ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Aplicación GIS utilizando mapas en multi-capas aplicado a la planificación urbana

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniería en Telemática

Presentado por:

Andrés Darío Robles Chiriguaya

GUAYAQUIL - ECUADOR

Año: 2023

DEDICATORIA

El presente proyecto lo dedico en principio a la mujer más importante que ha marcado mi vida, mi madre abuela Tula. A lo largo de mi crecimiento académico y personal, ella fue un pilar fundamental, la que me enseñó que a pesar de las adversidades que se presenten en el camino siempre debo estar firme y nunca dar el brazo a torcer. Por último, al resto de mis familiares y amigos que siempre me dieron ánimos de aliento cuando estuve a punto de tirar la toalla en esta travesía.

AGRADECIMIENTOS

Quiero darle las gracias a Dios, quien es el único que sabe lo duro que fue esta travesía, gracias por haber escuchado mis oraciones en los momentos que te pedía que me dieras fuerzas cuando prácticamente estaba decidido a dejar pasar la oportunidad de culminar mis estudios, gracias Padre, porque solo Tú conoces mi situación actual y mis más profundos sentimientos. Por último a mi amigo e Ingeniero Jose Luis Salazar, quien fue la persona que me daba aliento y quien me brindó ideas en la elaboración de este proyecto de grado.

DECLARACIÓN EXPRESA

”Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; Andrés Darío y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual”

Andrés Darío Robles

Andrés Darío Robles Chiriguaya

EVALUADORES

Firmado electrónicamente por:
IGNACIO MARIN GARCIA

IGNACIO MARÍN, PhD.

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
WASHINGTON ADRIAN VELASQUEZ VARGAS

WASHINGTON VELÁSQUEZ, PhD.

PROFESOR TUTOR

RESUMEN

Debido al conocimiento limitado y de uso de las aplicaciones web GIS (Sistema de Información Geográfico), existen territorios en diferentes partes del mundo que no cuentan con un plan de gestión y organización adecuados que satisfagan a las verdaderas necesidades de la población de dichas zonas.

Este proyecto tiene como principal objetivo crear una aplicación web GIS que sea amigable con el usuario final, de tal forma que su uso sea concreto y preciso con el fin de que logre captar interés por parte del usuario final. Las categorías que intervienen en esta aplicación son parques, carreteras, bosques y electricidad, la idea es permitirle al usuario seleccionar un área de estudio y mostrar mediante capas la distribución geográfica exacta en el mapa mediante la superposición de capas y mostrando la información de cada una. La aplicación fue desarrollada en lenguaje de programación HTML (Lenguaje de Marcas de Hipertexto), lo que permitió la fácil integración con LeafLet. Esta librería permitió la creación de mapas interactivos, debido a que brinda herramientas que permitieron realizar las acciones necesarias para la interacción con el usuario. Para obtener los datos reales de las categorías que se implementaron, se decidió realizar peticiones POST a OpenStreetMAP, una base de datos colaborativa que permite a la comunidad subir información libremente. La API (Interfaz de programación de aplicaciones) de Overpass Turbo fue el encargado de hacer de puente entre la aplicación y OSM (OpenStreetMAP). Los resultados obtenidos dieron para realizar un pequeño caso de estudio, aplicando métricas sencillas que permitieron evaluar la efectividad del uso de las herramientas tecnológicas.

Palabras Clave: GIS, OpenStreetMAP, mapa interactivo, API, Overpass Turbo

ABSTRACT

Due to limited knowledge and use of web GIS (Geographical Information System) applications, there are territories in different parts of the world that lack adequate management and organization plans to meet the true needs of the population in those areas.

This project's main objective is to create a user-friendly web GIS application in such a way that its use is concise and precise, aiming to capture the interest of the end-user. The categories involved in this application include parks, roads, forests, and electricity. The idea is to allow the user to select a study area and display the exact geographical distribution on the map through layer overlay, showing the information for each one.

The application was developed in HTML (HyperText Markup Language) programming language, which facilitated seamless integration with LeafLet. This library enabled the creation of interactive maps by providing tools that allowed for necessary interactions with the user. To obtain real data for the implemented categories, it was decided to make POST requests to OpenStreetMAP, a collaborative database that allows the community to freely upload information. The Overpass Turbo API (Application Programming Interface) acted as a bridge between the application and OSM (OpenStreetMAP).

The results obtained were used to conduct a small case study, applying simple metrics to evaluate the effectiveness of using technological tools.

Keywords: GIS, OpenStreetMAP, interactive maps, API, Overpass Turbo

ÍNDICE GENERAL

ABREVIATURAS

AJAX	Asynchronous JavaScript and XML
API	Interfaz de Programación de Aplicaciones (del inglés Application Programming Interface)
GIS	Sistema de Información Geográfico (del ingés: eographical Information System)
GPS	Sistema de Posicionamiento Global (del inglés: Global Psistioning System)
GUI	Interfaz gráfica de usuario (del inglés: Graphical User Interface)
HTML	Lenguaje de Marcas de Hipertexto (del inglés: HyperText Markup Language)
OGC	Consorcio Geoespacial Abierto (del inglés: Open Geospatial Consortium)
OSM	OpenStreetMAP
POT	Planes de Ordenamiento Territorial (del Inglés: Territorial Planning Plans)
RAM	Memoria de acceso aleatorio (del inglés Random Access Memory)
SMDB	Sistema de Gestión de Bases de Datos (del inglés: Database Management System)
VsCode	Visual Studio Code

SIMBOLOGÍA

km^2	Kilómetros cuadrados
v	Voltios
ms	Milisegundos
%	Porcentaje
calles/km^2	Calles por kilómetros cuadrados
peticiones/s	Peticiones por segundos

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CODIGOS DE PROGRAMA

CAPÍTULO 1

1. INTRODUCCIÓN

La planificación urbana constituye un elemento de máxima relevancia en la elaboración de una guía de decisiones acertadas que permitan alcanzar futuros sostenibles en los diversos territorios geográficos del planeta Tierra. Este imperativo surge como consecuencia del acelerado crecimiento poblacional experimentado en los últimos años. En la actualidad, se dispone de herramientas tecnológicas que asisten a los planificadores urbanos en el análisis y evaluación de datos, los cuales les posibilitan comprender la situación de una ubicación específica. Estas herramientas son conocidas como Sistemas de Información Geográfica (GIS).

Un Sistema de Información Geográfica permite la recopilación, gestión y análisis de datos geoespaciales mediante la interpretación de mapas temáticos (Constantinidis and Pérsico, 2021). De esta manera, se logra seleccionar la información pertinente para abordar problemas específicos.

Es importante tener presente que un mapa temático está compuesto por una tabla de contenido que posibilita al lector añadir capas de información a un mapa base que representa ubicaciones del mundo real (Esri, 2011).

Estas herramientas permiten la integración de datos geográficos con información demográfica, económica y ambiental, lo cual facilita la comprensión de la distribución espacial de los recursos, servicios y necesidades de la comunidad.

Las simulaciones obtenidas mediante los mapas temáticos nos proporcionan datos relevantes acerca del medio ambiente, los patrones de movimiento dentro de la ciudad y la calidad de vida, lo cual contribuye a la toma de decisiones fundamentadas en evidencia empírica.

1.1 Descripción del problema

La disponibilidad y precisión de información geográfica son de vital importancia para una planificación urbana eficiente (Batty, 2013). Sin embargo en muchas ocasiones existe un conocimiento limitado y real del uso de estas aplicaciones.

En la tesis realizada por Arias y Olave se da un claro ejemplo de lo mencionado anteriormente e indican que los municipios y despachos de muchas ciudades de Colombia no cuentan con un buen proceso de gestión y planificación de tomas de decisiones y como resultado se obtiene Planes de Ordenamiento Territorial (POT) muy alejados a la realidad y necesidades de la población (Arias and Olave, 2015).

Además, la integración de las aplicaciones GIS en los procesos de toma de decisiones de la planificación urbana es otro desafío importante. La información geoespacial generada a través de las aplicaciones GIS debe ser compartida y comunicada de manera efectiva a los responsables de la toma de decisiones para garantizar su utilización en el diseño y la implementación de políticas urbanas. La falta de integración y colaboración entre los profesionales de GIS y los responsables de la toma de decisiones puede dificultar la adopción de los resultados generados por estas aplicaciones en la planificación urbana (Campbell, 1996).

1.2 Justificación del Problema

Es notable los grandes avances que han surgido a lo largo del tiempo con respecto a los Visores de información geográfica e infraestructuras de datos espaciales. Como se nos indica en (Puebla, 2010) , los visores geográficos están prácticamente al alcance de todo el público, siendo Google Maps y Google Earth los preferidos, incluso hasta para científicos y técnicos que se encuentran renuentes a utilizar software GIS.

La idea de este proyecto es implementar una aplicación web GIS que sea de fácil entendimiento y atractiva para los planificadores urbanos. La aplicación contará con información actualizada periódicamente, gracias al uso de tecnologías de código abierto. Esto garantizará que no sea necesario pagar ningún valor económico por su uso. En la actualidad, con la sobrepoblación existente, es necesario tener en cuenta numerosos factores al realizar ampliaciones o construcciones de nuevas infraestructuras para

atender las necesidades de toda la comunidad en las diferentes ciudades. Por lo tanto, contar con información actualizada y precisa es fundamental. La aplicación web GIS proporcionará a los profesionales la información necesaria para abordar estos desafíos. Así pues, gracias a esta aplicación, los planificadores urbanos podrán tomar decisiones fundamentadas y eficientes, optimizando los recursos disponibles y asegurando un desarrollo urbano sostenible.

1.3 Objetivos

El presente proyecto tiene como objetivo general:

- Desarrollar una herramienta que permita la toma de decisiones y facilite la gestión de información espacial relacionada con el diseño y desarrollo de áreas urbanas, garantizando un futuro sostenible y eficiente.

Los objetivos específicos a cumplir son:

- Obtener los datos que proporciona OpenStreetMAP de las diferentes capas implementadas, con el fin de que se identifique como se encuentra distribuida una zona en específico.
- Implementar el uso de capas mediante la integración de OpenStreetMAP, seleccionando y personalizando las capas adecuadas y evaluando su efectividad para un mejoramiento de la visualización y comprensión de la información geográfica.
- Diseñar una aplicación web que garantice al usuario la elección de un área de estudio en específico, donde la interacción sea de manera clara y sencilla.

1.4 Alcance y Limitaciones

- Este proyecto será en principio una aplicación que permitirá al usuario observar mediante capas como se encuentran distribuidos los espacios territoriales de una ciudad o área en específico.

- Para esta aplicación se ha decidido implementar al menos cuatro capas que permitirán observar de qué manera se encuentran distribuidas dentro de un espacio territorial.

- La aplicación web mostrará la información que se encuentren hasta el momento en la base de datos de OpenStreetMAP, ya que, al ser de colaboración abierta se tendrá que esperar que la comunidad actualice la información correspondiente.

1.5 Estado del arte

En esta sección se exponen diversos proyectos vinculados con el proyecto actual de grado titulado ”Aplicación GIS utilizando mapas en multi-capas aplicado a la planificación urbana”.

En su tesis titulada ”Diseño e implementación de un Mapa Interactivo utilizando Web Mapping y Base de Datos Espacial: Ciudad de Quevedo”, Erazo (Moreta, 2009) tiene como objetivo desarrollar una aplicación web que contenga un mapa del cantón Quevedo (Figura 1.1). Esta aplicación permite visualizar datos relevantes para futuros trabajos en el área, como la división político-administrativa, la red de calles, los recorridos de autobuses urbanos y lugares de interés, como alojamientos, bancos, instituciones educativas y espacios públicos. Además, ofrece la posibilidad de realizar búsquedas de lugares, intersecciones de calles y rutas más cortas (tanto peatonales como vehiculares) entre dos puntos. Para determinar las rutas más cortas, se utiliza el algoritmo de Dijkstra, el cual encuentra el camino más eficiente desde un punto de origen hacia el resto de los puntos en un grafo con pesos en cada arista.

El desarrollo de este sitio web se ha basado en la técnica AJAX, lo cual evita la necesidad de recargar por completo la página cada vez que el usuario realiza una acción, lo que mejora considerablemente el rendimiento de la aplicación. Para crear este sitio, se utilizaron herramientas de código abierto, incluyendo el sistema operativo, el servidor web, los lenguajes de programación, el servidor de mapas y el sistema de gestión de bases de datos.

Los datos necesarios se almacenaron en una base de datos PostgreSQL con extensión espacial. Para recopilar los puntos de interés, se recorrió la ciudad de Quevedo con la ayuda de un GPS, recolectando toda la información alfanumérica disponible. Para

la construcción de la red de calles, el autor utilizó la aplicación de escritorio ArcGIS, mientras que para los recorridos de autobuses urbanos se basó en los permisos de operación otorgados por el Consejo Provincial de Tránsito y Transporte Terrestre de Los Ríos. Para la búsqueda de rutas, se emplearon las funciones proporcionadas por pgRouting, un proyecto de código abierto mantenido por PostLBS, que ofrece capacidades de enrutamiento para PostgreSQL/PostGIS.

Como resultado, se obtiene un mapa interactivo que permite visualizar información básica del cantón de Quevedo.

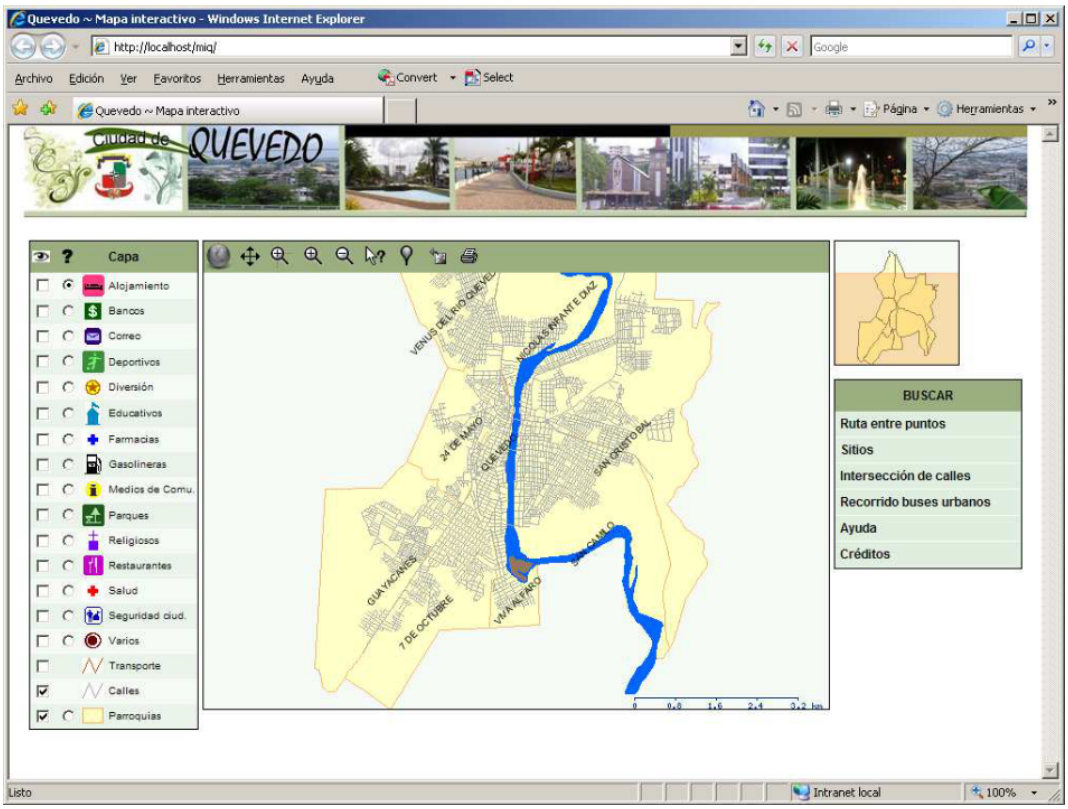


Figura 1.1: Diseño de la página web del mapa interactivo de la ciudad de Quevedo realizado por Erazo.

(a) Fuente: https://repositorio.usfq.edu.ec/bitstream/23000/984/1/95301.pdf

Por otro lado (Iglesias, 2014) en su tesis titulada ”Diseño e implementación de una aplicación web para la gestión de información geográfica del departamento de desarrollo forestal de la CONAFOR estado de México” desarrolló una aplicación web mapping como un sistema de visualización catográfica, edición y consulta de información. Dicha data espacial y alfanumérica fue proporcionada por los programas de desarrollo forestal de la CONAFOR (Desarrollo Forestal de la Comisión Nacional Forestal).

El desarrollo se ha llevado a cabo utilizando el enfoque de modelo cascada, también conocido como lineal secuencial, el cual se basa en el ciclo de vida del software. Este enfoque proporciona métodos y técnicas para la creación de software, específicamente aplicaciones de mapeo web. En este proceso, se plantean los requisitos que el sistema debe cumplir, se recopilan los datos, se diseñan la base de datos y la interfaz de usuario, se implementa y se evalúan los resultados obtenidos.

Dentro del sistema se encuentra una Base de Datos que almacena los registros de los apoyos otorgados. Para su funcionamiento, se utilizan diversas aplicaciones, entre ellas un Gestor de Bases de Datos PostgreSQL en conjunto con la extensión espacial PostGIS. Además, se emplea un servidor de mapas llamado Map Server, así como un Framework diseñado para la representación espacial en la web. Para las tareas de edición cartográfica, se utilizan los editores ArcGIS y Quantum GIS.

El resultado final del diseño de la aplicación se lo observa en la figura 1.2.

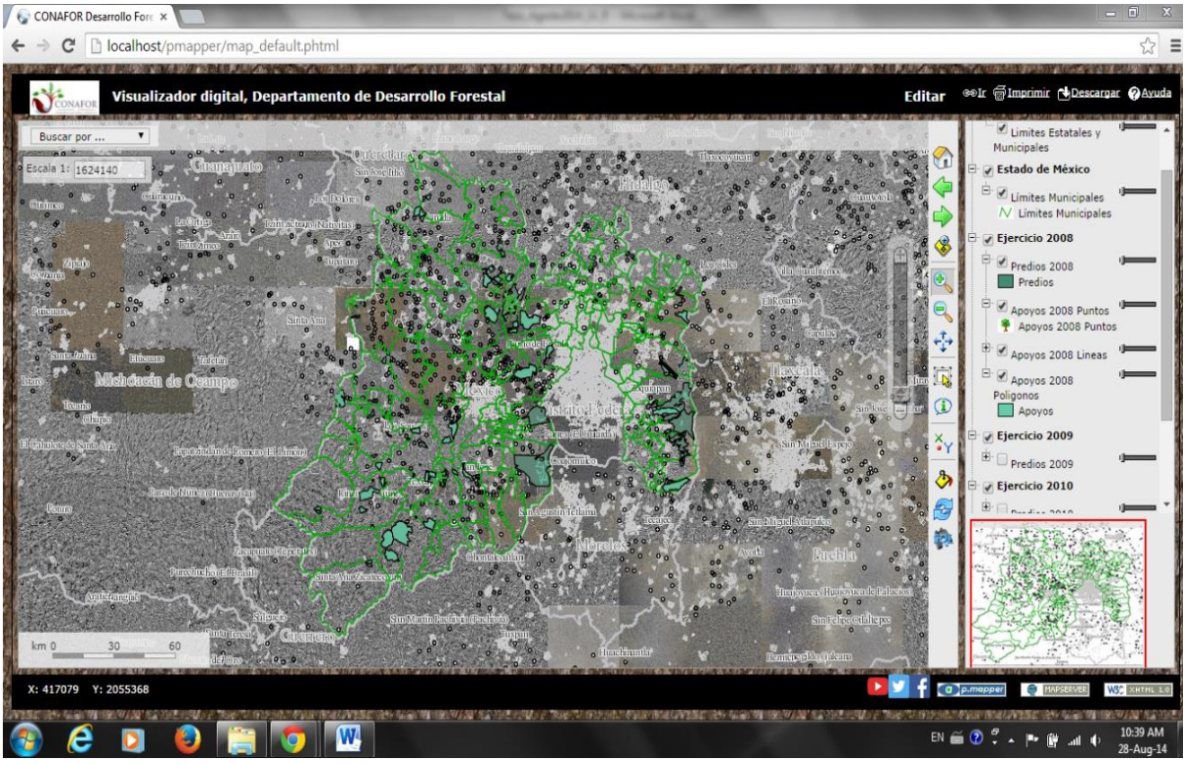


Figura 1.2: Diseño de la página web realizada por Peña

(a) Fuente: http://surl.li/kspnd

1.6 Marco Teórico

A continuación se revisan los conceptos más importantes, los cuales se encuentran relacionados estrechamente con la realización de este proyecto.

1.6.1 Sistemas de Información Geográfica

La información geográfica desempeña un papel fundamental en una amplia variedad de actividades, y los Sistemas de Información Geográfica (GIS) son la herramienta básica para su manejo y utilización. En el contexto de los países en desarrollo, el uso combinado de GIS, datos geoespaciales de libre acceso y estándares abiertos se presenta como una solución óptima que puede contribuir de manera positiva a su progreso. Esta combinación representa la mejor alternativa posible para trabajar con información geográfica y ofrece un enfoque prometedor en el avance de estos países (Olaya, 2009).

Se debe entender a un GIS como un elemento complejo que engloba demasiados elementos (datos, procesos, visualización, tecnología y el factor organizativo) los cuales se encuentran conectados y cumplen con una función en específico (Olaya, 2014).

Podemos definir de una manera más sólida a un GIS como un sistema que combina la tecnología informática, las personas y la información geográfica (Esri, n.d.), con el objetivo principal de capturar, analizar, almacenar, editar y representar datos con referencia geográfica (Korte, 2011).

1.6.2 Componentes de un Sistema de Información Geográfica

Un Sistema de Información Geográfica (GIS) consta de los mismos elementos que cualquier sistema de información (figura 1.4), con un enfoque en la toma de decisiones. Sin embargo, su característica distintiva radica en la capacidad de gestionar datos espaciales de manera integrada con los datos descriptivos. Los elementos son los siguientes:

- **Los equipos o ”hardware”:** El hardware es lo que conforma los elementos físicos del ordenador. En tanto que el software y el tipo de tarea a realizar trabajan

directamente en conjunto con el hardware. Algunas veces, es importante contar con una gran cantidad de datos y por ende un almacenaje adecuado para ello, por lo que el ordenador debe estar equipado con un disco duro de una capacidad determinada. Si la tarea a realizar necesita un procesamiento de datos complejo, el ordenador debe contar con una memoria RAM y una velocidad de cálculo específica. Por último, las características del ordenador deben permitir la visualización de imágenes en pantalla, sobre todo para casos donde se requiera una reproducción gráfica o cartográfica con un nivel mínimo de detalle (Preciado, 2020).

- **El componente operativo o ”software”:** Son conjuntos de algoritmos que permiten acceder, presentar, analizar y sintetizar los datos en una base de datos, ya sea en función de sus características espaciales o no espaciales. Estos programas se integran habitualmente con el sistema operativo de la computadora, trabajando conjuntamente con otras aplicaciones en una sola sesión de trabajo (Navarro et al., 2011).

- **Datos:** Dentro de un GIS, los componentes fundamentales son los datos geográficos y los datos tabulares relacionados. La recolección de estos datos suele ser un proceso largo que puede retrasar el desarrollo del producto final, pero justifica en gran medida la inversión realizada. Estos datos pueden obtenerse a través de fuentes internas o proveedores comerciales de datos. Para organizar y gestionar eficientemente estos datos, la mayoría de los GIS implementan un Sistema de Gestión de Bases de Datos Espaciales (SMDB), lo que facilita su creación y mantenimiento. Esto permite un manejo más efectivo de los datos en general (Quintana, 2015).

- **Los métodos:** Se refieren a las instrucciones escritas que están destinadas a los operadores o usuarios con el fin de facilitar el manejo eficiente y seguro de un SIG. Estos procedimientos abarcan tanto aspectos técnicos como de uso, abarcando los manuales técnicos y de usuario para los paquetes y programas desarrollados por los usuarios (Saavedra, 1992).

- **Los usuarios:** El personal encargado del manejo de datos gráficos está debidamente capacitado y cuenta con una comprensión sólida de los principios y técnicas asociadas. Por lo general, los analistas y diseñadores de sistemas

están formados por profesionales multidisciplinarios que no solo resuelven los desafíos relacionados con la entrada y manipulación de datos, sino que también son capaces de conceptualizar y realizar análisis eficientes de las bases de datos integradas y de las modelaciones desarrolladas utilizando tecnologías multicriterio (Saavedra, 1992).

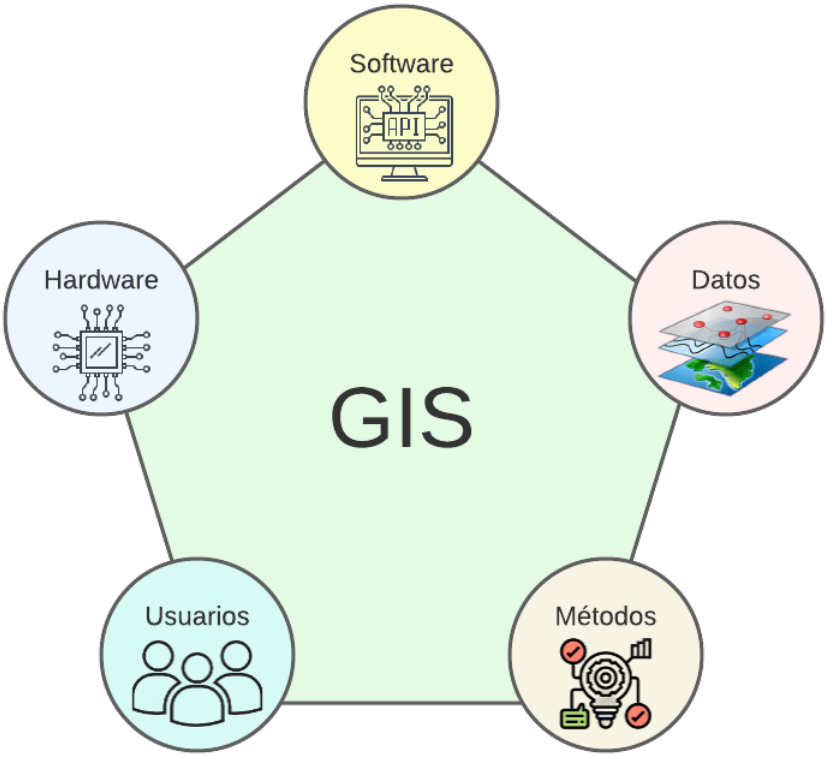


Figura 1.3: Componentes de un Sistema de Información Geográfico

1.6.3 Tareas de un Sistema de Información Geográfica

Tiene como objetivo desarrollar 5 metas las cuales son:

- **Ingreso:** Los datos geográficos primero deberan ser digitalizados en el formato apropiado, esto no es más que la conversión de formato análogos a archivos computacionales (Reino, 2014).
- **Manipulación:** Antes de integrar la información en el sistema actual, es necesario realizar un análisis para garantizar su compatibilidad. En caso de que no sea

compatible, se deben realizar las transformaciones necesarias para que sea utilizable. Estas transformaciones pueden ser temporales, con el propósito de visualización, o permanentes, con el fin de realizar análisis más profundos (Reino, 2014).

- **Manejo y Administración:** En proyectos GIS a pequeña escala, puede resultar útil almacenar la información geográfica como archivos simples. Sin embargo, a medida que el volumen de datos aumenta, es recomendable utilizar un sistema de administración de bases de datos (DBMS) que permita gestionar de manera eficiente y organizada dicha colección de datos. En el ámbito de los GIS, el diseño más utilizado y beneficioso es el diseño relacional. En este diseño, los datos se almacenan conceptualmente como una colección de tablas interconectadas a través de campos comunes. El diseño relacional se ha popularizado por su flexibilidad y su capacidad de distribución en aplicaciones tanto dentro como fuera de los GIS (del Río San José, 2010).

- **Consulta y Análisis:** Los Sistemas de Información Geográfica (GIS) ofrecen la posibilidad de realizar consultas simples y avanzadas, lo que brinda a los técnicos y especialistas información relevante y oportuna. La verdadera potencia de la tecnología GIS se despliega al utilizarla para analizar datos geográficos, identificar patrones y tendencias. Las herramientas de análisis sofisticadas que brinda permiten obtener información valiosa a partir de los datos geográficos (Reino, 2014).

- **Visualización:** En la actualidad, están surgiendo numerosos programas especializados en la visualización y consulta de datos espaciales, conocidos como aplicaciones de mapeo de escritorio. Estas aplicaciones representan un complemento a los SIG más que ser SIG por sí mismas. Sin embargo, gran parte de la popularización de los SIG se debe a este tipo de aplicaciones, ya que han facilitado la introducción de la dimensión espacial de la información en entornos de trabajo donde previamente no existía una tradición en este aspecto, como en empresas y otros contextos laborales (Sarría, 2006).

1.6.4 Web Mapping

La información cartográfica disponible en la web, conocida como mapas en línea, puede ser procesada, diseñada y visualizada a través de la red informática global. El Web Mapping se enfoca en suministrar estos datos en formatos compatibles con las especificaciones del Consorcio Geoespacial Abierto (OGC) para lograr la interoperabilidad de los datos espaciales. Es importante tener en cuenta que un servidor de Web Mapping no es un Sistema de Información Geográfica (GIS) en sí mismo, sino que se basa en la información generada por un GIS para respaldar la cartografía y los atributos asociados, permitiendo su visualización a través de una interfaz gráfica de usuario. Hasta el momento, no todas las capacidades de análisis espacial ofrecidas por un SIG de escritorio pueden compararse con las que proporciona una aplicación de Web Mapping (Pérez et al., 2015).

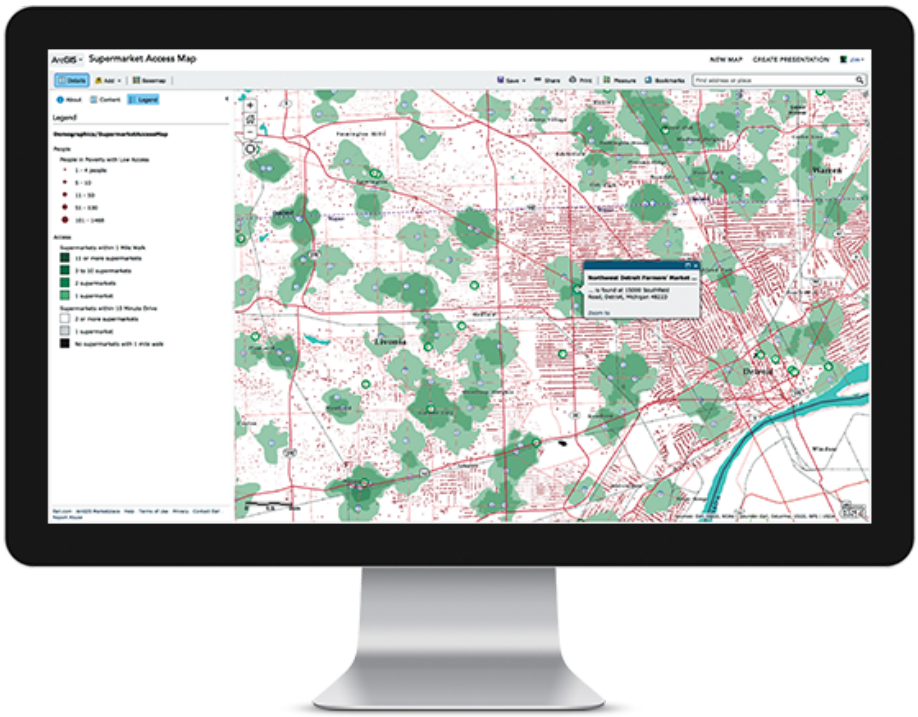


Figura 1.4: Web Mapping.

(a) Fuente: https://www.esri.com/about/newsroom/insider/the-role-of-web-maps/

1.6.5 FrontEnd

Dentro del contexto del desarrollo de aplicaciones web, se implica el uso de tecnologías con las cuales el usuario interactúa de manera directa. Estas tecnologías generalmente

se crean utilizando los lenguajes de HTML, CSS y JavaScript, y también se utilizan herramientas de diseño gráfico como Photoshop o Fireworks. El propósito es construir la interfaz gráfica de usuario (GUI) con el objetivo de proporcionar una experiencia de uso altamente valorada por el usuario final, lo que puede requerir investigación, estudios y pruebas en algunos casos. Asimismo, en el desarrollo de aplicaciones web, es posible crear el front-end de la aplicación sin necesidad de contar con un back-end que interactúe con la base de datos (Caballero, 2016).

1.6.6 BackEnd

Se conoce como BackEnd a la parte de un software que se encarga de acceder a los datos y que no es visible para el usuario final. Esta capa contiene la lógica de la aplicación que maneja los datos. Es importante mencionar que los datos de una aplicación se almacenan en una base de datos dentro de un servidor (Ibarra et al., 2021). El responsable del BackEnd es la persona que trabaja en el lado del servidor y se forma como desarrollador de aplicaciones web o de aplicaciones multiplataforma. Debe estudiar diversos lenguajes de programación necesarios para llevar a cabo su trabajo, y esto dependerá de la empresa en la que esté empleado, ya que cada una puede requerir habilidades específicas. También debe comprender cómo interactuar con distintas bases de datos, conocer las diferencias entre ellas y entender las características de las más utilizadas (Ibarra et al., 2021).

Es relevante destacar que ser un experto en BackEnd no implica desconocer por completo el trabajo realizado por el FrontEnd, sino más bien tener los conocimientos necesarios para colaborar eficientemente en equipo, ya que ambas partes se complementan.

1.6.7 Librerías

Las librerías son conjuntos de clases o métodos que ofrecen funcionalidades a otras aplicaciones. A diferencia de los frameworks, no se centran en controlar el flujo interno de datos, utilizar herencia o patrones de diseño. Algunos ejemplos de estas librerías son Jquery y Mootools para JavaScript, así como Twitter Bootstrap para CSS (Caballero, 2016).

1.6.8 JavaScript

JavaScript representa un lenguaje de secuencias del lado del usuario empleado en navegadores de internet. Su enfoque primordial es asistir a los programadores en la interacción con la página web y el navegador en sí. Aunque está inspirado en ciertos aspectos por el lenguaje de programación Java, no puede ser etiquetado como una variante ”liviana” de Java, pese a compartir enfoques de programación y cierta sintaxis (Dimes, 2015).

1.6.9 Cliente web

El cliente web es un software con el que el usuario interactúa para pedir al servidor web que le envíe los recursos que necesita a través de HTTP. En las aplicaciones web, la parte del cliente generalmente se compone del código HTML que constituye la página web, además de alguna programación ejecutable en el navegador, que puede ser JavaScript o VBScript, o bien pequeñas aplicaciones (applets) creadas en Java (Mora, 2001).

1.6.10 Servidor

El servidor web es una aplicación que está constantemente en espera de recibir solicitudes de conexión a través del protocolo HTTP de parte de los clientes web. En sistemas Unix, a menudo se presenta como un ”demonio”, una entidad que opera en segundo plano de manera autónoma para manejar estas peticiones. En contraste, en sistemas Microsoft Windows, se presenta como un ”servicio” que se ejecuta como un proceso en segundo plano. Esta aplicación es fundamental para atender las solicitudes entrantes de los usuarios y proporcionarles los recursos web que necesitan (Mora, 2002).

1.6.11 OpenStreetMAP

OpenStreetMap (OSM) es un proyecto colaborativo que tiene como objetivo crear mapas libres y editables. Utilizando una plataforma diseñada según los principios de la Web 2.0, los usuarios no solo reciben contenido, sino que también pueden crear y compartir su propia información. OSM funciona como un servicio de mapas mundial en formato Wiki, donde los mapas se crean utilizando datos geográficos capturados con dispositivos GPS

móviles, ortofotografías y otras fuentes de información libre. Esto significa que cualquier usuario puede contribuir a crear, completar y corregir los mapas. La filosofía subyacente en OSM se basa en el aprendizaje colaborativo, donde cada persona aprende más al interactuar con otros miembros de la comunidad que si trabajara individualmente. En un enfoque colaborativo, el resultado del trabajo realizado en un grupo tiene un valor superior a la simple suma de los esfuerzos individuales. En el caso de OSM, este enfoque colaborativo es esencial para lograr el objetivo de crear una cartografía completa y detallada de todo el mundo. Solo a través de un sistema de trabajo colaborativo es posible abordar este desafío de mapear todo el mundo (Córdovez, 2009).

1.6.12 Leaflet

Leaflet.js es una biblioteca de código abierto en JavaScript que simplifica la creación de mapas en la web. Esta tecnología permite a los usuarios acceder y utilizar los códigos fuente, lo que significa que pueden ver cómo funciona y también contribuir al proyecto corrigiendo o mejorando el código. Los archivos JavaScript proporcionan diversas funciones para presentar el mapa en una página web. Leaflet.js se puede implementar tanto en navegadores de escritorio como en dispositivos móviles, lo que permite a los usuarios compartir fácilmente los mapas en cualquier lugar. El objetivo principal de Leaflet.js es facilitar el desarrollo de mapas, centrándose en el rendimiento y la usabilidad. Cuenta con una API muy sólida que hace que sea más fácil para los usuarios adaptarla a diversas situaciones. Además, Leaflet.js ofrece características como superposiciones de capas, ventanas emergentes, zoom, formas y panorámicas. Una de las principales ventajas de Leaflet.js es su capacidad para ampliar su funcionalidad mediante complementos de terceros, lo que permite mejorar aún más las capacidades de esta biblioteca (Abdillah et al., 2021).

1.6.13 Visual Studio Code

Visual Studio Code (VS Code) es un popular editor de código fuente desarrollado por Microsoft [1].

VS Code proporciona diversas extensiones y complementos específicos para GIS

[1] Visual Studio Code: https://code.visualstudio.com/docs

que facilitan el desarrollo de aplicaciones web con componentes geográficos. Estas extensiones incluyen soporte para lenguajes de programación populares en el ámbito GIS, como JavaScript, TypeScript, HTML y CSS. Además, existen extensiones para trabajar con bibliotecas y frameworks geoespaciales como Leaflet, que permiten integrar mapas y datos geográficos fácilmente en la aplicación web.

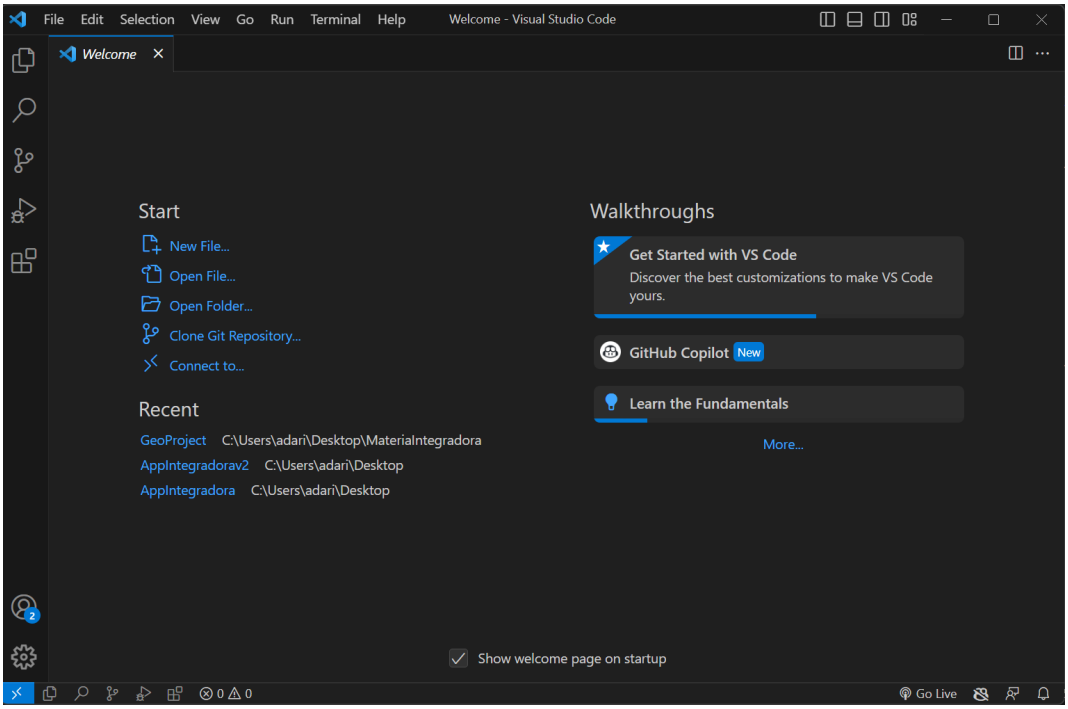


Figura 1.5: Entorno de Visual Studio Code

CAPÍTULO 2

2. METODOLOGÍA

En este capítulo se abordan temas relacionados con las diversas herramientas utilizadas. Este trabajo se lleva a cabo mediante etapas, en donde primero se fija un área en específico, luego se procede al procesamiento de datos existentes en dicho espacio territorial y finalmente se muestra los resultados en la interfaz de usuario. Para este trabajo se utiliza el método de investigación cuantitativa con datos secundarios, pues se abordan análisis con la utilización de datos ya existentes [1].

2.1 Arquitectura tecnológica

En la figura 2.1, se puede observar cómo se encuentran distribuidas las diferentes herramientas tecnológicas y la función que desarrollan en cada procedimiento.

Por parte del cliente web, se encuentra la aplicación web como tal, escrita en lenguaje de programación JavaScript. Asimismo, entra en juego Leaflet, que es el que permite la visualización del mapa, así como la integración de algunas librerías necesarias para el diseño de la aplicación web y facilitando el uso de eventos que permitirán la visualización de los datos de interés.

Por la parte del servidor, la herramienta que se encargará de realizar las peticiones hechas por el usuario será la API de Overpass, la cual está asociada directamente con la base de datos de OpenstreetMAP. De esta manera, el usuario podrá contar con datos de su interés en diferentes áreas de estudio.

[1] Tipos de investigación: https://www.uv.mx/apps/bdh/investigacion/unidad1/investigacion-tipos.html

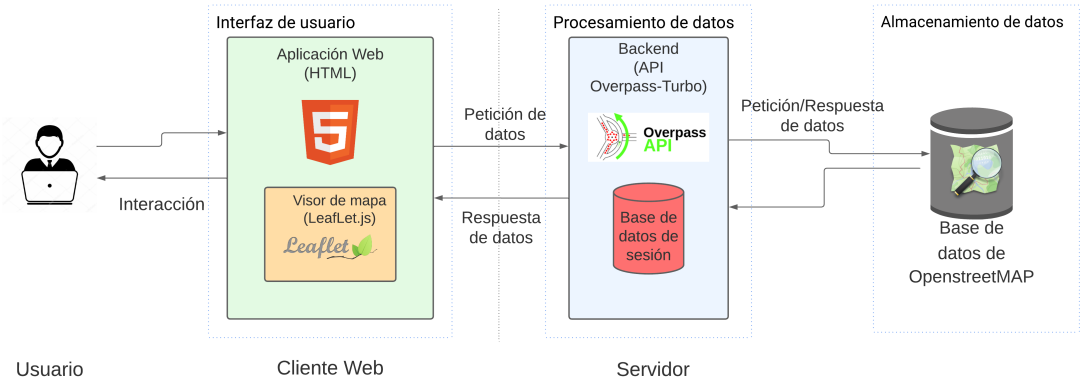


Figura 2.1: Arquitectura Tecnológica

2.2 Interfaz de usuario

La elaboración del diseño de la intefaz de usuario, así como la distribución de los respectivos contenedores ”div” de la aplicación web está desarrollada en lenguaje HTML. Para que el usuario esté en la capacidad de observar los mapas se incorporó la librería de JavaScript llamada Leaflet. Esta elección se basa en las múltiples y valiosas herramientas que Leaflet ofrece, ya que, simplifican el trabajo al momento de desarrollar las diferentes funciones que ayudan a cumplir con los objetivos específicos de este proyecto.

La elección de utilizar HTML como el lenguaje principal para la composición de la interfaz de usuario permite una estructura clara y coherente. Esto es esencial para garantizar la accesibilidad y la usabilidad en diversas plataformas y dispositivos. La combinación de HTML con JavaScript (Leaflet) proporciona un marco sólido para la implementación de funciones interactivas y dinámicas en la aplicación web, debido a que, facilita la creación de una experiencia visual envolvente para los usuarios.

2.3 Procesamiento de datos

El tema de procesamiento de datos en esta aplicación se basa en la interacción del usuario con la aplicación. Al momento en que se dibuja un polígono o también denominado área de estudio para el contexto de este proyecto, se desencadenan

múltipes funciones capaces de realizar peticiones POST que permitirán devolver información geoespacial desde la base de OpenStreetMAP.

Luego con la información ya obtenida se procesan los datos manipulándolos, transformándolos y dándoles diferentes aspectos que los puedan distinguir uno del otro para luego colocarlos de manera visual en el mapa.

2.4 Almacenamiento de datos

Para el almacenamiento de datos como tal, fueron implementadas dos bases de datos, una de ellas es la base de datos propia de OSM, de donde se extraerá la información de las cuatro categorías con sus respectivas etiquetas, y finalmente una base de datos propia de la aplicación en donde se albergaron los datos computados de las categorías.

2.4.1 Base de datos de sesión

La información de cada área de estudio se irá almacenando en una base de datos interna de la aplicación a la cual se la ha denominado como ”base de datos de sesión”, con el fin de luego poder acceder a la misma y extraer la información recolectada para presentarlas en tablas, facilitando la visualización y análisis de los datos recopilados. Debido a que esta base de datos sirve para almacenar la información de la sesión actual, una vez que se refresque la página de la aplicación los datos almacenados serán borrados.

2.4.2 Base de datos de OpenStreetMAP

Para acceder a la información geoespacial en este contexto, se optó por utilizar OpenStreetMap (OSM) como fuente de datos. Esta elección se basa en su gratuidad y en la ausencia de restricciones en cuanto al número de solicitudes que se pueden realizar. La API seleccionada para acceder a estos datos es Overpass Turbo.

El enfoque de la base de datos de OSM se basa en el uso de palabras clave que se

asignan a una amplia gama de categorías geográficas [2]. Estas palabras clave permiten clasificar y etiquetar de manera efectiva elementos geoespaciales. Esta estructura resulta fundamental para el desarrollo del proyecto en cuestión, ya que facilita la obtención y organización de información relevante.

Figura 2.2: Estructura de una petición a Overpass Turbo.

(a) Fuente: https://overpass-turbo.eu/

2.5 Métricas de evaluación de la aplicación

Para medir el rendimiento de la aplicación al momento de hacer las peticiones POST a la API de Overpass Turbo y luego obtener la información de OSM, se decidió utilizar la aplicación ”Postman”, en donde se analizaron las siguientes métricas:

- **Total de peticiones enviadas:** se refiere al número de peticiones que son enviadas a la API durante un determinado tiempo.
- **Peticiones por segundo (peticiones/s):** se refiere al número de peticiones POST enviadas en un segundo.
- **Tiempo promedio de respuesta (ms):** se refiere al promedio de tiempo de todas las solicitudes a la API
- **Porcentaje de error (%):** se refiere al porcentaje de datos perdidos durante las peticiones

[2]OpenStreetMAP: https://wiki.openstreetmap.org/wiki/Map_features

2.6 Métricas para el caso de estudio

Una vez concluida la elaboración de la aplicación web GIS, es necesario evaluar la información entregada por la base de datos de OpenStreetMAP, para lo cual se han considerado las siguientes métricas:

- **Distribución de calles por tipo de superficie (%):** se refiere a realizar un conteo general por tipo de superficie de todas las calles que contenga el área de estudio seleccionada por el usuario, permitiendo así evaluar que tipo de riesgo conlleva cada tipo de superficie con respecto a los accidentes de tránsito.
- **Densidad de calles por kilómetros cuadrados (calles/km2):** se refiere a la densidad total de calles que tiene el área de estudio. Se utilizó la ecuación 2.1.

$$Densidad\ de\ calles = (Total\ de\ calles)/(Area\ total)\ [calles/km^2] \quad (2.1)$$

- **Capacidad Energética de la zona (v):** se refiere a brindar información de la parte eléctrica de acuerdo a los datos entregados por OSM, que estén comprendidos dentro del área seleccionada.
- **Espacios verdes y parques (%):** Porcentaje de área total ocupada por espacios verdes dentro del área de estudio. Para este punto se implementa la ecuación 2.2

$$Porcentaje\ area\ total\ de\ espacios\ verdes = (Area\ de\ parques/Area\ total) * 100\% \quad (2.2)$$

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

En este capítulo se describen las configuraciones respectivas de las herramientas tecnológicas que se observan en la figura 2.1. La integración de Leaflet con OSM mediante el uso de la API de Overpass Turbo, en donde, se construyó la estructura de las peticiones para las cuatro diferentes categorías empleadas en la aplicación. También se muestran los códigos de las funciones más importantes que permiten garantizar al usuario final una interacción clara y sencilla con la aplicación web. La elaboración del frontend como la del backend fueron realizadas con el editor de código de fuente Visual Studio Code.

3.1 Integración de Leaflet con OpenStreetMAP

Las principales ventajas que brinda Leaflet es su perfecta integración con OpenStreetMap (OSM). Leaflet pone a disposición del desarrollador herramientas y funciones necesarias para visualizar, interactuar y personalizar de manera fácil y eficiente los mapas de OSM [1].

La manera en que se integra Leaflet y OSM se basa en la capacidad de Leaflet de consumir los datos proporcionados por OSM y representarlos en forma de mapas interactivos en una página web. Leaflet accede a la base de datos de OSM mediante el uso de la API de Overpass, donde luego son mostrados en forma de mapas interactivos.

Para utilizar Leaflet y OSM, es necesario agregar la biblioteca Leaflet al código HTML de la página web y configurar un contenedor HTML donde se mostrará el mapa. Luego, se crea un objeto de mapa utilizando la función L.map() de Leaflet y se especifican las opciones de visualización, como la ubicación inicial y el nivel de zoom, tal como se

[1] Leaflet: https://leafletjs.com/

muestra en la figura 2.3.

```
</html>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.9.4/leaflet.css" integrity="sh
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw.js" integrity="sha512-o


<script>
    var map = L.map('map').setView([51.505, -0.09], 13);

var osm1 = L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);
```

Figura 3.1: Función básica para inicialización del mapa

Leaflet también ofrece una variedad de capas y controles adicionales que se pueden agregar al mapa para mejorar su funcionalidad [2]. Por ejemplo, para la elaboración de este proyecto se agregaron controles de zoom, controles de capas y herramientas de dibujo de polígonos para interactuar con el mapa y realizar acciones específicas, así como las opciones de editar y borrar los polígonos (Figura 3.2).

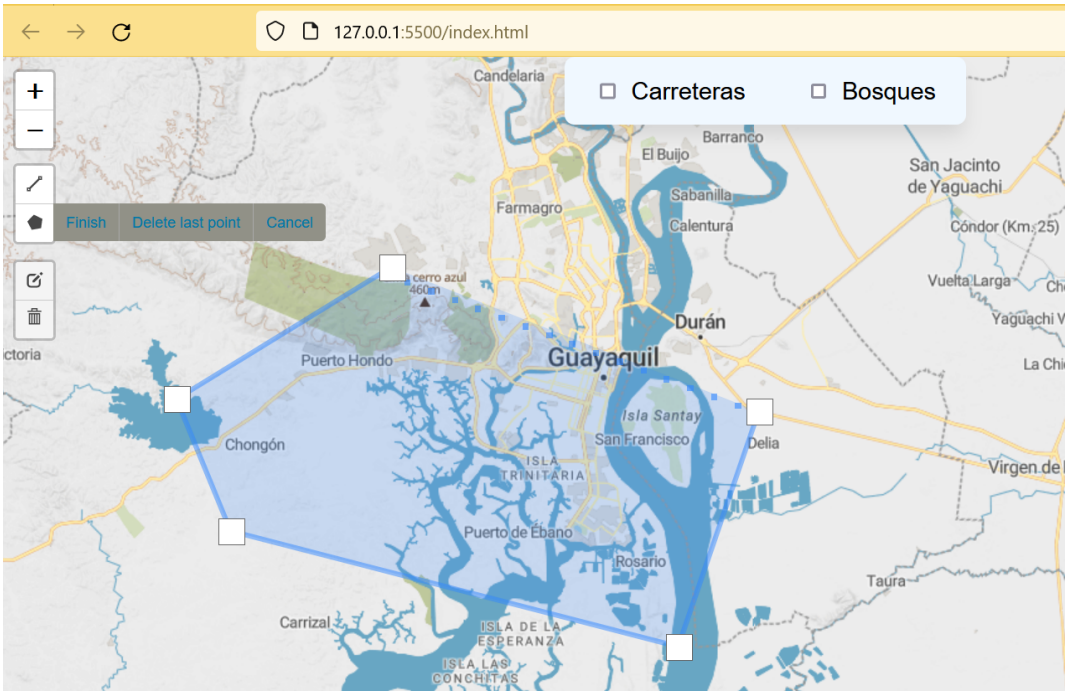


Figura 3.2: Controles de dibujo y zoom

Otra funcionalidad que ofrece esta librería es la capacidad de visualizar los datos que se obtuvieron mediante OSM. Por ejemplo, al dar un clic en un polígono se podrá obtener información adicional, como el nombre de una ubicación o detalles específicos

[2]Leaflet: https://leafletjs.com/reference.html

de un elemento geoespacial (Figura 3.3).

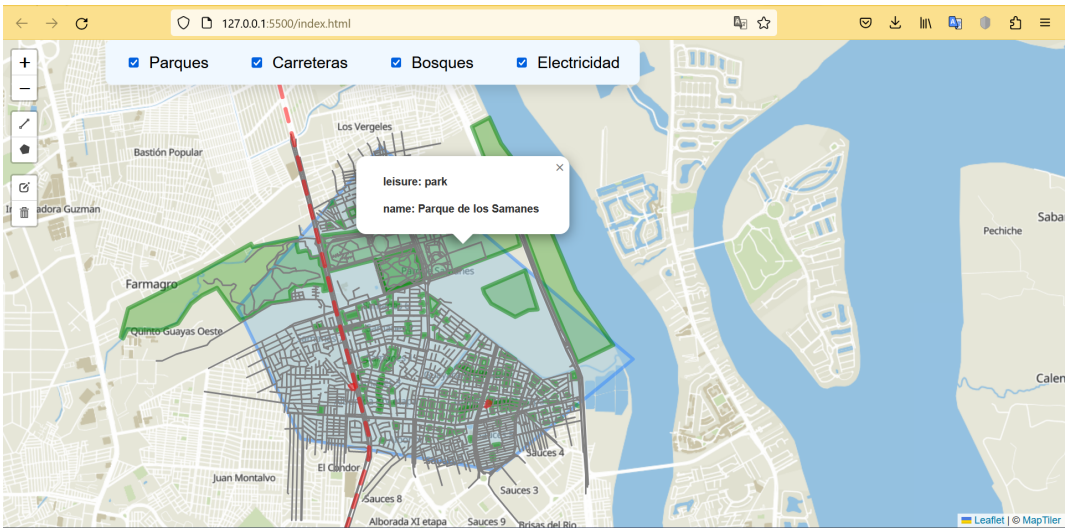


Figura 3.3: Información adicional de puntos de interés

En resumen, Leaflet y OSM trabajan juntos para brindar una solución eficiente y personalizable para visualizar e interactuar con mapas en aplicaciones web. La integración entre estas dos herramientas permite a los desarrolladores crear fácilmente mapas interactivos y acceder a los datos geoespaciales de OSM para una amplia variedad de proyectos y aplicaciones.

3.2 Overpass API

La API de Overpass es una poderosa herramienta que permite acceder y obtener información geoespacial detallada de OpenStreetMap (OSM).

El funcionamiento de la API de Overpass es relativamente sencillo. Basta con enviar solicitudes mediante peticiones POST [3], que contienen una consulta en el estilo de Overpass QL (Query Language). Esta estructura especifíca los criterios de búsqueda, como ubicaciones geográficas, tipos de elementos (nodos, caminos, áreas), etiquetas específicas, y otros parámetros para filtrar los datos deseados.

[3]Overpass API: https://wiki.openstreetmap.org/wiki/Overpass_API

La API procesa la consulta y devuelve los resultados en formato XML, que posteriormente deben ser parseados. Los datos obtenidos incluyen información detallada sobre los objetos geoespaciales, como coordenadas geográficas, etiquetas y otros atributos relevantes.

3.3 Elaboración del FrontEnd

Para el diseño de la vista principal de la página web se implementaron dos contenedores div, del lado izquierdo se procedió a mostrar el mapa y del lado derecho se encuentran los checkboxs que se encargarán de activar o desactivar las capas de interés ya sean estas carreteras, bosques, parques o electricidad. La información de cada área de estudio también se muestran en la parte derecha en forma de tablas dinámicas.

3.3.1 Visor de mapas

El encargado de crear el contenedor donde se podrá mostrar el mapa en la página web es Leaflet, para lo cual es necesario agregar su CDN en nuestro archivo donde se escribió todo el código de la aplicación. La versión de la librería en cuestión que se utilizó fue la 1.9.4.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.9.4/leaflet.js"
  integrity="sha512-BwHfrr4c9kmRkLw6iXFdzcdWV/PGkVgiIyIWLLlTSXzWQzxuSg4DiQUCpauz/EWjgk5TYQqX/kvn9pG1NpYfqg
  crossorigin="anonymous" referrerpolicy="no-referrer"></script>
```

Figura 3.4: CDN de la librería de Leaflet

Para que el usuario pueda cambiar el estilo de la capa base del mapa es necesario buscar por internet páginas que brinden este servicio, para este caso se ha utilizado MapTiler que es una herramienta para crear y usar mapas personalizados [4] que ofrece una amplia paleta de estilos de capa para la creación de mapas interactivos. Es necesario crear una cuenta para generar una API KEY que permitirá usar los beneficios de MapTiler, cabe recalcar que el uso de esta plataforma es gratuito.

[4] MapTiler: https://maptiler.es/cloud/

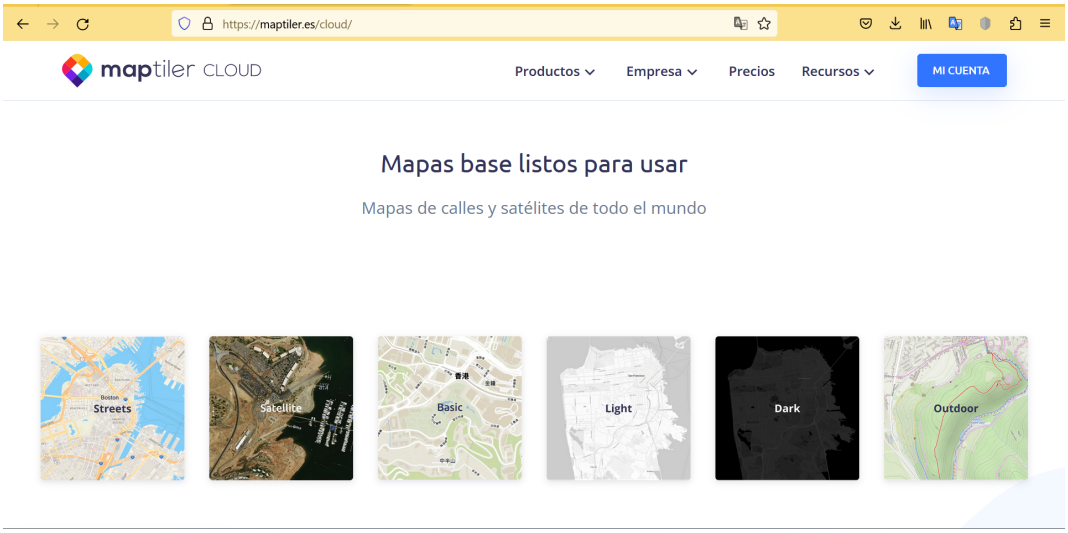


Figura 3.5: Página principal de MapTiler.

(a) Fuente: https://maptiler.es/cloud/

El código 3.1 define una constante llamada ‘layerMapBasic‘ que almacena la URL de un servicio de mapas llamado ”basic-v2” proporcionado por MapTiler. Se utiliza una plantilla de cadena para insertar una clave de API en la URL. Luego, se crea una capa de mosaicos utilizando ‘L.tileLayer‘, que utiliza la URL definida anteriormente como fuente de los mosaicos. Se establecen algunas opciones para la capa, como la atribución de los datos del mapa a MapTiler, el nivel máximo de zoom, el tamaño de los mosaicos y otros ajustes. Finalmente, la capa se agrega al mapa, que debe ser un objeto previamente definido en el código.

Código 3.1: Mosaico de capas

```
const API_KEY = '********************'
var map = L.map('map').setView([-2.1830430546019954, -79.8952696091102], 13);
const layerMapBasic = 'https://api.maptiler.com/maps/basic-v2/{z}/{x}/{y}.png?key=${API_KEY}'
    L.tileLayer(layerMapBasic, {
        attribution : '&copy; <a href="https://www.maptiler.com/">MapTiler</a>',
        maxZoom: 18,
        tileSize : 512,
        zoomOffset: -1,
        crossOrigin: true
    }).addTo(map);
```

3.3.2 Control de estilos capas

Es posible agregar más de un estilo de capa para el mapa base, con el objetivo de que el usuario pueda alternarlas entre si. Se agregaron dos capas más teniendo así un total de tres.

El código 3.2 presenta la forma correcta de crear un layer control y al mismo tiempo como crear los objetos que contienen los estilos de mapa.

Código 3.2: Layer Control

```
const layerOpenStreetMap = ‘https://api.maptiler.com/maps/openstreetmap/{z}/{x}/{y}@2x.jpg?
key=${API_KEY}‘
const osmOPM = L.tileLayer(layerOpenStreetMap, {
    attribution : ’&copy; <a href=”https :// www.maptiler.com/”>MapTiler</a>’,
    maxZoom: 18,
    tileSize : 512,
    zoomOffset: −1,
    crossOrigin: true
})
const layerSatellite = ‘https: // api.maptiler.com/maps/satellite/{z}/{x}/{y}@2x.jpg?key=${API_KEY}‘
const osmSatellite = L.tileLayer( layerSatellite , {
    attribution : ’&copy; <a href=”https :// www.maptiler.com/”>MapTiler</a>’,
    maxZoom: 18,
    tileSize : 512,
    zoomOffset: −1,
    crossOrigin: true
})
L.control.layers({”Basic”: osmBasic, ”OpenStreetMap”: osmOPM, ”Satelite”: osmSatellite}).addTo(map)
```

3.3.3 Control de dibujo

Uno de los objetivos específicos de este proyecto indica que el usuario final se sentirá en libertad de seleccionar un área de estudio en cualquier parte del mapa, por lo que, para poder cumplir con este objetivo se procedió a utilizar una herramienta, la cual permite dibujar polígonos dando múltiples clicks sobre el mapa. Para contar con esta

herramienta es necesario agregar la CDN respectiva en su versión 1.0.4, como se muestra a continuación:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw-src.css'
  integrity="sha512-vJfMKRRm4c4UupyPwGUZI8U651mSzbmmPgR3sdE3LcwBPsdGeARvUM5EcSTg34DK8YIRiIo+oJwNfZPMKEQyug==
  crossorigin="anonymous" referrerpolicy="no-referrer" />
```

Figura 3.6: CDN del plugging Draw Control

El código que nos permite activar las opciones de dibujo es el siguiente:

Código 3.3: Draw Control

```
var drawControl = new L.Control.Draw({
    draw: {
      polygon: true,
      rectangle: false,
      circle : false,
      marker: false,
      circlemarker: false
    },
    edit: {
      featureGroup: drawnItems
    }
  });
  map.addControl(drawControl);
```

Es importante recalcar que solo dejaremos en ”true” la opción de ”polygon”, ya que, es la única que cumple con las condiciones antes mencionadas.

Figura 3.7: Diseño del panel del draw control.

3.3.4 Generación de marcadores, polígonos y polilíneas

Para representar de manera visual la ubicación respectiva de las capas de carreteras, parques, bosques y lo relacionado a la energía eléctrica es importante tener en cuenta como son asignadas dicha información en la base de datos de OSM. Cuando el usuario termina de dibujar el polígono o área de estudio, sobre la capa base del mapa se dibujará en las respectivas coordenadas de latitud y longitud lo siguiente:

- Un marcador circular si el tipo de dato es ”node”
- Un polígono si el tipo de dato es ”way”

El caso de las polilíneas son especialmente asociadas a las carreteras o calles, para lo cual estas se construyen a través de cada posición de la geometría.

A continuación en el código se muestra la función que hace posible la generación de marcadores circulares y polígonos para la capa de los parques.

Código 3.4: Función para agregar marcadores para parques

```
export function addMarkersPark(parks, leaflet_id_marked) {
    if (parks.length === 0) return;
    for (const element of parks) {
        // Extraer la informacion de las ubicaciones
        const tags = element.tags
        createHeader("park", leaflet_id_marked, tags)
        const information = createTextInformation(tags)

        if (element['type'] == 'node') {
            const latlonPost = [element['lat'], element['lon']]
            const marker = L.circleMarker(latlonPost, {
                color: 'green', weight: 5,
                opacity: 0.5,
            }).addTo(memoryMarked[leaflet_id_marked]["layer"]).bindPopup(`${information || 'Desconocido'}`);
            memoryMarked[leaflet_id_marked]["parks"].push(marker)
        } else if (element['type'] == 'way') {
            const geo = element.geometry;
```

```
                if (geo === undefined) continue
                const arrLat = []
                const arrLonLat = []
                for (const pos of geo) {
                    arrLat.push([pos['lat'], pos['lon']])
                    arrLonLat.push([pos['lon'], pos['lat']])
                }
                const tags = element.tags
                createHeader("park", leaflet_id_marked, tags)
                const information = createTextInformation(tags)
                // console.log("Arreglo de wood",arrLonLat)
                const current = turf.polygon([arrLonLat])
                // console.debug("Multipolygon: ", current)
                const area = calculateAreaPolygonInKilometerSqueere(current.geometry)
                tags["area"] = area
                const poly = L.polygon(arrLat, {
                    color: 'green',
                    weight: 5,
                    opacity: 0.5,
                }).addTo(memoryMarked[leaflet_id_marked]["layer"]).bindPopup(`${information || 'Desconocido'}`)
                memoryMarked[leaflet_id_marked]["parks"].push(poly);
            }
            databaseSession["park"]["row"][leaflet_id_marked].push(tags)
        }
    }
```

3.3.5 Generación de tablas de información

Para que el usuario pueda observar la información general de las áreas de estudio o polígonos dibujados se diseñó el código 3.5. Tiene como función principal mostrar en una ventana emergente los datos recolectados en forma de tablas. La idea de esta función es trabajar con los ID únicos que se le asignan a cada polígono al momento de su creación y luego verificar si se encuentra en databaseSession, que es una base de datos interna

de la aplicación, si el polígono existe se extrae los valores computados y se los agrega a un arreglo llamado htmlInfoPolygon, en base a este arreglo se generan las tablas con la ayuda de la biblioteca SweetAlert.

Código 3.5: Función para mostrar tablas de información

```
function showDataPolygon(element) {
    console.log("show data polygon: ", element)
    const cardPolygon = element.target;
    console.log("show polygon: ", cardPolygon)
    let polygonID = cardPolygon.id
    console.log("Computed after: ", polygonID)
    if (polygonID === undefined || polygonID.length === 0) {
        const parent = cardPolygon.parentElement;
        polygonID = Number(parent.id)
    }


    const htmlInfoPolygon = []
    for (const key of Object.keys(databaseSession)) {
        console.log("Computed key: ", key, " Polygon:", polygonID)
        const computed = databaseSession[key]["computed"][polygonID]
        console.log('Computed values: ', computed)
        if (computed === undefined) continue
        const rows = createRowTable(computed)
        const zone = `
            <div class="zone table-title ">
                <h4>${key}</h4>
                <table>
                    <thead>
                    <tr>
                        <th>Zona</th>
                        <th>Tipo</th>
                        <th>Cuenta</th>
                    </tr>
                    </thead>
```

```
                    <tbody>
                        ${rows}
                    </tbody>
                </table>
            </div>
        `
        htmlInfoPolygon.push(zone)
    }

    Swal.fire ({
        title : `Datos del Poligono ${polygonID}`,
        width: 700,
        padding: '3rem',
        html: htmlInfoPolygon.join('\n'),
        showConfirmButton: false,
    })
}
```

3.4 Elaboración del BackEnd

A continuación se explicarán los detalles de como se desarrolló el backend para que la aplicación pueda extraer la información desde la base de datos de OSM y posteriormente como se los manejaron para finalmente mostrarlos de manera visual en el mapa interactivo.

3.4.1 Controles de eventos

La funcionalidad de esta aplicación web va a depender de tres importantes eventos, los cuales fueron asignados a las opciones que ofrece el panel del draw control (figura 3.7), los cuales son ”Draw Polygon”, ”Edit Layers” y ”Delete Layers”.

- **Draw Polygon:** El siguiente fragmento de código 3.6 entra en acción cuando se crea una área de estudio o polígono. Entre las acciones más importantes es generar un ID único para el polígono el cual lo asigna a la capa correspondiente. Además,

crea un objeto en un diccionario para almacenar información relacionada con el polígono, como un grupo de capas para marcadores y listas para diferentes tipos de datos geográficos. El polígono se agrega a un conjunto de capas y se realiza la extracción de su área en kilómetros cuadrados. El formato de la geometría del polígono se ajusta para realizar consultas específicas en la base de datos OverPass. Luego, se realizan búsquedas de datos, como parques, carreteras, bosques y elementos de energía, en la zona del polígono. Si no se encuentran datos en ninguna categoría, se muestra un mensaje de error, pero si se encuentran datos, se agregan marcadores y líneas correspondientes a cada categoría. Finalmente, se realiza el renderizado y se completa el proceso de carga de datos en el visor de mapa.

Código 3.6: Draw Polygon

```
map.on(L.Draw.Event.CREATED, async function (event) {
    const layer = event.layer;
    // Le cambiamos el estilo al polygono
    layer.setStyle({
        fillOpacity : 0.5,
        fillColor : "#9FCAE0"
        });
    const polygonID = L.Util.stamp(layer); // Genera un ID único para el polígono
    const leaflet_id_marked = `${polygonID}`
    layer._leaflet_id = polygonID; // asignamos el id

    console.log('polygon id:', leaflet_id_marked)
    memoryMarked[leaflet_id_marked] = {
        "layer": L.layerGroup().addTo(map),
        "parks": [],
        "highway": [],
        "wood": [],
        "power": [],
    } // creamos un layerGroup para guardar los marcadores del poligono
    drawnItems.addLayer(layer);
    console.log(layer.toGeoJSON())
```

```
        const area = calculateAreaPolygonInKilometerSqueere(layer.toGeoJSON().geometry)
        addPolygonRow(leaflet_id_marked, area)
        const poly = formatGeoJSONtoOverPassPolygon(layer.toGeoJSON())
        loadDataMap()
        const parks = await findPark(overpassQueryPark(poly));
        const highway = await findHigway(overpassQueryHighway(poly))
        const woods = await findWood(overpassQueryWood(poly))
        const power = await findPower(overpassQueryPower(poly))
        const allZoneEmpty = parks.length === 0 && highway.length === 0 & woods.length
        === 0 & power.length === 0
            if (allZoneEmpty) {
                errorModal("Error, no se encontraron datos")
                return
            }
        addMarkersPark(parks, leaflet_id_marked);
        addPolylineHigway(highway, leaflet_id_marked)
        addPolylineWood(woods, leaflet_id_marked)
        addPowerMarker(power, leaflet_id_marked)
        renderComputedValue(leaflet_id_marked)
        finishLoadDataMap()
    });
```

- **Edit Layers:** El código 3.7 establece un manejador de eventos que responde a las ediciones de polígonos. Cuando el usuario finaliza las ediciones en un polígono existente, se ejecutan una serie de acciones. Estas van a permitir la actualización de información, como el cálculo del área del polígono editado, la eliminación de marcadores anteriores, la búsqueda y visualización de nuevos datos geográficos en el mapa, como parques, carreteras, bosques y elementos de energía, y la manipulación de la presentación visual de los resultados. En caso de no encontrarse nuevos datos en ninguna de estas categorías, se muestra un mensaje de error.

Código 3.7: Edit Layers

```
map.on(L.Draw.Event.EDITED, async function (event) {
const layers = event.layers;
layers.eachLayer(async function (layer) {
    /*
        Cuando editamos el poligono, extraemos su id y limpiamos esa capa de marcadaroes,
        realizamos la peticion para buscar nuevamente los parques, hoteles, hospitales
        y agregamos los marcadores.
    */
    console.log(layer)
    const polygonID = layer._leaflet_id ;
    console.log("ID del polígono editado:", polygonID);
    const area = calculateAreaPolygonInKilometerSqueere(layer.toGeoJSON().geometry)
    editPolygonRow(polygonID, area)
    memoryMarked[polygonID]["layer"].clearLayers();
    memoryMarked[polygonID]["polygon"] = layer
    memoryMarked[polygonID]["arrLatLong"] = layer.toGeoJSON().geometry;
    const poly = formatGeoJSONtoOverPassPolygon(layer.toGeoJSON())
    loadDataMap()
    const parks = await fetchOverpassTurbo(overpassQueryPark(poly));
    const highway = await fetchOverpassTurbo(overpassQueryHighway(poly))
    const woods = await fetchOverpassTurbo(overpassQueryWood(poly))
    const power = await fetchOverpassTurbo(overpassQueryPower(poly))
    const allZoneEmpty = parks.length === 0 && highway.length === 0 & woods.length === 0
    & power.length === 0
    if (allZoneEmpty) {
        errorModal("Error, no se encontraron datos")
        return
    }
    addMarkersPark(parks, polygonID);
    addPolylineHigway(highway, polygonID)
    addPolylineWood(woods, polygonID)
    addPowerMarker(power, polygonID)
    calculateAndStoreComputedValuesForPolygon(polygonID)
```

```
            finishLoadDataMap()

        });
});
```

- **Delete Layers:** Cuando se elimina un polígono, el código 3.8 realiza las siguientes acciones: itera a través de cada capa de polígono eliminada, extrayendo su ID único y limpiando la capa de marcadores asociados. Luego, elimina tanto el ID como el objeto de marcadores relacionados con ese polígono de una estructura de datos llamada 'memoryMarked'. Además, elimina registros relacionados con el polígono.

Código 3.8: Delete Layers

```
map.on(L.Draw.Event.DELETED, function (event) {
    const layers = event.layers;
    layers.eachLayer(async function (layer) {
        console.log(layer)
        var polygonID = layer._leaflet_id ;
        console.log("ID del polígono editado:", polygonID);
        memoryMarked[polygonID]["layer"].clearLayers();
        delete memoryMarked[polygonID];
        deletePolygonRor(polygonID);
        deleteDataToDatabaseSession(polygonID);
    });
})
```

3.4.2 Base de datos de sesión

El código 3.9 presenta la estructura que tiene la base de datos interna de la aplicación donde se guardan la información de las etiquetas y el conteo general de las diferentes categorías.

Código 3.9: Base de datos de sesión

```
export const databaseSession = {
"power": {
    "header": new Set(["length", "area"]),
    "row": {},
    "computed": {}
},
"wood": {
    "header": new Set(["length", "area"]),
    "row": {},
    "computed": {}

},
"park": {
    "header": new Set(["length", "area"]),
    "row": {},
    "computed": {}

},
"highway": {
    "header": new Set(["length", "area"]),
    "row": {},
    "computed": {}

},
};
```

Esta base de datos almacenará información de las categorías de parques, bosques, carreteras y electricidad, cada una cuenta con tres propiedades que son:

- **header:** esta propiedad crea un objeto de tipo conjunto, el cual a su vez cuenta con dos atributos que se utilizan como encabezados en la tabla, los cuales son "length" y "area".

- **row:** este objeto se lo inicializa vacío y tiene como principal función guardar la

información de cada fila de las categorías.

- **computed:** este objeto se lo inicializa vacío y tiene como principal función guardar datos computados específicos de cada categoría.

3.4.3 Generación de consultas a Overpass Turbo

El código 3.10 define una función llamada **fetchOverpassTurbo**, la cual tiene como función realizar consultas de datos geoespaciales a la API de Overpass Turbo a través del acceso a la base de datos OpenStreetMAP.
Cuando la función es invocada mediante una consulta, se procede a construir una solicitud POST a la API, donde también se incluye la consulta codificada en el cuerpo de la solicitud. Luego, se espera la respuesta de la API y esta se convierte en objeto tipo JSON. Finalmente, devuelve la respuesta en un arreglo de elementos, que contiene la información geoespacial según el tipo de consulta que se haya realizado.

Código 3.10: Solicitudes POST a Overpass Turbo

```
export async function fetchOverpassTurbo(overpass_query) {
    const response = await fetch(URL_API, {
        method: 'POST', headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `data=${encodeURIComponent(overpass_query)}`
    })
    const data = await response.json()
    return data.elements
}
```

A continuación se presentan las estructuras de las consultas que trabajan en conjunto con el código 3.10 para las diferentes categorías de la aplicación.

Código 3.11: Estructura de la categoría parques

```
export function overpassQueryPark(polygon) {
    return `[out:json];
      (
        node["leisure"="park"](poly:"${polygon}");
```

```
        way["leisure"="park"](poly:"${polygon}");
      );
      out body geom;`
}
```

Código 3.12: Estructura de la categoría carreteras

```
export function overpassQueryHighway(polygon) {
    return `
          [out:json];
      (
        way["highway"](poly:"${polygon}");
        relation ["highway"](poly:"${polygon}");
      );
      out body geom;`
}
```

Código 3.13: Estructura de la categoría bosques

```
export function overpassQueryWood(polygon) {
    return `
    [out:json];
    (
      way["natural"="wood"](poly:"${polygon}");
      relation ["natural"="wood"](poly:"${polygon}");
    );
    out body geom;
    `;
}
```

Código 3.14: Estructura de la categoría electricidad

```
export function overpassQueryPower(polygon) {
    return `
    [out:json];
    (
```

```
    way["power"="line"](poly:"${polygon}");
    node["power"="transformer"](poly:"${polygon}");
    node["power"="tower"](poly:"${polygon}");
    node["power"="terminal"](poly:"${polygon}");
    way["power"="substation"](poly:"${polygon}");
    node["power"="substation"](poly:"${polygon}");
     relation ["power"="substation"](poly:"${polygon}");
    node["power"="portal"](poly:"${polygon}");
    way["power"="portal"](poly:"${polygon}");
    node["power"="generator"](poly:"${polygon}");
    way["power"="generator"](poly:"${polygon}");
    );
    out body geom;
     `;
}
```

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS

En este capítulo se analizarán los resultados de la aplicación web totalmente operando y el tipo de acciones que puede realizar el usuario final, así como el análisis de la información que nos brinda la base de datos de OpenStreetMAP al momento de seleccionar un área de estudio, mediante la implementación de un simple caso de estudio con el uso de las métricas descritas en la sección 2.6. Finalmente se mostrará el rendimiento de la aplicación al momento de realizar una petición a la base de datos de OSM mediante la API de Overpass Turbo. Los datos fueron obtenidos mediante el uso del software ”Postman”.

4.1 Plataforma Desarrollada

La figura 4.1 muestra el aspecto principal de la aplicación web, mostrando en primeras instancias la distribución de las diferentes herramientas con las que el usuario final estará en la capacidad de realizar diversas acciones.

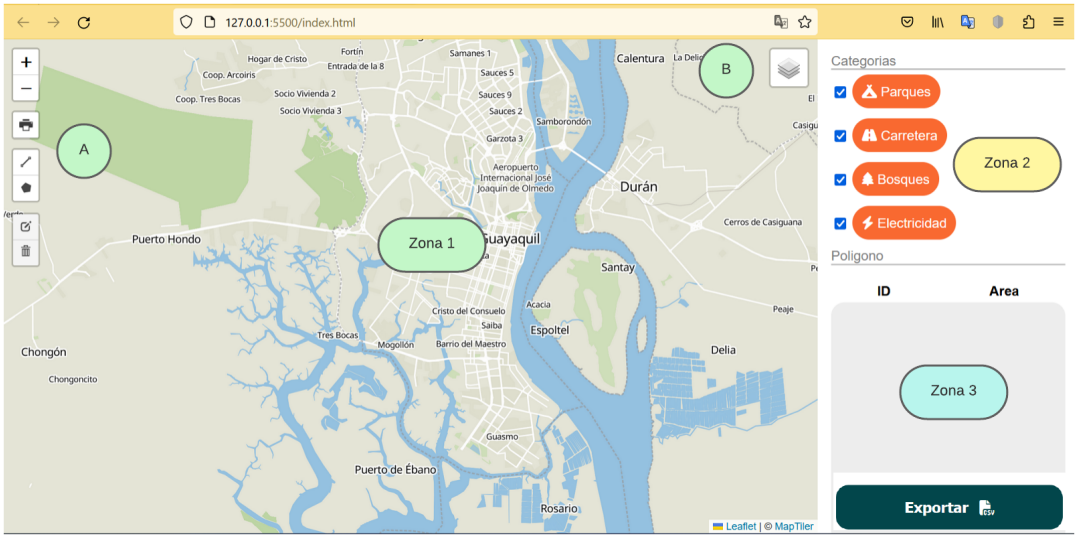


Figura 4.1: Diseño de la página principal de la aplicación GIS

La figura 4.1 está dividida en tres zonas, las cuales serán descritas a continuación:

- **Zona 1:** Vista general de la capa base del mapa.
- **Zona 1.A:** En la parte izquierda de la pantalla encontraremos la barra de herramientas en donde el usuario tendrá la capacidad de realizar acciones tales como zoom, imprimir la vista del mapa, dibujar, editar y eliminar un polígono o área de estudio.
- **Zona 1.B:** En la parte superior derecha se encuentra localizado el controlador de capas.
- **Zona 2:** Se encuentran las categorías de las capas, las cuales son controladas por checkboxs.
- **Zona 3:** En esta zona se observará los ID de los polígonos así como el área total de los mismos. En la parte inferior de esta zona se encuentra el botón que tiene la función de exportar la información en formato CSV.

En la subsección 3.1.4 se detalló la manera de como se generan de manera visual las diferentes categorías de la aplicación, a continuación se explica los estilos que los diferencian uno del otro.

- Polilíneas de color gris, las cuales representan a las calles.

- Polígonos y marcadores circulares de color verde que representan a los parques.
- Polígonos de color amarillo con bordes púrpura que representan a los bosques.
- Líneas entre cortadas de color rojo que representan a la red eléctrica.

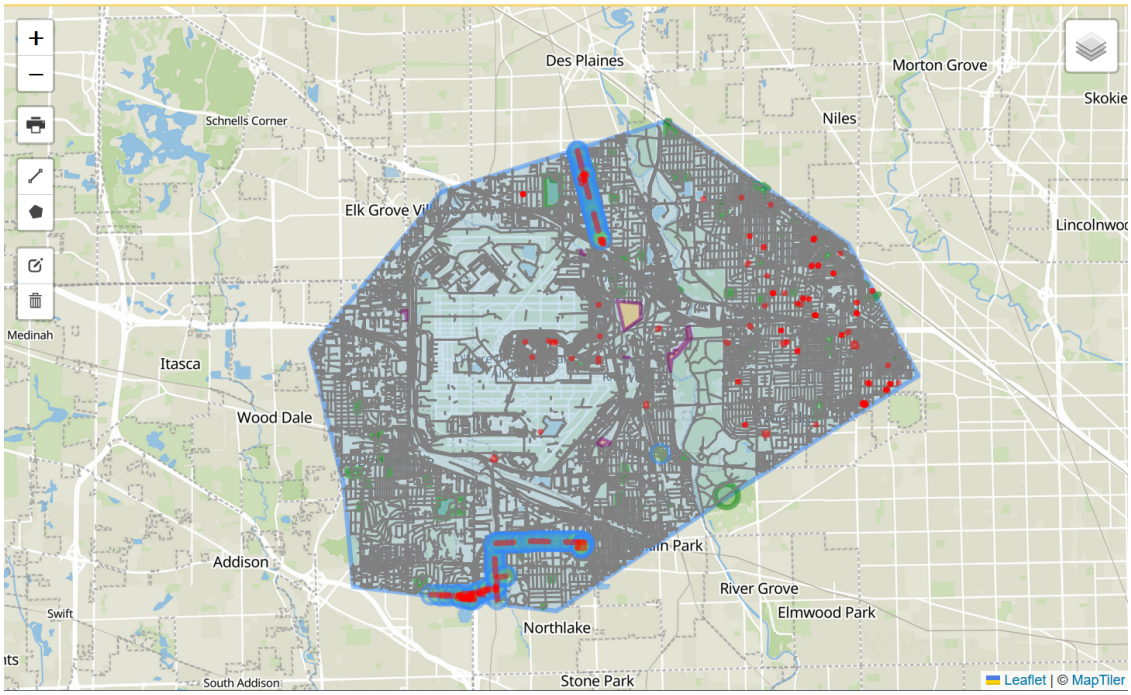


Figura 4.2: Generación visual de marcadores, polígonos y polilíneas.

4.2 Casos de uso

En la figura 4.3 se observan las interacciones que el usuario está en capacidad de realizar con la aplicación web mapping. Estos casos de uso, están divididas en tres grupos los cuales son despliegue de información, visualización y acciones. Cada grupo cuenta con las diferentes actividades destinadas a cumplirse para la satisfacción del usuario.

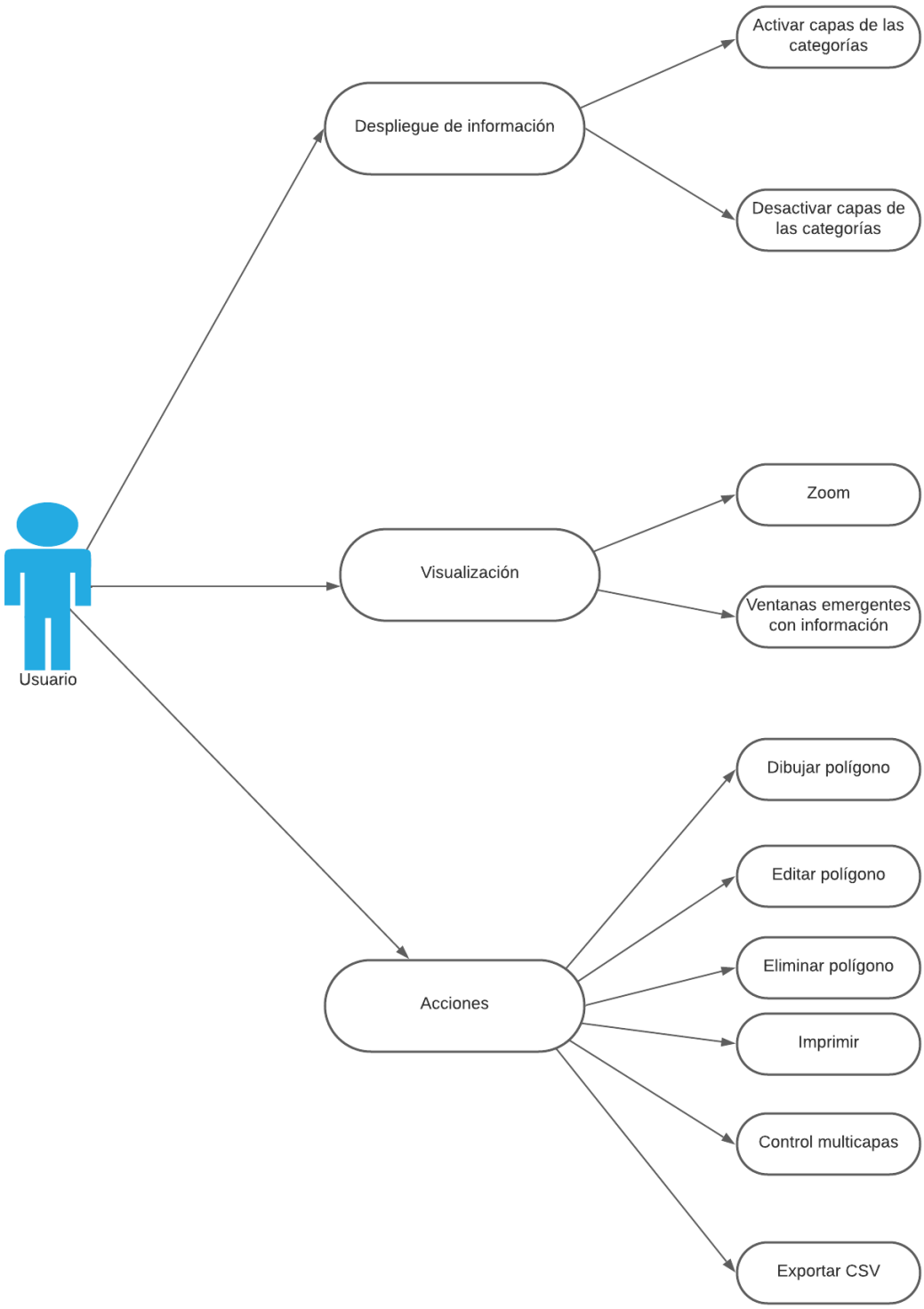


Figura 4.3: Diagrama de casos de uso.

A continuación en las siguientes subsecciones se explica de manera breve las acciones que el usuario puede realizar en la aplicación web.

4.2.1 Caso de uso ”Dibujar Polígono”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de dibujar un área de estudio en cualquier parte del mapa. Esta acción es posible mediante la ayuda de la herramienta ”Draw Control” de la librería de Leafet. Una vez que se termina de dibujar el polígono, la información se cargará y se mostraá dentro del área de interés.

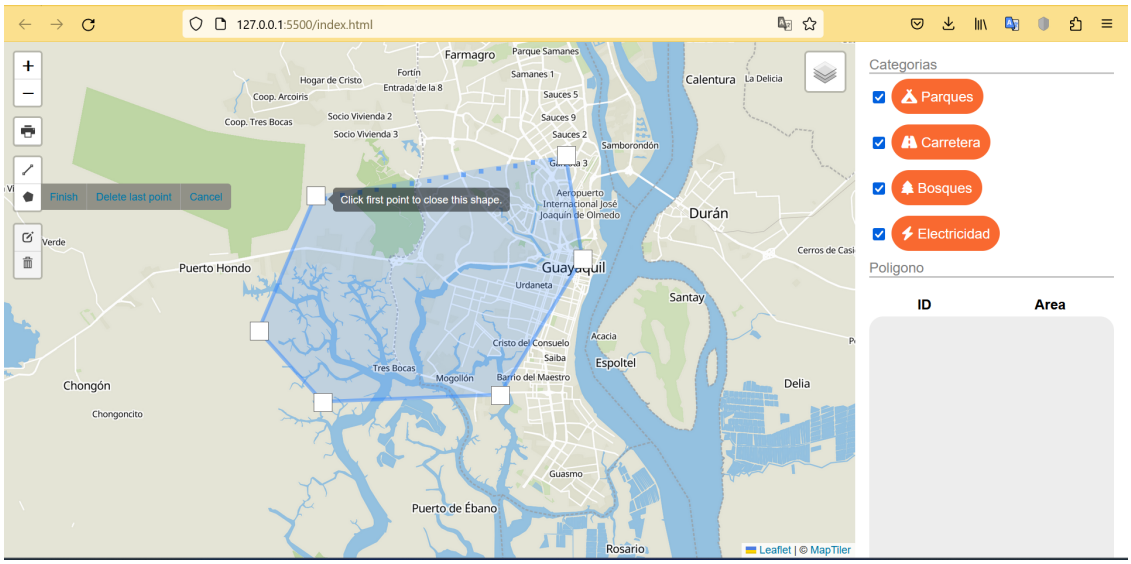


Figura 4.4: Caso de uso dibujar polígono

4.2.2 Caso de uso ”Editar Polígono”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de editar un área de estudio o polígono ya creado anteriormente, estando en la capacidad de poder agrandar o achicar el polígono. La acción descrita es posible mediante la ayuda de la herramienta ”Draw Control” proporcionada por Leaflet. Una vez que se termina de editar el polígono, la información se cargará y se mostraá dentro del área de interés.

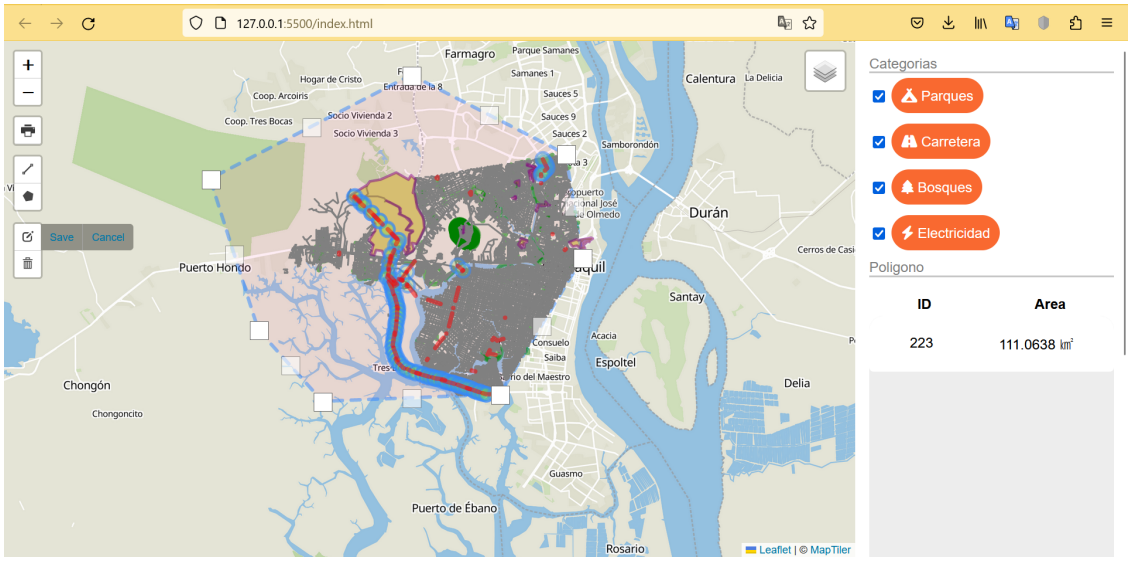


Figura 4.5: Caso de uso editar polígono

4.2.3 Caso de uso ”Eliminar Polígono”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de eliminar un área de estudio o polígono ya creado anteriormente, una vez que es eliminado del mapa, la información ya no se podrá recuperar. La acción descrita es posible mediante la ayuda de la herramienta ”Draw Control” proporcionada por Leaflet.

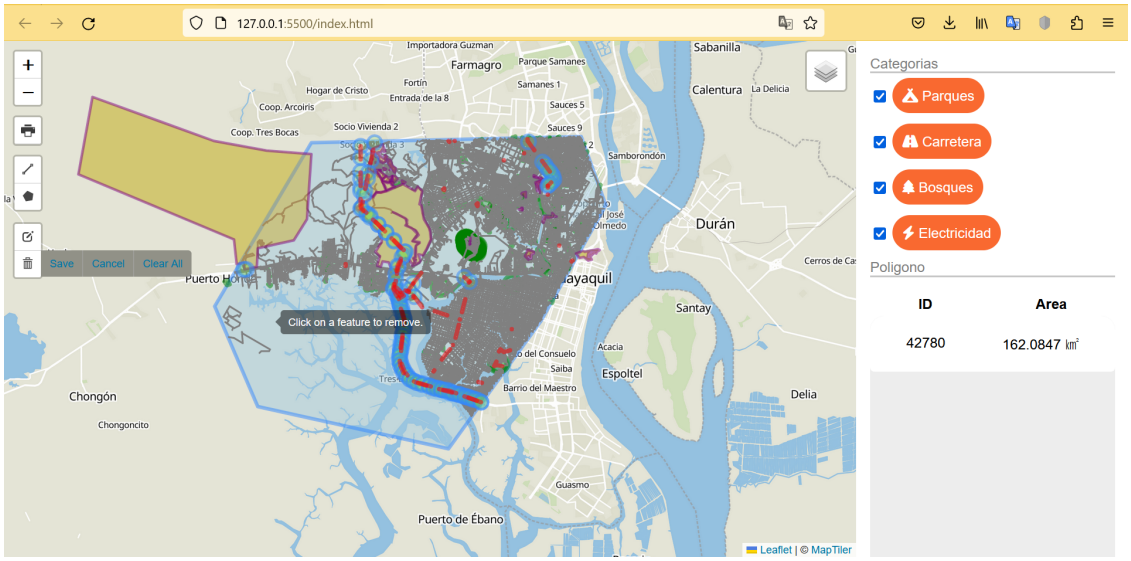


Figura 4.6: Caso de uso eliminar polígono

4.2.4 Caso de uso ”Zoom”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de poder alejar o acercar la vista de la capa base del mapa con o sin los marcadores de las categorías activadas para tener una mejor visión de la distribución geográfica de los componentes que conforman el área de estudio.

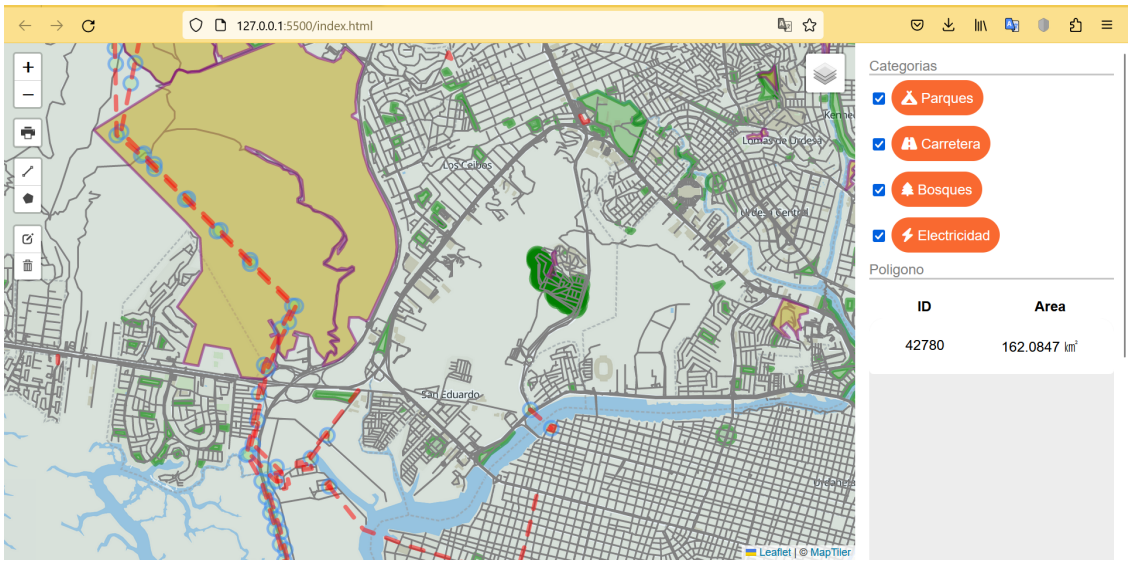


Figura 4.7: Caso de uso zoom

4.2.5 Caso de uso ”Activar/Desactivar marcadores de categorías”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de poder activar o desactivar las capas de información de las carreteras, parques, bosques y red eléctrica que son representadas por marcadores circulares, polígonos y polilíneas según el tipo de información. La acción descrita se da gracias al uso de checkboxs.

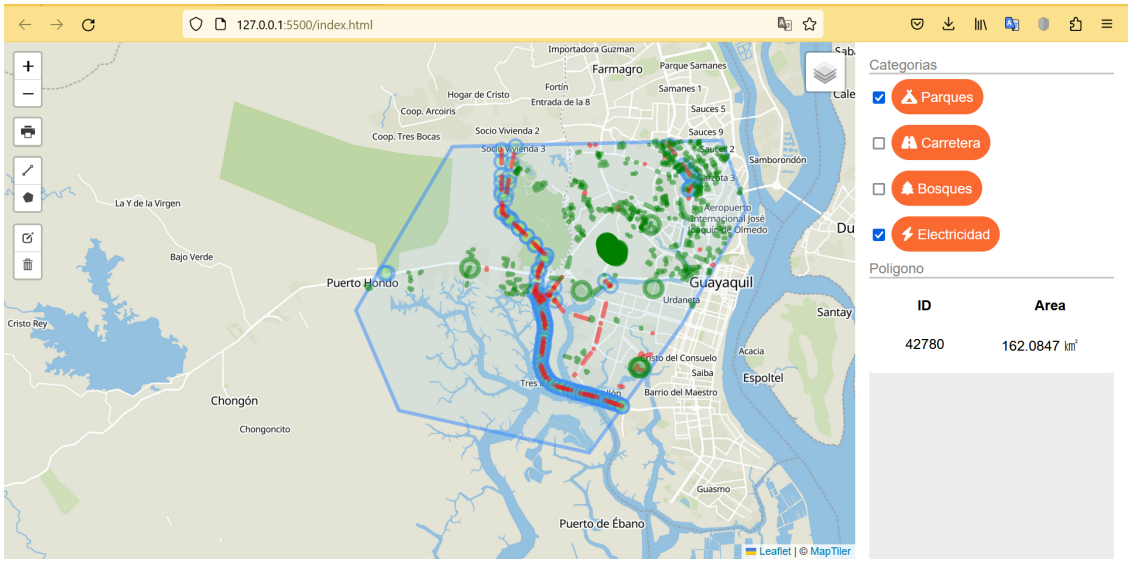


Figura 4.8: Caso de uso activar/desactivar marcadores de categorías

4.2.6 Caso de uso ”Imprimir Pantalla”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de poder imprimir la vista actual del mapa, con la distribución geográfica de las categorías activadas o las que sean de interés por parte del usuario. La acción es posible gracias a la implementación de un botón que es brindada por Leaflet.

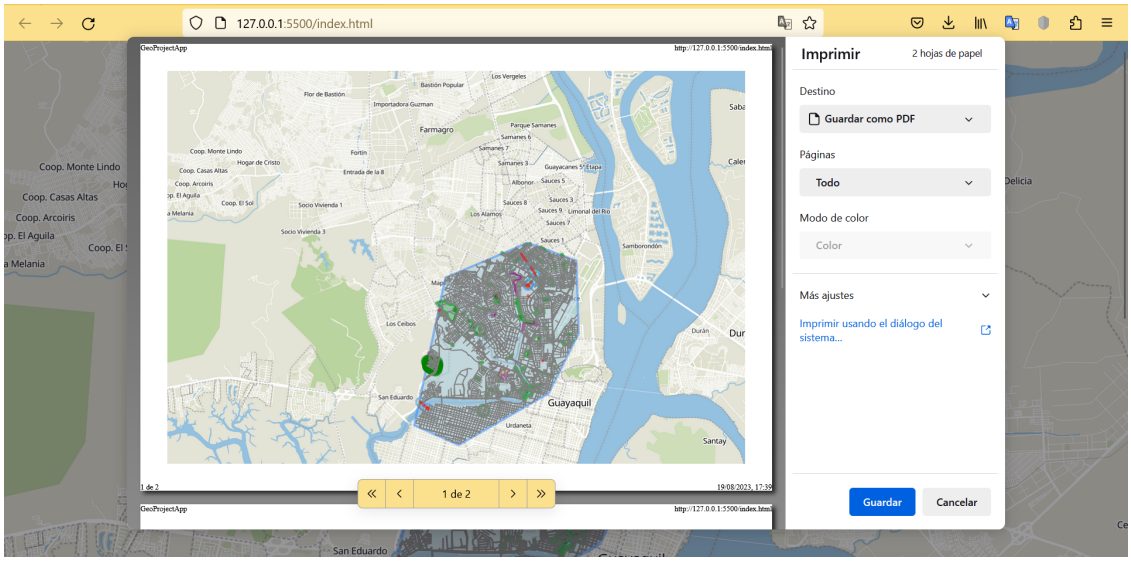


Figura 4.9: Caso de uso imprimir

4.2.7 Caso de uso ”Exportar a CSV”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de generar un archivo CSV, en donde se encuentra toda la información general de todos los polígonos o áreas de estudios dibujados anteriormente. La acción es posible gracias a la implementación de un botón donde se diseñó la función descrita anteriormente.

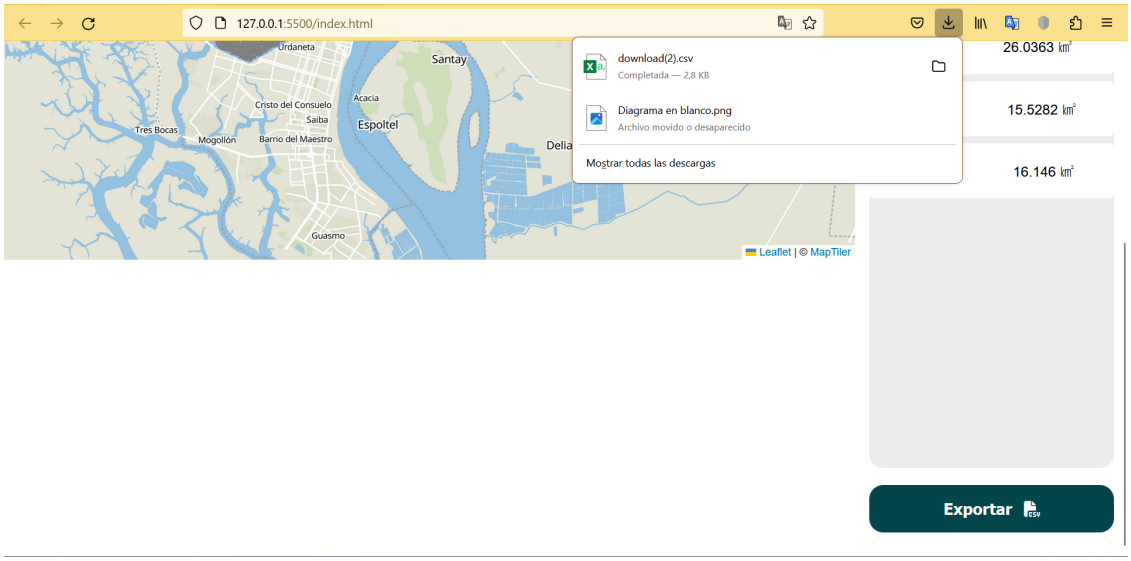
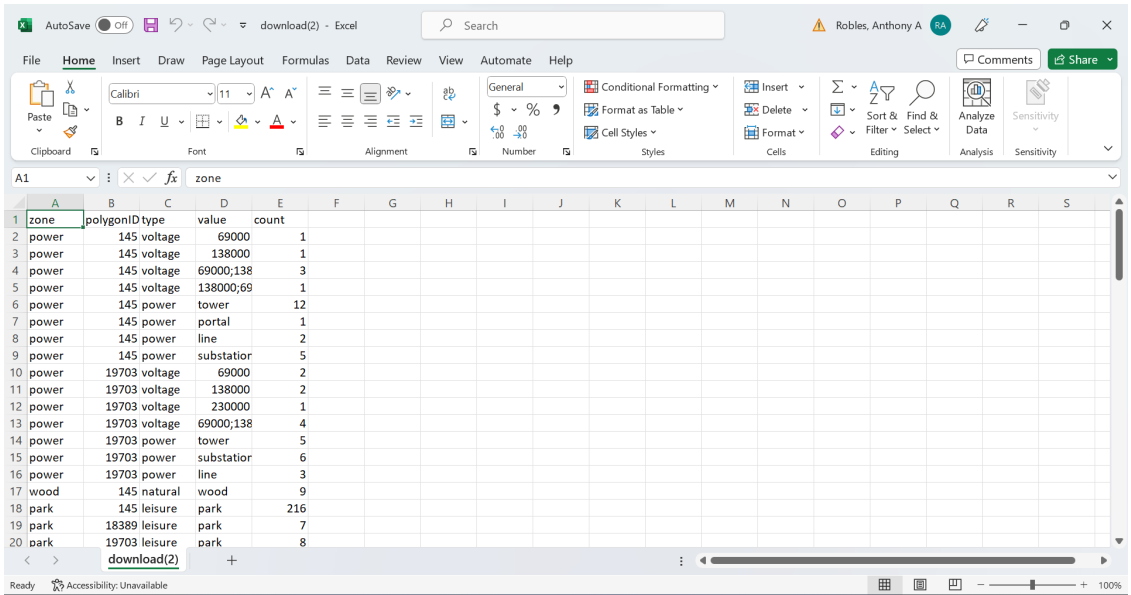


Figura 4.10: Caso de uso expotar a CSV



zone	polygonID	type	value	count
power	145	voltage	69000	1
power	145	voltage	138000	1
power	145	voltage	69000;138	3
power	145	voltage	138000;69	1
power	145	power	tower	12
power	145	power	portal	1
power	145	power	line	2
power	145	power	substatior	5
power	19703	voltage	69000	2
power	19703	voltage	138000	2
power	19703	voltage	230000	1
power	19703	voltage	69000;138	4
power	19703	power	tower	5
power	19703	power	substatior	6
power	19703	power	line	3
wood	145	natural	wood	9
park	145	leisure	park	216
park	18389	leisure	park	7
park	19703	leisure	park	8

Figura 4.11: Documento generado

4.2.8 Caso de uso ”Control Multilayer”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de cambiar la capa base del mapa entre tres opciones, las cuales son básica, satélite y la brindada por OSM. La acción es posible gracias a la implementación de un control multilayer brindada por Leaflet.

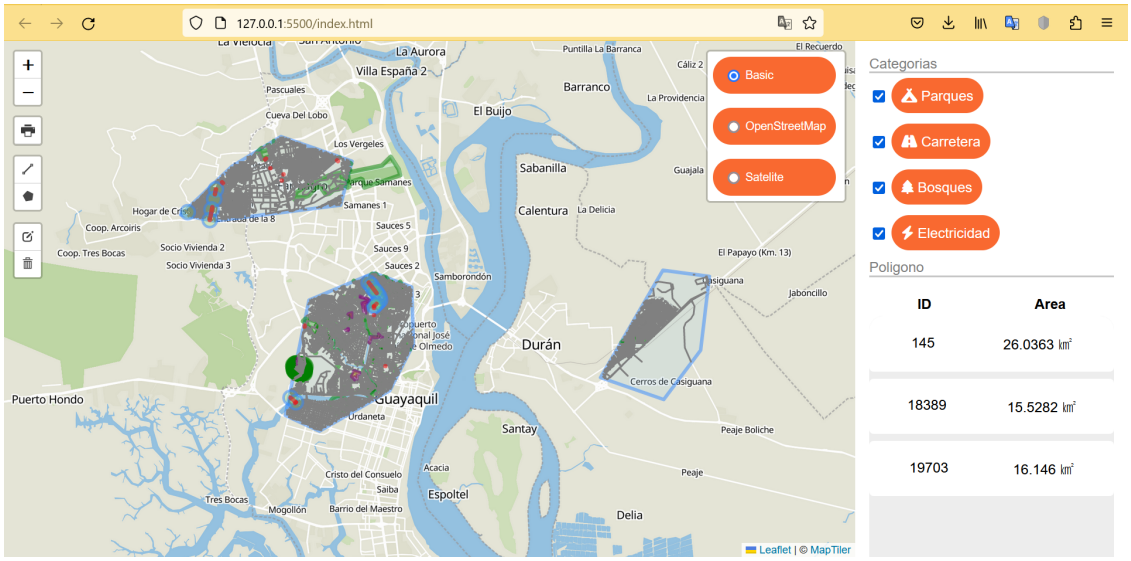


Figura 4.12: Capa base de mapa ”Basic”

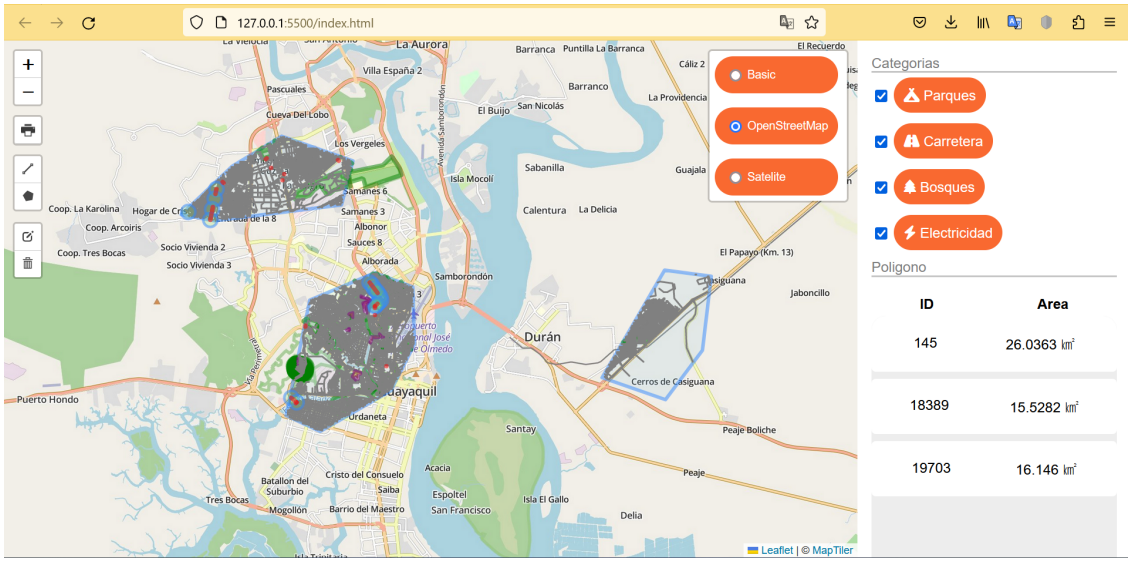


Figura 4.13: Capa base de mapa ”OpenStreetMAP”

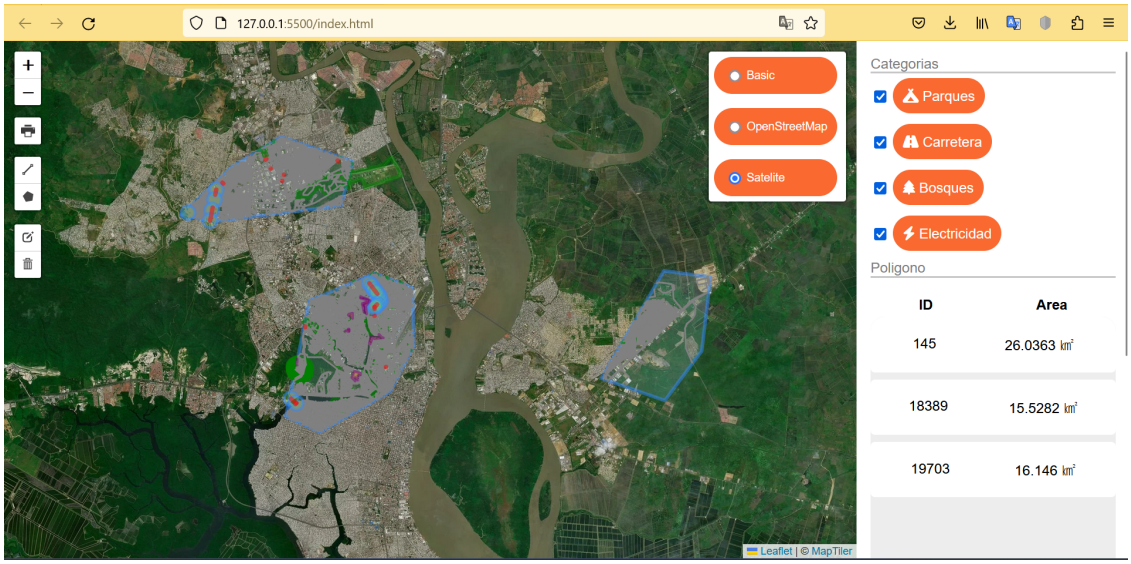


Figura 4.14: Capa base de mapa ”Satelite”

4.2.9 Caso de uso ”Despliegue de información”

Este caso de uso tiene como actor principal al usuario final, el propósito es brindarle la capacidad de desplegar la información de los marcadores de las diferentes categorías que están presente en esta aplicación. La acción es posible dando click sobre los marcadores, en donde aparecerá una pantalla emergente con las etiquetas respectivas entregadas por OSM.

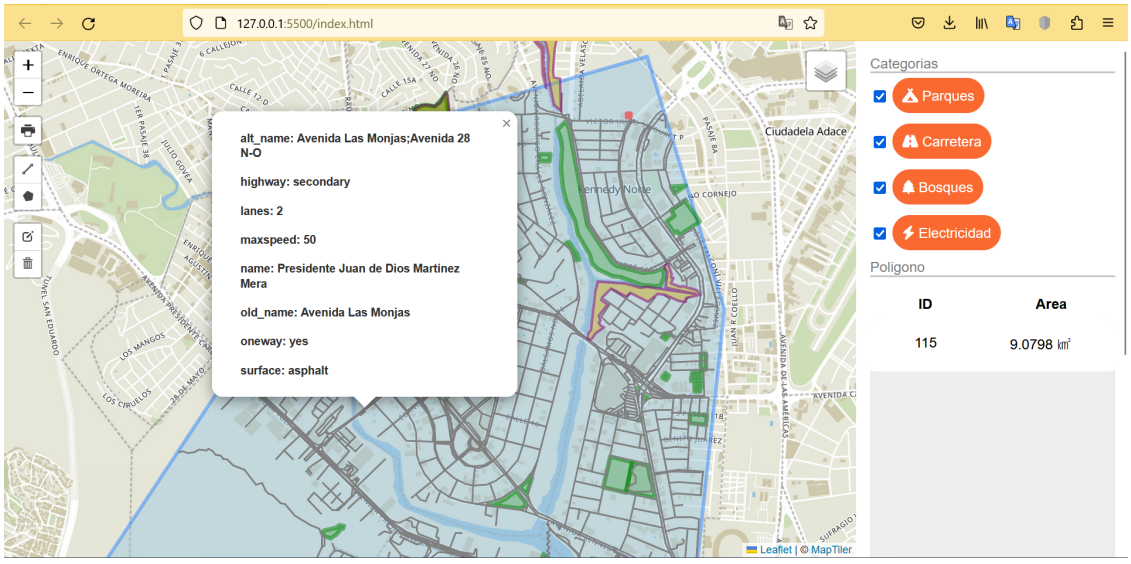


Figura 4.15: Despliegue de información sobre polilínea

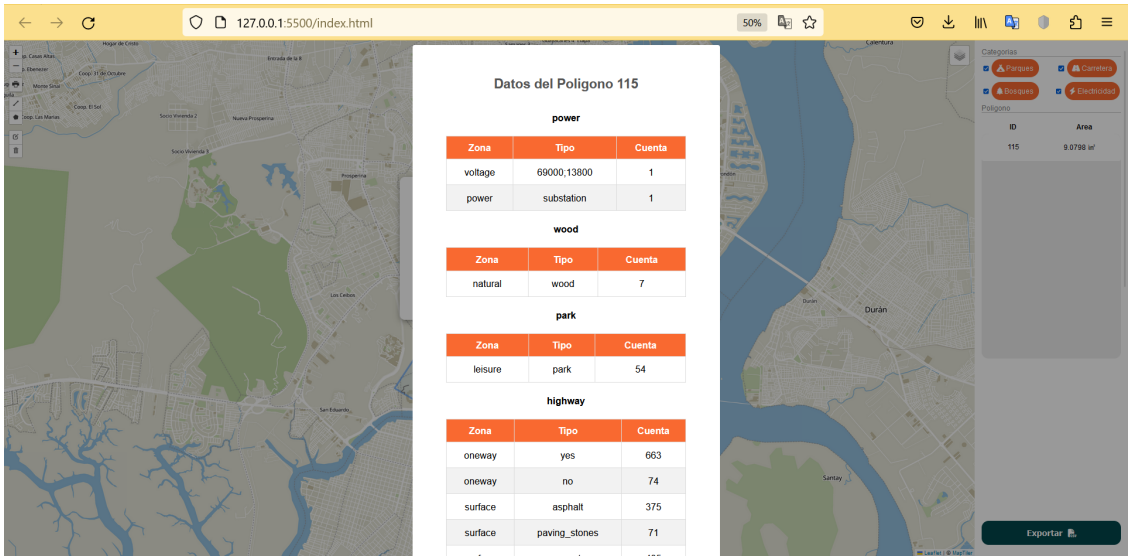


Figura 4.16: Despliegue de información general del área de estudio

4.3 Evaluación del rendimiento de la aplicación

En esta sección se evaluó el rendimiento de la aplicación, se presentan dos escenarios, en donde intervienen cierta cantidad de usuarios tratando de usar la aplicación. Se podrá notar que llega a un punto en donde el servidor colapsa en cierto número de usuarios.

4.3.1 Escenario 1

Para este primer caso se consideró una muestra de 20 usuarios, los cuales iban haciendo peticiones gradualmente por 1 minuto a la API de Overpass Turbo, debido a que es la que hace posible la obtención de los datos. Los resultados obtenidos en este primer escenario fueron muy satisfactorios, el número total de peticiones fue de 303 y las peticiones por segundo fueron de 4.03 peticiones/s, dando un porcentaje de error de los paquetes perdidos de 0.33% y con un tiempo promedio de respuesta de 1,073 ms.

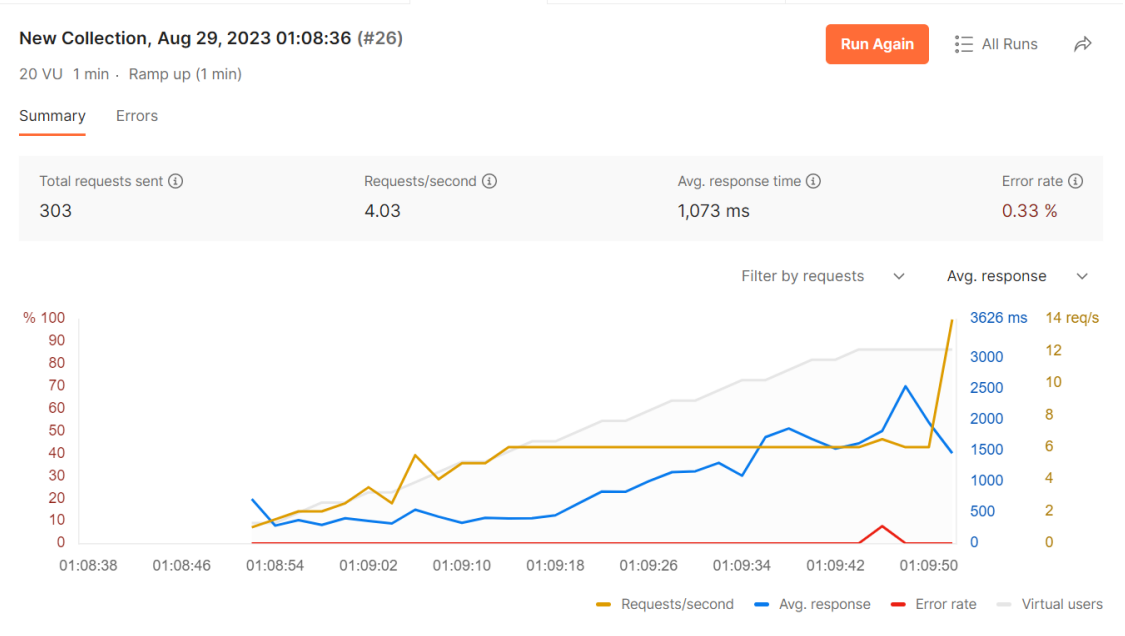


Figura 4.17: Resultados de la prueba de escenario 1

4.3.2 Escenario 2

En el segundo escenario se realizó la misma prueba pero con la diferencia que ahora los usuarios tratando de enviar las peticiones POST aumentaron a 100 personas durante 1 minuto. En este caso los resultados obtenidos no fueron lo suficientemente satifactorios, ya que, el promedio de datos perdidos fue de 76.90% con un tiempo de promedio de 1,008 ms y un total de 19.83 peticiones por segundo. El porcentaje elevado de error se debe a problemas por parte del servidor, en donde Postman indica que existe un error en tiempo de ejecución relacionado con la solicitud de datos desde OSM, esto debido a una limitación de la tasa de acceso. Se sugiere verificar el estado de la cuota de acceso de la dirección IP que está realizando la solicitud.

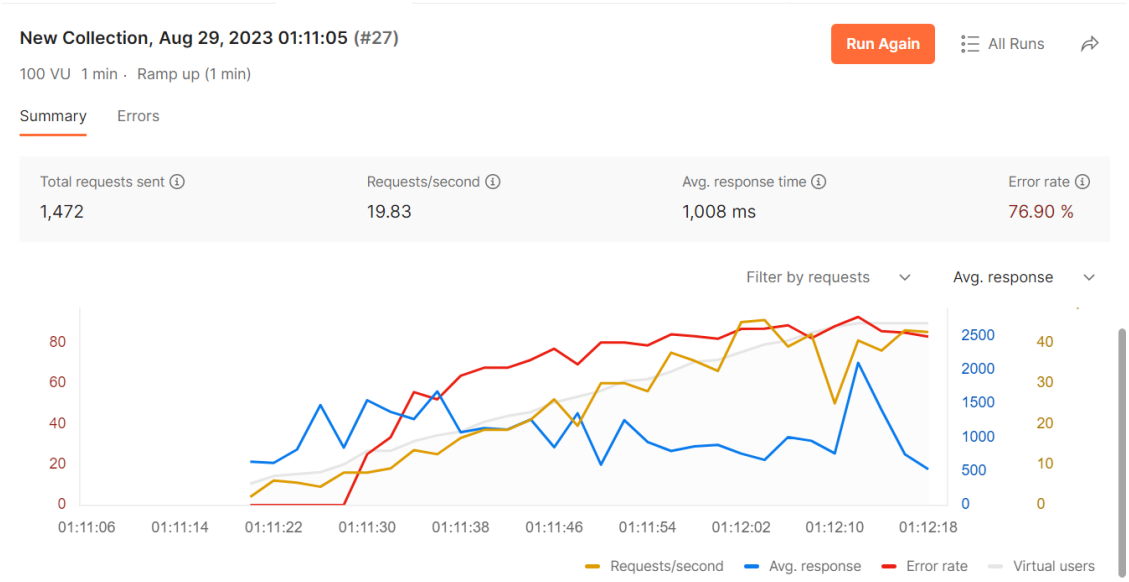


Figura 4.18: Resultados de la prueba de escenario 2

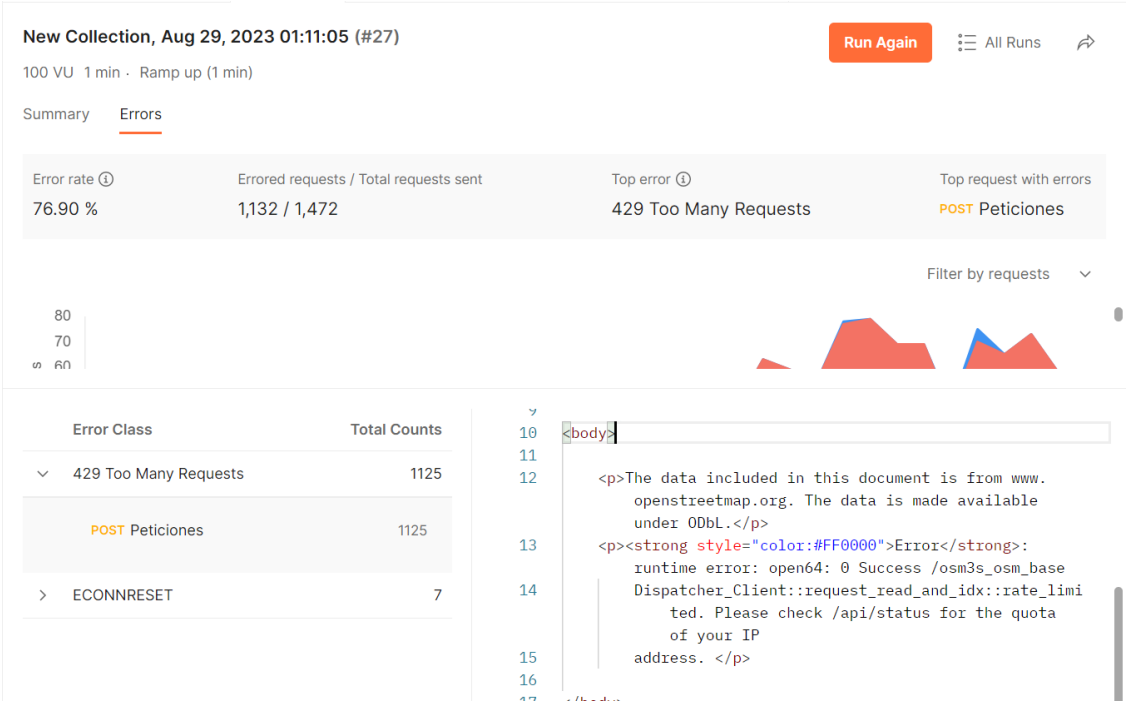


Figura 4.19: Errores del escenario 2

4.4 Caso de estudio

En esta sección se procede analizar los resultados de la aplicación como tal, analizando los datos que nos devuelve la base de datos de OSM de acuerdo a las métricas

establecidas en la sección 2.5.

En la figura 4.20 se observa que se procedió a seleccionar un área de estudio. Debido a que será un análisis general, los checkboxs de las cuatro categorías estarán activados.

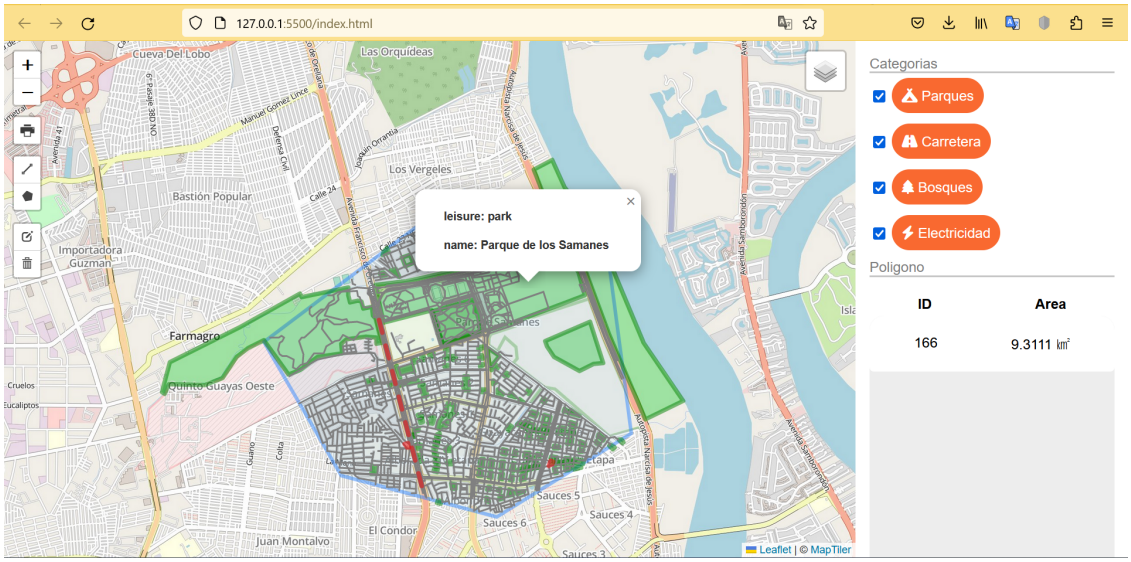


Figura 4.20: Selección del área de estudio

Como dato importante se observa a primera vista el área total del polígono, localizada en la zona 3 de acuerdo a la distribución de la pantalla que nos muestra la figura 4.1. Luego, se procede a desplegar la información general del área de estudio, tal como lo muestra la figura 4.21 y 4.22.

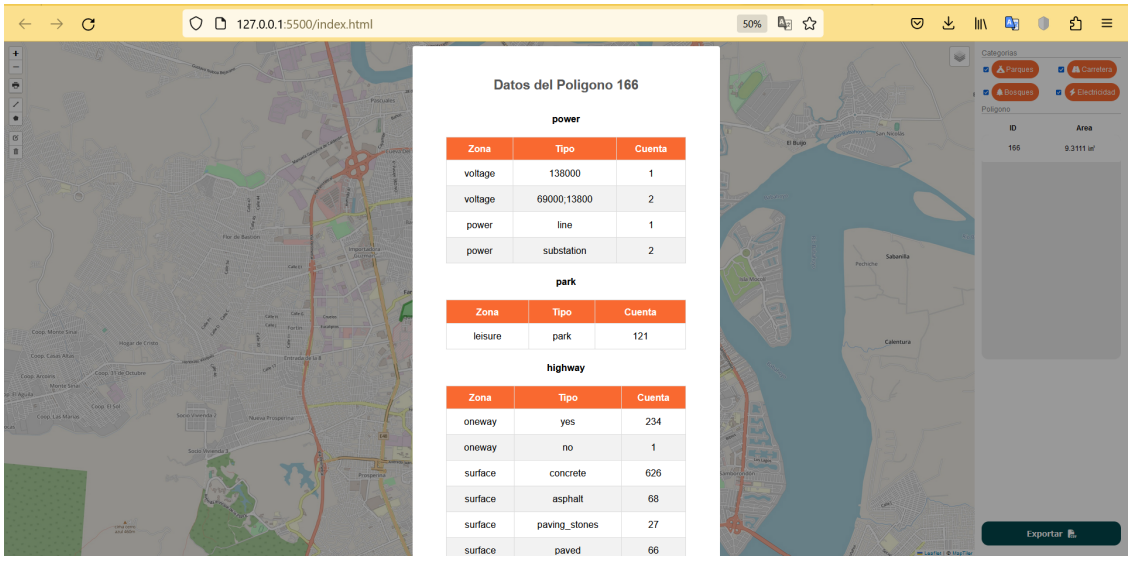


Figura 4.21: Información general del área de estudio

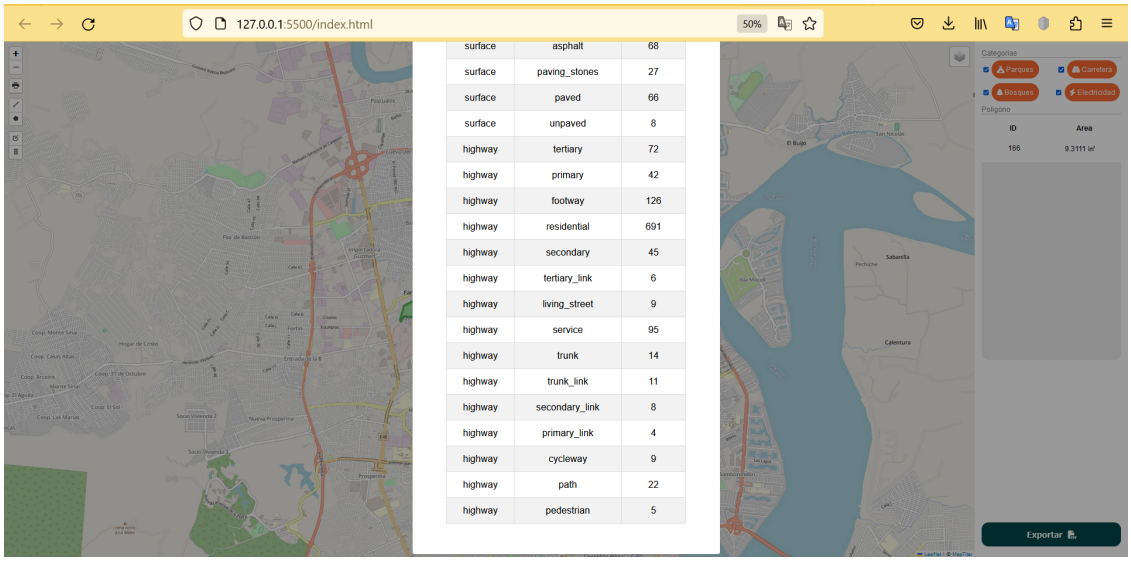


Figura 4.22: Información general del área de estudio

4.4.1 Distribución de calles por tipo de superficie

La tabla 4.1 nos indica el porcentaje por superficies que están presentes dentro del área seleccionada. La pavimentación se ha convertido en un elemento clave para la viabilidad de carreteras, calles y todo tipo de vías, ya que proporcionan mucha durabilidad y una superficie de rodamiento estable para los vehículos [1]. Analizando los datos obtenidos se concluye que el área tiene un alto porcentaje de que no existan accidentes de tránsito causados por el tipo de superficies.

Tabla 4.1: Tipos de superficie de las calles

Superficie	**Cantidad**	**Porcentaje**
Concreto	626	79%
Asfalto	68	9%
Piedra pavimentada	27	3%
Pavimentada	66	8%
Sin pavimentar	8	1%
Total	795	100%

[1] Tipos de pavimento: https://involucrasl.es/todos-los-tipos-de-pavimentacion-que-tienes-que-conocer/

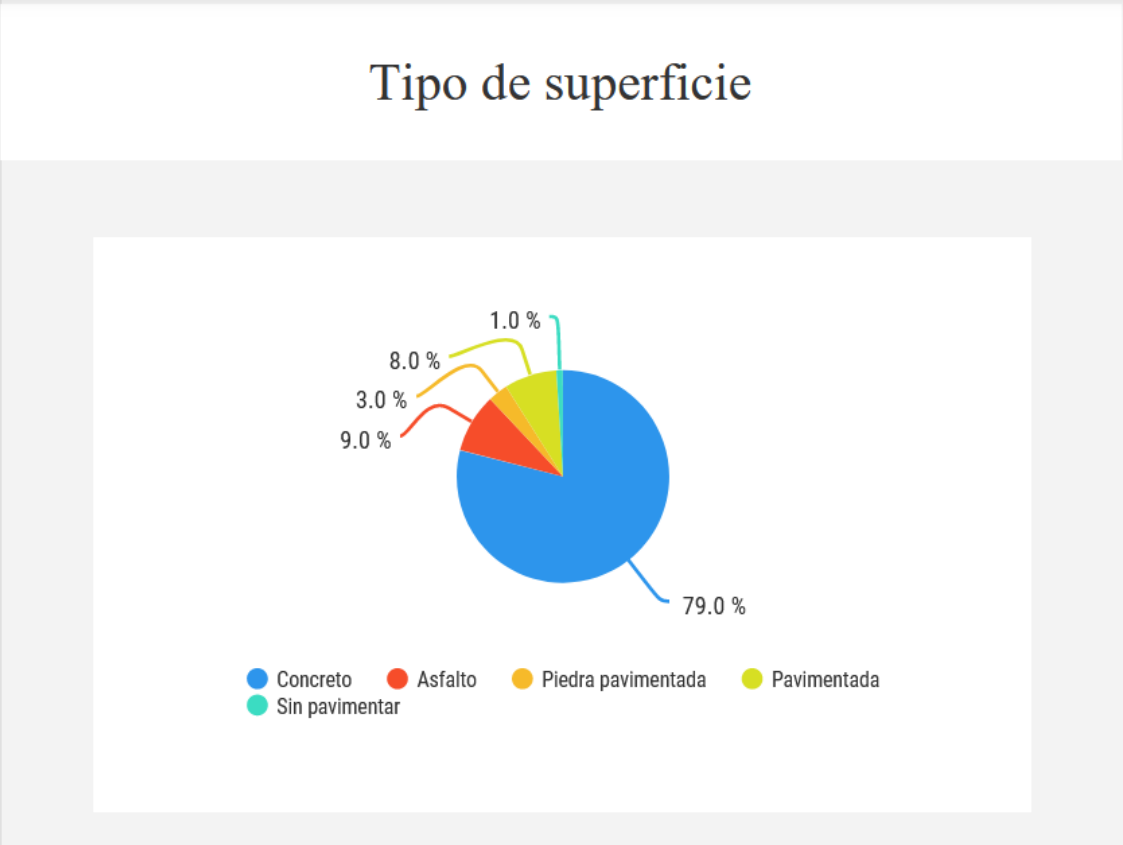


Figura 4.23: Tipos de superficie

4.4.2 Densidad de calles por kilómetros cuadrados

El número total de carreteras incluyendo las analizadas en la subsección 4.3.1 es de 1954, recordando que el área total de la zona de estudio es de 9.3111 kilómetros cuadrados, se procede aplicar la ecuación 2.1. Por lo que se obtiene lo siguiente:

$$Densidad\ de\ calles = (Total\ de\ calles)/(Area\ total) \tag{4.1}$$

$$Densidad\ de\ calles = 1954/9.3111\ km2 \tag{4.2}$$

$$Densidad\ de\ calles = 209.85\ calles/km^2 \tag{4.3}$$

Una vez hecho el cálculo respectivo se deduce que en la zona de estudio existen 209.85 calles por cada kilómetro cuadrado.

4.4.3 Capacidad energética de la zona

Si se observa la figura 4.21 en la sección de power, se observa que existen tres subestaciones.

- **69000-13800v** estos valores de voltajes obtenidos por OSM, nos indica que las subestaciones trabajan con diferentes valores entre rangos de 13800 v y 69000 v. Lo que significa que cuando existe alta demanda la subestación puede suministrar energía a 69000 v para satisfacer la necesidad de energía en la zona. En momentos de menor demanda, podría suministrar energía a un voltaje menor, como 13800 v, para conservar recursos y eficiencia.
- **138000v** el voltaje obtenido por esta subestación tiene una capacidad considerable con las dos primeras, por lo que puede abastecer una zona más amplia.

4.4.4 Espacios verdes y parques

Nuevamente, en la figura 4.21 nos dirigimos a la sección de park y se observa que existen 121 parques, lastimosamente los datos que se obtuvieron de OSM no son suficientes para determinar el porcentaje de área total de los espacios verdes, ya que, al dar click sobre los marcadores de los múltiples parques que se pueden observar, se puede notar que la información es nula (figura 4.24). Sin embargo, existe un parque en donde se muestra el nombre, el cual es el Parque Samanes.

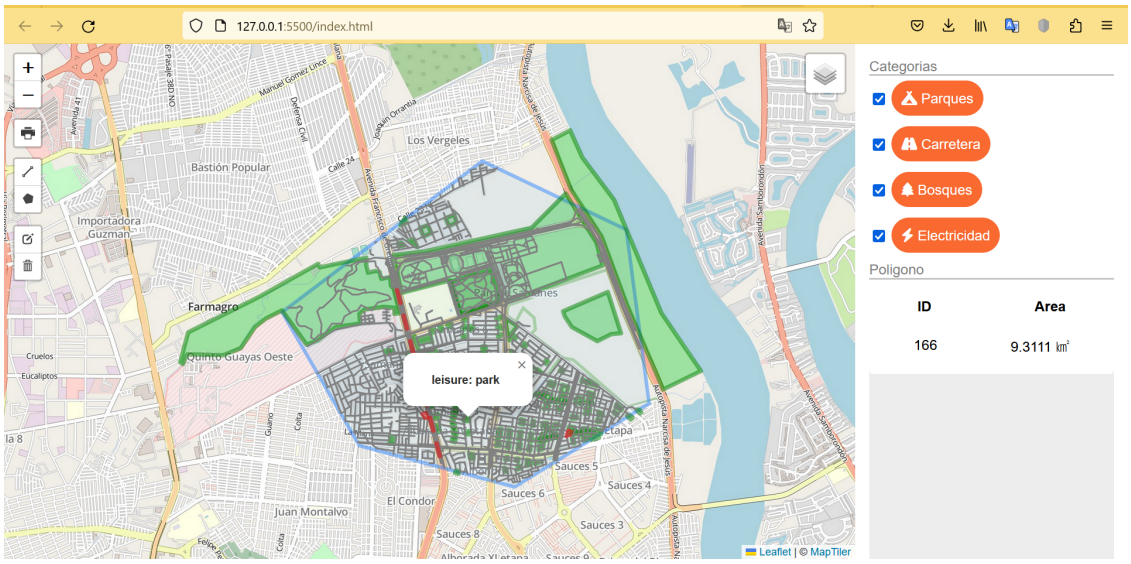


Figura 4.24: Falta de información de parques.

El área comprendida del parque en cuestión es de 379.79 hectáreas [2], realizando la respectiva conversión se obtiene 3.7979 kilómetros cuadrados. Aplicando la ecuación 2.2 se puede obtener el porcentaje que ocupa este parque en el área de estudio.

$$Porcentaje\ area\ total\ de\ espacios\ verdes = (Area\ de\ parques / Area\ total) * 100 \tag{4.4}$$

$$Porcentaje\ area\ total\ de\ espacios\ verdes = (3.7979\ km2 / 9.3111\ km2) * 100 \tag{4.5}$$

$$Porcentaje\ area\ total\ de\ espacios\ verdes = 40.78\% \tag{4.6}$$

[2]Parque Samanes: https://ec.viajandox.com/guayaquil/parque-samanes-A3227

CAPÍTULO 5

5. CONCLUSIONES Y LINEAS FUTURAS

En este capítulo se redacta las conclusiones que se obtuvieron luego de haber implementado las herramientas tecnológicas para poder satisfacer los objetivos específicos planteados en la sección 1.3.2. Así como recomendaciones para el usuario final y futuras mejoras que se le pueden implementar a la aplicación.

5.1 Conclusiones

- En base a los resultados finales de productividad, la elección de haber implementado la biblioteca de Leaflet para el desarrollo de esta aplicación fue un acierto. Debido a que esta librería brinda numerosas herramientas con las que el desarrollador puede trabajar con el fin de crear mapas interactivos y de fácil comprensión para poder entregar datos geoespaciales de forma intuitiva.

- En base a las pruebas realizadas con la herramienta de control de dibujo de Leaflet se puede concluir que esta ha demostrado ser muy valiosa para cumplir el tercer objetivo específico de esta aplicación, brindando la capacidad de que el usuario final esté en total libertad de trazar polígonos de cualquier forma directamente sobre el mapa. Además la capacidad de que venga incorporado la opción de editar y eliminar ayudan a que se pueda implementar más opciones para el resultado final, ya que, en un principio no estaba en mente poder editar ni eliminar las áreas de estudio.

- En base a los resultados obtenidos para el análisis de caso de estudio, al trabajar con un proyecto colaborativo y una base de datos geoespacial de código abierto que permite a personas de todo el mundo crear, editar y compartir información cartográfica como lo es OpenStreetMAP, se vieron reflejado que existen lugares

en donde la información es casi nula, un claro ejemplo se lo puede observar en la figura 4.21. Causando así que no se pueda realizar un análisis más profundo sobre ciertas categorías implementadas en este proyecto. Sin embargo, la información de coordenadas es precisa, lo que garantiza que sobre el mapa se dibuje la distribución territorial exacta de las capas de las categorías, garantizando así cumplir con el primer objetivo específico.

- En base a los datos obtenidos por Postman, se llega a concluir que la aplicación no soportaría 100 usuarios tratando de enviar las solicitudes POST a Overpass Turbo para la obtención de los datos de un área de estudio.

- La fácil integración que tiene Leaflet con otras tecnologías permitieron la conexión con la base de datos de OpenStreetMAP mediante el uso de la API de Overpass Turbo. En principio la idea era descargar archivos en formato GeoJSon desde el propio sitio web de Overpass, pero al intentar generar la consulta para todo el mapa, este daba problemas ya que era demasiado flujo de datos, así que se optó por la implementación directa de la API que mediante peticiones POST realizaba las consultas a la base de datos de OSM.

5.2 Recomendaciones

- Se recomienda una conexión estable a internet para evitar demoras al momento de realizar una petición.

- Al momento de realizar una consulta mediante la creación de las áreas de estudio, se recomienda que el polígono no sea exageradamente grande, debido a que se pueden presentar demoras en obtener los resultados deseados.

- En la carpeta del proyecto existe un archivo llamado overpassQuery.js que es donde se aloja todo lo relacionado a las consultas, en caso de que el código se quiera editar para obtener categorías diferentes a las que presenta en principio esta aplicación web, se recomienda ir a este archivo y cambiar la consulta. Hay que recordar que el tipo de consulta varía para todas las categorías que ofrece OSM, por lo que, hay que tener en cuenta usar las keys correctas.

- Se recomienda que cada usuario se cree una cuenta en MapTiler para que de esta forma adquieran su propia API key, la cual se deberá reemplazar en el archivo configMap.js, recordar que este servicio es el que nos brinda las 3 capas bases disponibles de esta aplicación.

5.3 Líneas Futuras

A cntinuación se presentan posibles mejoras que harían de la aplicación un poco más completa.

- En primer lugar y como un punto fuerte sería buscar la implementación de otra fuente de base de datos más completa. Aunque si existen gratuitas lo más probable es que sea igual o peor que OSM, lo más razonable sería invertir en la API de Google.
- Implementar el uso de servicio de bases de datos externas que permitan alojar usuarios donde se guarden las consultas que se han realizado, para que de esta forma se tenga un registro del historial de áreas de estudio.
- Implementar la opción de poder visualizar rutas de caminos más cortas para poder desplazarse de un punto a otro punto, para ello se debería aplicar el uso de GPS que permita realizar esta operación.
- Al momento de desplegar la tabla con la información general por polígonos, esta nos muestra los datos computados de todas las categorías, indistintamente si el checkbox de una categoría en concreto este activada o no. Una posible mejora sería que se muestren los datos de acuerdo a los checkboxs que estén habilitados en ese momento.

BIBLIOGRAFÍA

Abdillah, M. Z., Nawangnugraeni, D. A., & Yuniarto, A. H. P. (2021). Geographic information system (gis) for mapping greenpark using leaflet js. *Jurnal Teknik Informatika Kaputama*, *5*(2), 259–266.

Arias, B., & Olave, I. (2015). *Implementación de un sig para la zona urbana del municipio de corinto, cauca con base en el producto de la revisión y formulación del p.b.o.t* (Doctoral dissertation). Universidad de Manizales. https://ridum.umanizales.edu.co/handle/20.500.12746/2349

Batty, M. (2013). Big data, smart cities and city planning. *Dialogues in Human Geography*, *3*(3).

Caballero, J. J. V. (2016). Modelo de procesos para el desarrollo del front-end de aplicaciones web. *Interfases*, (9), 187–208.

Campbell, S. (1996). Green cities, growing cities, just cities? urban planning and the contradictions of sustainable development. *Journal of the American Planning Association*, *62*(3). https : / / my . vanderbilt.edu/greencities/files/2014/08/Campbell1.pdf

Constantinidis, B., & Pérsico, M. E. (2021). *Gis aplicado al planeamiento urbano* (Doctoral dissertation). Universidad de Belgrano. http://repositorio.ub.edu.ar/handle/123456789/9246

Córdovez, R. R. T. (2009). *Openstreetmap: Aprendizaje colaborativo en urbanismo a través de la web 2.0*.

del Río San José, J. (2010). *Introduccion al tratamiento de datos espaciales en hidrología*.

Dimes, T. (2015). *Javascript una guía de aprendizaje para el lenguaje de programación javascript*.

Esri. (2011). *Gis for urban and regional planning* (Vol. 1).

Esri. (n.d.). *Https://www.esri.com/en-us/what-is-gis/overview*.

Ibarra, S. G. P., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el frontend al backend. *XXIII Workshop de Investigadores en Ciencias de la Computación*, 347–350.

Iglesias, G. E. P. (2014). *Diseño e implementación de una aplicación web para la gestión de información geográfica del departamento de desarrollo forestal de la conafor estado de méxico* (Doctoral dissertation). Universidad Autónoma del Estado de México. http://ri.uaemex.mx/bitstream/handle/20.500.11799/30870/PE%c3%91A-GABRIELA-LGI-2014.pdf?sequence=1&isAllowed=y

Korte, G. (2011). *The gis book* (Vol. 5).

Mora, S. L. (2001). *Programación en internet: Clientes web* (Vol. 1). Editorial Club Universitario.

Mora, S. L. (2002). *Programación de aplicaciones web: Historia, principios básicos y clientes web* (Vol. 1). Editorial Club Universitario.

Moreta, O. R. E. (2009). *Diseño e implementación de mapa interactivo utilizando web mapping y base de datos espacial: Ciudad de quevedo* (Doctoral dissertation). Universidad San Francisco de Quito. https://repositorio.usfq.edu.ec/bitstream/23000/984/1/95301.pdf

Navarro, A. P., Plana, A. B., Bollas, A. M., Olivella, R., Olmedillas, J. C., & Lloret, J. R. (2011). *Introducción a los sistemas de información geográfica y geotelemática*.

Olaya, V. (2009). Sistemas de información geográfica. *Cuadernos internacionales de tecnología para el desarrollo*, *8*(15).

Olaya, V. (2014). *Sistemas de información geográfica* (Vol. 1). https://hdl.handle.net/11537/25452

Pérez, A. A., Mazo, W. M., & Pineda, O. P. (2015). *Diseño e implementación de un sig bajo una plataforma web mapping para el ingenio mayagüez sa* (Doctoral dissertation). Universidad de Manizales. https://ridum.umanizales.edu.co/bitstream/handle/20.500.12746/2117/Mantilla_Arroyo_Perez_2015.pdf?sequence=1&isAllowed=y

Preciado, J. M. S. (2020). *Sistemas de información geográfica*.

Puebla, J. G. (2010). Las tecnologías de la información geográfica en la planificación urbana y la ordenación del territorio: Viejos retos, nuevas direcciones. *Ciudad Y Territorio Estudios Territoriales*, *42*, 431–443. https://recyt.fecyt.es/index.php/CyTET/article/view/76009

Quintana, H. I. C. (2015). *Planificador gis multimodal de rutas* (Doctoral dissertation). Pontificia Universidad Católica de Valparaíso. http://opac.pucv.cl/pucv_txt/txt-8500/UCE8993_01.pdf

Reino, A. E. G. (2014). *Aplicación de gis en la implementación del sistema de control geo referenciado para la red distribución del sistema de agua potable de patamarca patrono san andrés* (Doctoral dissertation). Universidad San Francisco de Quito. https://repositorio.usfq.edu.ec/bitstream/23000/3373/1/110679.pdf

Saavedra, N. S. (1992). Los sistemas de información geográfica (sig) una herramienta poderosa para la toma de decisiones. *Ingeniería e Investigación*, (28), 31–40.

Sarría, F. A. (2006). Sistemas de información geográfica. *Universidad de Murcia*, *239*.

APÉNDICES

A Manual de usuario

1. Dirigirse a la página web https://geoprojectapp.000webhostapp.com/

2. Posicionar en el mapa el área de estudio en donde se va a trabajar.

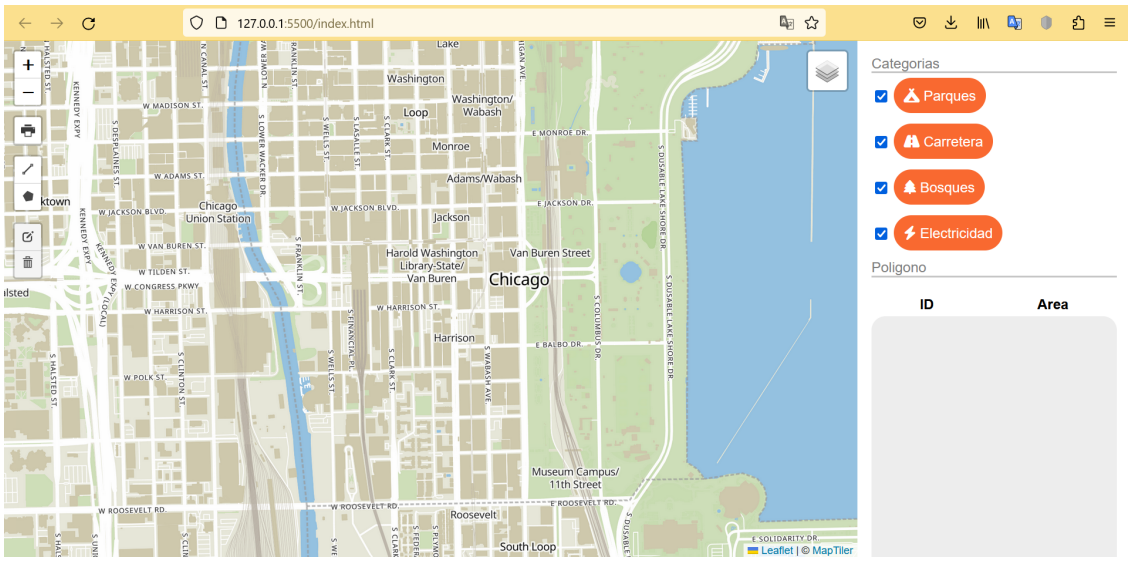


Figura 1: Elección de área de estudio

3. Haz clic en el botón ”Dibujar Polígono”. La forma correcta de dibujar es haciendo clic en el mapa a medida que vas formando el polígono. Para cerrar el área, simplemente haz clic en el punto inicial donde comenzaste a dibujar.

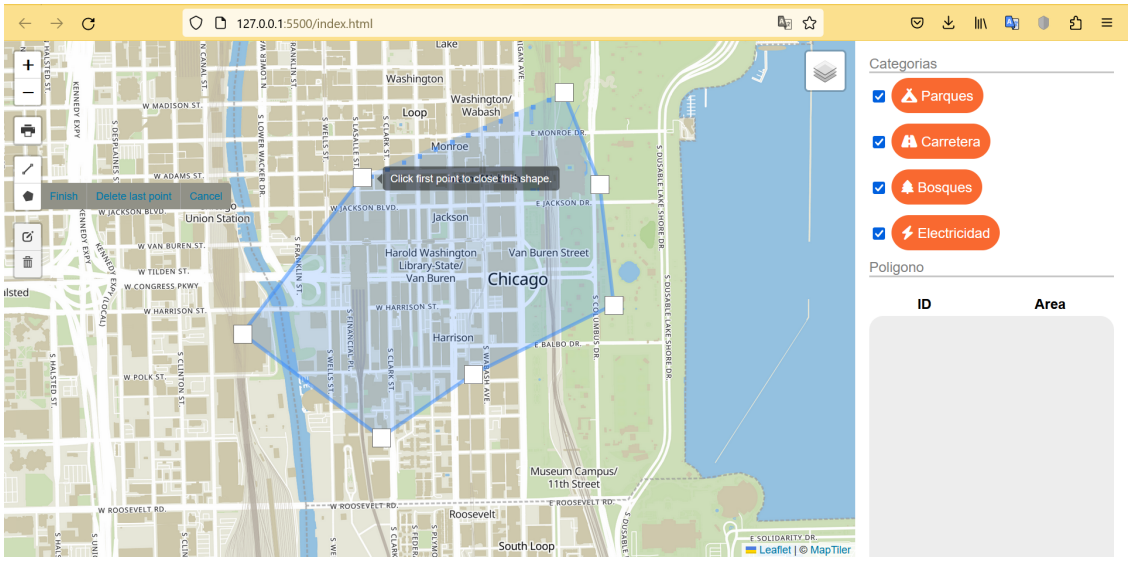


Figura 2: Elección de área de estudio

4. Haz click en el botón editar si deseas agrandar o disminuir el área de estudio.

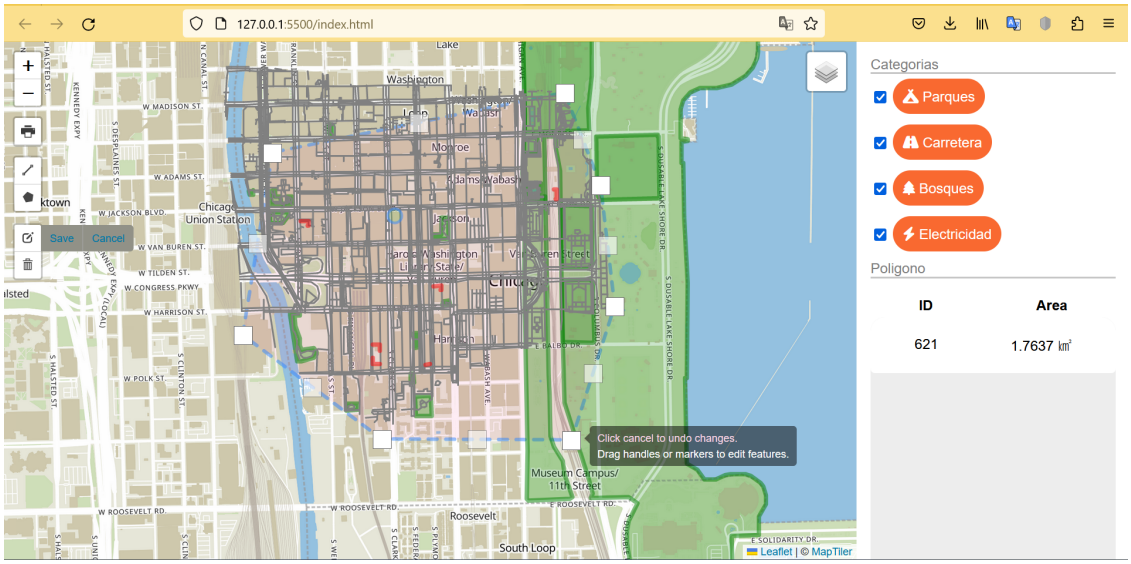


Figura 3: Editar polígono

5. En la parte superior derecha encontrarás los checkbox de las diferentes categorías, activa o desactiva los de tu preferencia.

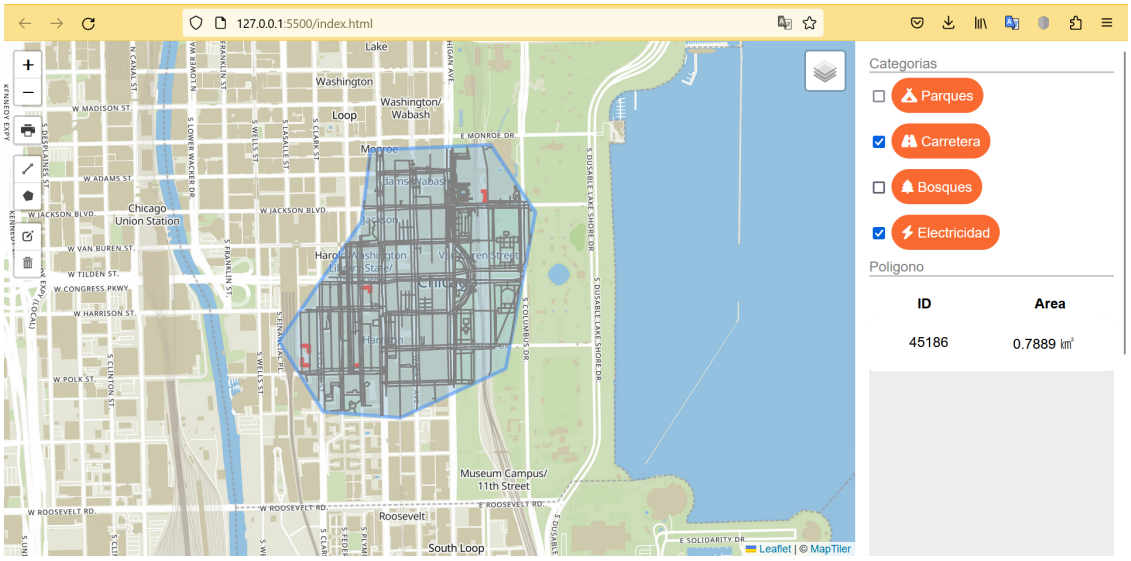


Figura 4: Checkboxs de las categorías

6. Si deseas visualizar la información directamente desde el mapa, haz clic sobre los el marcador de interés, luego aparecerá una ventana emergente con la información.

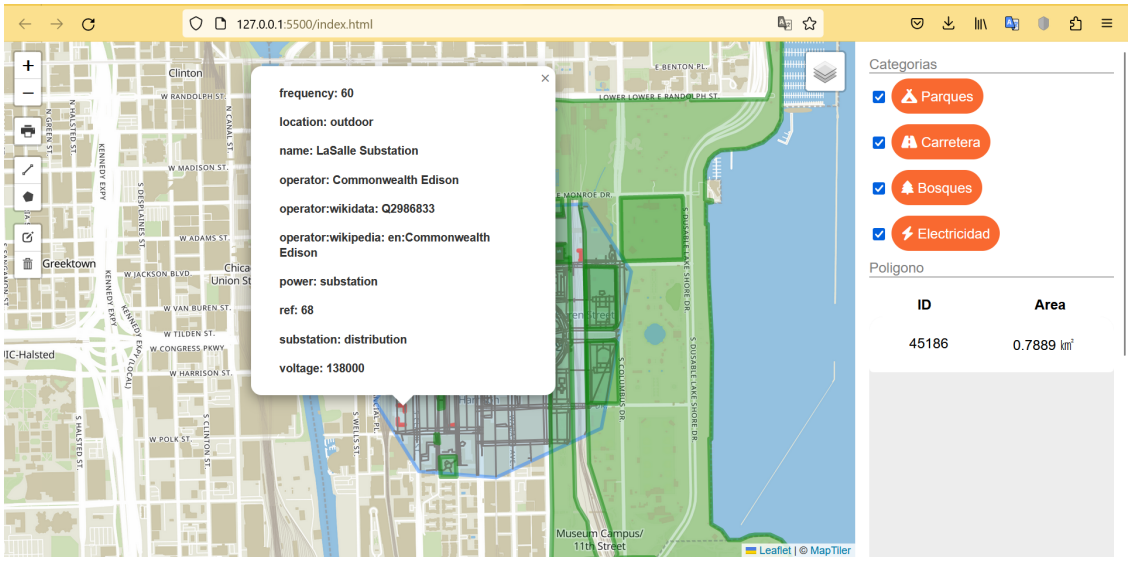


Figura 5: Despliegue de información

7. Si deseas visualizar la información general, haz clic sobre el ID del polígono, esto permite que se active la ventana emergente donde se encuentra la información.

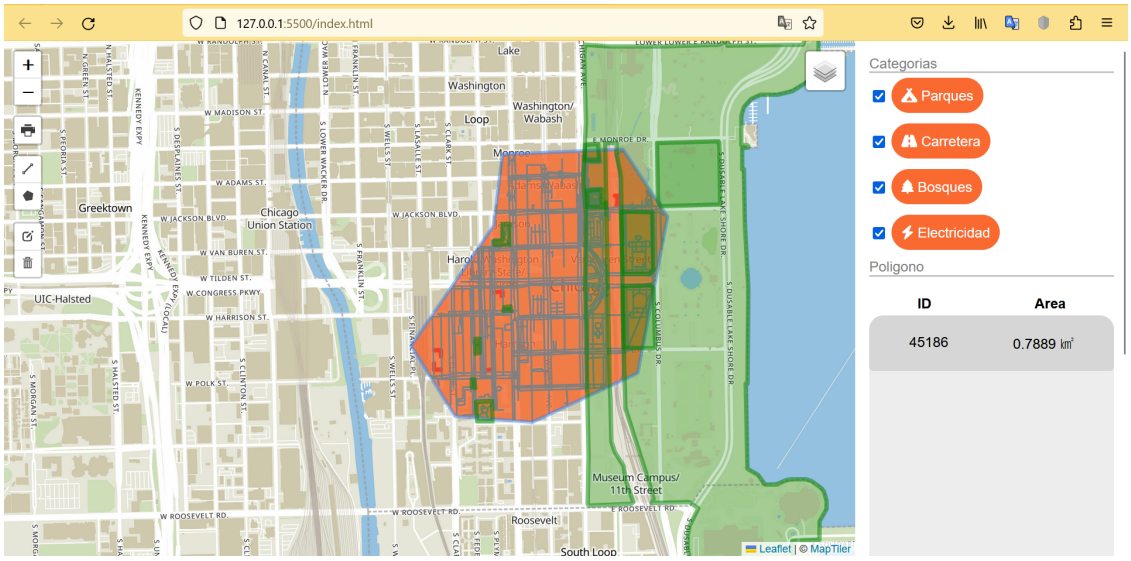


Figura 6: Clic en ID del polígono

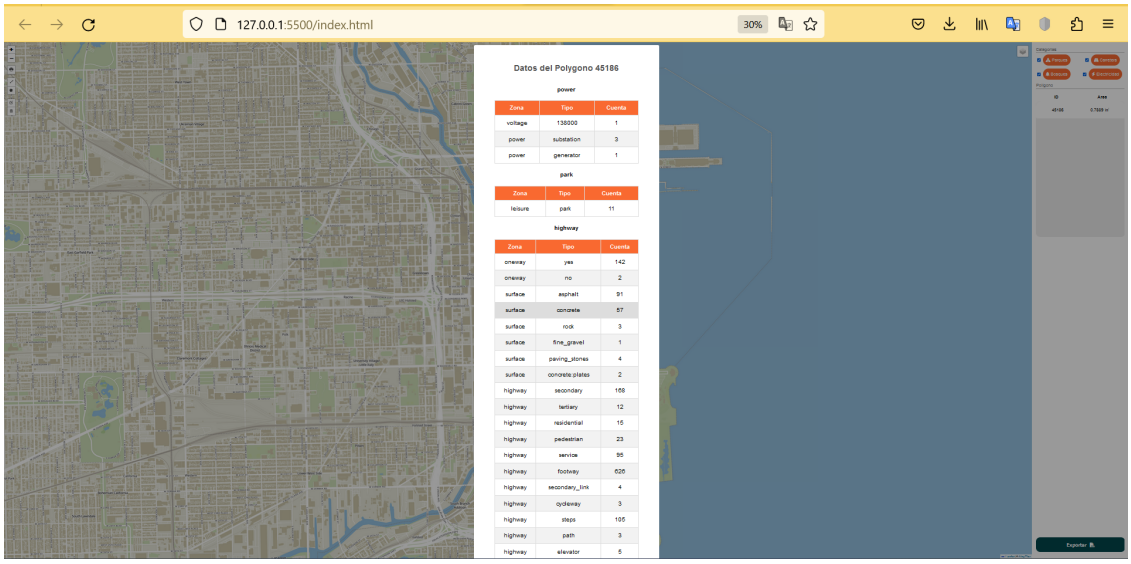


Figura 7: Despliegue de información general del polígono

8. Para exportar la tabla de información general, haz clic en el botón ”Exporta CSV”, luego comenzará la descarga.

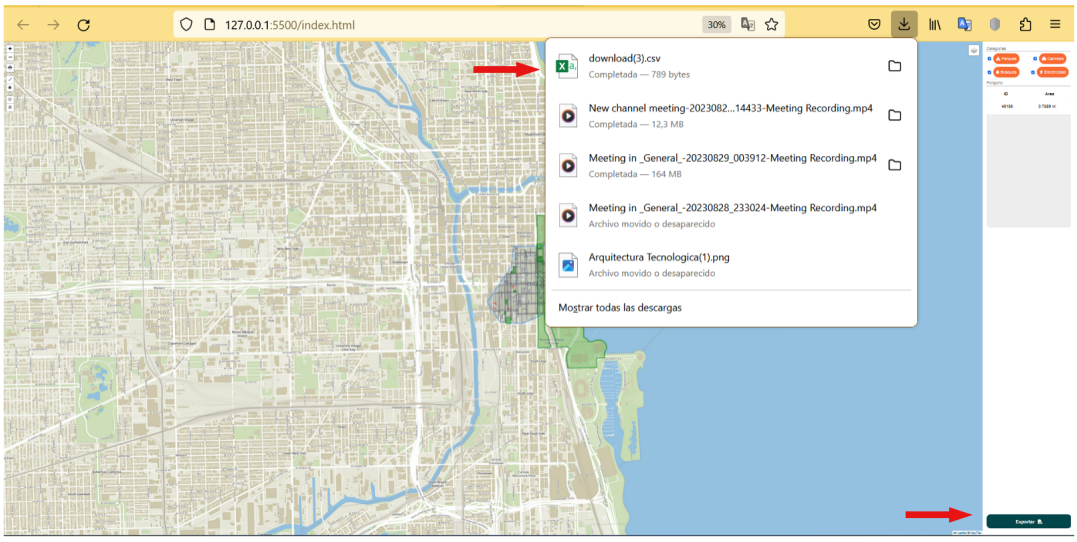


Figura 8: Exportar información a formato CSV

B Detalles del costo de la aplicación

Este trabajo tiene un valor de $10 por hora, a continuación se detalla lo realizado y el valor total de la aplicación.

Tabla 1: Tabla de costos

Actividad	**Horas**	**Total por horas**
Lectura sobre información de las diferentes tecnologías que se van a implementar	6	$60
Desarrollo del visor de mapas	10	$100
Desarrollo de las diferentes funciones integradas con Leaflet	37	$370
Desarrollo de las tablas con la información general de los polígonos y base de datos de sesión	15	$150
Desarrollo de la funcionalidad del botón ”Exportar CSV”	3	$30
Total	71	$710