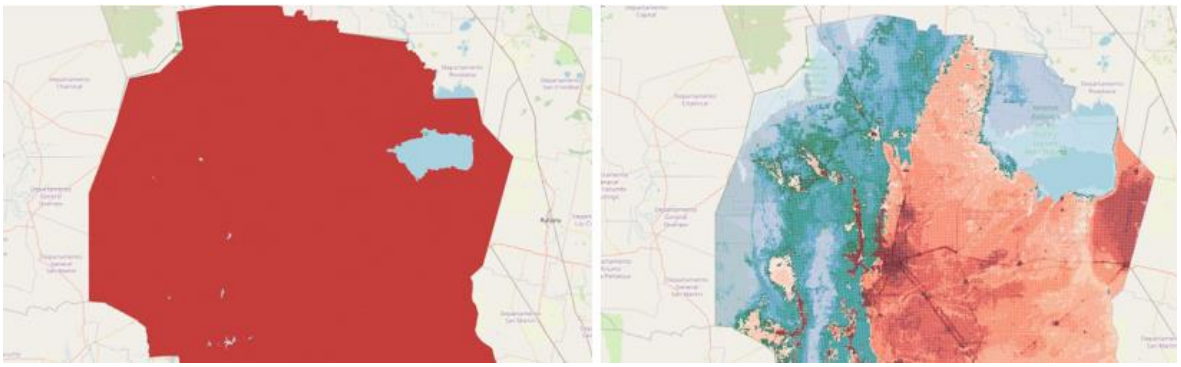
2. ***Publicación de Estilos:*** El servidor permite publicar los estilos creados. Esta funcionalidad es importante porque asegura que las teselas vectoriales se rendericen de manera consistente y como se pretende visualizar en diferentes plataformas o herramientas. Los estilos publicados pueden ser utilizados por clientes GIS como QGIS (Figura 8).

Figura 8.
Publicación de estilos



3. ***Soporte para MapLibre Style Specification:*** El servidor MVT es compatible con la especificación de estilos de MapLibre. Esto permite que clientes como MapLibre carguen un *proyecto de estilo completo*, que incluye no solo la apariencia de las capas, sino también las fuentes de datos, metadatos, sprites, glifos, niveles de zoom y la vista inicial del mapa (centro y zoom predeterminado). (Figura 9).
4. ***Servidor Integrado de Glifos y Sprites:*** MVT Server cuenta con un servidor integrado de glifos y sprites. Estos son componentes esenciales para aplicar estilos personalizados, permitiendo el uso de iconos personalizados y fuentes de texto. Debes organizar tus archivos de sprites y glifos en una estructura de directorios específica y configurar tu archivo JSON de estilo de MapLibre para que apunte a las URL de estos servicios proporcionados por el servidor MVT Server. El servidor puede servir glifos generados a partir de fuentes utilizando herramientas externas como fontnik, y aunque la versión actual solo soporta una fuente en el array text-font de un estilo, el servidor verifica la existencia de la fuente a través del panel de administración.

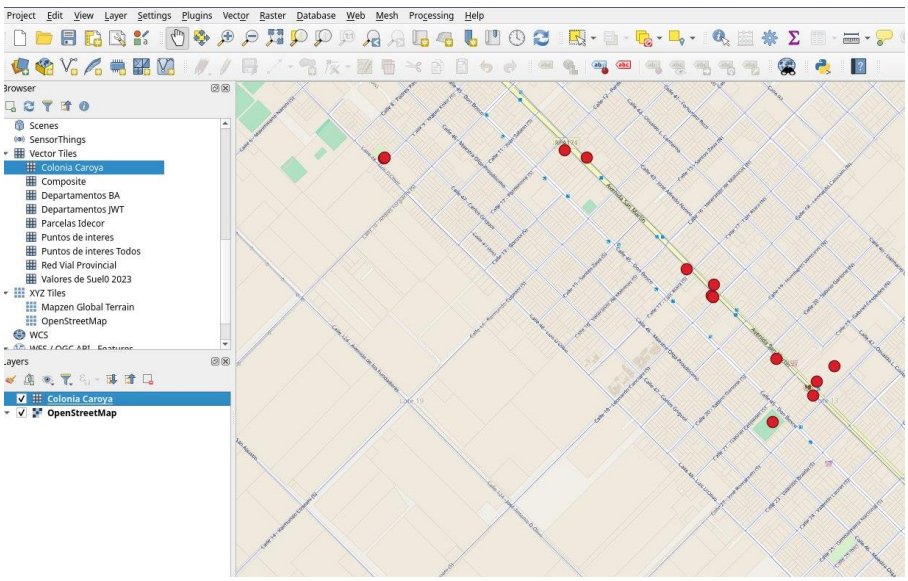
Figura 9.
Layer (capa) publicada Sin Estilo y Con Estilo asignado



Ventajas del Uso de MVT Server

- Generación de teselas vectoriales al vuelo: Utilizando PostgreSQL/PostGIS, permite la publicación de cualquier tabla o vista con un campo de geometría como teselas vectoriales (Figura 10).
- Administración web: MVT Server incluye una interfaz de administración para gestionar usuarios, grupos, capas, estilos, categorías y más.
- Almacenamiento en caché: Integrado con soporte para disco o servidor Redis, lo que mejora la eficiencia en la entrega de teselas.
- Múltiples variantes de fuente: Permite recuperar teselas a través de diferentes configuraciones como capa única, multicapa por composición y multicapa por categoría.
- Servidor de glifos y sprites: Integrado para estilos personalizados, lo que facilita la visualización coherente de los datos.
- Control de acceso: Por capa a través de Autenticación Básica o JWT, asegurando la seguridad en el acceso a los datos.
- Compatibilidad: Los estilos pueden ser consumidos en herramientas como QGIS o MapLibre.

Figura 10.
Uso de Vector Tiles en QGIS



Ejemplos Prácticos

Acceso a teselas vectoriales

Los clientes pueden recuperar teselas vectoriales usando diferentes rutas base dependiendo del tipo de fuente deseada:

- Capa única: /services/tiles/{layer}/{z}/{x}/{y}.pbf
- Múltiples capas: /services/tiles/multi/{layers}/{z}/{x}/{y}.pbf
- Por categoría: /services/tiles/category/{category}/{z}/{x}/{y}.pbf

MapLibre, OpenLayers y Leaflet son herramientas de código abierto compatibles con MVT Server para la gestión y visualización de datos geoespaciales. Es importante señalar que Leaflet necesita complementos adicionales para admitir de manera nativa las teselas vectoriales.

Comparativa Con Otros Servidores

Aunque MVT Server es un servidor Rust eficiente y con muchas funcionalidades, no aparece en el benchmark comparativo de otros servidores como Martin o BBOX. Sin embargo, su capacidad de generación de teselas al vuelo, administración web integrada, soporte para caché y la posibilidad de configurar estilos y almacenarlos internamente para ser utilizados en distintas capas lo hacen una opción robusta para aplicaciones web y de escritorio que consumen datos geoespaciales dinámicos.

Casos de Uso en IDECOR

En IDECOR se están creando varias implementaciones que incluyen el uso de MVT Server. Actualmente, se están llevando a cabo desarrollos informáticos que incluyen colectores de en colectores de datos y paneles de control (dashboards). Se busca integrar estos servicios con otros ya publicados, trabajando para incluir información de MVT Server en un módulo accesible al público.

Conclusión

MVT Server se posiciona como una herramienta robusta y versátil para el manejo de teselas vectoriales, ofreciendo capacidades avanzadas como la generación dinámica de datos, soporte para múltiples formatos y la integración con diversos clientes y servicios. Su implementación en proyectos como los de IDECOR destaca su potencial para satisfacer las necesidades de aplicaciones que requieren datos geoespaciales dinámicos, eficientes y accesibles. Además, el alineamiento con estándares como OGC API Tiles refuerza la interoperabilidad y el uso eficiente de esta tecnología en la distribución y consulta de información geoespacial. A medida que la tecnología de mapas vectoriales sigue evolucionando, herramientas como MVT Server continúan marcando el camino hacia soluciones más innovadoras y eficaces en la distribución de información geográfica.

Agradecimientos

Agradecemos a Fadel Jachuf, desarrollador freelance, por sus colaboraciones en el desarrollo de “MVT Server”; a Nicolas Oller, administrador de bases de datos geoespaciales del equipo de IDECOR; y al equipo de IDECOR dirigido por el Ing. Hernán Morales por su apoyo constante a los proyectos de desarrollo.

Referencias

Doug Lau. (n.d.). **mvt: Rust library for encoding mapbox vector tiles**. https://github.com/DougLau/mvt

Geographic Information Systems Stack Exchange. (2025, 18 de febrero). **Best way to serve vector tiles from data that changes frequently?** https://gis.stackexchange.com/questions/361741/best-way-to-serve-vector-tiles-from-data-that-changes-frequently

Mo, J. (2024, 7 de febrero**). Deploying Rust web applications. Blog de Shuttle.dev**. https://www.shuttle.dev/blog/2024/02/07/deploy-rust-web

mvt-proj. (2025, 19 de abril). **mvt-rs: Simple and high-speed vector tile server developed in Rust**. https://github.com/mvt-proj/mvt-rs

go-spatial. (2025, 8 de enero). **tegola: Tegola is a Mapbox Vector Tile server written in Go**. https://github.com/go-spatial/tegola

(n.d.). **MapLibre GL JS vs. Leaflet: Choosing the right tool for your interactive map**. https://blog.jawg.io/maplibre-gl-vs-leaflet-choosing-the-right-tool-for-your-interactive-map/

Mapbox. (n.d.). **Vector tiles standards | Tilesets | Mapbox Docs**. Mapbox Docs. https://docs.mapbox.com/data/tilesets/guides/vector-tiles-standards/

maplibre. (n.d.). **Using with MapLibre - Martin Tile Server Documentation**. Martin Tile Server Documentation. https://maplibre.org/martin/using-with-maplibre.html

maplibre. (2025, 7 de abril). **maplibre/martin: Blazing fast and lightweight PostGIS, MBtiles and PMtiles tile server, tile generation, and mbtiles tooling.** https://github.com/maplibre/martin

MapTiler. (n.d.). **Vector Tiles from PostgreSQL | MapTiler Server**. https://www.youtube.com/watch?v=iWGbAfK2wzQ

PostGIS Project. (n.d.). **ST_AsMVT**. PostGIS Documentation. https://postgis.net/docs/en/ST_AsMVT.html

QGIS Project. (n.d.). **Working with Vector Tiles**. En *QGIS Documentation (QGIS 3.40)*. https://docs.qgis.org/3.40/en/docs/user_manual/working_with_vector_tiles/vector_tiles.html

Rechsteiner, F. (2024, mayo). **FabianRechsteiner/vector-tiles-benchmark: Performance comparison of open source vector tiles server solutions for providing geodata from PostGIS databases**. https://github.com/FabianRechsteiner/vector-tiles-benchmark

Villar Iglesias, A. (2018, 17-19 de octubre). **Servicio de Mapas de Teselas Vectoriales** [Presentación]. IDEE.

https://www.idee.es/resources/presentaciones/JIIDE18/JIIDE2018_Presentacion_MULTIESCALA.pdf