UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA DE GENERACIÓN AUTOMÁTICA DE HEATMAPS SOBRE DATOS GEOLOCALIZADOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN COMPUTACIÓN

CONSTANZA CATALINA ESCOBAR FOSTER

PROFESOR GUÍA:
JOCELYN SIMMONDS WAGEMANN

MIEMBROS DE LA COMISIÓN:
JORGE PÉREZ ROJAS
DIONISIO GONZÁLEZ GONZÁLEZ

SANTIAGO DE CHILE
2018

Resumen

Unholster es una empresa que se dedica al desarrollo de software a medida para clientes en distintos rubros, con énfasis en el uso y análisis de datos. Dentro de sus proyectos internos, existe uno que consiste en visualizar datos geográficos interesantes, como, por ejemplo, el padrón electoral chileno, con el objetivo de descubrir patrones o relaciones en los datos. Estas visualizaciones muestran la concentración de puntos geográficos en las distintas calles de un área de interés, y se destacan conjuntos de calles con un alto número puntos. Sin embargo, el proceso para generar estos mapas es lento, con muchas etapas ejecutadas en forma manual, lo que hace poco viable la generación de nuevas visualizaciones, u ofrecer este sistema como un servicio.

En este trabajo de título se propone crear un sistema que automatice la creación de estas visualizaciones, y que además disminuya el tiempo de generación. Además, este sistema tendrá una interfaz web que permita a distintos usuarios subir sus propios conjuntos de datos a visualizar, dando la posibilidad a que esta plataforma sea ofrecida como un servicio de parte de Unholster a sus clientes.

Para esto, primero se estudió el proceso manual de generación de visualizaciones, identificando las distintas etapas y las oportunidades de mejora. Con esta información se diseñó un nuevo proceso que permite acelerar las operaciones a realizar, basándose en un modelo de datos que separa las calles y manzanas de los datos que se visualizarán. Además se introdujeron mejoras, como acotar el número de objetos que participan en las operaciones que realiza el sistema. Se establecieron las acciones que los usuarios pueden realizar en el sistema, y se definió una interfaz web para que la interacción sea simple y amigable.

Una vez definido el nuevo proceso y modelo de datos, se implementó una aplicación que expone una API y una interfaz web al usuario, y que implementa las operaciones descritas en este trabajo basándose en el modelo de datos planteado. Así, el sistema es capaz de generar visualizaciones para distintos tipos de datos geográficos, con una mínima participación de los usuarios y alcanzando los objetivos de tiempo planteados. Tanto el proceso como el sistema fueron validados por los directores de Unholster, quienes se mostraron satisfechos al ver que la aplicación cumple con los objetivos y los requisitos de usabilidad y tiempo de ejecución.

Finalmente, este trabajo representa un primer prototipo a lo que se espera sea un nuevo producto dentro de la empresa. Ahora generar visualizaciones de mapas es un proceso rápido y mayormente automático, por lo que puede ser ofrecido a clientes que quieran explorar y analizar sus datos geográficos en forma visual, abriendo una nueva oportunidad de negocio para Unholster.

A mis padres Jeannette y Pablo, y mi hermana Andrea
me han apoyado de forma constante toda mi vida.

A mi querido Ariel
por creer en mi, y en nuestros sueños.

Y a mi, hace 5 años
porque hemos llegado lejos, aunque parecía imposible.

Agradecimientos

Gracias a mi familia, por darme la oportunidad y las facilidades para estudiar y perseguir todas las oportunidades que se me han presentado, siempre dándome su apoyo, respaldo y amor. Ustedes me inculcaron que soy capaz de lograr todo lo que me proponga si me empeño, y hoy creo ser capaz de cumplir todas mis metas.

A todos los amigos que hice durante mis años en la Universidad, no podría haberlo logrado sin las risas, quejas, anécdotas, innumerables almuerzos, clases y tareas juntos. En especial gracias a Karina, Eduardo, Emilio, al Consejo de Ancianos, y los DCC's más viejos que me incluyeron como computina cuando apenas estaba dando un ramo de la carrera.

Gracias a todos en Unholster por el apoyo constante y el recibimiento, por darme la oportunidad de realizar este trabajo y de desarrollar al máximo mis capacidades; el aprendizaje ha sido inmenso desde que me uní al equipo como una alumna en práctica.

Muchas gracias a mi profesora guía, Jocelyn Simmonds, por la paciencia que ha tenido conmigo todo este tiempo, por las risas y la preocupación. Nuestras reuniones semanales durante esta memoria fueron una gran fuente de motivación para trabajar y avanzar en este proyecto. También al profesor co-guía Jorge Pérez, por sus ideas para mejorar el trabajo hecho en esta memoria.

Finalmente, gracias a la persona más importante que he conocido, Ariel, por todo el amor, la paciencia y apoyo que siempre me da. Por impulsarme a cumplir mis metas, y construir entre los dos el futuro que soñamos. La vida adulta se nos viene encima, pero estando juntos creo que todo es posible.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Antecedentes

Unholster es una empresa chilena fundada el 2008, se dedica al desarrollo de software a medida para clientes en distintos rubros, con énfasis en el uso y análisis de datos. La mayoría de estos proyectos puede clasificarse en una de dos categorías: por un lado proyectos que se enfocan en automatizar procesos críticos de empresas, que manejan sus flujos de información de forma manual, con muchos actores involucrados, para pasar a sistemas automáticos con trazabilidad de datos. Por otro lado, proyectos con clientes que manejan grandes cantidades de información, con distintos orígenes, donde se requiere implementar una gobernanza de datos, con una plataforma que permita utilizarlos para los distintos procesos de la empresa y además hacer análisis sobre ellos.

Dentro de la empresa también se desarrollan proyectos de interés interno, uno de ellos consiste en visualizar datos geográficos interesantes, tanto para uso de Unholster como para sus clientes. Estos datos son de todo tipo, refiriéndose a personas naturales, instituciones, lugares, entre otros. Un ejemplo son los datos contenidos en el padrón electoral chileno, donde se encuentra la dirección electoral de los votantes del país, junto con información como la circunscripción a la que pertenecen, el distrito, número de mesa, entre otros. Si bien esta información es por si misma valiosa, no siempre es fácil de consultar, en especial si los parámetros de consulta incluyen componentes geográficas, por ejemplo si se quiere saber datos sobre los votantes en un área específica.

En estos casos, es deseable disponer de estos datos visualmente en un mapa mediante simbología o heatmap, que es una visualización sobre mapas donde se muestra información usando una serie de colores. Esto se ejemplifica en la Figura 1.1, de esta forma se puede hacer una revisión por sector/lugar de los datos consultados. En el caso de la Figura 1.1, se identifican áreas con una alta concentración de votantes dentro de distintas comunas del distrito electoral n°9[1]. Esta información es especialmente útil para los candidatos políticos

[1]El distrito 9 se compone de las siguientes comunas: Conchalí, Renca, Huechuraba, Cerro Navia, Quinta Normal, Lo Prado, Recoleta e Independencia.

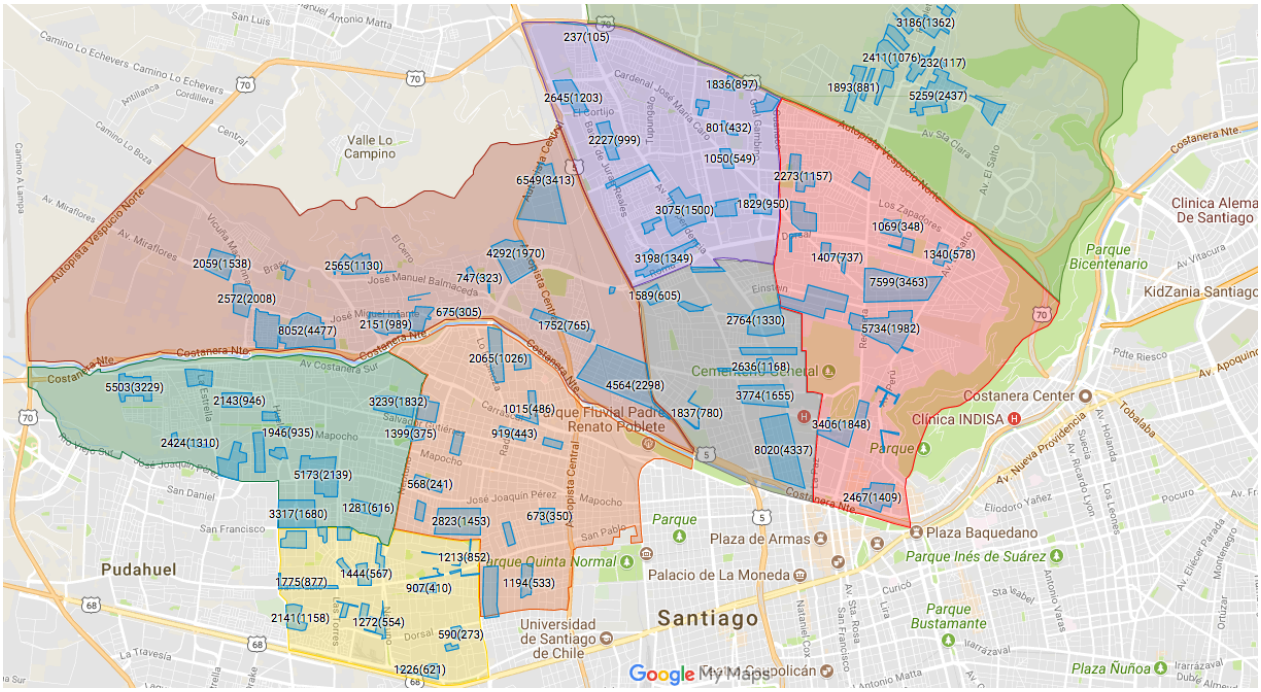


Figura 1.1: Mapa que muestra las comunas del distrito n°9 y, en azul, áreas con una alta concentración de votantes.

en estas comunas, ya que les permite planificar de mejor manera sus campañas electorales.

Si bien existen soluciones comerciales para generar mapas a partir de datasets con componentes geográficas, éstas son caras y por lo general debe generarse cada mapa en forma individual, siendo difícil generar un mismo mapa usando distintos conjuntos de datos. Por otro lado, la generación manual de estos mapas en Unholster es extremadamente lenta, tomando un día de trabajo de una persona, aproximadamente, para la vista de una comuna.

1.2. Motivación

Al momento de comenzar esta memoria, los mapas como el de la Figura 1.1 se generaban manualmente en Unholster; es decir, no había un proceso automático para procesar los datos y dibujar los mapas. Este proceso manual es lento: generar el mapa de una comuna toma un día de trabajo de una persona; esta limitación es muy fuerte, generar el mapa de una ciudad como Santiago o una región podría tardar semanas. No es escalable, ya que se debe hacer un tratamiento de los datos que se quieren integrar al mapa, identificar visualmente las áreas del mapa que se desea visualizar, y marcar las zonas con una mayor concentración de datos. Si los datos o los mapas de base utilizados cambian, se debe ejecutar el proceso desde el principio.

Al visualizar datos en un mapa, los requerimientos sobre los datos a mostrar vienen de los clientes; este es un problema dinámico, pues los usuarios finales pueden querer ver y comparar diferentes segmentos de usuarios, o bajo distintas condiciones, de forma iterativa.

Es aceptable que la generación de estos mapas tarde horas, pero no semanas para una ciudad completa, como ocurría con el proceso manual, ya que se pierde el propósito de generar distintos mapas para su comparación y análisis.

El proceso manual carece de herramientas de análisis al generar los mapas, es decir, al iniciarse este trabajo no se podía mostrar todo lo que se quisiera graficar en la plataforma, con una o varias agrupaciones recomendadas que permitan al usuario final iterar con ellas para encontrar la que más acomode sus objetivos de negocio. Como ejemplo, se podría querer visualizar la población de votantes del año 2013 y comparar con los votantes del año 2009; para luego comparar con los votantes registrados para el año 2017. Con el proceso manual se deben generar dos visualizaciones distintas, con todo el tiempo que esto significa, en lugar de generar una visualización que permita agregar o quitar conjuntos de datos, como los votantes de un año específico.

Como parte del proceso manual mencionado anteriormente, se encuentra la identificación de intersecciones entre las calles para generar las distintas cuadras, de manera similar a como se ve en la Figura 1.2: se deben encontrar todas las intersecciones de calles, marcadas como puntos en la figura, para así identificar los segmentos que las componen. Sin embargo, este proceso no identifica las manzanas que estos segmentos crean. Sin tener esta información disponible, se pierden muchas potenciales visualizaciones en que las agrupaciones podrían hacerse por manzanas en lugar de por calles o cuadras, como ocurre en el caso de el Censo, donde los resultados son entregados según cada manzana encuestada.

Figura 1.2: Cada línea entre dos puntos es un segmento de calle o cuadra, dato que se puede guardar en una base de datos para su posterior análisis.

Por último, dado que todo el proceso se hace en forma manual y particular a la visualización a generar, no es posible ofrecer a clientes la posibilidad de crear mapas a partir de sus propios datos, ya que esto requeriría que entreguen esta información a Unholster. Sería provechoso tener una API (*Application Programming Interface*) que exponga las funcionalidades del sistema, y permita a los usuarios interactuar con este en forma directa; así Unholster puede ofrecer esta interfaz a sus clientes como un servicio.

1.3. Objetivos

A continuación se presentan el objetivo general y los objetivos específicos de este trabajo de título.

1.3.1. Objetivo general

El objetivo de esta memoria es diseñar, construir y validar un sistema que automatice el proceso de generación de heatmaps a partir de datos geolocalizados, que disminuya el tiempo de creación de un mapa en un 75 % con respecto a lo que demora el proceso manual.

Además, para facilitar el acceso de los usuarios al sistema, se deben implementar una API y una interfaz web de esta.

1.3.2. Objetivos específicos

- Estudiar el proceso actual de generación de mapas e identificar oportunidades de mejora y automatización de este.
- Diseñar e implementar un proceso automático para generación de heatmaps sobre mapas, dado algún conjunto de datos de entrada.
- Validar funcionalidad del proceso según tiempos de ejecución, los datos de entrada, y satisfacción del cliente de este proyecto.

1.4. Estructura de la memoria

El resto del documento se estructura como sigue: en el Capítulo 2 se presentan conceptos de Sistemas de Información Geográfica relevantes durante el desarrollo de este trabajo de título, se hace una revisión del proceso actual en Unholster y se analizan distintas soluciones comerciales al problema presentado. Luego en el Capítulo 3 se revisan los requerimientos del proyecto y se plantea y analiza la solución escogida, mientras que en el Capítulo 4 se explica como fue realizada la implementación de esta. En el Capítulo 5 se revisa la validación de la solución implementada, y finalmente, en el Capítulo 6 se exponen las conclusiones del trabajo realizado y se presentan lineamientos para futuras mejoras en el proyecto.

Capítulo 2

Marco Teórico

Los conceptos relacionados a los Sistemas de Información Geográfica que son utilizados en este trabajo de título no son de conocimiento general, por lo que en la Sección 2.1 de este Capítulo se explican de manera breve algunos de los elementos que se utilizarán en capítulos posteriores. También se presenta, en la Sección 2.2, la situación en Unholster con respecto a la generación manual de visualizaciones, se describe el proceso seguido por medio de un ejemplo, y los problemas de este. Por último, en la Sección 2.3, se revisan algunos de los sistemas de información geográfica disponibles en el mercado, cuáles son sus ventajas, y por qué no se ajustan al problema presentado en este trabajo.

2.1. Sistemas de Información Geográfica

Los Sistemas de Información Geográfica (GIS por sus siglas en inglés), proveen herramientas para trabajar con datos geográficamente referenciados, ya sea almacenar, editar, analizar o mostrar esta información. Los datos que se manejan en un sistema GIS tienen información de localización (coordenadas geográficas) y atributos como el nombre del objeto real al que están asociados, o alguna especificación de este.

2.1.1. Conceptos fundamentales

Existen dos formas de modelar objetos del mundo real en un GIS: como raster o vectores [16]. Un raster es una cuadrícula de celdas o pixeles, donde cada celda contiene valores sobre un atributo del objeto a representar, siendo este valor una categoría o un número. Por ejemplo si se quiere representar la altura de un terreno, cada celda contendrá la altura en ese punto. Este concepto no se discute en mayor profundidad dado que no será usado en este trabajo de memoria.

Los modelos vectoriales se usan principalmente para representar objetos discretos y con límites bien definidos, como casas, ríos o calles. Existen tres tipos de geometrías vectoria-

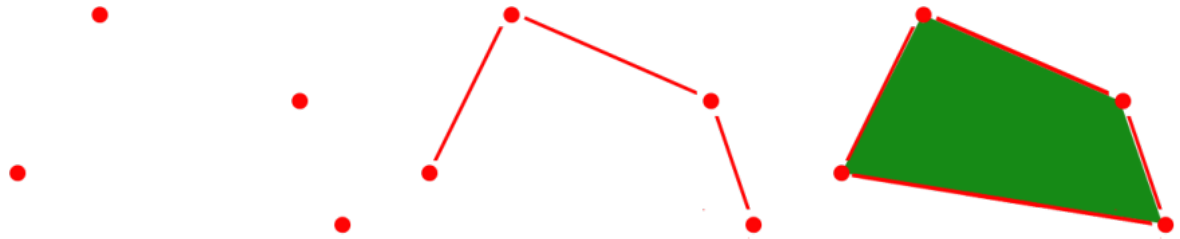
les [27], ejemplificadas en la Figura 2.1.

Punto: Es un par de coordenadas geográficas. Se usa para representar objetos pequeños y bien definidos, como una casa o un árbol, o una ciudad dependiendo de la escala de la visualización.

Polilínea: Es una secuencia de puntos conectados en orden por líneas, formando una figura abierta. Se usa para representar objetos lineales, como calles o ríos.

Polígono: Son puntos conectados en orden por líneas, pero a diferencia de la polilínea, forman una figura cerrada, la que puede ser cóncava o convexa. Se usa para representar áreas en una visualización.

Al polígono rectangular de menor tamaño que encierra todos los puntos de una geometría, según los ejes coordenados, se le llama *Bounding Box*; este término también es ampliamente usado en un GIS, sin embargo durante este trabajo se les llama simplemente cajas.



(a) Puntos (b) Polilínea (c) Polígono

Figura 2.1: Geometrías para representar datos vectoriales.

Para dibujar los modelos presentados en un mapa, se utilizan capas. Una capa usa símbolos para representar un grupo de objetos, los cuales tienen un mismo tema, geometría y conjunto de atributos. Por ejemplo, todos los objetos de una cierta capa son casas, que son representados con puntos, y en sus atributos tienen el nombre del dueño de la casa y el año de su construcción. Basándose en los atributos, cada objeto se puede representar de una manera distinta, por ejemplo, utilizando una variedad de colores.

Adicionalmente, para este trabajo se considerarán las siguientes definiciones:

Segmento: Es el trazo de calle comprendido entre dos intersecciones, que convencionalmente se conoce como *una cuadra*, representado por una polilínea.

Manzana: Es un polígono formado por segmentos contiguos, es decir, cuyos extremos coinciden.

2.1.2. Herramientas GIS

En esta sección se mencionan algunas de las herramientas y Sistemas de Información Geográfica usadas durante el desarrollo de esta memoria, y que se mencionan a lo largo de

este documento.

PostGIS [5]: Es una extensión para PostgreSQL que añade soporte para objetos geográficos, permitiendo ejecutar consultas de ubicación en SQL. Para lograrlo incluye nuevos tipos a la base de datos, así como funciones e índices sobre estos tipos.

El tipo de dato más relevante para este trabajo es `geometry`, en cuatro de sus formas, `geometry(Point)`, `geometry(LineString)`, `geometry(Polygon)` y `geometry(MultiPolygon)`. Las tres primeras representan a su tipo de geometría correspondiente de los mencionados en la sección anterior, y la última, `geometry(MultiPolygon)` representa una figura formada por distintos polígonos.

GeoDjango [11]: Es una extensión para el framework de desarrollo web Django, que añade funcionalidades para almacenar y manipular datos geográficos. También extiende su mapeo objeto-relacional, añadiendo compatibilidad con PostGIS y sus funciones.

GeoJSON [10]: Es un formato basado en la notación de objetos de Javascript (JSON [15]), diseñado para representar elementos geográficos junto con sus atributos. Está soportado por múltiples GIS, así como librerías para representar mapas.

QGIS [21]: Es un GIS de escritorio, gratis y de código abierto, que soporta trabajar con distintos formatos de datos geográficos, incluido PostGIS. Permite crear distintas visualizaciones, explorar datos y construir mapas exportables.

2.2. Generación de mapas en Unholster

Como se comentó anteriormente, Unholster tiene un proceso manual para generar las visualizaciones de mapas que se requieran. Estos mapas están limitados a un área específica y para un conjunto de datos específico, ya que cambiar alguna de estas variables requiere generar un mapa nuevo.

Para explicar este proceso, se usará como dataset de ejemplo el padrón electoral chileno, ya que tiene un gran número de datos, todos con su ubicación, e información anexa: sexo del votante, número de mesa, tipo de mesa, circunscripción y distrito electoral. Este dataset también se usará como ejemplo a lo largo de todo esta memoria.

2.2.1. Proceso manual

Los pasos que se siguen en la generación manual son los siguientes:

1. Se obtienen datos geográficos (calles) de OpenStreetMap [19], seleccionando solo un área de interés sobre la que se generará la visualización, por ejemplo, una de las comunas

del distrito electoral n°9.

2. Los datos de calles son transformados y normalizados a segmentos usando software externo, osm2pgrouting [6], que recibe como entrada un archivo obtenido de OpenStreetMap y los convierte a datos de tipo geométrico. Estos datos se almacenan en una base de datos PostgreSQL, extendida con PostGIS, y son utilizados para generar solo una visualización de la zona escogida. Los segmentos se guardan en tablas ad hoc en una base de datos, según la visualización que se generará.

3. Se hace una limpieza de los datos puntuales que se quieren visualizar, en este caso los votantes del distrito electoral n°9, verificando que las direcciones a las que están asociados existan, y se les asignan sus coordenadas geográficas usando el servicio de Google Maps [13]. Se importan a la misma base de datos que los segmentos, y de igual manera se guardan en tablas ad hoc según la visualización a generar.

4. Para ubicar los puntos geolocalizados en una calle específica, se busca su segmento de calle más cercano, ordenando todos los segmentos disponibles según su distancia al punto, y se le asigna al punto aquel que esté más cerca.

5. Para cada segmento, se cuenta con una consulta SQL todos los puntos que tiene asignados y se le asocia este total. Luego se ponderan los segmentos según esta suma, de manera que los que tienen una mayor población tengan un valor mayor.

6. Se visualiza el mapa con la población en QGIS, de forma que los segmentos de mayor rango se dibujen con un mayor grosor de línea, y se identifican visualmente los conjuntos de segmentos cercanos con mayor población.

7. Se delimitan manualmente ciertos conjuntos de alta densidad de habitantes, creando "barrios"[1] entre algunas calles.

8. Se exportan los límites de los barrios, junto con la metadata asociada, para su posterior visualización en programas como Google Maps u OpenStreetMap, que permiten a cualquier usuario hacer análisis sobre los datos presentados.

2.2.2. Problemas del proceso manual

Luego de conversar con los encargados de generar las visualizaciones siguiendo el proceso anterior, se encontraron múltiples problemas que esta memoria apunta a mejorar. Estos pueden dividirse en tres áreas: tiempo, reusabilidad y precisión.

Tiempo: Al inicio del proyecto en Unholster, ejecutar todo el proceso manual demoraba cerca de un día de trabajo para generar solo una visualización de una comuna. Luego de ganar cierto entrenamiento en seguir estos pasos, una visualización demoraba entre 3 y 4 horas, para comunas pequeñas como Talca o Linares. Parte importante de este

[1]Estos barrios no necesariamente se corresponden con barrios geográficos reales, la palabra se usa únicamente para denominar estos conjuntos de puntos.

tiempo es la asignación de puntos a su segmento más cercano, que demora alrededor de 2 horas.

Reusabilidad: Los segmentos de calles no pueden reutilizarse de una visualización a otro, aunque correspondan al mismo sector, lo que obliga a rehacer todo el proceso. Esto sucede porque el conteo de puntos que tienen asociados modifica las características de los segmentos. Dado que no existe un modelo de datos bien definido, las tablas para puntos y segmentos se crean de forma ad hoc según la visualización, generando una gran cantidad de datos duplicados.

Precisión: Los barrios interesantes se descubren por inspección visual, provocando inexactitudes en la visualización generada y enlenteciendo el proceso.

2.3. Soluciones existentes

Existen diversas plataformas que permiten hacer análisis de datos geográficos y entregar estos como un servicio. A continuación se presenten tres de los sistemas más conocidos, y se analizan sus ventajas y desventajas.

2.3.1. ArcGIS

ArcGIS [8] fue uno de los primeros GIS existentes en el mercado, y actualmente es uno de los líderes en cuanto a este tipo de software. Si bien su foco principal es en análisis offline e integración de datos geográficos con otras plataformas, también cuenta con un producto online (ArcGIS online) que permite generar mapas y dejarlos a disposición de otros en un sitio web.

Una de sus principales ventajas es la gran cantidad de características que ofrece, ya que es posible hacer casi cualquier cosa referente a sistemas geográficos, desde cargar datos propios para analizarlos y visualizarlos en un sitio web, hasta desarrollar sistemas de geolocalización y visualización de rutas y lugares cercanos, usando datos que son parte de ArcGIS.

Su desventaja principal es el alto costo de la plataforma, cuya suscripción anual por un servidor para uso en ambiente comercial cuesta 56.325 USD [4], equivalente aproximadamente a $36.000.000; un precio excesivamente alto para el bajo número de clientes de Unholster que usarían esta plataforma. Otra desventaja es el entrenamiento que se requiere para poder usarla en su totalidad, y en general el tamaño del software, pues gran parte de sus características no sirven para una aplicación como la que se busca desarrollar en este trabajo. Algunas de ellas son la colaboración en tiempo real sobre los mapas, integración en páginas web, integración con otras aplicaciones del ecosistema ArcGIS, entre otros.

2.3.2. CARTO

Este sistema de software como servicio provee herramientas GIS en línea [2], enfocándose principalmente en análisis de datos y visualización de mapas. Permite generar mapas personalizados, y su interfaz apunta a usuarios principiantes o no desarrolladores, de forma que puedan usar herramientas avanzadas de GIS, sin dejar de lado a usuarios avanzados que pueden manipular los datos de formas más complejas.

Por lo mencionado anteriormente, una de sus ventajas es su facilidad de uso. Otra es la existencia de una API en la plataforma, que permite usar los mapas generados en distintas aplicaciones web, o hacer consultas sobre los datos.

Su principal desventaja es su precio, ya que para acceder a las funciones más avanzadas como la API, se debe tener un plan de empresa. Si bien este precio se debe cotizar, como referencia se tiene que el costo para un individuo es de 1.639 USD anual [3], aproximadamente $1.050.000. Dado que probablemente el precio final para una empresa sea mayor que para un individuo, y considerando el bajo número de clientes de Unholster para esta plataforma, no parece ser una opción rentable.

2.3.3. Mapbox

Es un proveedor de mapas personalizados online [17], dando énfasis en que los mapas puedan usarse en cualquier plataforma, y en la facilidad para generarlos; en la misma línea de CARTO, no es necesario ser un experto o programador para poder utilizar la plataforma.

Su ventaja principal es la gran personalización y diseño de los mapas, junto con la posibilidad de uso en plataformas móviles y web, lo que permite integrar los mapas generados en otras aplicaciones.

Su desventaja es el precio, pues el sistema de cobros depende de la cantidad de usuarios que acceden al mapa; el costo base para una empresa es de 5.988 USD anual [18], aproximadamente $3.900.000, más un costo mensual según la cantidad de usuarios que acceden a los mapas y las distintas vistas de mapas pueden ver, lo que puede ser difícil de estimar anticipadamente. Además no cuenta con una API o alguna forma de acceder a los datos de forma automatizada, por ejemplo de parte de un cliente que quiere consultar los mapas generados con sus datos.

2.3.4. Discusión

Como se vio en las subsecciones anteriores, la mayor desventaja de las tres opciones presentadas es su costo para uso comercial; si bien se espera que el producto desarrollado sea comercializable, actualmente el número de clientes que tiene la empresa no justifica una inversión tan alta, y probablemente significaría una pérdida. Cabe mencionar, que con una solución ya desarrollada el número de clientes podría aumentar, pero las ganancias que esto

traería no pueden predecirse en este momento.

Adicionalmente, todas estas plataformas permiten generar visualizaciones a partir de un único dataset, pero no es posible generar una "plantilla" de visualizaciones en las que cualquier usuario pueda subir su propio dataset y se genere la visualización esperada, algo más parecido al sistema que Unholster espera poder ofrecer a sus clientes. Usando alguna de las plataformas presentadas, cada cliente tendría que configurar las visualizaciones por su cuenta, eliminando el propósito de este proyecto y el valor agregado que Unholster busca darle a sus clientes.

Capítulo 3

Análisis y Diseño de la Solución

Como se mencionó en los capítulos anteriores, el problema que se busca resolver corresponde a la automatización y tiempo de ejecución del proceso de generación de mapas en Unholster. Para ello se identificaron tres ejes principales del problema, el tiempo que demora el proceso, la nula reutilización de información y el descubrimiento de características interesantes de los mapas por inspección visual.

Para solucionar estos problemas, primero, en la Sección 3.1, se mencionarán cuáles son las principales historias de usuario y requisitos que el sistema propuesto debe cumplir. Posteriormente en la Sección 3.2, se explica la solución planteada en el desarrollo de este trabajo, cómo aborda las historias y requisitos planteados, y cuáles son los puntos importantes considerados en el diseño expuesto.

3.1. Definición del problema

El problema a resolver fue definido en dos partes: por un lado con historias de usuario que narran lo que los usuarios esperan del sistema, y por otro los principales requisitos funcionales y no funcionales, que describen los requerimientos técnicos que el sistema debe cumplir. Tanto las historias de usuario como los requisitos, detallados a continuación, fueron acordados y validados con el Director de Operaciones de Unholster.

3.1.1. Historias de Usuario

Los usuarios de estas historias son principalmente data scientists y clientes de Unholster, como empresas que quieren analizar a sus usuarios según su ubicación, o algún recurso con geolocalización. Todos los usuarios desarrollan las mismas historias, que se detallan a continuación en forma general, para después ejemplificar con una situación concreta.

Historias de usuario en forma general

HG1 Un usuario quiere subir al sistema un archivo con datos que contengan localización (latitud y longitud), y otros parámetros.

HG2 Un usuario quiere visualizar el número de datos pertenecientes a cada manzana o segmento de las ubicaciones asociadas a los datos.

HG3 Un usuario quiere modificar la visualización generada en HG2, al filtrar los datos según atributos del dataset subido.

Historias de usuario: ejemplo del padrón electoral

Las historias anteriores, reescritas tomando como dataset el padrón electoral, quedan de la siguiente forma:

HP1 Un usuario quiere subir al sistema el padrón electoral chileno geolocalizado, que contiene el domicilio electoral de cada votante con sus coordenadas geográficas, y otros datos como la circunscripción a la que pertenece, su número de mesa y el tipo de mesa.

HP2 Un usuario quiere visualizar el número de votantes pertenecientes a cada manzana o segmento de las ubicaciones asociadas a los datos.

HP3 Un usuario quiere modificar la visualización generada, al filtrar los datos según atributos del padrón, como mostrar solo el número de votantes de una circunscripción en particular.

Dado que estas historias no son suficientes para asegurar una solución al problema planteado, se presentan también los principales requisitos funcionales y no funcionales del sistema, que complementan las historias de usuario ya expuestas.

3.1.2. Requisitos

Si bien para este proyecto existen requisitos típicos de cualquier sistema de información, como administración y permisos de usuarios, se destacan los siguientes requisitos pues son únicos a este proyecto con respecto a un sistema de información común.

Requisitos funcionales

RF1 El sistema debe exponer una API para llevar a cabo acciones como subir archivos con datos y obtener mapas enriquecidos.

RF2 El sistema debe contar con una interfaz web con la que interactúen los usuarios.

RF3 Los mapas que consume el sistema deben ser usados como base para las distintas visualizaciones.

Los requisitos **RF1** y **RF2** fueron pedidos explícitamente por Unholster, ya que la estructura de API e interfaz web como sistemas separados se usa en muchos de sus proyectos, con buenos resultados. El requisito **RF3** surge en respuesta a la gran cantidad de información duplicada que se produce como consecuencia del proceso manual, como se vio en la Sección 2.2.

Requisitos no funcionales

RNF1 El sistema debe generar de manera automática las visualizaciones, sin necesitar input manual más que la elección de los datos a visualizar, y tomando en cuenta los filtros elegidos por el usuario.

RNF2 El sistema debe asignar los datos a sus segmentos y manzanas respectivos en a lo más dos horas.

RNF3 El sistema debe generar distintas visualizaciones del mismo dataset en menos de un día de trabajo.

Estos requisitos aseguran que el prototipo a desarrollar represente una mejora en los tiempos de procesamiento en cuanto al sistema manual, ya que disminuye la interacción del usuario en el proceso y acelera las operaciones que se deben realizar. En el caso de RNF2, el límite de tiempo está definido según lo que demora esta acción en el proceso manual, como se vio en la Sección 2.2.

3.2. Solución propuesta

Se identificaron tres procesos que el sistema debe llevar a cabo, y que conforman los componentes principales de este proyecto, enumerados a continuación. Si bien estos no dan garantías de tiempo, con respecto a **RNF2** y **RNF3**, se espera que los cambios en las etapas del proceso baje los tiempos de operación lo suficiente para cumplir con estos requisitos.

1. Procesar los datos geográficos de las ciudades (calles) y separarlos en los segmentos que conforman las cuadras y los polígonos que representan las manzanas.
2. Importar al sistema los datos que se quieren visualizar, representando cada dato como un punto con ubicación geográfica, y asignar a cada dato el segmento y manzana al que pertenece.
3. Generar mapas enriquecidos para su posterior visualización.

En las siguientes subsecciones se explica la solución propuesta con este nuevo proceso, dividida en las distintas componentes que forman el sistema: el modelo de datos, el detalle de los procesos que ejecutará el sistema, y las distintas interfaces web.

3.2.1. Modelo de datos

Si bien no existía un modelo de datos estándar en el proceso manual, surgió la necesidad de tener uno para soportar los procesos mencionados anteriormente. Se diseñó un modelo en que los elementos guardados no quedan sujetos a una visualización en particular, sino que son reusables y se pueden tener como base para todos los mapas que se generen.

La Figura 3.1 muestra las tablas principales de este modelo de datos; las líneas entre las tablas representan las relaciones entre entidades según los campos desde donde sale cada una; por ejemplo, el campo `id` en la tabla *Dataset* corresponde a la llave foránea `dataset_id` en la tabla *Punto*.

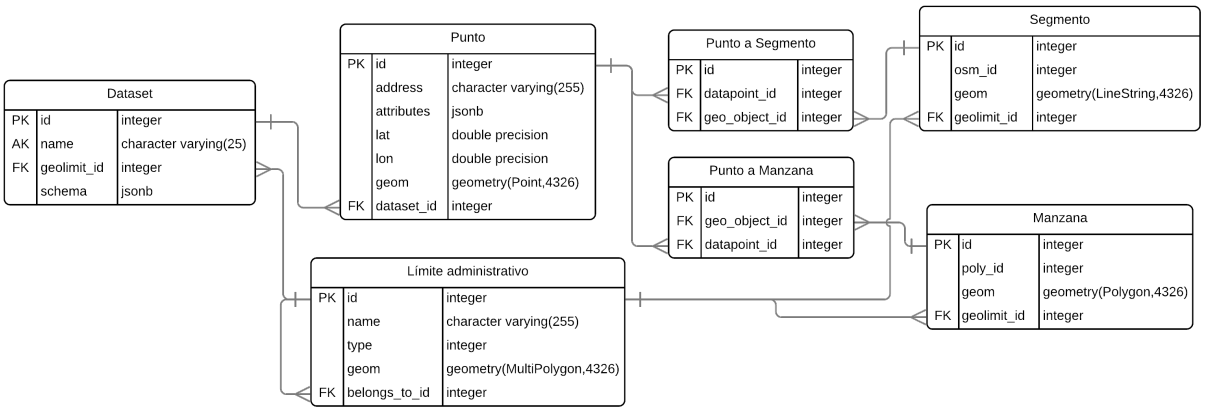


Figura 3.1: Tablas principales del modelo de datos para el diseño de la solución.

El tipo `geometry` de algunos atributos del modelo corresponde a tipos geométricos específicos de PostGIS, y están explicados en la Sección 2.1.2.

El modelo de la Figura 3.1 es una representación simplificada del modelo de datos, algunas tablas y campos no se muestran pues no son de gran importancia en el diseño de la solución, y existen para dar soporte al sistema en general. Además, por claridad, se le dieron nombres simples a las entidades en este esquema; el modelo completo con los nombres originales se encuentra en el anexo A de este documento.

En la siguiente lista se describen brevemente cada una de las entidades de la Figura 3.1:

Límite administrativo: Estos objetos son polígonos que representan el borde de las divisiones político-administrativas del país, pues cubren el área que encierran estos bordes. Se relacionan unos con otros en función de su pertenencia, como se ve en el campo `belongs_to_id` de la tabla. Por ejemplo, la comuna de Providencia pertenece a la provincia de Santiago, que a su vez pertenece a la Región Metropolitana.

Dataset: Representa un conjunto de puntos pertenecientes a un mismo archivo. Este objeto guarda también la división administrativa a la que pertenecen sus datos, para no calcular constantemente esta relación al consultar dentro de qué división administrativa se ubican los datos. En el caso del padrón electoral, un dataset correspondería al listado

de votantes de una comuna en particular. En el campo `schema` se guarda el nombre y tipo de los atributos extra del dataset. Por ejemplo, en el caso del padrón, un dato contendrá además de su ubicación, la circunscripción del votante, su número de mesa, entre otros.

Punto: Es una ubicación geográfica que pertenece a un dataset, representa un dato. Puede tener atributos extra, guardados en `attributes` como JSON, para mayor flexibilidad, según el tipo de dato al que corresponda.

Segmento: Es un trozo de calle entre dos intersecciones, representado como una polilínea (`geometry(LineString)`). Según su ubicación geográfica, pertenece a alguna división administrativa, guardando esta información como parte del objeto.

Punto a Segmento: En esta tabla se guarda la relación entre un punto y el segmento en el que se ubica, en forma separada tanto de puntos como segmentos. Así, cuando los puntos o los segmentos cambien y la información en esta tabla se modifique, la otra entidad no se ve afectada. Con esto el sistema cumple con **RF3**, ya que permite la reusabilidad de los segmentos.

Esto permite también asignar un número ilimitado de puntos a un segmento, sin importar el dataset al que pertenecen o sus parámetros, y de esta forma facilita la reutilización de los segmentos para cualquier mapa que se quiera generar.

Manzana: Es un polígono que representa un bloque o manzana. Al igual que los segmentos, tiene como atributo la división administrativa a la que pertenece.

Punto a manzana: Guarda la relación entre un punto y la manzana en la que se ubica, de manera análoga a *Punto a Segmento*. También permite la reusabilidad de las manzanas, pues no quedan asociadas de manera directa a los puntos de distintos datasets.

Con el modelo de datos definido de esta forma, disminuirá la cantidad de información duplicada al crear nuevas visualizaciones, ya que ahora los segmentos y manzanas no estarán sujetos al dataset con el que se trabaja, y los datos no dependen de el mapa base al que pertenecen. Considerando estas mejoras, los procesos definidos a continuación operan sobre el modelo ya presentado, haciendo uso en algunos casos de la flexibilidad de sus componentes.

3.2.2. Procesos

Después de un análisis más profundo de los procesos presentados al principio de esta sección, se puede ver que parte de los procesos 1 y 2 se puede ejecutar de forma paralela, pues el primero trabaja sobre los mapas base de las visualizaciones obtenidos de internet, y el segundo sobre los datos subidos por los usuarios. Considerando este paralelismo, estas operaciones fueron separadas en subprocesos, y se muestran en la Figura 3.2, donde se ilustra el proceso completo; los rectángulos representan las entradas y salidas del sistema, y los óvalos identifican cada operación. Todas estas acciones trabajan sobre el modelo de datos presentado en la subsección anterior.

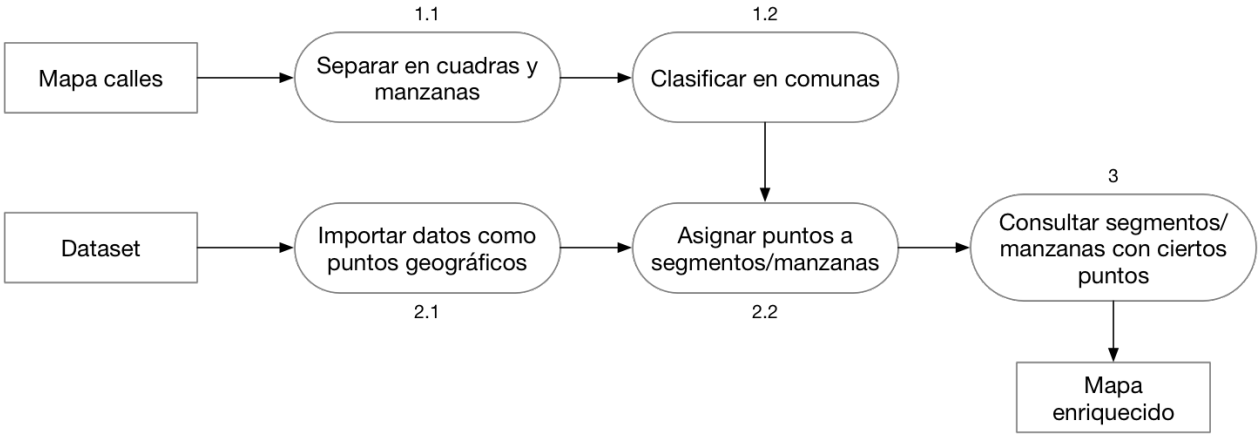


Figura 3.2: Diagrama de procesos del sistema.

A continuación se explica cada proceso en el orden general en el que se llevan a cabo.

Subproceso 1.1: Separar en cuadras y manzanas

Este proceso toma como entrada un archivo con la representación de las calles que se usaran como base en el sistema, obtenido de algún sitio web como OpenStreetMap; por ejemplo, las calles de Chile representadas como polilíneas. Las calles son separadas en los segmentos que corresponden a cada cuadra, y luego a partir de estos se identifican los polígonos que forman, es decir, las manzanas. Estos datos se guardan en las tablas *Segmento* y *Manzana*, según corresponda.

Este operación solo debe llevarse a cabo cuando se necesite actualizar el mapa base que usa el sistema, lo que podría significar que se ejecute cada 6 meses, en lugar de cada vez que se necesita generar un mapa como ocurre en el proceso manual. Además, como se identifican manzanas y segmentos para todo el mapa (en este caso Chile), cuando se necesite generar mapas para distintas comunas no será necesario correr este paso de nuevo, a diferencia del proceso manual, que como se vio en la Sección 2.2, requería obtener las calles para cada visualización a generar.

Subproceso 1.2: Clasificar en comunas

Para mejorar la velocidad de las consultas se decidió clasificar los segmentos y manzanas generados en el paso anterior, según la división administrativa más pequeña a la que pertenecen, que en el caso de Chile corresponde a la comuna. Así, cuando se consulte los datos de una comuna, provincia o región se pueden descartar rápidamente los segmentos o manzanas que no pertenecen a este. Al igual que el proceso anterior, esta clasificación debe hacerse sólo cuando se actualice el mapa base del sistema.

Subproceso 2.1: Importar datos como puntos geográficos

Toma como entrada un archivo con los datos que se desea visualizar en el sistema. Los datos deben tener ubicación geográfica (latitud y longitud) y opcionalmente pueden tener otros atributos asociados; en el caso del padrón electoral, los atributos serán todos los datos que son parte de este, como la circunscripción electoral, el número y tipo de mesa, etc.

Todos los datos de un archivo se agrupan bajo un mismo dataset, el que a su vez está asociado a una división administrativa, por ejemplo, los votantes de la provincia de Santiago; esta información también se guarda en el sistema al momento de importar los datos.

Con esto se cumple con **HG1**. Además, este proceso se ejecuta solo una vez para cada dataset, y luego esta información no vuelve a modificarse.

Subproceso 2.2: Asignar puntos a segmentos y/o manzanas

Una vez que los segmentos y manzanas fueron clasificados en comunas, y los datos de un dataset fueron importados, se puede asignar a cada punto el segmento o manzana en que se ubica. Es decir, para un punto geográfico dado, identificar el segmento de cuadra que le es más cercano, o la manzana que lo contiene, según la visualización que se desea generar. Es posible asignar solamente las manzanas si no se tiene interés en la visualización de segmentos.

Como se ve en la Figura 3.3, al punto indicado en rojo se le asigna el segmento del mismo color, pues es el más cercano. En el caso de las manzanas, a cada punto se le asigna la manzana dentro de la que se encuentra.

En la Figura 3.3 se ven algunos puntos sobre las líneas dibujadas; esto es solo por el tamaño del dibujo, pues los puntos geográficos no intersectan las líneas. Las direcciones de los puntos siempre se ubican a un costado u otro de las calles, según su numeración, por lo que siempre será posible clasificar los puntos en las distintas manzanas.

Tal y como en el proceso anterior, esta asignación se lleva a cabo solo una vez luego de importar un dataset. Sin embargo, debe repetirse al cambiar el mapa base, pues los segmentos o manzanas de referencia podrían haber cambiado.

Subproceso 3: Consultar segmentos y/o manzanas con ciertos puntos

Debido al diseño de los procesos anteriores, para obtener el mapa de los datos a visualizar no se requiere realizar nuevos cálculos o clasificaciones, si no que basta con hacer consultas a la base de datos para obtener la información. Estas consultas se construyen a partir de la comuna que se pretende visualizar, y las características propias del dataset elegido, es decir, el schema guardado al importar estos datos.

Llevando esta consulta al padrón electoral, esto significa que si se quiere visualizar, por ejemplo, el número de hombres que vota en la comuna de Santiago, se deben consultar los

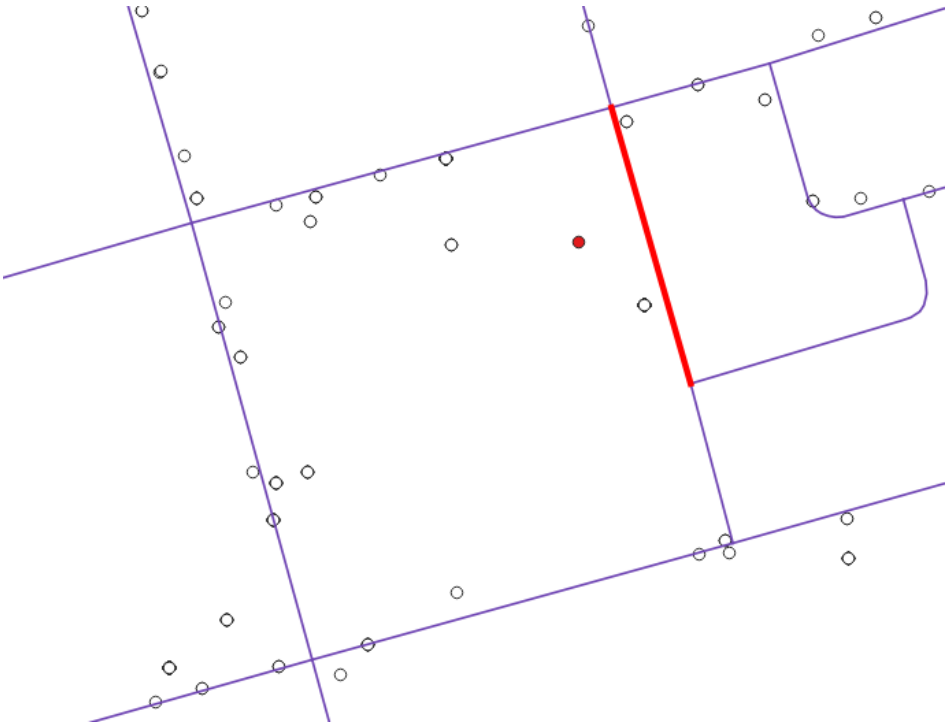


Figura 3.3: Asignación punto-segmento.

puntos que cumplen esta condición, es decir, son hombres votantes de esa comuna, y luego visualizar los segmentos o manzanas con esta información.

Con este proceso se cumplen **HG2** y **HG3**, pues el usuario puede generar las visualizaciones con estas consultas. Por otro lado, dado que no se deben hacer nuevos cálculos, las consultas para generar las distintas visualizaciones es rápida, y se pueden generar varias distintas sin demasiado esfuerzo, en poco tiempo.

La propuesta de procesos expuesta en esta sección permite que el sistema cumpla tanto con las historias de usuario como los requisitos, si bien no da garantías de tiempo en el caso de los requisitos no funcionales. Estas operaciones, junto con el modelo de datos, representan la base del prototipo a desarrollar, a lo que se debe agregar la interfaz para el usuario. Esta se separa en una API del sistema e interfaces web, y se explican a continuación.

3.2.3. API e Interfaz web

Se diseñaron distintos endpoints, direcciones web que entregan o reciben contenido, que son expuestos por el sistema como una API, y permiten interactuar con el sistema e iniciar los distintos procesos. Este diseño sigue la estructura general de las API diseñadas en Unholster, para un mejor entendimiento de sus funcionalidades dentro de la empresa.

Algunos de los endpoints presentados exponen información del sistema, como los datasets subidos, y otros permiten iniciar alguno de los procesos mencionados anteriormente. También se diseñó una aplicación web que implementa la API desarrollada, y presenta a los usuarios

una interfaz para manejar el sistema. Así se cumple con los requisitos **RF1** y **RF2**.

Endpoints del sistema

A continuación se presentan algunos de los endpoints de la API más relevantes para un usuario. Se diseñaron otros de carácter informativo, que solo exponen la información que se encuentra en la base de datos sin realizar cálculos adicionales, como los datasets que se han subido y su estado. Estos se encuentran detallados en el anexo B de este documento.

1. **Subir un archivo:** Permite al usuario subir un archivo con el dataset que se usará en las visualizaciones.
2. **Importar un archivo:** Inicia el importado de el dataset de un archivo subido previamente al sistema a la base de datos, como puntos geográficos.
3. **Asignar segmentos:** Inicia la asignación de puntos a sus segmentos correspondientes, para un dataset particular.
4. **Asignar manzanas:** Inicia la asignación de puntos a las manzanas que los contienen, para un dataset particular.
5. **Obtener mapa:** Permite al usuario obtener la visualización del mapa enriquecido con el dataset subido, de acuerdo a los filtros determinados.

Cada uno de los endpoints se relaciona directamente con alguno de los procesos explicados anteriormente:

- Los endpoints 1 y 2 conforman el proceso 2.1, importar datos como puntos geográficos.
- Los endpoints 3 y 4 inician el proceso 2.2, asignar puntos a segmentos y/o manzanas, respectivamente.
- El endpoint 5 obtiene la visualización del mapa generada con el proceso 3, consultar segmentos y/o manzanas con ciertos puntos.

Interfaz web

Se crearon mockups para validar, junto con el Director de Operaciones de Unholster, el aspecto de la interfaz web, las acciones que los usuarios podrían realizar, y la información que pondrá a disposición de estos.

En esta sección, se presentan los mockups de dos de las principales interfaces del sistema: *Carga de datasets* (ver Figura 3.4) y *Mapas* (ver Figuras 3.5 y 3.6), que permiten al usuario subir archivos con datos a visualizar, y generar las visualizaciones de los mapas, respectivamente. El resto no se muestran en este informe pues son puramente informativas y no agregan interacciones al usuario.

En la Figura 3.4 se muestra la vista *Carga de datasets*, donde los usuarios pueden ver el estado de los datasets subidos previamente al sistema, e información asociada a ellos, como la fecha en que fueron subidos, el número de datos que contienen, el estado de la operación

de importar el archivo, entre otros. También permite subir nuevos datasets, en un cuadro de diálogo que aparece al presionar el botón “Cargar nuevo dataset”.

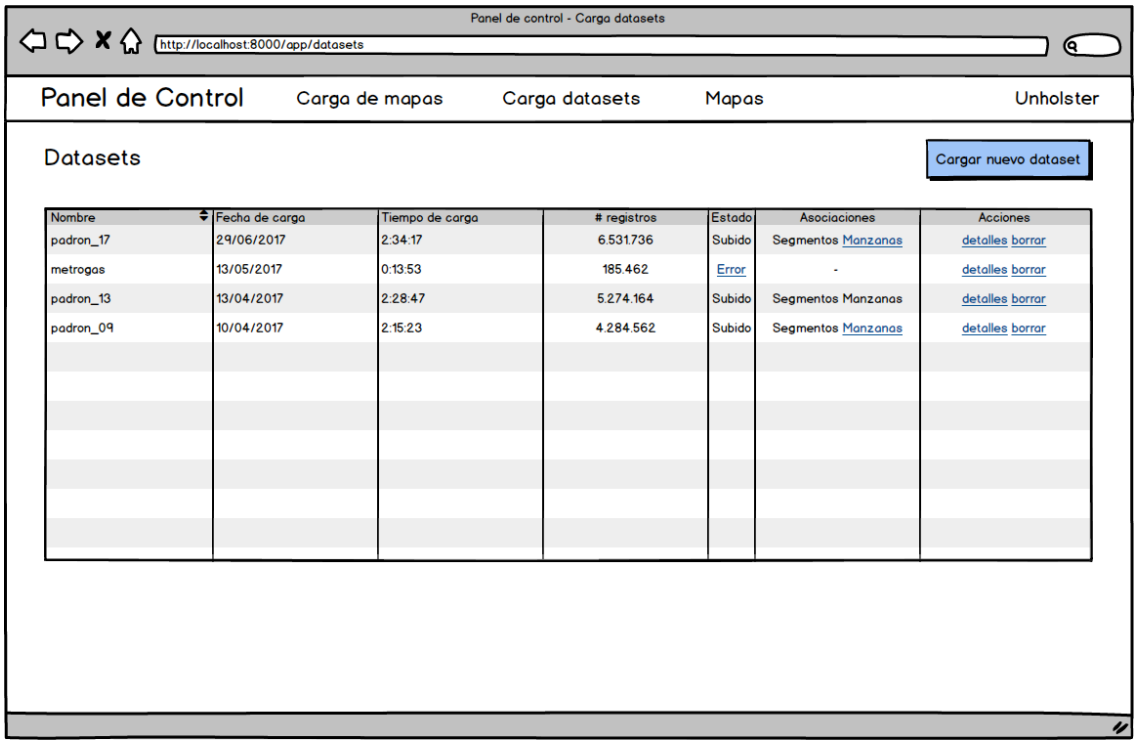


Figura 3.4: Vista *Carga de datasets*.

Adicionalmente, para cada dataset se permite al usuario iniciar el proceso de asignación de segmentos o manzanas, seleccionando el enlace correspondiente en la columna “Asociaciones”. Una vez que el usuario inicia un proceso se deshabilita esta opción, impidiendo que una asignación pueda ser iniciada múltiples veces.

También se diseñó una vista llamada *Carga de mapas*, similar en lógica a la vista *Carga datasets*, con la diferencia de que se listan los distintos mapas base cargados en lugar de los datasets. Sin embargo, se decidió pronto en el desarrollo que esta vista no sería implementada, pues se consideró de baja prioridad dado que representa los procesos 1.1 y 1.2, que no se ejecutarán constantemente, como se discutió anteriormente.

Por último, la vista *Mapas* se muestra en las Figuras 3.5 y 3.6, según se genera la visualización basada en segmentos o manzanas. Los usuarios pueden seleccionar el tipo de visualización a generar, el dataset en el que se basará el sistema, la comuna de interés y opcionalmente, parámetros extra con los que filtrar los datos.

Los colores usados en las visualizaciones de mapas son ilustrativos de lo que se espera ver, teniendo en cuenta que la interfaz final deberá mostrar los datos en una escala de colores, según la visualización generada

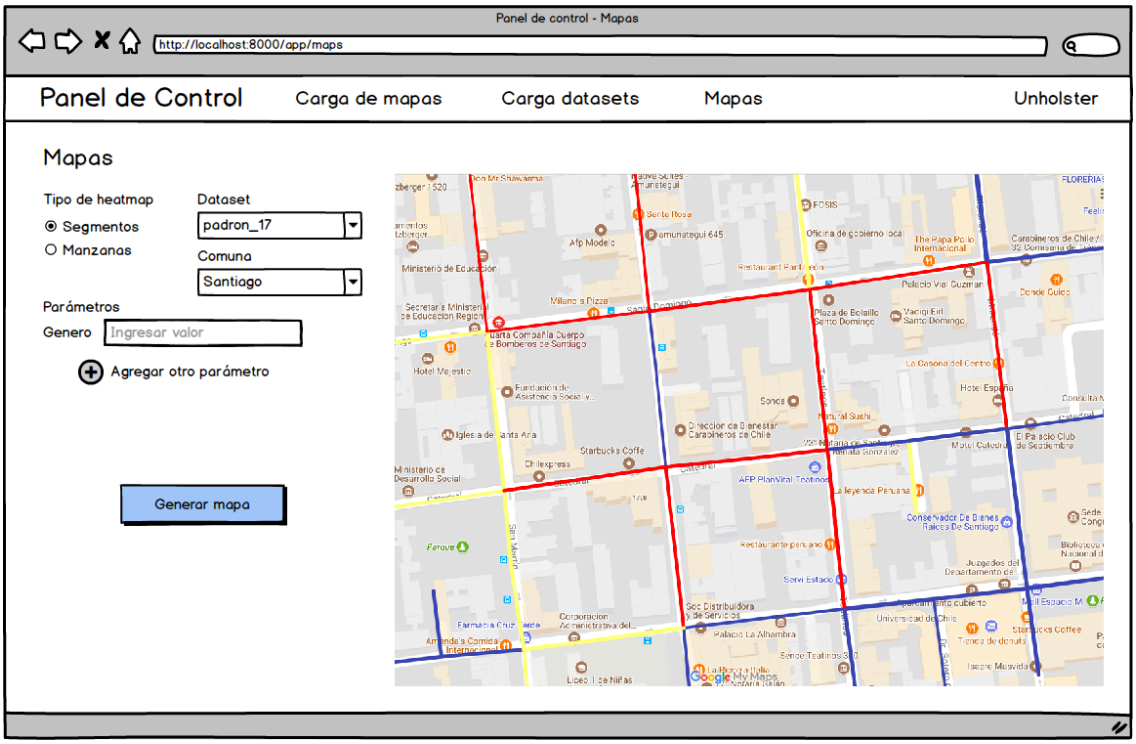


Figura 3.5: Vista *Mapa* con visualización basada en segmentos.

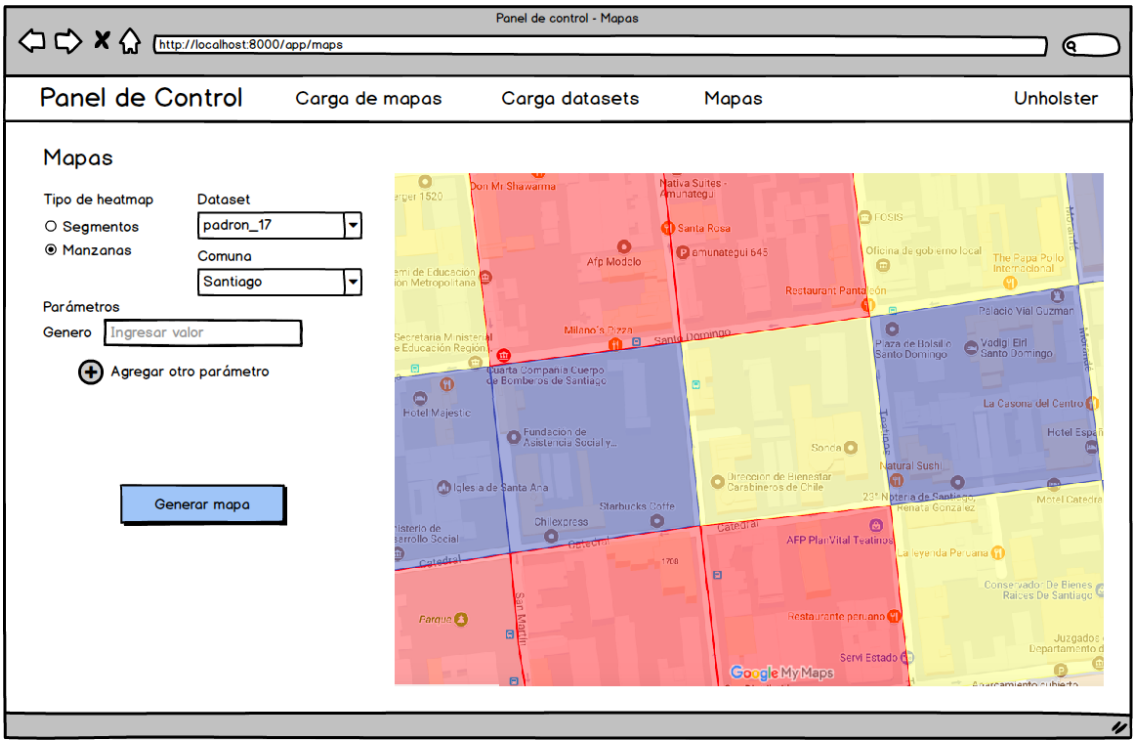


Figura 3.6: Vista *Mapa* con visualización basada en manzanas.

3.3. Mejoras al proceso de asignación

Posterior a la realización de este trabajo de título, surgieron nuevas formas de mejorar el proceso de asignación de manzanas y segmentos, que se explican a continuación. En conjunto, estas mejoras permiten generalizar la solución presentada a datasets que no indican la división administrativa a la que pertenecen, permitiendo una mayor libertad a los usuarios al momento de subir datos al sistema.

Estas mejoras, propuestas por los profesores miembros de la comisión, se mencionan como trabajo futuro sobre la solución propuesta, pues ayudarán de forma importante a reducir el tiempo de procesamiento de los datos subidos.

3.3.1. Jerarquización del territorio

Si bien en la solución presentada se tiene en cuenta las relaciones de pertenencia de las distintas divisiones (manzanas pertenecen a comunas, comunas pertenecen a provincias, etc.), no es fácil recorrerlas desde un territorio mayor a uno menor. Si se cambia la dirección de estas relaciones, se obtiene una estructura de árbol como en la Figura 3.7, que simplifica el recorrido hasta una manzana.

Teniendo un punto cualquiera, que no tiene una comuna asociada, encontrar la manzana a la que pertenece bajo la solución presentada tomará 160.140 operaciones, el número total de manzanas. Con la estructura jerarquizada, y sabiendo que en promedio cada región tiene 4 provincias, cada provincia 7 comunas, y cada comuna 464 manzanas, el número de operaciones necesarias para ubicar el punto baja a 490, en promedio. Esto es mucho menor al total de manzanas, y hacer la consulta de esta forma reduce en forma importante el tiempo requerido para ubicar un punto cualquiera, haciendo posible la aplicación de este sistema a un problema generalizado.

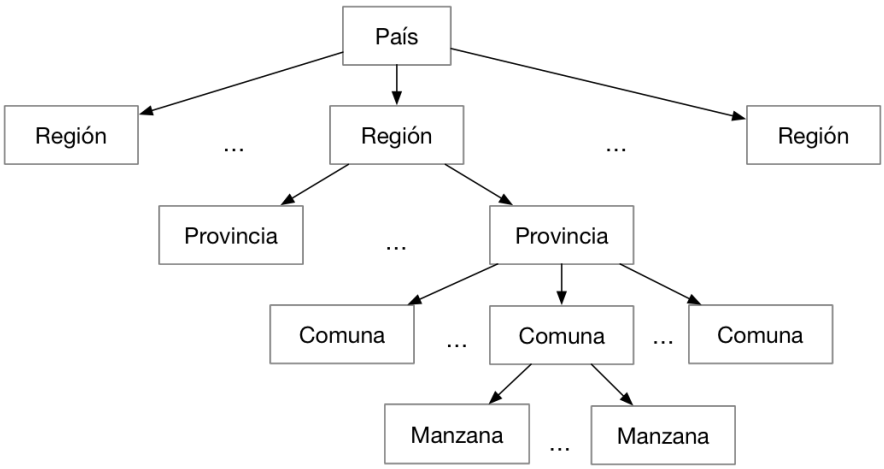


Figura 3.7: Árbol de territorios jerarquizados.

La búsqueda entre las manzanas de una comuna puede acelerarse al consultarlas de manera ordenada usando la técnica "sweep line": se ordenan las manzanas según una de sus

coordenadas geográficas, por ejemplo su latitud, y luego se hace un barrido en la misma dirección. En el ejemplo propuesto, el barrido se puede hacer en dirección norte-sur hasta encontrar una grupo de manzanas cuyo borde inferior tiene una latitud mayor que la del punto buscado, por lo que se deben consultar solo las que se encuentran en dicha franja. Este proceso puede acelerarse aún más haciendo una búsqueda binaria sobre las franjas de manzanas.

3.3.2. Restricción de segmentos posibles

En lugar de asignar los segmentos tomando en cuenta todos los posibles dentro de un territorio acotado, es posible restringir los segmentos posibles a sólo aquellos que forman el borde de la manzana que contiene a un punto.

El proceso de asignación de manzanas es más rápido que el de asignación de segmentos debido al menos número de las primeras, por lo que correr ese proceso como paso previo no constituye un aumento en el tiempo. Luego para cada punto, sus posibles segmentos más cercanos serán, en la mayoría de los casos solo 4, que conforman el borde de la manzana que lo contiene. Esta consulta será mucho más rápida que consultar por todos los segmentos de una comuna, como en la solución propuesta.

3.3.3. Paralelización de asignaciones

Dado que cada punto existe en forma independiente uno de otro, las asignaciones para cada uno pueden ejecutarse en forma completamente independiente, incluso en procesos separados dentro del sistema. Esto permitiría paralelizar los procesos de asignación de puntos y manzanas, según el número de recursos que se tenga disponible.

Si bien el desarrollo requerido para esto podría no ser simple, el tiempo total requerido para completar las asignaciones podría bajar al orden de segundos, superando con creces las expectativas iniciales de este trabajo de título.

3.4. Resumen

Con las historias de usuario y requisitos validados por el cliente, se presentó un diseño del sistema a desarrollar que cumple con las características pedidas; en particular esta propuesta considera un modelo estandarizado para todos los datasets que se quiera visualizar, y permite reutilizar la información disponible en distintas visualizaciones de una misma área, a diferencia del proceso manual.

También se presentaron interfaces de aplicación y web del prototipo, que permitirán a los usuarios hacer uso del sistema propuesto. Finalmente, se presentaron algunas ideas que

surgieron de forma posterior a este trabajo de título, que representan mejoras a la solución planteada y quedan como trabajo futuro de esta memoria.

En los capítulos siguientes se discuten las decisiones tomadas en la implementación de esta propuesta, los resultados obtenidos, y la validación del sistema según los objetivos específicos de esta memoria.

Capítulo 4

Implementación

Tomando en consideración las decisiones y el diseño propuesto en el Capítulo 3, se describe a continuación cómo se realizó la implementación de los procesos descritos, y las tecnologías usadas para el desarrollo del sistema en general. Primero, en la Sección 4.1, se revisan los aspectos generales del sistema, las tecnologías utilizadas, la metodología de trabajo que se siguió, y los distintos componentes del prototipo desarrollado. Luego en la Sección 4.2 se discute en detalle la implementación de los procesos presentados en el Capítulo 3, junto con las decisiones tomadas y los problemas que debieron ser solucionados.

4.1. Aspectos generales

Para el desarrollo del sistema se siguió un modelo de desarrollo incremental, y se trabajó en forma paralela tanto en el backend como la interfaz web del prototipo. Esto permitió que la interfaz web se mantuviera como una prioridad, y se le dedicara tiempo suficiente a su desarrollo. Al finalizar cada nueva característica del sistema se presentaban los avances al Director de Tecnología de Unholster, encargado del proyecto de parte del cliente, para revisar el grado de avance y algunas de las decisiones tomadas.

La estructura del prototipo se explica en cada una de las subsecciones siguientes, según un modelo de tres capas típico de una aplicación web, donde se tiene una capa de datos, de lógica y una de presentación.

4.1.1. Datos

Se eligió PostgreSQL 10 como sistema administrador de bases de datos, porque es una tecnología de código abierto y tiene buen soporte para datos geográficos, tanto para su almacenamiento como operación, a través de la extensión PostGIS (versión 2.4), como fue mencionado en la Sección 2.1.2.

4.1.2. Lógica

Para implementar todas las operaciones del sistema detalladas en este trabajo, se decidió usar Python y Django. Esta decisión se fundamentó en dos ámbitos importantes, por un lado las tecnologías que se usan en Unholster, y por otro las necesidades del proyecto.

La mayoría de los proyectos en Unholster se basan en Python, y aquellos que requieren una aplicación web implementan también Django, por lo que la mayoría de los desarrolladores son versados en el lenguaje y sus librerías. Esto es importante, pues se espera que este trabajo continúe como un proyecto dentro de la empresa, que deberá ser mantenido y extendido; al ceñirse a las tecnologías en las que se especializa Unholster, este proceso será más simple.

Elegir estas tecnologías también se justifica en que se requería una API web para exponer los endpoints de las operaciones, lo que se logra de forma simple usando dos librerías para Django: GeoDjango, que extiende el mapeo objeto-relacional de Django a objetos geográficos, como se vio en la Sección 2.1.2, y Django REST, que permite construir APIs web de forma rápida y reusable a partir de objetos de Django.

Dado el tiempo que toman ciertas operaciones, como importar un archivo a la base de datos o asignar puntos a sus segmentos o manzanas, ejecutarlas dentro del thread principal de la aplicación que expone la API dejaría al usuario esperando una respuesta por tanto tiempo como demoren los procesos. Para evitar este problema se usa Celery [23], que permite ejecutar tareas de forma asíncrona en procesos separados. Así, cuando un usuario inicia un proceso, recibirá una respuesta del sistema sobre el inicio de este, y en un proceso distinto, manejado por Celery, se ejecutará.

Si bien el sistema de usuarios no se encuentra contemplado dentro de el prototipo a desarrollar, dado que se usa Django como tecnología base del proyecto, será poco esfuerzo agregar esta característica, ya que está integrada por defecto en el sistema Django.

4.1.3. Presentación

Se implementó una interfaz web separada del sistema principal, que permita a los usuarios hacer uso del sistema de una forma amigable desde un navegador. La aplicación esta implementada con ReactJS, y consume la API del sistema principal para funcionar.

Esta aplicación muestra el mapa que se quiere visualizar; para ello usa la librería Leaflet [1], y dibuja el mapa obtenido desde el sistema como una capa vectorial. Para ilustrar las distintas cantidades de puntos en cada segmento o manzana, estas capas se visualizan utilizando una escala de colores; la clasificación de cada elemento se hace usando el método Jenks [9], que genera intervalos basados en las diferencias naturales inherentes a los datos. Estos minimizan la diferencia de los elementos que pertenecen a intervalo, y maximiza las diferencias entre los distintos rangos. Esta clasificación es específica a los datos y no sirven para comparar mapas creados a partir de información distinta.

En la elección de los colores para los mapas se usaron los sitios ColorBrewer [7] y Viz

Palette [24], que permiten generar escalas de colores para visualizaciones, teniendo en cuenta el número de categorías, el tipo de datos, el color de fondo, entre otros.

4.2. Procesos

Los procesos presentados en el Capítulo 3 están implementados en su totalidad en la capa de Lógica, y se explican en detalle a continuación, siguiendo el mismo orden en que se explican en la Sección 3.2.2.

4.2.1. Separar en cuadras y manzanas

El mapa de base usado se obtiene del sitio Geofabrik [12], donde diariamente se suben archivos con todos los datos de OpenStreetMap respecto al área pedida. El mapa usado corresponde a todo el territorio chileno, e incluye información como las distintas calles y avenidas que existen, edificios, transporte público, entre otros.

Para importar el mapa a la base de datos, inicialmente se usó la herramienta utilizada en el proceso manual, osm2pgrouting; sin embargo esta herramienta sólo funciona para áreas pequeñas, como una comuna, al intentar importar las calles de todo el país la herramienta usa toda la memoria disponible en el equipo y termina su ejecución en error.

Luego de una pequeña investigación sobre las herramientas disponibles, se probó con Imposm 3 [25], que importa distintos datos de OpenStreetMap a bases de datos PostgreSQL. Este programa es altamente configurable, y permite elegir el tipo de datos que se importarán, como calles, establecimientos, líneas de transporte público, etc. Sin embargo, Imposm3 importa las calles como polilíneas completas en la base de datos, lo que después requiere de una consulta manual para separar cada calle en los segmentos correspondientes, con el alto costo que esto significa.

Finalmente, se decidió usar Osm2po [20], que a diferencia de osm2pgrouting e Imposm3, permite manejar archivos de gran tamaño, incluso la copia completa de la información de OpenStreetMap del mundo entero, demorando alrededor de dos minutos en generar e importar a la base de datos todos los segmentos, en el caso de Chile. Es importante notar que esta herramienta no considera las calles exclusivas de peatones, y sólo trabaja con aquellas por las que pueden circular vehículos.

En cuanto a las manzanas, estas se generan en la base de datos sin usar herramientas externas, con una consulta que demora alrededor de 50 minutos para todo el territorio chileno. Estas se forman generando los polígonos más pequeños que forman las intersecciones de segmentos.

Así, el proceso de importar los mapas desde que se descarga el archivo base hasta que se tienen los segmentos y manzanas en la base de datos toma alrededor de una hora. Considerando que los mapas no cambian constantemente, el tiempo que demora importarlos es

despreciable con respecto al tiempo de vigencia de los mismos, que puede ser de unos seis meses.

Para finalizar, si bien estos procesos no están automatizados dentro del sistema, al ser operaciones simples su integración será rápida, y no se hizo dentro del trabajo de esta memoria por la baja prioridad de este proceso, ya que solo fue necesario ejecutarlo una vez para obtener la base del resto del sistema.

4.2.2. Clasificar en comunas

Al principio del desarrollo de este trabajo, no se consideraba incluir las divisiones político-administrativas del país. Sin embargo, al hacer pruebas del tiempo que tomaba el proceso de asignar puntos a segmentos y manzanas, se hizo necesario limitar el área involucrada, pues mientras más segmentos o manzanas se consideran, más tiempo demora la asignación, llegando a tardar más de 4 horas la asignación de puntos para la Región Metropolitana.

Se decidió entonces agregar los límites administrativos como polígonos en la base de datos, y se diseñó una consulta para asignar a cada segmento y manzana la comuna a la que corresponde. Los datos se obtuvieron del sitio de la Biblioteca del Congreso Nacional [26]; se eligió este sitio pues se considera como una fuente oficial de información, a diferencia de otros como OpenStreetMap. Si las comunas, provincias o regiones cambian, estos datos deberán ser actualizados en el sistema. Se eligió la comuna como unidad de clasificación porque es la unidad más pequeña, y si se quiere saber la provincia o región en que se encuentra un segmento, basta con seguir la relación de pertenencia de la comuna.

La clasificación se hace de la misma forma tanto para segmentos como para manzanas, respetando los nombres de las distintas tablas: para cada segmento, se compara con todas las comunas disponibles y se le asigna aquella que lo contiene geográficamente como una llave foránea. En el caso particular de los segmentos, se toma como referencia el punto medio de la polilínea para decidir en los casos en que cruzan dos comunas, principalmente porque la alternativa es tener casos especiales para cada límite comunal, que no siempre están bien definidos por las municipalidades y haría más compleja la asignación. Para las manzanas se toma como referencia el centro del polígono, por las mismas razones.

El proceso completo, considerando segmentos y manzanas, demora aproximadamente 3 horas. Igual que en el proceso anterior, esta operación solo ocurre una vez, al cambiar los mapas base cada cierta cantidad de meses.

4.2.3. Importar datos como puntos geográficos

Dado que, a diferencia del proceso manual, los datos a visualizar deben subirse al sistema, se acordó junto al Director de Operaciones de Unholster, un formato específico que deben cumplir los archivos que se suban: los archivos deben estar en formato `.tsv`, donde cada línea representa un dato, y contener 4 columnas: `address`, `lat`, `lon` y `attributes`.

address	lat	lon	attributes
Coquimbo 920, Santiago, Región Metropolitana, Chile	-33.4565573	-70.6471909	{"sexo": "V", "mesa_numero": 52, "distrito": 10, "mesa_tipo": "V", "circunscripcion": "Parque Almagro"}

Tabla 4.1: Ejemplo de la información en una línea del padrón electoral en el formato planteado.

Las primeras tres columnas `address`, `lat`, `lon` son obligatorias para todos los datos, y contienen la dirección, latitud y longitud de cada uno, respectivamente. La columna `attributes` contiene cualquier otro atributo que tengan los datos, como la circunscripción o el número de mesa, en el caso del padrón electoral. Estos atributos se describen en un objeto JSON, como se ve en la Tabla 4.1; si un dato no tiene atributos, tendrá un objeto vacío.

Este formato permite que todos los archivos sigan la misma estructura, independiente del tipo de datos que contienen, y los atributos que estos tengan. Además, queda en forma clara para los usuarios cuáles son los campos esperados por el sistema. Si para algunos usuarios este formato fuera complejo de generar, por no estar familiarizados con las estructuras JSON por ejemplo, se puede construir una herramienta que construya los archivos a partir de planillas Excel. Por último, el formato JSON de los atributos permite mantener la información de tipos de los datos, lo que posteriormente permite filtrar los datos aprovechando estas propiedades, por ejemplo, en un dato de tipo numero, se podrá filtrar por mayor o menor que, en vez de solo con igualdad como en caso de strings.

Para acelerar el proceso de asignación entre puntos y segmentos o manzanas, todos los puntos de un dataset deben pertenecer a la misma comuna, lo que debe ser indicado al momento de importarlos. Si bien es posible identificar la comuna a la que pertenece un punto de forma automática, sin que el usuario lo indique, esto podría tomar demasiado tiempo; en esta primera versión del sistema se decidió que el usuario explicite la comuna, en lugar de la región o provincia, para acelerar el proceso de importado lo más posible. Asimismo, si el usuario ingresa una comuna equivocada para los datos, el sistema no lo verifica y asignará los puntos de forma incorrecta. Corregir este comportamiento se consideró de baja prioridad para esta versión y se dejó como trabajo futuro.

Cada dato en la base de datos guarda su ubicación, la geometría que corresponde a sus coordenadas (en este caso un punto), y un campo atributos donde se almacena cualquier información extra que se quiera utilizar en las visualizaciones. En el caso del padrón, en este campo van datos como la circunscripción, el número de mesa, el tipo de mesa, etc., para cada votante.

El campo `atributos` corresponde a un campo de tipo JSON en la base de datos, como se ve en la Figura 4.1. Este campo permite que el modelo de datos sea lo suficientemente flexible como para permitir cualquier tipo de atributos para distintos datos, pues de otra forma esta información sería guardada en campos individuales, forzando la implementación del sistema para aceptar un solo tipo de dataset. Además, guardar los atributos en un campo

JSON permite usar las herramientas de búsqueda de Postgres dentro de cada objeto; por ejemplo, para un dataset con atributos como los de la Figura 4.1, es posible filtrar los datos en que `distrito` sea 10.

```
                attributes
-------------------------------------------
 {                                        +
     "sexo": "V",                         +
     "distrito": 10,                      +
     "mesa_tipo": "V",                    +
     "mesa_numero": 47,                   +
     "circunscripcion": "Parque Almagro"
 }
```

Figura 4.1: Ejemplo del campo `attributes` de un dato del padrón electoral en la base de datos.

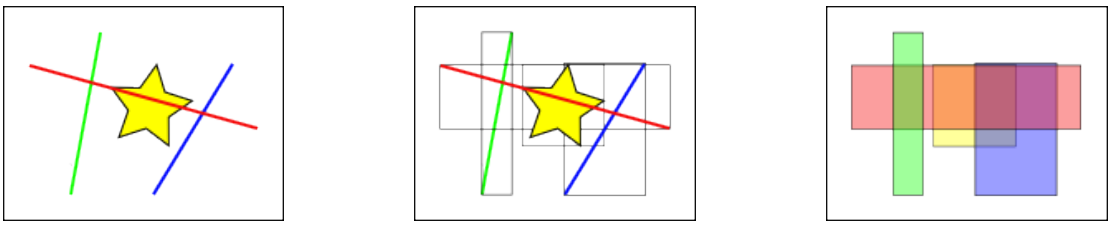
Dentro del dataset también se guarda un listado de todos los atributos disponibles para los datos que contiene, y el tipo de estos, en el campo `schema`. Así, cuando se necesite hacer consultas sobre los datos según dichos atributos, el sistema puede identificar qué campos en la consulta son válidos y cuáles no, y el tipo de comparación disponible: igualdad para strings, igualdad y comparaciones numéricas para enteros.

4.2.4. Asignar puntos a segmentos y/o manzanas

Como se vio en la Sección 2.2, en el proceso manual se asigna a cada punto su segmento más cercano. En la práctica, esto significa ejecutar una consulta que calcula la distancia entre un punto y los segmentos disponibles, para todos los puntos. Luego se ordenan los segmentos según esta distancia, y se escoge el de menor valor. Este es un proceso lento, pues el sistema ocupa tiempo de cómputo en cada distancia, y es uno de los pasos que consume más tiempo en el proceso manual, por lo que se buscó una mejor forma de realizar estas asignaciones.

El problema de encontrar los k elementos más cercanos a otro es conocido como *K-Nearest Neighbors* [22], su solución en general requiere mantener un índice de las cajas que encierran las geometrías a consultar, similar a lo que se ve en la Figura 4.2; usando este índice se puede consultar de forma rápida las distancias entre objetos. Sin embargo, dado que el índice actúa sobre las cajas y no los objetos en sí, es posible que la respuesta sea errónea; en el caso de la Figura 4.2, sólo la línea roja intersecta a la estrella, pero tanto la caja azul como la roja intersectan la caja amarilla.

Para obtener la respuesta final, se usa este índice de cajas para reducir el número de geometrías que se consultarán, y luego se calculan las distancias reales entre los objetos restantes. Por ejemplo, para encontrar el segmento más cercano a un punto, primero se usa el índice de cajas para elegir los 100 segmentos más cercanos, y para este subconjunto se calculan las distancias reales para obtener la distancia final.



(a) (b) (c)

Figura 4.2: Ejemplo de intersección de figuras. En (a) se ven las figuras originales, en (b) con sus bounding boxes o cajas y en (c) solo se ven las cajas.

Este enfoque de calcular los vecinos más cercanos en dos pasos está implementado en PostGIS desde la versión 2.2, en versiones anteriores sólo se hacía usando el índice, y para encontrar los verdaderos vecinos más cercanos se debían calcular las distancias por separado.

Así, para cada punto se ordenan los segmentos usando el operador de PostGIS que implementa este comportamiento [14], se elige el primero, y cada par punto-segmento se guarda en la tabla *Punto a Segmento*. Para el padrón electoral de la comuna de Santiago, esta operación demora aproximadamente 30 minutos; comunas con menos votantes como Recoleta demoran 15 minutos o menos.

Dado que en el proceso manual de generación de mapas no se incluían mapas basados en las manzanas, no existe una línea base para comparar los tiempos que toma asignar a cada punto su manzana correspondiente. En este caso, la consulta sólo revisa si los polígonos disponibles contienen al punto en cuestión, escogiendo aquel que cumple esta condición. Esta asignación toma considerablemente menos tiempo que la asignación punto-segmento, en parte por el número reducido de manzanas en relación a los segmentos de una comuna, y también porque la asignación de manzanas requiere solo una pasada por los datos, en lugar de dos como para los segmentos.

La restricción de limitar los segmentos, manzanas y puntos a aquellos pertenecientes a una comuna representa un aumento considerable en la velocidad de asignación con respecto a limitar según una provincia o región, pues estos casos las asignaciones demoran horas en lugar de minutos, como ocurre con las comunas. Otra ventaja de restringir las asignaciones esta unidad, es que en caso que se requiera procesar múltiples comunas, como en el caso de una región, las asignaciones podrán correrse en paralelo, según las distintas comunas, disminuyendo el tiempo para procesar la región desde horas a minutos.

4.2.5. Consultar segmentos/manzanas con ciertos puntos

Para obtener las visualizaciones con el número de puntos asociados a los segmentos o manzanas, se debe consultar por un dataset en particular, junto con el área a la que pertenece e indicando el tipo de mapa que se quiere obtener, usando los segmentos o las manzanas.

Tomando como ejemplo el padrón electoral para explicar de mejor manera este proceso, si se quiere generar un mapa de la comuna de Santiago, considerando solo los votantes de

la circunscripción "El Centro" de esta comuna, el sistema ejecuta la consulta en dos partes: primero, selecciona los puntos que cumplen con los filtros pedidos, y luego calcula, para cada segmento o manzana, cuántos de estos puntos tiene asignados.

Siguiendo el ejemplo para la primera parte, el sistema selecciona los puntos que pertenecen al dataset asociado a la comuna de Santiago, filtrados según los parámetros indicados; en este caso se seleccionan solo aquellos puntos cuya circunscripción es "El Centro". Los filtros usados deben corresponder a alguno de los atributos del dataset. El sistema valida que esto se cumpla consultando el esquema de atributos guardado en el campo `schema` creado al importar los datos. Si en la consulta se indican filtros inválidos, el sistema responde con un error 400 — Bad Request.

Luego de esto, para todos los segmentos o manzanas correspondientes a la comuna de Santiago, según sea el caso, se calcula el número de puntos que tienen asignados, y que se encuentran en el conjunto filtrado en el paso anterior. La información de los segmentos o manzanas es enriquecida con el número de puntos contados, y finalmente se entrega esta información al usuario.

Un ejemplo de esta consulta se muestra a continuación, escrita como pseudocódigo y usando los nombres de tablas y campos vistos en la Sección 3.2.1. En esta caso, el dataset de la comuna de Santiago tiene `id = 4` y la comuna de Santiago, como división administrativa, tiene `id = 373`.

```
SELECT segmento,
       COUNT(punto_a_segmento.datapoint_id) FILTER(
           WHERE punto_a_segmento.datapoint_id IN (
               SELECT punto.id FROM punto
               WHERE punto.dataset_id = 4
                  AND (punto.attributes -> 'circunscripcion') = 'El Centro'
           )
       ) AS point_count
FROM segmento LEFT OUTER JOIN punto_a_segmento
    ON (segmento.id = punto_a_segmento.geo_object_id)
WHERE segmento.geolimit_id = 373
GROUP BY segmento.id
```

4.3. Resumen

En este capítulo se revisaron las decisiones tomadas en la implementación del prototipo propuesto por este trabajo de memoria, así como las tecnologías elegidas para cada capa del sistema.

Uno de los puntos más importantes discutidos en este capítulo es la separación de segmentos, manzanas y puntos en las comunas a las que pertenecen, lo que acelera en gran medida el proceso de asignación, y permite paralelizar el mismo para datasets de más de una comuna.

Además, el diseño de la base de datos presentado en el Capítulo 3 permite que las consultas, al generar las visualizaciones, sean rápidas y así se puedan tener distintos mapas en muy poco tiempo, filtrando de distintas formas los datasets subidos.

Capítulo 5

Validación

Para validar el trabajo realizado efectivamente representa una mejora con respecto a la situación inicial se realizaron distintas instancias de validación, tanto para la interfaz de usuario como para el proceso en sí. Primero, en la Sección 5.1 se muestran las vistas más relevantes de la interfaz web y su relación con los mockups presentados en el Capítulo 3. Luego, en la Sección 5.2, se explica el proceso usado para medir el tiempo que demoran las distintas operaciones del sistema, con respecto a los requisitos no funcionales de este proyecto, y los resultados obtenidos al hacer pruebas sobre ellas. Finalmente, en la Sección 5.3, se presenta la validación hecha con miembros de Unholster a los que se les demostró el sistema.

5.1. Interfaz Web

Las vistas desarrolladas contienen la información e interacciones necesarias para cumplir con los requerimientos del proyecto, dado que se trata de un prototipo. Ambas vistas se basan en los mockups presentados en la Sección 3.2.3, con ciertas diferencias en funciones, de acuerdo a lo implementado.

5.1.1. Administración de datasets

Esta vista permite a los usuarios subir nuevos datasets al sistema, obtener información sobre datasets ya subidos e iniciar los procesos de asignación de segmentos y manzanas para cada uno.

En la Figura 5.1 se muestra la pantalla principal de esta vista, donde se listan los distintos datasets ingresados al sistema. Para aquellos que ya fueron importados, se lista también el número de puntos que tienen asociados; en el caso de el dataset *renca*, este número no se muestra porque en ese momento el sistema aún está importando los datos.

La secuencia de la interfaz al iniciar la asignación de segmentos y manzanas se muestra

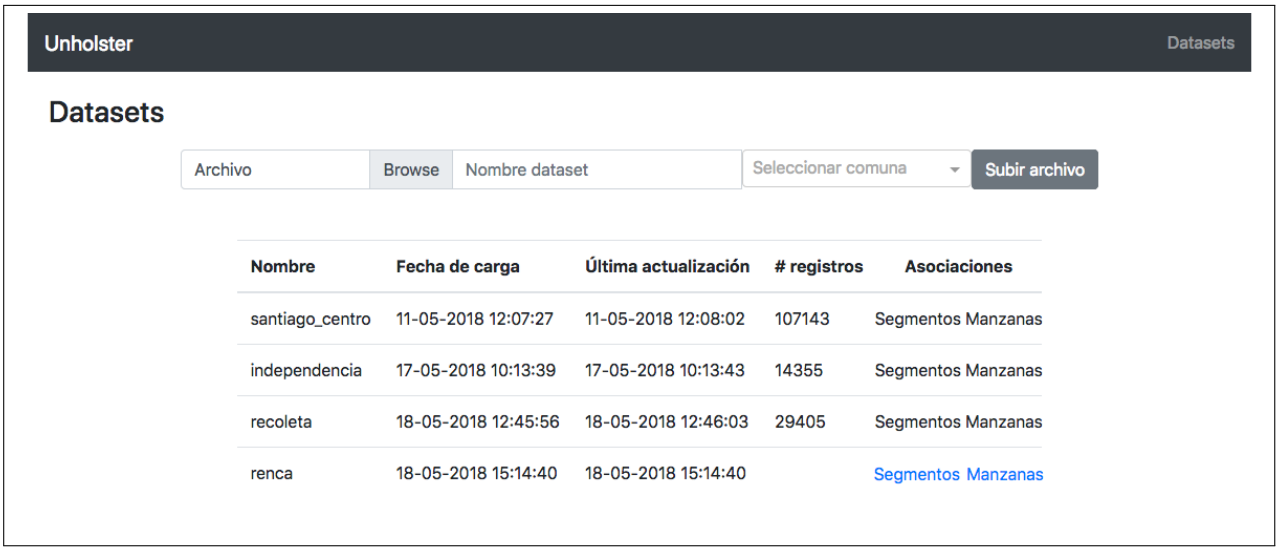


Figura 5.1: Vista datasets.

en la Figura 5.2: sólo se muestra una porción de la vista, porque el resto de la imagen no cambia. Como se ve en la Figura 5.2a, inicialmente el dataset *renca* tiene dos enlaces, *Segmentos* y *Manzanas*. Cuando el usuario selecciona uno de ellos, en este caso *Segmentos*, se inicia el proceso de asignación de segmentos en el sistema y el enlace cambia, mostrándose el texto *Procesando*, como en la Figura 5.2b. Esto previene que el usuario inicie el proceso de asignación múltiples veces al hacer clicks repetidos en el enlace.

Finalmente, cuando la asignación se completa, en lugar del enlace inicial se muestra *Segmentos* como texto plano, como se ve en la Figura 5.2c, indicando al usuario que el proceso fue realizado.

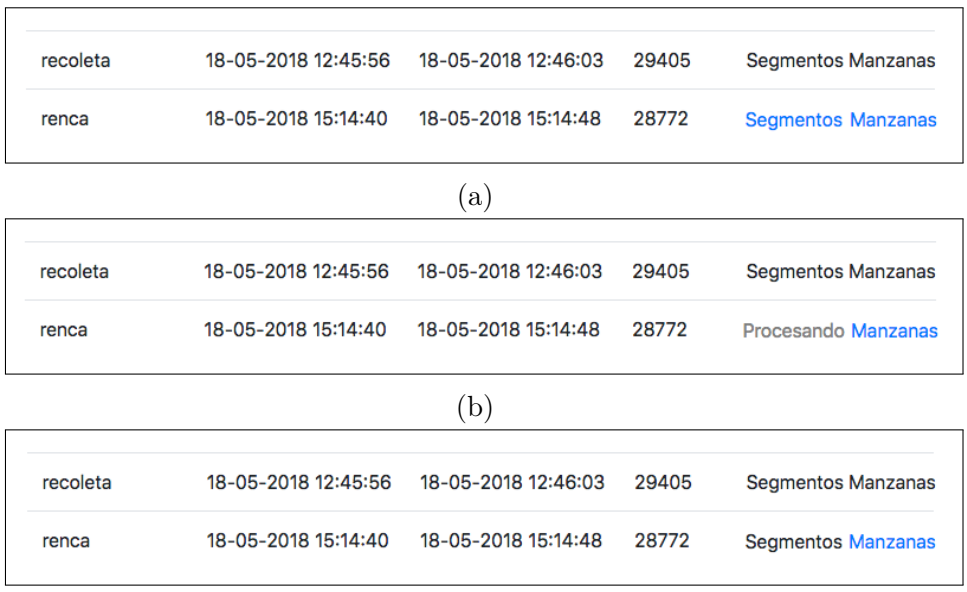


Figura 5.2: Secuencia proceso de asignación.

Esta vista se basa directamente en el mockup de la Figura 3.4 del Capítulo 3, con algunas diferencias respecto a la cantidad de información mostrada y acciones disponibles, pues no todas fueron implementadas en el prototipo desarrollado.

5.1.2. Generación de mapas

La vista principal de la aplicación corresponde a la de visualización de mapas. Aquí se permite al usuario seleccionar el dataset a usar, el tipo de visualización, basada en segmentos o manzanas, y los filtros que desee aplicar a los datos.

En la Figura 5.3 se muestra la visualización de segmentos generada al usar como dataset el padrón electoral de la comuna de Santiago, sin filtrar los datos. El mapa es navegable, y al seleccionar un segmento se muestra el número de puntos que tiene asociados, según los filtros establecidos.

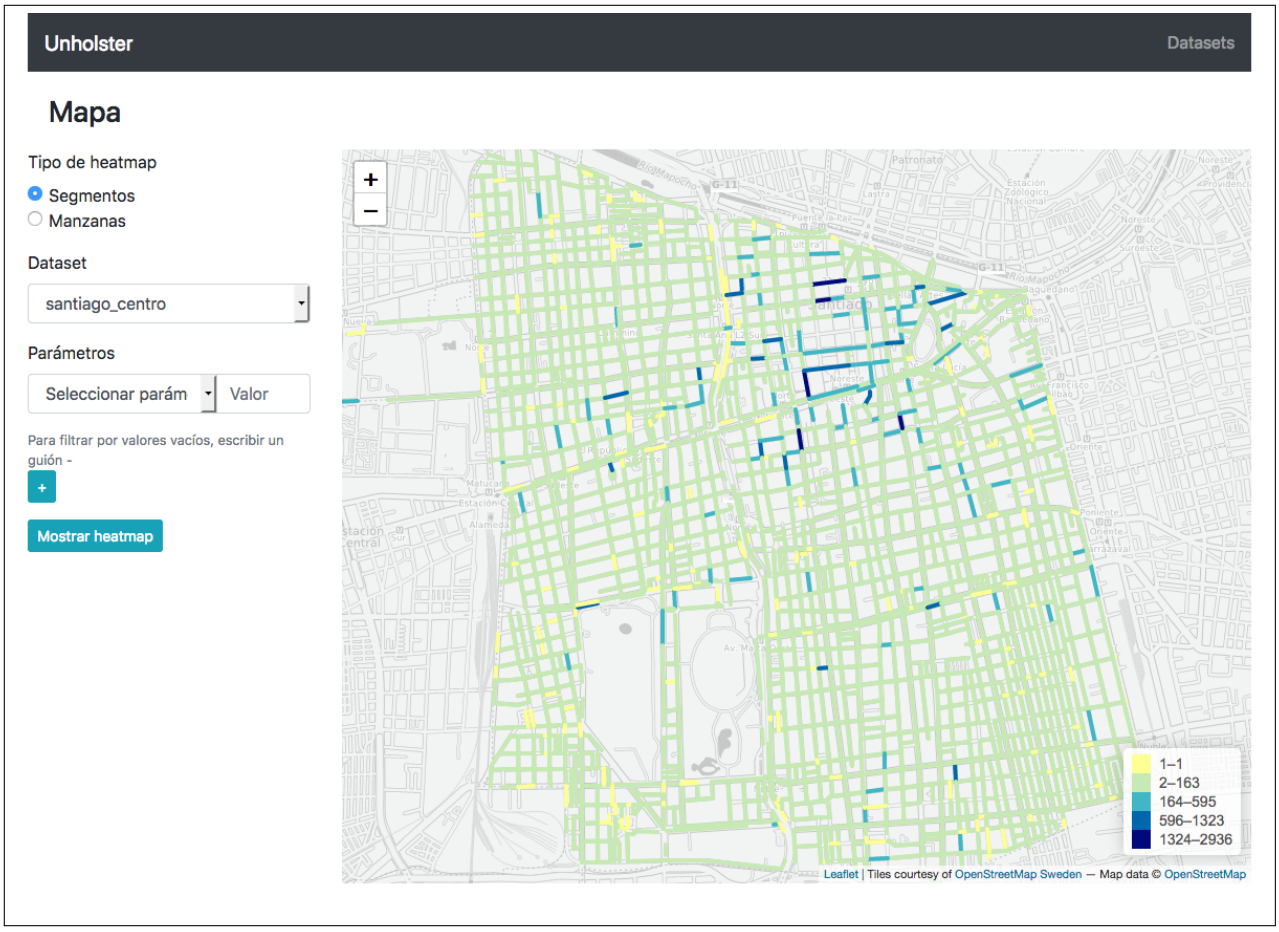


Figura 5.3: Visualización del padrón electoral como segmentos.

Para el mismo dataset, la visualización basada en manzanas se muestra en la Figura 5.4. Además para esta figura se aplicó un filtro sobre los datos, utilizando sólo aquellos cuya circunscripción es “El Centro”.

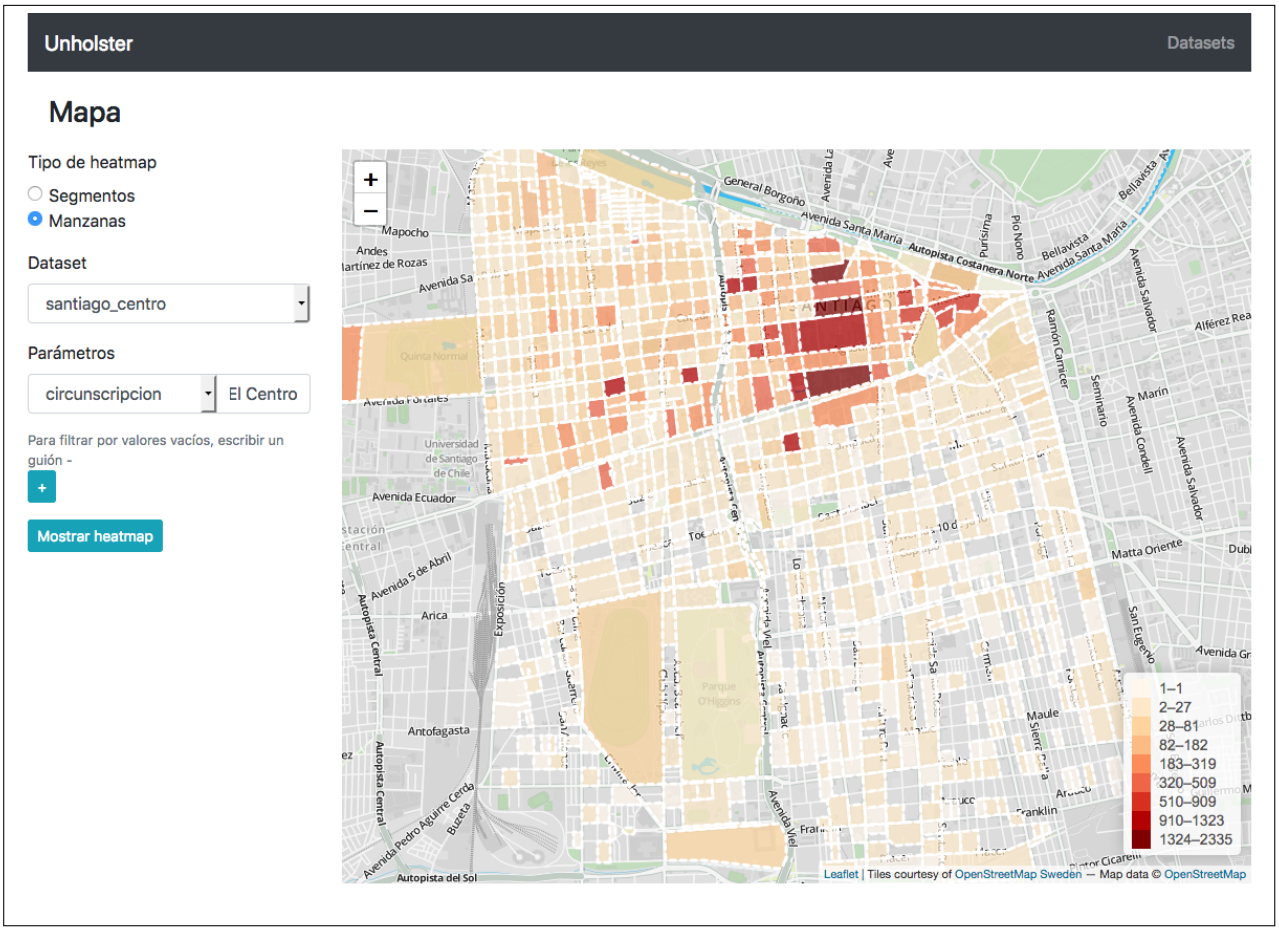


Figura 5.4: Visualización del padrón electoral como manzanas.

Como en el caso de la vista de datasets, las vistas de visualización de mapas se basan en los mockups de las Figuras 3.5 y 3.6 del Capítulo 3, con ciertas diferencias en la presentación, como la escala de colores usada y la leyenda en la esquina inferior derecha de los mapas.

5.2. Tiempos de ejecución

Durante la implementación de este sistema siempre se usaron como datasets de prueba los padrones de Santiago, Recoleta, Renca e Independencia. El primero se eligió principalmente por la complejidad de la comuna en cuanto al número de segmentos y manzanas, junto al alto número de votantes que residen en ella; así al medir el tiempo que demoran los procesos se podía tener la confianza de que representarían una cota superior para todo el resto de las comunas. El resto de los datasets se eligió porque ya habían sido geolocalizados y procesados anteriormente en Unholster, al hacer las visualizaciones manualmente. Si bien hay comunas con mayor población que Santiago, no tienen la misma complejidad de segmentos y por lo tanto la asignación punto-segmento será menos compleja.

Con el objetivo de comprobar las mejoras en el tiempo de los procesos, se probó el sistema tomando datos que hubiesen sido usados en el proceso manual para generar una visualización

Dataset	Nº puntos	Nº segmentos	Tiempo total [m]
Talca	17696	8795	7,8
Curicó	15721	6087	5,3
Linares	11255	2720	1,5

Tabla 5.1: Tiempo total que demora importar, asignar segmentos y asignar manzanas, para cada dataset.

específica. En particular, se usaron 3 comunas de la región del Maule: Curicó, Talca y Linares. Para cada una, se ejecutaron los procesos de importar datos, asignar segmentos y asignar manzanas, 10 veces. El tiempo total de estas operaciones se muestra en la Tabla 5.1. No se consideró el tiempo que demoró importar el mapa base y generar los segmentos y manzanas, pues esto ya se había realizado durante la implementación del proyecto y no era necesario repetirlo.

En el proceso manual, generar el mapa correspondiente a estas tres comunas a la vez demoraba entre 3 y 4 horas, alrededor de medio día de trabajo. Como se puede ver en la Tabla 5.1, el tiempo que demoran los procesos disminuyó considerablemente, y teniendo en cuenta que la generar la visualización solo demora unos pocos segundos al hacer la consulta, se puede afirmar que el trabajo realizado cumple con los requisitos **RNF2** y **RNF3**, de disminuir el tiempo que demora asignar los puntos a sus segmentos y manzanas, y en generar las visualizaciones.

Adicionalmente se probó correr el proceso de asignación punto a segmento, usando la consulta del proceso manual. Es decir, teniendo como base el nuevo modelo implementado, se calcularon todas las distancias entre un punto y los segmentos disponibles, sin hacer uso del índice de la base de datos. El tiempo que demoró esta operación fue de aproximadamente el doble que usando la consulta optimizada, es decir, el doble de los tiempos mostrados en la Tabla 5.1, que de todas maneras son menores a los que tomaba generar una visualización con el proceso manual.

5.3. Valoración Unholster

Finalmente, para validar la satisfacción del cliente con el proyecto, se hizo una presentación y demostración del sistema a los directores de Unholster, explicando los cambios realizados al proceso de generación de mapas, los resultados obtenidos y el funcionamiento del sistema desarrollado.

La respuesta en general fue muy positiva, y se mostraron satisfechos con el sistema desarrollado; todos los comentarios emitidos en esa ocasión apuntaron a mejoras y extensiones posteriores a este trabajo de memoria. Adicionalmente y por escrito, la respuesta de todos

en conjunto se presenta en forma textual a continuación:

> *Creemos que la presentación del prototipo sobrepasa las expectativas iniciales que teníamos en todos los aspectos presentados, desde el punto de vista funcional, arquitectónico y desde el punto de interfaz (aunque preliminar).*

También se presentó el sistema al resto de los desarrolladores de Unholster, incluyendo a aquellos encargados de generar los mapas antes de el desarrollo de este trabajo de título; la respuesta también fue positiva y no presentaron mayores reparos con el sistema.

5.4. Resumen

De lo expuesto en este capítulo, se satisfacen las historias de usuario y los requisitos no funcionales presentados en la Sección 3.1: con la interfaz mostrada en la Sección 5.1 los usuarios pueden subir datasets y generar distintas visualizaciones con ellos. Así mismo, de los resultados de la Sección 5.2 se constata que el sistema cumple las restricciones de tiempo impuestas: demorar menos de dos horas en la generación de mapas y construir distintas visualizaciones en poco tiempo.

Además se dan por cumplidos el objetivo general y los objetivos específicos de este trabajo de título, pues se construyó el sistema de acuerdo al diseño planteado, y en este capítulo fue validado según los parámetros propuestos.

Conclusión

El trabajo realizado en esta memoria permitió acelerar y automatizar la creación de visualizaciones de mapas en Unholster, luego de un exhaustivo análisis de las operaciones llevadas a cabo, identificando oportunidades de mejora. Se rediseñó el proceso completo y se propuso un modelo de datos generalizado, lo que permitió sistematizar gran parte de las operaciones. Además, se propuso e implementó una aplicación web en la que se ejecutan estos procesos, con la que los usuarios pueden interactuar, sentando las bases para un nuevo producto dentro de la empresa.

En base a lo expuesto en esta memoria, se cumplieron los objetivos específicos en su totalidad. Se estudió el proceso anterior de generación de mapas, y se identificaron las mejoras posibles y los procesos que se pueden paralelizar. Tomando esto como base, se diseñó un nuevo proceso de generación de mapas, generalizable a cualquier tipo de datos de entrada que cumpla con ciertas características base, y se implementó casi en su totalidad un sistema automático basado en este diseño. En cuanto a la validación del sistema, el cliente quedó satisfecho con el prototipo desarrollado, pues cumple con sus expectativas, y además cumple con las restricciones de tiempo impuestas a los procesos.

Considerando lo anterior, el objetivo general de este trabajo se cumplió. Ahora Unholster cuenta con un sistema capaz de generar de forma autónoma y en poco tiempo, múltiples visualizaciones de distintos datasets geolocalizados. Esta herramienta, además, permite trabajar con cualquier tipo de dataset que contenga datos geográficos, sin importar otras características de estos. A diferencia del proceso manual, el sistema es capaz de visualizar datos con distintas características, permitiendo a Unholster comercializar este servicio de visualización y análisis a sus clientes, ya que podrán trabajar con sus propios datos.

Es importante notar que, al principio de este trabajo de memoria, se consideraba que el proceso que consumía más tiempo en la generación de mapas era la asignación de puntos a segmentos, pues demoraba aproximadamente dos horas. Bajo esa perspectiva, era fundamental mejorar la eficiencia de esta operación para asegurar el éxito del proyecto. Sin embargo, una vez finalizado el desarrollo se comprobó que esto no era cierto, y que fue el rediseño de los procesos y el nuevo modelo de datos lo que redujo el tiempo de ejecución. Al utilizar la consulta de asignación antigua con la nueva arquitectura, los tiempos, si bien mayores a los de la consulta optimizada, no superaron una hora en promedio.

La metodología de desarrollo utilizada en esta memoria dio buenos resultados, y se pudo completar casi todo el trabajo planificado en el tiempo previsto. El proyecto tuvo cambios de liderazgo dentro de Unholster, pasando de estar a cargo del Director de Operaciones al

Director de Tecnologías, quienes tienen estilos distintos para manejar los proyectos de la empresa. Esto implicó un cambio en la planificación de las características del sistema, siendo necesario tomar una mayor iniciativa en cuanto a la dirección del desarrollo en cada paso.

Trabajo futuro

Entre los pasos a seguir después de este trabajo de memoria se encuentra la inclusión en el sistema de los procesos que se encargan de importar los mapas base, y separarlos en segmentos y manzanas clasificados por comunas. Este proceso se dejó fuera de este prototipo debido al ajustado período de tiempo disponible para su desarrollo. Si bien este procedimiento solo debe ejecutarse de forma ocasional, es más eficiente tener una forma automática de hacerlo, en lugar de que alguien se encargue de correr las consultas necesarias por separado. Estos procesos se dejaron adecuadamente documentados en los documentos anexos al código desarrollado como parte del proyecto, por lo que se espera que su inclusión al sistema no sea complicada.

Otras mejoras que se pueden hacer al sistema incluyen la generalización de los límites administrativos a los que pueden pertenecer los datasets; es decir, que un usuario suba un archivo correspondiente a una provincia, en lugar de una comuna, y el sistema sea capaz de dividir el dataset en las comunas correspondientes, para luego correr la asignación de puntos y segmentos de cada comuna en forma paralela, como se menciona en la Sección 3.3. También la posibilidad de visualizar la densidad de datos por segmento o manzana, en lugar de los totales; así la información mostrada estará normalizada, y se podrá hacer un mejor análisis sobre los mapas mostrados.

Junto con las adiciones mencionadas, otra mejora que puede hacerse al sistema es la opción de visualizar dos datasets de forma simultánea, mostrando una capa del mapa para cada uno, o combinando sus resultados. De esta forma podrán hacerse comparaciones históricas, por ejemplo, del padrón electoral, o comparar datasets de distintas fuentes y analizar las relaciones que puedan surgir.

Este trabajo de memoria es una primera aproximación a un sistema de generación automática de mapas de este tipo, mostrando las ventajas que se pueden obtener al sistematizar todos los procesos que se hacían en forma manual, pues permite enfocar los esfuerzos en el análisis de los datos y visualizaciones más que en la generación de cada mapa. Se espera que este prototipo sea desarrollado hasta llegar a ser una plataforma web completa, en que distintos usuarios puedan administrar los datos que se suben y luego los analicen usando las distintas visualizaciones generadas, sin tener que esperar horas entre una y otra.

Bibliografía

[1] Vladimir Agafonkin. *Leaflet - a JavaScript library for interactive maps*. 2018. URL: https://leafletjs.com/ (visitado 31-05-2018).

[2] CARTO. *CARTO — Location Intelligence Software*. 2017. URL: https://carto.com/ (visitado 22-11-2017).

[3] CARTO. *Pricing — CARTO*. 2018. URL: https://carto.com/pricing/ (visitado 28-03-2018).

[4] Dirección ChileCompra. *Mercado Público - La plataforma de compras públicas y oportunidades de negocio del Estado de Chile*. 2018. URL: http://www.mercadopublico.cl/TiendaFicha/Ficha?idProducto=1390016 (visitado 28-03-2018).

[5] PostGIS Project Steering Committee. *PostGIS — PostGIS Feature List*. 2017. URL: http://www.postgis.net/features/ (visitado 12-09-2017).

[6] pgRouting Community. *pgRouting/osm2pgrouting: Import tool for OpenStreetMap data to pgRouting database*. 2018. URL: https://github.com/pgRouting/osm2pgrouting (visitado 25-05-2018).

[7] Mark Harrower Cynthia Brewer y The Pennsylvania State University. *ColorBrewer: Color Advice for Maps*. 2018. URL: http://colorbrewer2.org (visitado 01-06-2018).

[8] Esri. *ArcGIS*. 2017. URL: https://www.arcgis.com/features/index.html (visitado 12-09-2017).

[9] Esri. *FAQ: What is the Jenks optimization method?* 2018. URL: https://support.esri.com/en/technical-article/000006743 (visitado 20-06-2018).

[10] Internet Engineering Task Force. *RFC 7946 - The GeoJSON Format*. 2018. URL: https://tools.ietf.org/html/rfc7946 (visitado 21-06-2018).

[11] Django Software Foundation. *GeoDjango | Django documentation | Django*. 2018. URL: https://docs.djangoproject.com/en/2.0/ref/contrib/gis/ (visitado 14-06-2018).

[12] Geofabrik GmbH. *OpenStreetMap Data Extracts*. 2018. URL: http://download.geofabrik.de/south-america/chile.html (visitado 24-05-2018).

[13] Google. *Google Maps*. 2018. URL: https://www.google.com/maps/ (visitado 14-06-2018).

[14] The PostGIS Development Group. <->. 2018. URL: https://postgis.net/docs/geometry_distance_knn.html (visitado 18-06-2018).

[15] The PostgreSQL Global Development Group. *PostgreSQL: Documentation: 10: 8.14. JSON Types*. 2018. URL: https://www.postgresql.org/docs/10/static/datatype-json.html (visitado 28-05-2018).

[16] Paul A. Longley y col. *Geographic information systems and science*. Wiley, 2011. ISBN: 9789470721445.

[17] Mapbox. *Mapbox*. 2017. URL: https://www.mapbox.com/ (visitado 22-11-2017).

[18] Mapbox. *Pricing | Mapbox*. 2018. URL: https://www.mapbox.com/pricing/ (visitado 28-03-2018).

[19] Colaboradores de OpenStreetMap. *OpenStreetMap*. 2017. URL: https://www.openstreetmap.org (visitado 12-09-2017).

[20] *osm2po - openstreetmap converter and routing engine for java*. 2017. URL: http://osm2po.de/ (visitado 10-11-2017).

[21] QGIS. *Welcome to the QGIS project!* 2017. URL: http://qgis.org/en/site/ (visitado 12-09-2017).

[22] Nick Roussopoulos, Stephen Kelley y Frédéric Vincent. «Nearest Neighbor Queries». En: *SIGMOD Rec.* 24.2 (mayo de 1995), págs. 71-79. ISSN: 0163-5808. DOI: 10.1145/568271.223794. URL: http://doi.acm.org/10.1145/568271.223794.

[23] Ask Solem y contributors. *Homepage | Celery: Distributed Task Queue*. 2018. URL: http://www.celeryproject.org/ (visitado 31-05-2018).

[24] Elijah Meeks Susie Lu. *Viz Palette*. 2018. URL: http://projects.susielu.com/viz-palette (visitado 01-06-2018).

[25] Oliver Tonnhofer. *Imposm3 — Imposm3 3.0.0a documentation*. 2018. URL: https://imposm.org/docs/imposm3/latest/ (visitado 25-05-2018).

[26] BCN Transparente. *Mapas vectoriales*. 2017. URL: https://www.bcn.cl/siit/mapas_vectoriales/index_html (visitado 23-11-2017).

[27] Michael Zeiler y Jonathan Murphy. *Modeling our world: the ESRI guide to geodatabase design*. ESRI Press, 2010. ISBN: 978-1-58948-278-4.

Anexo A

Modelo de datos

En la Figura A.1 se muestra el modelo de datos completo usado en el sistema desarrollado. Las tablas *api_ auxiliaryfile* y *api_ operation* almacenan información necesaria para el funcionamiento del sistema, como los archivos subidos y las operaciones ejecutadas, pero no tienen relación con el diseño presentado en este trabajo de título.

La tabla *maps_ geolimit* guarda la información de los límites administrativos en el sistema, incluyendo las relaciones de jerarquía, a través del campo `belongs_to_id` que referencia al campo `id` de la misma tabla.

Las tablas *maps_ block* y *maps_ segment* contienen los datos de manzanas y segmentos, respectivamente. Si bien la información que se almacena es análoga entre ambas tablas, *maps_ segment* tiene más campos al almacenar también los datos de las calles que se obtuvieron de OpenStreetMap. Ambas tablas se relacionan con *maps_ geolimit* a través del campo `geolimit_id`.

La tabla *datasets_ dataset* almacena información sobre los datasets subidos al sistema, y se relaciona con la tabla *maos_ geolimit* a través del campo `geolimit_id`. Los puntos que son parte de los datasets se guardan en la tabla *datasets_ datapoint*; la relación de pertenencia de un punto a un dataset se tiene en el campo `dataset_id` de esta tabla.

Finalmente, en la tabla *maps_ pointtoblock* se guarda la información de las relaciones entre puntos y manzanas; se relaciona con la tabla *maps_ block* con el campo `geo_object_id` y con la tabla *datasets_ datapoint* con el campo `datapoint_id`. La tabla *maps_ pointtosegment* guarda las relaciones entre puntos y segmentos de forma análoga a *maps_ pointtoblock* en todos los aspectos.

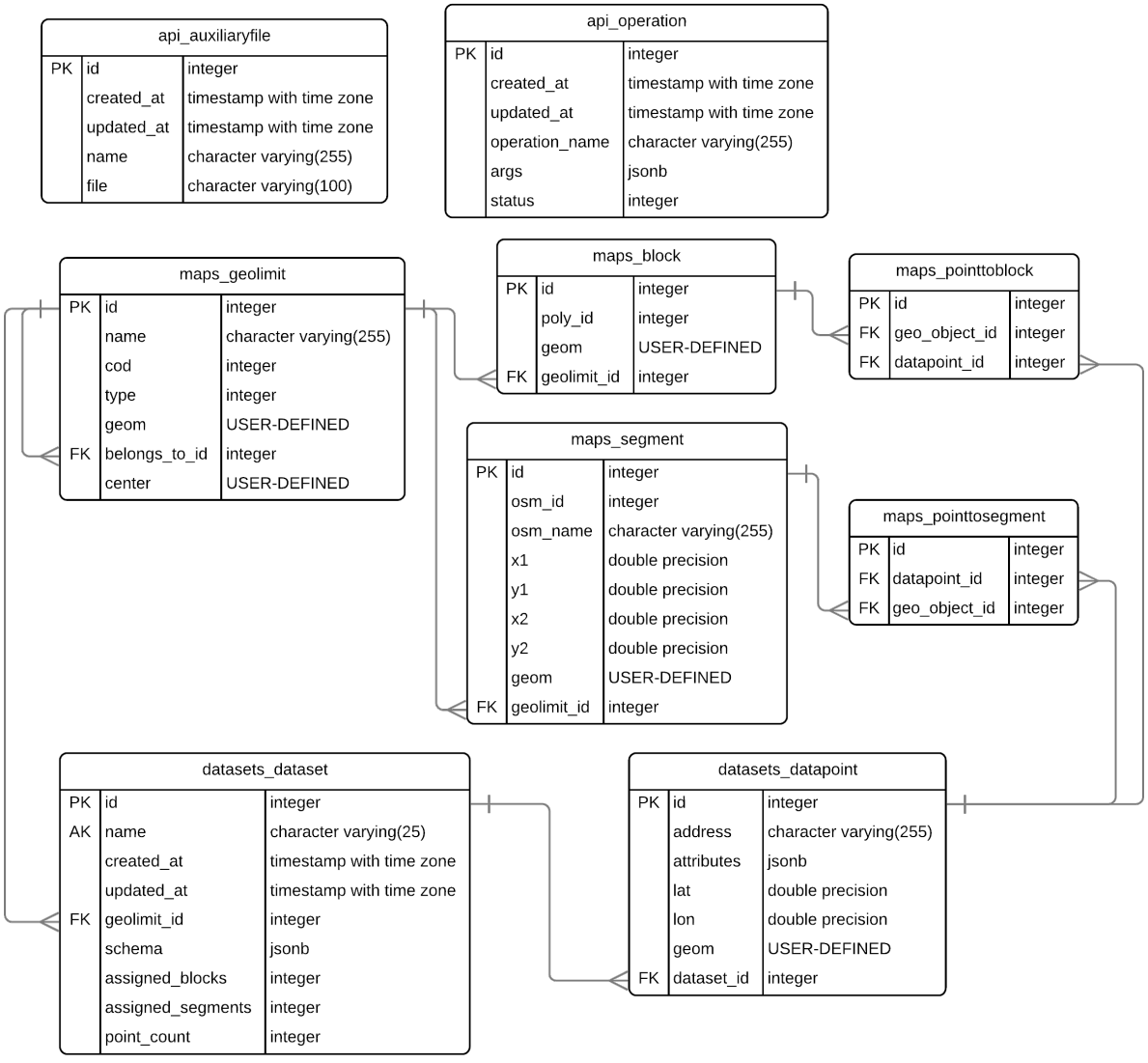


Figura A.1: Diagrama del modelo de datos del sistema.

Anexo B

Endpoints de la aplicación

A continuación se listan todos los endpoints disponibles en el sistema:

Información de límites administrativos: Entrega una lista de las áreas geográficas de interés que existen en el sistema, como regiones o comunas, que pueden usarse para delimitar los datasets que se suban al sistema.

Información de datasets: Entrega una lista con los datasets existentes en el sistema e información de estos, incluyendo la división administrativa a la que pertenece.

Visualización generada: Entrega un GeoJSON que representa al mapa con los segmentos o manzanas enriquecidos, según el dataset escogido y los parámetros indicados.

Resultado de operaciones: Entrega información sobre el estado de una operación, como subir un dataset o asignar segmentos a sus puntos.

Subir archivos: Permite subir archivos al sistema.

Cargar dataset: Importa un dataset a partir de un archivo subido al sistema previamente, indicando su id.

Procesar un dataset (segmentos): Asigna segmentos a los puntos del dataset para ser usado en visualizaciones de segmentos, indicando el id del dataset.

Procesar un dataset (manzanas): Asigna manzanas a los puntos del dataset para ser usado en visualizaciones de manzanas, indicando el id del dataset.