BondisUY

Marcos Pulido mpulidouy@gmail.com
Andrés López andorkan@gmail.com
Agustina Corvo aguscorvo@gmail.com
Bruno Fernández bruferper@gmail.com
Ricardo Machado rickymach1012@gmail.com

Abstract

With the steady increase of websites, software applications and diverse systems that use geographic data to communicate their business operations as well as public information more effectively, new standards and technologies are emerging to face the challenges aroused by these systems.

This work focuses on studying OGC's WFS and WMS standards, as well as researching and putting into practice operations with the PostGIS extension and a Geographic Information Server, such as Geoserver, together with Wildfly, an Application Server.

Simultaneously, it focuses on implementing systems integration, through a layered architecture and Web Services, with the standards already mentioned. It's based on Java EE for the development of the Business and Data layers, while using Javascript, JSP, OpenLayers, JQuery and AJAX for the Presentation layer.

The main purpose of this work is to develop an application to query and modify

geographic information about Montevideo's public transport system. The application's name is "BondisUY".

Con el constante crecimiento de sitios, aplicaciones y diversos sistemas informáticos que utilizan datos geográficos para comunicar de forma más eficaz su operativa de negocio y/o información a la ciudadanía, surgen nuevos estándares y tecnologías para resolver los desafíos que presentan estos sistemas.

El presente trabajo se centra en estudiar los estándares WFS y WMS de OGC, como también investigar e implementar operaciones con la extensión PostGIS y un servidor de información geográfica, como ser Geoserver, junto a Wildfly, como servidor de aplicación.

A su vez, se centra también en implementar la integración de sistemas y su comunicación, a través del trabajo en una arquitectura en capas y la utilización de servicios web, con los estándares ya mencionados. Se trabaja con Java EE para el desarrollo de la capas lógica y de datos del sistema, mientras que se implementa Javascript, JSP, OpenLayers, JQuery y AJAX para el desarrollo de la capa de presentación.

Esto se logra a través del desarrollo de una aplicación para consultar y modificar información geográfica vinculada al sistema de transporte público de Montevideo. Dicha aplicación se llama "BondisUY".

Palabras clave

GIS, Servicios Web GIS, Servicios Web, WMS, WFS, OpenLayers, JSP, PostGIS, Geoserver, Wildfly, Java EE.

1. Introducción

El presente trabajo se enmarca como proyecto

final de la materia “Taller de Sistemas de Información Geográficos” de la Facultad de Ingeniería de la Universidad de la República.

El desafío planteado fue el de desarrollar una aplicación que brindara información actualizada a la población sobre el sistema de transporte público en Montevideo.

Se investigó las fuentes de información geográfica a utilizar, evaluando si incluir recursos ya disponibles de forma abierta, o cargar los propios. Se definió utilizar las capas disponibles del sistema de información geográfica de la Intendencia de Montevideo[1].

Además del desafío de obtener toda esta información del sistema de transporte capitalino, se presentaba también el desafío de modificar y persistir dicha información en la base de datos. Este desafío se solucionó con la implementación de OpenLayers, en comunicación con Geoserver.

La elección de herramientas a utilizar se definió de forma rápida, debido a buenas experiencias pasadas con algunas de ellas, como también el hecho de haber trabajado durante el curso con las que aún no se conocían, y por último, la necesidad de aprovechar el tiempo para comenzar cuanto antes con la implementación de la solución final.

El resto del documento se organiza de la siguiente manera. La sección 2 presenta el marco conceptual del presente proyecto. En la sección 3, se presenta la descripción del problema planteado, mientras que en la sección 4 se presenta su solución. La sección 5 presenta un esquema de la arquitectura del sistema y en la sección 6 su implementación. La evaluación de la solución se presenta en la sección 7. Por último, en la sección 8 se presenta el desarrollo del proyecto y en la sección 9 las conclusiones del trabajo.

2. Marco conceptual

Mapa
Es una representación a escala de una selección de entidades sobre la superficie de la Tierra. A su vez, en Matemática y Computación, se utiliza como una transferencia de información de una forma en otra.

Entidad
Conjunto o subconjunto de datos en el mapa, generalmente ubicados dentro de cierta categoría (ej: calles, vías de tren, manzanas, parcelas, entre otras).

SIG
Sistema empleado para describir y categorizar la Tierra y otras geografías con el objetivo de mostrar y analizar la información a la que se hace referencia espacialmente. Este trabajo se realiza fundamentalmente con los mapas[2].

Geometría
Atributo que representa la información espacial de una entidad. Las geometrías básicas existentes son Point, LineString y Polygon.

Capa
Conjunto de entidades agrupadas según una temática definida.

Vector
Representación geométrica de características geográficas en forma precisa y compacta, utilizando puntos, líneas y polígonos.

Utilizar vectores en este trabajo permitió realizar distintas operaciones geográficas con las capas y cálculos (ej: calcular los recorridos y paradas más cercanos a cierta ubicación o zona geográfica), agregando así distintas funcionalidades a la aplicación.

Open Geospatial Consortium (OGC)
Consorcio internacional que nuclea más de 500 empresas, organizaciones gubernamentales, institutos de investigación y universidades con el objetivo de que la información geoespacial sea fácil de encontrar, accesible, interoperable y reutilizable[3].

OGC Web Services (OWS)
Nombre genérico con el que se agrupa a todos los estándares de Web Services de OGC.

WMS
Define un protocolo para obtener mapas dinámicos a partir de información geográfica distribuida. Definido en el estándar ISO 19128[4]. Sólo permite consulta de datos, y no su escritura/modificación.

WFS
Define un protocolo para consultar y modificar información geográfica. Definido en el estándar ISO 19142[5].

SRID
Un identificador de referencia espacial (SRID) es un identificador único asociado con un sistema de coordenadas específico, tolerancia y resolución[6].

3. Descripción del problema

El problema planteado consiste en desarrollar un sistema georeferenciado con información de paradas, líneas, horarios y sus recorridos. Una herramienta que permita tanto consultar dicha información en tiempo real por usuarios anónimos, como también actualizarla por parte de administradores.

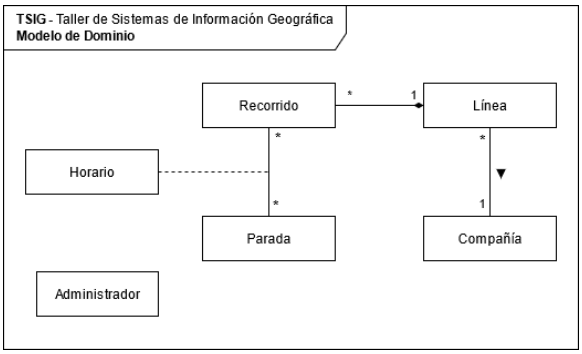
La herramienta deberá proveer un mapa con información para el usuario final del transporte público, pudiendo realizar filtros para buscar recorridos, líneas o paradas que cumplan ciertas condiciones.

Las funcionalidades solicitadas fueron las siguientes:

- Login del usuario administrador para acceso a backend.
- Alta, baja y modificación de líneas, paradas y recorridos por parte de administrador.
- Consulta de información actualizada de líneas, paradas y recorridos por parte de usuarios anónimos

4. Solución planteada

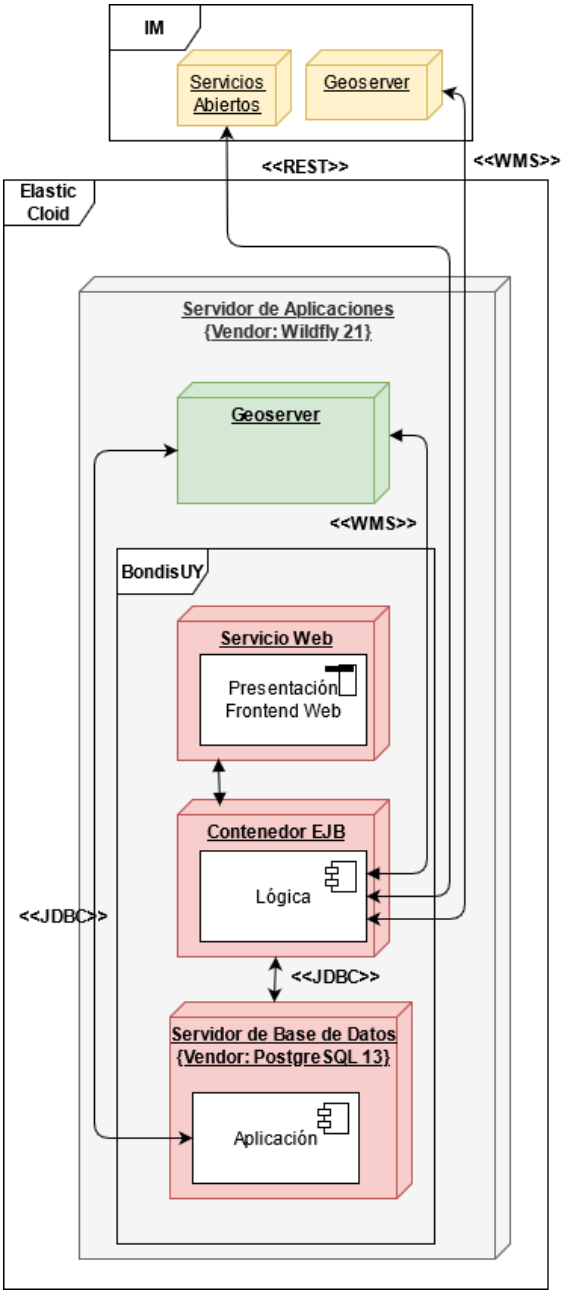
La solución lograda fue un sistema con la siguiente estructura de conceptos, de forma de reflejar lo mejor posible la realidad planteada en la letra del proyecto:



Como se muestra en el Modelo de Dominio, se entiende que existen paradas y recorridos vinculados de N a N, con horario como concepto asociativo. Los recorridos pertenecen a una línea, mientras que las líneas tienen varios recorridos. La línea pertenece a una compañía, pero cada compañía puede tener varias líneas. Y por último, existen Administradores para gestionar todo esto.

5. Arquitectura del sistema

A continuación, se detalla la arquitectura implementada con el siguiente esquema:



6. Implementación

Dentro de las tecnologías utilizadas en el desarrollo de la solución se encuentran, por un lado, Java EE con JDK v11 para implementar la lógica de negocio junto con un servidor de base de datos PostgreSQL v13 con la extensión PostGIS v3.0.3 para realizar la persistencia de los datos.

En lo que respecta a la presentación, se integró el framework Bootstrap v4.6.0 en páginas JSP junto con JQuery v3.4.1 y Ajax para optimizar la experiencia de usuario. En cuanto al manejo de los mapas, se emplea la librería OpenLayers v6.5.0.

Como servidor de aplicaciones se utilizó Wildfly v21.0.0 y Geoserver v2.19.0 para el uso de operaciones geoespaciales.

Para utilizar datos ya existentes, se definió utilizar los datos abiertos del SIG de la Intendencia de Montevideo. Para lograr eso, se definió consumir (a través de servicios web REST) los datos relacionados a calles, con la finalidad de contar con funcionalidades, como ser: buscar direcciones, esquinas y calles.

A su vez, se definió importar los datos de la ubicación de paradas de ómnibus, líneas, recorridos y sus respectivos horarios (a través de scripts en Python) y cargarlos de forma manual en la base de datos de PostGIS (a través de scripts en SQL), al realizar el despliegue del ambiente.

Aprovechando la funcionalidad disponible a partir de la versión 2.1.0 de Geoserver, se agregaron algunas capas a través de vistas SQL[7]. Esto permitió contar con capas sin tener que generar dichas vistas ni tablas a nivel de la base de datos.

Todas las capas geográficas utilizan el SRID 32721 en la base de datos, pero se proyectan a otro SRID al momento de consumirlas desde el frontend (ver sección 6.2 Problemas encontrados).

Para el despliegue de todos los servidores se eligió la plataforma Elastic Cloud de Antel.

6.1 Productos y Herramientas

La elección de herramientas a utilizar se definió de forma rápida, debido a buenas experiencias pasadas con algunas de ellas, como también el hecho de haber trabajado durante el curso con las que aún no se conocían, y por último, la necesidad de aprovechar el tiempo para comenzar cuanto antes con la implementación de la solución final.

Tabla 1. Evaluación de Productos y Herramientas

Producto	Puntos Fuertes	Puntos Débiles	Evaluación General
Java EE	Lenguaje ampliamente utilizado, potente y documentado.	- Problemas para integrar objetos geográficos. - Consume muchos recursos.	Lenguaje apto y conveniente para el sistema solicitado.
Wildfly	Servidor de aplicaciones potente y maduro.	Un poco lento y no tan liviano como otros (Tomcat por ejemplo).	Herramienta apta y conveniente para el sistema solicitado.
JSP	Fácil integración de código Java en formato HTML	Difícil dar seguimiento a los errores (debug).	Herramienta apta y conveniente para el sistema solicitado.
Bootstrap	-Disponibilidad de muchos estilos y variedad de recursos. -Soportado por la mayoría de navegadores -Estilos Responsive	-Muchos ajustes si se precisa customizar el diseño. -Salida verborrágica en código html generado.	Herramienta apta y conveniente para el sistema solicitado.
Ajax	-Tecnología madura y fácil de implementar.	-Riesgos de problemas de incompatibilidad por navegadores que no usen Javascript.	Herramienta apta y conveniente para el sistema solicitado.
	-Mejora la experiencia de usuario (UX).		
JQuery	Librería de Javascript sumamente compatible y versátil.	No hay un estándar. Hay muchas versiones de JQuery, algunas más compatibles que otras.	Herramienta apta y conveniente para el sistema solicitado.
Postgres & PostGIS	Motor y extensión sumamente útiles para manejar información geográfica	No se encontraron desventajas relevantes.	Herramienta apta y conveniente para el sistema solicitado.
GeoServer	Interfaz gráfica sencilla de utilizar y amigable.	No se encontraron desventajas relevantes.	Herramienta apta y conveniente para el sistema solicitado.
OpenLayers	-Amplia documentación y disponibilidad de librerías. -Ofrece muchas funcionalidades.	Más complejo de utilizar que otras herramientas similares.	Herramienta apta y conveniente para el sistema solicitado.

6.2 Problemas Encontrados

1. Durante la etapa de desarrollo de una aplicación para otra materia, se presentó un inconveniente al intentar usar objetos geográficos en Java. Puntualmente, se quería que determinada entidad tuviera un Point como propiedad geom.

   Para intentar solucionar eso, se utilizó Hibernate Spatial y la librería Geolatte. La creación del punto se logró hacer sin problemas, pero luego, cada vez que se quería consultar información de un determinado elemento de dicha entidad, se generaba un error que hacía referencia a un PGObject. Dado que no se lograba resolver el problema, se

terminó resolviendo con la creación de una tabla aparte para la geometría.

Luego de esa experiencia, al abordar este proyecto se decidió resolver el desafío de los objetos geométricos a través de la base de datos. Es decir, se generan tablas para las entidades Parada y Recorrido con todas sus propiedades menos la geometría. Después, desde PostgreSQL, se agregan las geometrías, las cuales son manipuladas desde GeoServer.

2. Otro de los inconvenientes encontrados fue a la hora de manejar los SRID, dado que al intentar utilizar el sistema de proyección original de las capas (es decir, el 32721), OpenLayers generaba problemas y no permitía la correcta visualización de la información geográfica. Esto se solucionó manteniendo la proyección original en la base de datos pero proyectando al SRID 4232 cada vez que se intenta visualizar los datos en el frontend, como también volviendo a proyectar de 4232 a 32721 cada vez que se intenta persistir en la base de datos.

3. A su vez, otro inconveniente encontrado fue a la hora de realizar el despliegue de la aplicación, tanto para el ambiente de Desarrollo como el de Producción.

   Se notó que al utilizar Wildfly y Geoserver de forma separada se generaba un error por intentar realizar solicitudes de origen cruzado. Es decir, que se estuviera intentando acceder desde Wildfly a recursos (por API) del Geoserver, en un servidor (dominio) aparte.

   El error generado era "*Access to XMLHttpRequest at […] has been blocked by CORS policy [...]* ".

   Para estos casos, el World Wide Web Consortium (W3C) recomienda el mecanismo de Intercambio de Recursos de Origen Cruzado (CORS en inglés), utilizando las cabeceras CORS[8].

   Sin embargo, esto fue difícil de implementar y el problema se solucionó utilizando el binario de Geoserver en el propio Wildfly, de forma tal que éste lo despliegue y se acceda a todo (tanto Wildfly como Geoserver) a través del mismo puerto (8080 en el caso de este proyecto).

4. Por último, otro problema encontrado durante todo el tiempo de desarrollo del proyecto fue para poder desplegar de una manera fácil y rápida un ambiente (ya sea de desarrollo como también de producción), dadas las complicaciones descritas anteriormente.

   Se intentó configurar ambientes de desarrollo con contenedores en Docker. Sin embargo, no se logró automatizar las configuraciones iniciales ya mencionadas, quedando como un punto a trabajar a futuro (ver sección 9).

7. Evaluación de la solución

Puntos fuertes

- Información precisa y confiable, al utilizar datos oficiales del SIG de la Intendencia de Montevideo.
- Fácil de utilizar, con interfaz simple e intuitiva, facilitando la experiencia de usuario (UX).

- Performante. Brinda la información de forma rápida y sin grandes demoras de carga.

Puntos débiles

- Se delegan operaciones de visualización de datos en el mapa del lado del cliente.

8. Desarrollo del proyecto

El proyecto se desarrolló sin un cronograma específico, pero con una clara división de tareas, de acuerdo a las siguientes áreas de trabajo:

- Desarrollo backend
- Desarrollo frontend
- Infraestructura
- Testing
- Documentación

Cada subgrupo se organizó de forma independiente, pero en constante comunicación y coordinación con el resto del equipo a través de Slack y reuniones virtuales por Google Meet.

9. Conclusiones y trabajo a futuro

Como trabajo adicional a lo ya logrado, se recomienda seguir trabajando en las siguientes funcionalidades:

- Integración con un servidor SMTP, de forma tal que se pueda enviar notificaciones por correo electrónico ante ciertos eventos (modificación de líneas, recorridos y/o paradas).
- Posibilidad de marcar ciertas líneas y recorridos como favoritos, de forma tal de poder recibir notificaciones solamente de la información de interés.
- En caso que el usuario no identifique líneas y/o recorridos como favoritos, generar notificaciones basado en el uso del usuario, tomando las líneas y recorridos más consultados como elementos para el envío de notificaciones ante cambios en los mismos.

A su vez, quedó pendiente la configuración del ambiente de la aplicación con contenedores, en una arquitectura de microservicios, y con la carga de datos y capas automatizada, de forma de facilitar tanto su despliegue, como también su mantenimiento y disponibilidad.

10. Referencias

[1] Sistema de Información Geográfica de la Intendencia de Montevideo
https://sig.montevideo.gub.uy/content/geoservicios-web
[Consulta: Junio 2021]

[2] Introducción a SIG
https://resources.arcgis.com/es/help/getting-started/articles/026n0000000t000000.htm
[Consulta: Junio 2021]

[3] About OGC
https://www.ogc.org/about
[Consulta: Junio 2021]

[4] ISO 19128:2005 - Geographic information — Web map server interface
https://www.iso.org/standard/32546.html
[Consulta: Junio 2021]

[5] ISO 19142:2010 - Geographic information — Web Feature Service
https://www.iso.org/standard/42136.html
[Consulta: Junio 2021]

[6] ¿Qué es un SRID?
https://desktop.arcgis.com/es/arcmap/10.4/manage-data/using-sql-with-gdbs/what-is-an-srid.htm
[Consulta: Junio 2021]

[7] SQL Views
https://docs.geoserver.org/latest/en/user/data/database/sqlview.html
[Consulta: Junio 2021]

[8] Control de acceso HTTP (CORS)
https://developer.mozilla.org/es/docs/Web/HTTP/CORS
[Consulta: Junio 2021]