Testeando mapas
Metamorphic testing of OpenStreetMap

Trabajo de Fin de Grado
Curso 2022–2023

Autores

Tyson Mendes de Graça
Roberto Anguís Martínez
Isabel Pérez Pereda

Director

Manuel Núñez

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Testeando mapas
Metamorphic testing of OpenStreetMap

Memoria presentada para optar
al título de Graduado en Ingeniería Informática

Autores
Tyson Mendes de Graça
Roberto Anguís Martínez
Isabel Pérez Pereda

Director
Manuel Núñez

Convocatoria: *Septiembre 2023*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

15 de Septiembre de 2023

Dedicatoria

Se lo dedico a mi madre, a mis hermanos, a mi pareja y a los de siempre.

Tyson Mendes

A mis padres, Juana Rosa y Pedro, y mis hermanos.

Roberto Anguís

A mis padres, en especial mi madre, y mi hermana Pilar por apoyarme sea cual sea la situación.

Isabel Pérez

Agradecimientos

De manera conjunta, nos gustaría agradecer a nuestro tutor Manuel Núñez por la paciencia y la confianza ciega depositada en nosotros, pero sobre todo por los consejos y la ayuda prestada en el desarrollo del proyecto. Agradecemos también a todos los profesores que hemos tenido a lo largo de estos años de carrera por las enseñanzas y a nuestros compañeros por sostenernos y acompañarnos día a día.

De manera más personalizada, nos gustaría agradecer el apoyo recibido por parte de nuestras familias que siempre han estado dispuestas a ayudar sin pedir nada a cambio. Gracias por proporcionarnos todas las herramientas necesarias para poder forjar nuestro propio camino.

Resumen

Testeando mapas

La culminación más deseada de toda investigación es poder generar una aplicación práctica que aporte una solución a algún problema conocido o simplemente mejore nuestra calidad de vida. Este es precisamente el objetivo de este TFG cuyo planteamiento inicial se basa en una investigación previa realizada sobre OpenStreetMap para evaluar la calidad de sus datos y tratar de buscar un método matemático que encuentre de forma eficaz la información errónea contenida en él. A través de un sistema de Pruebas Metamórficas (MT) perfectamente analizado y ajustado, este estudio propone una potente herramienta de búsqueda de fallos y, por consiguiente, una forma interesante de evaluar la corrección del sistema.

Nuestra aportación está orientada exclusivamente hacia el usuario y se centra en la creación de una aplicación que muestre visualmente lo que han logrado calcular las Pruebas Metamórficas (MT) obtenidas en dicha investigación. Para ello, hemos llevado a cabo un seguimiento exhaustivo en busca de una interfaz sencilla e intuitiva que facilite su accesibilidad y comprensión. De esta forma se amplia el rango de usabilidad y se consigue interpelar a un mayor y más diverso número de usuarios.

Palabras clave

Aplicación, OpenStreetMap, búsqueda de errores, usuario, interacción, interfaz intuitiva, accesibilidad.

Abstract

Metamorphic testing of OpenStreetMap

The most desired culmination of any research is to be able to generate a practical application that provides a solution to a known problem or simply improve our quality of life. This is precisely the objective of this TFG whose initial approach is based on a previous research carried out on OpenStreetMap to evaluate the quality of its data and try to find a mathematical method to efficiently find the erroneous information contained in it. Through a perfectly analyzed and tuned Metamorphic Testing (MT) system, this study proposes a powerful fault-finding tool, and therefore an interesting way to evaluate the correctness of the system.

Our contribution is exclusively user-oriented, and focuses on the creation of an application that visually displays what the Metamorphic Tests (MT) obtained in this research have managed to calculate. For this purpose, we have carried out an exhaustive research in search of a simple and intuitive interface that facilitates its accessibility and comprehension. In this way, the range of usability is widened and a greater and more diverse number of users is reached.

Keywords

Application, OpenStreetMap, error search, user, interaction, intuitive interface, accessibility.

Índice general

Índice de figuras

Capítulo 1

Introducción

El uso de software de código abierto es una estrategia cada vez más expandida y parece estar atravesando una de sus mejores etapas. Su particularidad se centra en poder ser utilizado y modificado por cualquier usuario sin ningún tipo de restricción, ya que es de libre acceso. Y, aunque al principio compartir y exponer un código que tanto esfuerzo y dedicación había supuesto generar parecía una auténtica locura, esta filosofía cada día tiene más adeptos. Se trata de una práctica que permite adaptarse a las necesidades específicas de cada usuario y, sobre todo, que mantiene vivo el código al estar en constante mantenimiento y actualización.

Algunos ejemplos mundialmente conocidos de software de código abierto son el sistema operativo Linux, la suite de oficina Apache OpenOffice, la plataforma educativa Moodle o el proyecto de cartografía digital OpenStreetMap (OSM). Este último representa una de las bases de datos geoespaciales de código abierto más importantes del mercado. Es una plataforma completamente abierta y colaborativa que, gracias a la diversidad y pluralidad de sus usuarios, proporciona un mapa considerablemente preciso y actualizado del mundo. Sin embargo, esta extensa red de colaboradores cuyas adiciones no son supervisadas también puede dar lugar a la aparición de errores que originan ciertas incoherencias.

Un estudio realizado en colaboración entre la Universidad de Almería y la Universidad Complutense de Madrid ha logrado generar un marco teórico que identifica la información errónea de cualquier OpenStreetMap, además de proporcionar su respectiva implementación de libre acceso. En este proyecto nos proponemos utilizar dicha investigación para crear una aplicación que permita encontrar y situar espacialmente cualquier tipo de error sobre el mapa. De esta forma, la identificación de incoherencias estará al alcance de cualquier usuario independientemente de sus conocimientos informáticos. Teniendo en cuenta que ya disponemos de una implementación del marco teórico, nos centraremos en cómo mostrar visualmente lo que la investigación hizo a nivel conceptual.

1.1. Motivación

El siglo XXI es el siglo de la modernidad, de la tecnología y de la inmediatez. Todo lo que no esté informatizado y publicado en la red no existe, y cuanto más sencillo sea el acceso mucha más repercusión tendrá. Este cambio de paradigma ha llenado nuestras vidas de aparatos electrónicos, redes sociales, inteligencia artificial y aplicaciones informáticas de todo tipo. Por ello, no hay mejor manera de darle vida a una investigación que a través de una aplicación que haga accesible toda la información descubierta.

En este caso, el estudio realizado presenta una forma rápida y sencilla de encontrar los errores presentes en un mapa OSM. Está basado en una herramienta más genérica que está demostrando su utilidad a la hora de realizar pruebas de software en sistemas que no disponen de oráculo o cuya ejecución resulta demasiado costosa computacionalmente. El desarrollo de una aplicación que muestre toda esta información de manera visual permitiría corregir fácilmente los errores y ampliar el campo de accesibilidad.

1.2. Objetivos

El objetivo principal del trabajo es crear una aplicación a nivel usuario que sea lo más sencilla e intuitiva posible. A nivel funcional debe poder encontrar la información defectuosa de cualquier mapa OSM y mostrarla de forma visual y precisa sobre el propio mapa. En cuanto a la parte interactiva, lo más importante es que resulte fácil de manejar para que pueda ser utilizada por todo tipo de público, sin importar edad o conocimientos. Para que el proyecto sea lo más completo y preciso posible, buscaremos además cumplir los siguientes objetivos específicos:

- Trasladar el espíritu de la investigación previa a la aplicación tratando de que el usuario la perciba como una potente herramienta de ayuda a la hora de buscar información errónea y, por consiguiente, de evaluar la calidad del mapa en cuestión.

- Generar una imagen visual lo más clara y limpia posible de los distintos errores encontrados en un determinado mapa OSM mostrando además su respectiva posición y sus atributos.

- Analizar la base de datos proporcionada por la investigación previa y estudiar tanto la forma más sencilla de acceder a ella como la más adecuada para integrar la implementación existente en la aplicación.

- Diseñar una interfaz gráfica que resulte intuitiva e interactiva para el usuario, priorizando siempre su comodidad y la perfecta comprensión de las funciones que tienen cada uno de sus elementos.

1.3. Plan de trabajo

Tarea a realizar	Inicio	Fin
Analizar la situación de partida del TFG	11/11/2022	15/12/2022
Analizar en profundidad la investigación previa	21/11/2022	06/02/2023
Entender cómo funcionan los mapas OSM	15/12/2022	06/02/2023
Estudiar la implementación de las pruebas metamórficas (MT)	15/12/2022	15/02/2023
Decidir el lenguaje de programación a utilizar	08/02/2023	27/02/2023
Estudiar la viabilidad de la integración de las MT en el lenguaje de programación escogido	08/02/2023	27/02/2023
Definir las líneas de trabajo de la aplicación	02/03/2023	09/03/2023
Desarrollar las bases de front y back end	10/03/2023	10/04/2023
Contrastar distintas opciones de diseño gráfico	13/04/2023	08/05/2023
Enlazar front end con back end	30/03/2023	21/08/2023
Evaluar el funcionamiento de la aplicación	24/04/2023	25/08/2023
Mejorar problemas de diseño y funcionalidad	07/08/2023	05/09/2023
Redactar la memoria del proyecto	24/08/2023	05/09/2023

1.4. Repositorio

El código de la aplicación realizada se encuentra en un repositorio público de Github. También aparece una carpeta "img" con las imágenes que utiliza la aplicación en términos de fondo, logotipo o icono. Hemos añadido además la carpeta "MT-OSM-master" con la implementación de las pruebas metamórficas (MT) y varios mapas OSM que nos han permitido evaluar la aplicación final. Por último, contiene un vídeo explicativo en el que se puede visualizar una ejecución de la herramienta creada y sus diversas funcionalidades. Al repositorio se puede acceder a través de la siguiente URL:

`https://github.com/tysonmdg/TFG`

1.5. Asignaturas relacionadas

En este proyecto se han utilizado herramientas de diversas asignaturas estudiadas a lo largo de la carrera universitaria. Entre ellas debemos destacar:

- Matemática Discreta y Lógica Matemática I y II;
- Fundamentos de la Programación I y II;
- Bases de Datos;

- Estructuras de Datos;
- Tecnología de la Programación I y II;
- Geometría Computacional (Doble Grado);
- Robótica (Optativa);
- Ingeniería de Comportamientos Inteligentes (Optativa).

Capítulo 2

Antecedentes

2.1. Funcionamiento de los mapas OSM

Los mapas OSM -abreviatura de OpenStreetMap- son una alternativa a Google Maps. Es una aplicación abierta y de participación voluntaria. Cualquier persona desde cualquier parte del mundo puede participar y actualizar o mejorar la información preexistente. Esta iniciativa permite crear colectivamente un mapa editable del mundo, aportando datos espaciales (geometrías) o datos textuales (etiquetas). Los usuarios pueden colaborar proporcionando información sobre calles, carreteras, edificios, zonas naturales y otros elementos geográficos, incluso sobre el estado de los mismos. Las fuentes también son diversas: GPS, fotografías aéreas, imágenes geolocalizadas, información propia por vecindad y conocimiento del lugar, entre otras muchas opciones. La información que van incorporando los usuarios se va conservando en la base de datos de OSM. A partir de ahí, los datos se procesan y visualizan en el mapa interactivo, quedando disponible para el público de forma global y genérica.

Para que todo esto pueda funcionar y sea factible interpretar los mapas, se establecen algunas reglas básicas que tienen que seguir los colaboradores. En los OSM se consideran dos tipos de geometrías: los nodos, que se utilizan para representar elementos como árboles y señales de tráfico; y las líneas, que pueden ser polilíneas, que son las que representan calles o carreteras, o polígonos cerrados, que se utilizan, entre otros, para edificios. En cuanto a los datos textuales, las etiquetas se definen como pares de referencia (clave, valor) que permiten adjudicar un valor a una determinada característica cuyo nombre se identifica con la clave. Estas etiquetas proporcionan información sobre los elementos de interés del mapa, como museos, hoteles, monumentos, hospitales, farmacias, etc. Pero también proporcionan información útil sobre las calles o carreteras, fundamentales a la hora de trazar rutas de un sitio a otro.

Para considerar los mapas OSM como una fuente confiable de datos geográficos es fundamental la evaluación de la calidad de los mismos. Valorando el esfuerzo de los muchos usuarios y su colaboración para detectar errores en etiquetas e inexactitudes en las geometrías, en los últimos años se han desarrollado diferentes herramientas

para reportar fallos en los mapas OSM. Entre los sistemas de validación de datos que permiten un gran número de verificaciones de carácter geométrico se encuentra el ampliamente utilizado Osmose. Otra herramienta disponible es Keep Right, que realiza varias comprobaciones de coherencia y dispone de mecanismos para verificar falsos positivos y etiquetarlos en el mapa como corregidos. El JOSM Validator, integrado con el editor JOSM, permite verificar los datos cargados en el editor, resaltar errores y advertencias, y además admite correcciones a demanda. Para la depuración de errores también podemos utilizar OSM Inspector, que forma parte de las herramientas de GeoFabrik y genera un mapa con errores clasificados en diferentes capas, aunque presenta menos controles de consistencia de geometría que los sistemas mencionados.

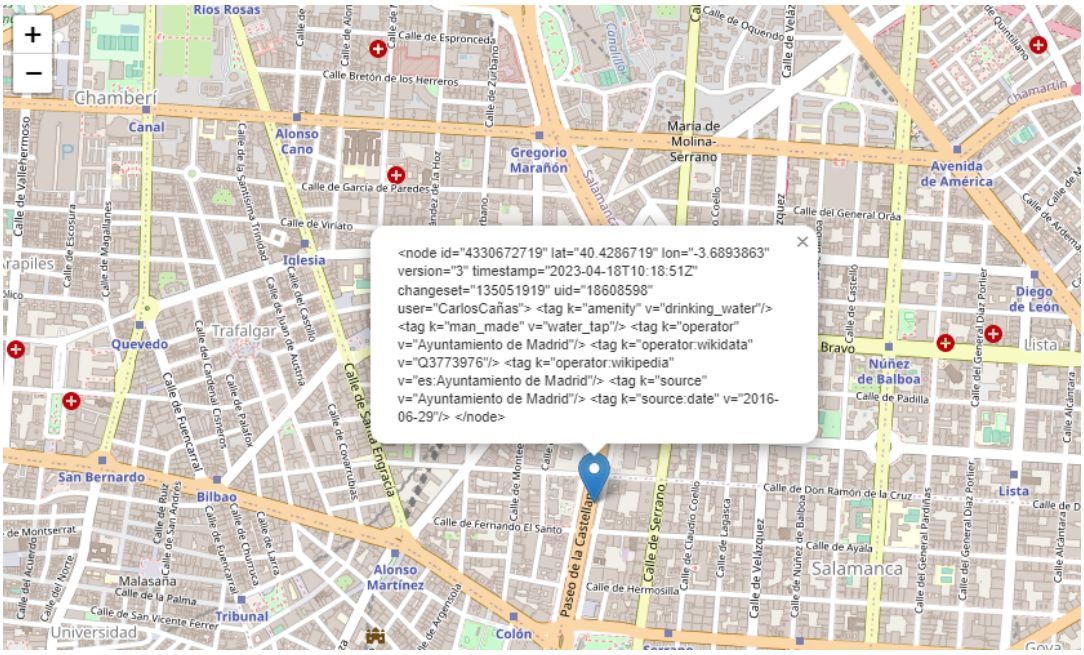


Figura 2.1: Ejemplo de mapa OSM

2.2. Análisis de la investigación previa

Para este proyecto partimos de la investigación "Metamorphic testing of OpenStreetMap" realizada en el año 2021 por Jesús M. Almendros-Jiménez, Antonio Becerra-Terón, Mercedes G. Merayo y Manuel Núñez como una colaboración entre la Universidad de Almería y la Universidad Complutense de Madrid, en relación con la evaluación de la calidad de los datos incluidos en los mapas OSM. Utiliza el concepto de Pruebas Metamórficas (MT) pero aplicado a la validación de datos, algo que ha empezado recientemente a generar interés en la comunidad científica. Para ello, hay que establecer una serie de Relaciones Metamórficas (MR) entre entradas y salidas para, mediante comparaciones, decidir si los datos son válidos o presentan fallos. Su peculiaridad es que pueden detectar situaciones que no representan un error como tal pero cuyo descubrimiento puede ayudar a entender mejor un deter-

minado comportamiento del sistema en prueba. En este caso, las MR definidas por los investigadores pueden identificar situaciones en las que el problema detectado no representa un error de geometría real sino de etiquetado.

Un ejemplo concreto que puede ayudarnos a entender mejor la utilidad de las MT es la aplicación de la función seno. En efecto, su periodicidad nos permite averiguar fácilmente si una determinada implementación suya P está bien construida. Ni siquiera es necesario saber qué valores devuelve P, solamente hace falta comparar $P(x)$ y $P(x + 2\pi)$. Si dichos valores son diferentes, como el periodo del seno es 2π, sabremos con total seguridad que la implementación no es válida. Esta afirmación puede realizarse gracias al conocimiento en detalle del comportamiento del seno, lo que hace innecesario disponer de valores exactos. Este tipo de razonamiento es el que se busca implementar a la hora de definir una MR, en este caso $MR(x, x + 2\pi, P(x), P(x + 2\pi))$. Además, para que funcionen de forma óptima, las entradas deben generarse de forma automática, distinguiendo entre una entrada de origen y una de seguimiento. Informalmente, la primera representa el valor escogido por el probador del sistema mientras que la segunda se utilizará junto a la de origen para, a través de sus respectivos resultados, compararlas en una MR. En el caso del seno, una entrada de origen podría ser el 0 y una de seguimiento 2π, comparándose los valores $P(0)$ y $P(2\pi)$. Lo más interesante de todo es que la mayoría de las veces una entrada de origen puede generar muchas entradas de seguimiento, como por ejemplo el 0 del cual se obtienen automáticamente 2π, 4π, -2π...

El enfoque que se pretende adoptar respecto a las MT se centra en la implementación de un análisis de calidad intrínseco en el que las MR se utilicen para evaluar la calidad de los mapas OSM. Esta propuesta supuso una nueva línea de investigación en métodos de evaluación de la calidad intrínseca en base a cuatro consideraciones indispensables. En primer lugar, la mayoría de las propiedades geométricas que se pueden evaluar mediante sistemas de validación de datos también se pueden expresar de forma matemática. En segundo lugar, la mayoría de ellos son propiedades sobre dos o más elementos OSM, es decir, son relaciones entre elementos OSM, por lo que se pueden expresar como MR. Como tercera consideración es importante deducir que dichas relaciones se pueden verificar automáticamente. Por lo tanto, las MT proporcionan un marco adecuado para el diseño de soluciones (semi)automáticas que comprueben si se mantienen dichas relaciones.

La estructura propuesta por la investigación se basa en dos conceptos principales e imprescindibles para su desarrollo. Por una parte, la revisión de los *datos primarios* que conforman la estructura de datos topológicos utilizada por el sistema OSM para representar tanto elementos espaciales como textuales. Por otra, la definición formal de las nociones fundamentales involucradas en las MT: relaciones metamórficas y entradas de origen y de seguimiento. El alcance de aplicación de las MR es un mapa específico y su objetivo es establecer relaciones que deban cumplir inequívocamente todos los elementos del mapa. Por consiguiente, estas MR van a evaluar la existencia de conexiones entre elementos específicos, enfocándose en las dos fuentes que son objeto de los máximos errores: las carreteras y las rotondas. Analicemos con más

detenimiento el trabajo Almendros-Jiménez et al. (2023) que es la base de este TFG.

NoDeadLock (vías sin puntos muertos) es el nombre que recibe la MR cuyo objetivo es detectar vías que posean dos segmentos diferentes con direcciones opuestas y no presenten ninguna salida alternativa, generando un punto muerto que puede llegar a generar una colisión. Para determinar la existencia de esta situación, se buscan vías con el mismo nombre y con el mismo nodo final. Si se encuentra alguna con esta característica, se puede suponer en un primer momento que no existe salida. Sin embargo, es necesario verificar si existe otra vía que atraviese el camino anterior y pueda llegar a ser una salida.

Otra MR analizada es **NoIsolatedWay** (caminos con entradas y/o salidas), que tiene como objetivo detectar aquellas vías que no se cruzan con ninguna otra, y por tanto están completamente aisladas. Para ello, se buscan vías que verifiquen que ninguno de sus nodos aparece en otra vía. Esta MR garantiza que todos los caminos tienen una salida o una entrada, de lo contrario no se podría acceder a ellas. De forma complementaria se definen las relaciones metamórficas **ExitWay** y **EntranceWay** (caminos con al menos una salida y una entrada) que verifican, respectivamente, que todas las vías tengan al menos una salida y una entrada. Dada una determinada carretera, hay que asegurarse de que al menos uno de sus nodos aparece en alguna otra vía pero no representa la salida (respectivamente entrada) de dicho camino.

Por otro lado, como las rotondas son uno de los elementos con mayor propensión a generar errores, resulta importante garantizar que el mapa no tenga rotondas sin salidas ni entradas. Lo primero ocurre cuando al menos uno de los nodos de los caminos que conforman la rotonda corresponde al primer nodo de otro camino, en cuyo caso se asume que la rotonda tiene una salida. Lo segundo surge cuando existe un camino cuyo último nodo se identifica con cualquiera de los que aparecen en la rotonda, formando una entrada. Para detectar estos errores se propusieron dos MR diferentes: **ExitRAbout** y **EntRAbout** (rotondas conectadas). También es cierto que, técnicamente, salvo que una vía tenga asociada la etiqueta "sin salida", debe estar conectada a otro camino. Esto se comprueba verificando que todas las vías que no tengan asociado dicho estado comparten el último nodo de otra vía. El objetivo de esta MR que recibe el nombre de **Connected** (caminos conectados) es asegurar que el mapa no tiene caminos desconectados.

El objetivo de **AreaNoInt** (área no intersectada) es detectar intersecciones entre las denominadas áreas, definidas como caminos cerrados, y el resto de vías. Se trata por tanto de verificar si alguno de los segmentos -formados a su vez por nodos consecutivos- que constituyen el área está siendo atravesado por un segmento perteneciente a otra vía. Para que la MR no encuentre falsos errores debe hacer una excepción con las etiquetas ferrocarril, autopista o vía fluvial, ya que pueden estar cruzadas por otras vías a otro nivel. Para no confundir intersección y superposición, se define **NoOverlap** (no hay carreteras ni edificios superpuestos) que asegura que las carreteras y los edificios no se superpongan respectivamente, es decir, que no

compartan ningún segmento.

El trabajo Almendros-Jiménez et al. (2023) implementa todas las relaciones metamórficas descritas y las aplica en mapas OSM reales. Como los datos de este tipo de mapas pueden exportarse en un formato XML, decidieron utilizar el lenguaje de consulta XQuery que resulta semánticamente muy similar a SQL. Es un lenguaje funcional y está basado en expresiones FLWOR ("For, Let, Where, Order by, Return"), un lenguaje de programación declarativa que le convierte en el candidato perfecto para la implementación de unas MR creadas a partir de lógica proposicional. Todas estas características facilitarían en gran medida la automatización de las MR ya que su traducción a dicho lenguaje se haría de forma casi directa.

2.3. Planteamiento posterior

Una vez estudiado el marco teórico y analizada la implementación de las MR que proporcionaba la investigación, la decisión más importante antes de comenzar a programar era si tratábamos de reutilizar dicho código o lo reimplementábamos desde cero en el lenguaje de programación que escogiéramos. Ciertamente, una reimplementación requería un estudio mucho más profundo de los mapas OSM para poder hacer un tratamiento óptimo de los datos y sus atributos, además de requerir de más tiempo para la elaboración de un código extra. Por otro lado, la reutilización tampoco era una vía sencilla ya que debíamos encontrar la forma de conectarnos a la base de datos, acceder a las XQuery y obtener los resultados para procesarlos en el lenguaje escogido para la aplicación. Y, aunque los dos caminos presentaban numerosas dificultades, decidimos optar por la segunda estrategia suponiendo que el acceso a las consultas nos resultaría mucho más sencillo y rápido de llevar a cabo que generar un código nuevo desde cero.

El siguiente paso era decidir el lenguaje de programación con el que desarrollar la aplicación, la cual queríamos en principio que fuera de escritorio. Tras sopesar las ventajas e inconvenientes, nos quedamos con dos opciones bastantes válidas: Python y JavaScript. Cuando empezamos a tener más consciencia de las herramientas que requería el proyecto, Python fue una de las primeras ideas que se pusieron sobre la mesa. En efecto, es un lenguaje de programación muy amplio y versátil y se utiliza esencialmente en aplicaciones científicas y especializadas, como es nuestro caso. Sin embargo, su uso profesional se centra sobre todo en el desarrollo back end, el área dedicada a la parte funcional de la aplicación y, por tanto, invisible para el usuario. Teniendo en cuenta la importancia que queríamos otorgarle a la interacción con este último, estudiamos la posibilidad de utilizar JavaScript. Centrada en el desarrollo de aplicaciones web, se puede emplear tanto para back end como para front end, que es el área encargada de crear todos los elementos visibles y sus interacciones.

La decisión se tomó en base a dos aspectos que resultaban esenciales para el correcto funcionamiento de la aplicación: la carga de los mapas OSM y la realización de consultas en la base de datos. Realizamos varias pruebas en ambos lenguajes en

busca de la forma más óptima y sencilla de enlazar la implementación preexistente de las relaciones metamórficas con los elementos visuales que necesitábamos para mostrar los mapas. Nos dimos cuenta enseguida de que Python cumplía con la mayoría de nuestros requisitos al disponer de bibliotecas como *BaseXClient*, *folium*, y una API de los mapas OSM. Y, aunque JavaScript nos ofrecía mayor comodidad respecto a la construcción de la interfaz gráfica de usuario, presentaba peores condiciones en cuanto a la comunicación con el servidor y estaba más enfocada en el desarrollo de aplicaciones web. Algunos detalles técnicos en los que no se ha profundizado se explicarán con mayor precisión en el capítulo 3.

Una vez definido el lenguaje de programación, había que comenzar a construir la base del proyecto, es decir las acciones que llevaría a cabo la aplicación. En ese momento era prioritaria la funcionalidad ya que representa la base de la estructura y sin ella deja de tener sentido cualquier intento de interacción con el usuario. Lo primero que intentamos fue mostrar visualmente un mapa OSM, y para ello escogimos la librería *folium*, que utiliza por defecto este tipo de mapas. Cuando conseguimos crear nuestro mapa interactivo, investigamos las herramientas de las que disponía dicha librería para marcar gráficamente los puntos que quisiéramos. Esto nos sería de gran utilidad a la hora de mostrar los errores encontrados. A partir de ahí nos centramos en la funcionalidad y, pensando que en un futuro nos resultaría útil tener acceso a los elementos del mapa, jugueteamos un poco con la API de OSM y tratamos de obtener información como la latitud y longitud de un punto o el nombre de un nodo.

La labor más complicada fue sin duda la unión entre el código proporcionado por la investigación previa y el resto de elementos de la aplicación. Nos dimos cuenta de que necesitaríamos una base de datos para cargar el mapa y realizar las consultas, y desde un primer momento optamos por BaseX pues era la utilizada por los investigadores. Este proceso se llevó a cabo gracias a la biblioteca *BaseXClient* mencionada anteriormente. Fuimos añadiendo funcionalidades de forma progresiva, comenzando por cargar el mapa, extrayendo a continuación los archivos que guardaban las consultas de las MR, y finalmente mostrando los resultados gráficamente en el mapa. También se hicieron varias pruebas respecto a la carga del mapa en la base de datos para tener distintas opciones, porque no es lo mismo cogerlo directamente del explorador que generarlo a partir de la API. Conforme el desarrollo del back end iba concluyendo se comenzaban a dar los primeros pasos en el front end y se trabajaba en el diseño de la interfaz gráfica.

Capítulo 3

Software y librerías utilizadas

3.1. Base de datos

Figura 3.1: Logotipo de la base de datos

Uno de los elementos más importantes de este proyecto es la base de datos en la que se almacenan los mapas y las relaciones metamórficas (MR). La base de datos es la que se encarga de recopilar de forma organizada la información y suele almacenarse, como cabe esperar, en un sistema informático. Además, está controlada por un sistema de gestión de bases de datos (DBMS) y dispone de un lenguaje de consulta. En este caso, la elección de dichos parámetros no fue propia ya que la implementación de las MR que proporcionaba la investigación había sido realizada en XQuery y sus respectivas evaluaciones en BaseX. Optamos por utilizar el mismo software para asegurarnos de que no hubiese problemas a la hora de acceder y realizar las consultas sobre la base de datos. Además, reutilizar el código nos permitía ganar tiempo y centrarnos en aspectos más relevantes como el diseño de la interfaz gráfica.

A nivel técnico también nos parecía una herramienta óptima para poder cumplir con los objetivos del trabajo. En efecto, BaseX es un sistema de gestión de bases de datos XML y procesador XQuery especializado en el almacenamiento, consulta y visualización de grandes documentos y colecciones XML. Además, se trata de un software de código abierto totalmente gratuito, manteniendo la línea colaborativa que iniciaron los OpenStreetMap. Un elemento claramente diferenciador respecto a otros DBMS es su interfaz gráfica de usuario, que permite explorar sus datos de forma interactiva así como realizar consultas XQuery en tiempo real.

Pero lo que terminó por convencernos fue su facilidad para integrarse en otros lenguajes gracias a su amplio número de APIs, entre las cuáles se encuentra una destinada a Python. Incluyendo *BaseXClient* en el preámbulo del código, tendremos acceso a todas las funciones de BaseX desde Python. Esto nos permitirá en particular abrir y cerrar un servidor que nos otorgue acceso a la base de datos tanto

en términos de lectura como de escritura. En cuanto a las consultas XQuery, la biblioteca proporciona un soporte completo para expresiones FLOWR y una serie de funciones para administrar la base de datos. Todas estas herramientas nos facilitaron enormemente el tratamiento de los mapas OSM ya que eran exportados con formato XML y aplicación de las relaciones metamórficas sobre ellos. BaseX fue de hecho una de las razones de mayor peso a la hora de decidir el lenguaje de programación de la aplicación.

3.2. Desarrollo de la aplicación

(a) Python (b) Jupyter (c) Spyder

Figura 3.2: Logotipo del lenguaje y entornos de desarrollo escogidos

En el capítulo 2 expusimos el proceso que llevamos a cabo hasta lograr escoger un lenguaje de programación óptimo para nuestras necesidades. Sin ánimo de repetirnos, vamos a tratar de profundizar en los aspectos técnicos que nos llevaron a tomar esta decisión. Como ya explicamos, estuvimos dudando entre Python y JavaScript, pero, teniendo en cuenta las funcionalidades que buscábamos cubrir, nos decantamos finalmente por la primera opción. En efecto, en esta fase del proyecto el front end pasó a un segundo plano y nos centramos en analizar cuál de los dos ofrecía mayor facilidad de comunicación entre servidores, bases de datos XQuery y mapas OSM.

En JavaScript, la forma más sencilla y comúnmente utilizada de generar mapas es a través de *Leaflet*, una librería de código abierto con una gran cantidad de propuestas y alternativas complementarias. Parecía la mejor opción en cuanto a visualización de mapas e interacción de los mismos con el usuario. Sin embargo, el acceso a BaseX y la ejecución de consultas XQuery entrañaba una gran complejidad que no era necesario asumir al disponer de otras posibilidades. Además, utilizando la biblioteca de Python conocida como *folium* conseguíamos obtener los mismos resultados respecto a la generación de mapas interactivos ya que su implementación también está basada en *Leaflet*. La decisión resultaba evidente: Python se adaptaba mejor a nuestras necesidades de manera global.

En cuanto a los entornos de desarrollo de Python, cada miembro del grupo escogió el que le resultaba más cómodo. Algunos se decantaron por Jupyter Notebook, que es una herramienta de código abierto que soporta distintos lenguajes de programación, entre ellos evidentemente Python. Resulta muy cómodo debido a que permite organizar el código en bloques independientes y ejecutarlos de forma separada, lo que lo hace idóneo para llevar a cabo procesos de análisis de datos con modelización incluida. El resto de integrantes del grupo utilizaron Spyder, un potente entorno de desarrollo interactivo y multiplataforma que también es de código abierto. Incluye funciones avanzadas de edición y depuración, así como un entorno informático numérico muy interesante.

3.2.1. Desarrollo back end

El desarrollo back end de la aplicación, tal y como hemos comentado en varias ocasiones, está basado en dos funcionalidades primordiales: el acceso a la base de datos y el tratamiento de los mapas OSM. Lo primero fue resuelto mediante el uso de la biblioteca *BaseXClient* que nos proporciona acceso a los datos y nos permite realizar consultas. En cuanto a los mapas, resultó algo más complicado encontrar la herramienta adecuada, y en un primer momento se llegó en pensar en *GeoServer*, un servidor de código abierto escrito en Java que permite mostrar y editar datos geoespaciales. Se realizaron varias pruebas, como por ejemplo la creación de un mapa o su respectiva carga en la base de datos, pero la herramienta estaba demasiado orientada en el desarrollo de cartografía web y preferimos utilizar la librería *folium*.

En efecto, *Folium* es una biblioteca de código abierto, como la mayoría de las herramientas empleadas en este proyecto, centrada en la creación de mapas interactivos. Por defecto utiliza OpenStreetMap como mapa base y su popularidad reside en ofrecer una gran cantidad de funcionalidades adicionales que facilitan el manejo de las interacciones con el mapa. Este amplio catálogo de opciones se organiza a través de "plugins", cada uno destinado a solucionar una funcionalidad diferente. En nuestro caso, teniendo en cuenta la importancia de mostrar correcta y claramente los errores en el mapa, hemos recurrido a *MarkerCluster*, encargado de clasificar los marcadores. También intentamos utilizar *Draw*, un plugin encargado de dibujar vectores editables en el mapa, permitiendo añadir todo tipo de formas. No obstante, en esta primera versión de la aplicación no ha sido factible adaptarla a nuestras necesidad y no está implementado.

3.2.2. Desarrollo front end

La interfaz gráfica de usuario (GUI) es uno de los aspectos del proyecto que requería mayor implicación y ha sido el que más cambios ha sufrido. Su diseño se trató de llevar a cabo con la biblioteca *Tkinter*, que es la que se suele utilizar por defecto en Python. La creación de ejecutables para las aplicaciones es muy accesible y es una herramienta sencilla de entender y dominar con ciertas limitaciones creativas. Resulta muy útil si se quiere crear una interfaz básica de forma rápida, pero para obtener resultados avanzados y de aspecto profesional, no es la más indicada. En

cuanto realizamos unas cuantas pruebas e intentamos cambiar atributos elementales como la tipografía, nos dimos cuenta de que iba a ser insuficiente. Sin embargo, el factor determinante para buscar una alternativa fue cuando tratamos de introducir el mapa OSM dentro de una ventana. Por más pruebas que realizamos no fue posible llevarlo a cabo, por lo que decidimos cambiar de biblioteca y así ganar también en independencia creativa.

La siguiente opción más utilizada para la elaboración de interfaces gráficas en Python era *PyQt5*, que está diseñada alrededor de señales y ranuras que permiten establecer una comunicación entre objetos. Esto otorga mucha flexibilidad y fluidez al código, sobre todo a la hora de crear eventos GUI. Además, dispone de una amplia gama de widgets editables que resuelven prácticamente todas las necesidades del diseño, entre ellas la integración de un mapa OSM en una ventana de aplicación. El único problema es que para conocer en profundidad todas sus facetas se requiere mucho tiempo ya que cada widget contiene multitud de detalles. Incluso así, la mayoría de los programadores reconoce que vale la pena invertir esfuerzo en adquirir estos conocimientos debido a los resultados muy profesionales que se obtienen.

Capítulo 4

Descripción de la aplicación

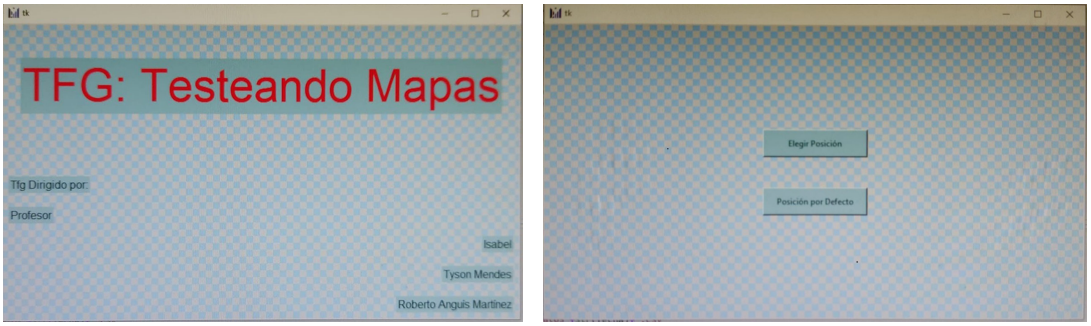
4.1. Diseño gráfico de la interfaz de usuario

El diseño de la interfaz es uno de los aspectos en los hemos encontrado mayor complejidad e inconvenientes a la hora de conseguir una estética sólida y depurada. Uno de los objetivos principales del proyecto era que el usuario pudiera manejar los datos de forma sencilla y que la interacción con el mapa fuera lo más intuitiva posible. Por ello, buscábamos un diseño limpio, natural y, sobre todo, claro, con elementos grandes y visibles que estuvieran perfectamente señalizados.

Al empezar el proyecto, tal y como mencionamos en el capítulo 3, nos decantamos por la librería *Tkinter* para la creación de la interfaz gráfica. Es el que se suele utilizar por defecto y uno de los más destacados a la hora de abordar el front end con Python. A base de prueba y error fuimos construyendo las primeras ventanas de la aplicación y definiendo la estructura. En la figura 4.1, podemos observar el primer diseño que desarrollamos tras varios intentos. Tratamos de representar una especie de portada de presentación del TFG seguida de una ventana de selección para que el usuario pudiera decidir dónde y cómo llevar a cabo las consultas de errores. Estéticamente, los colores resultaban demasiado tenues y homogéneos, y no llamaban la atención del usuario. La información no destacaba ya que carecía de contrastes y de una división clara del espacio. El fondo creaba cierta confusión al preservar la gama cromática de títulos y botones, lo que impedía la correcta identificación de los elementos importantes. En definitiva, estábamos más centrados en el qué que en el cómo, lo que nos llevó a cometer muchos fallos que tratamos de subsanar en los siguientes intentos.

A nivel conceptual, decidimos empezar a pensar en la aplicación como un objeto de mercado con posibilidad de expansión e incluso venta. Queríamos dejar de tratarlo como un Trabajo de Fin de Grado y enfocarnos en crear una imagen de marca reconocible por cualquier usuario en cualquier lugar. Lo primero que hicimos fue pensar en un nombre que hiciese referencia a mapas y errores y, tras una intensa lluvia de ideas, nos decantamos por "Mapflaws", cuya traducción literal es "defectos en el mapa". A continuación, tratamos de esbozar un logo cuyo diseño te permitie-

ra relacionarlo inmediatamente con aquellos símbolos del imaginario colectivo que aluden a mapas y búsquedas. Por último, nos centramos en el fondo, uno de los elementos que más protagonismo tendría y cuya estética debía encajar con la idea conceptual de la aplicación. Se nos ocurrió utilizar la imagen de un mapa y superponer un velo opaco de color de tal forma que se intuyesen las formas del mapa sin mostrarlas del todo.



(a) Presentación (b) Selección

Figura 4.1: Primer diseño de interfaz

Todos estos cambios pueden visualizarse en los diseños de la figura 4.2. En efecto, tras la primera experiencia decidimos hacer unos bocetos antes de introducir la idea en la aplicación. De esta forma podríamos discutir cualquier detalle sin necesidad de modificar el código constantemente. En cuanto al diseño, nos decantamos por colores básicos y sencillos, como pueden ser el blanco y el azul, introduciendo contrastes y dando a cada elemento su importancia y su lugar. La gama cromática debía respetarse a lo largo de la aplicación, por lo que buscamos un color que transmitiese confianza, honestidad, seguridad y mucha paz. Por ello, el fondo, que ocupa gran parte del diseño, es azul. El resto de elementos fueron colocados de la forma más estética posible intentando crear un conjunto armonioso.

(a) Presentación (b) Selección

Figura 4.2: Diseño final de interfaz

En lo que respecta a la ventana de selección, la cual puede observarse en la figura 4.2b, necesitábamos que los botones fueran más atractivos y llamativos ya que eran los elementos protagonistas del diseño. Como el fondo era azul, teníamos que escoger un color que aportase contraste sin perturbar la energía transmitida por la

imagen y optamos por el blanco. Manteniendo la esencia de los mapas, introdujimos una tipografía moderna y geométrica como es "Gilroy" y le añadimos el mismo tono del fondo. Para darle un aspecto más profesional y seguir con el trabajo de marca, decidimos colocar el logo en la esquina derecha de la ventana. De esta forma, el usuario irá tomando consciencia de la imagen de la aplicación y podrá, en un futuro, reconocerla automáticamente. Para conservar cierta coherencia estética y darle mayor conexión y compacidad al conjunto de ventanas, es importante mantener los aspectos básicos del diseño de una a otra, formando una unidad sólida.

4.2. Estructura y funcionalidad

La aplicación cuenta con un total de cinco ventanas que se van entrelazando mediante botones. La funcionalidad más importante reside en aquellas que muestran un mapa, pues es donde se accede a la base de datos y se llevan a cabo las consultas que se hayan solicitado. El resto tienen una función más bien estética o explicativa, la cual permite al usuario entender donde está y qué debe hacer en cada paso.

La primera ventana, tal y como muestra la figura 4.3, es una simple introducción a la aplicación. Su función es presentar la estética adoptada: el logo, la gama cromática, la tipografía, la distribución, las creatividades gráficas, en definitiva, la brand (imagen de marca). Y, aunque conseguimos recrear con todo lujo de detalles el diseño planteado en la figura 4.2a, decidimos añadir un objeto que informara al usuario de que todo estaba desarrollándose con normalidad. En efecto, para que pueda observar en detalle la presentación y no haya riesgo de que piense que tiene que accionar algo para activar el funcionamiento de la aplicación, decidimos colocar una barra de progreso. De esta forma, se pone en evidencia que el programa se está cargando y que en unos instante se abrirá una nueva ventana.

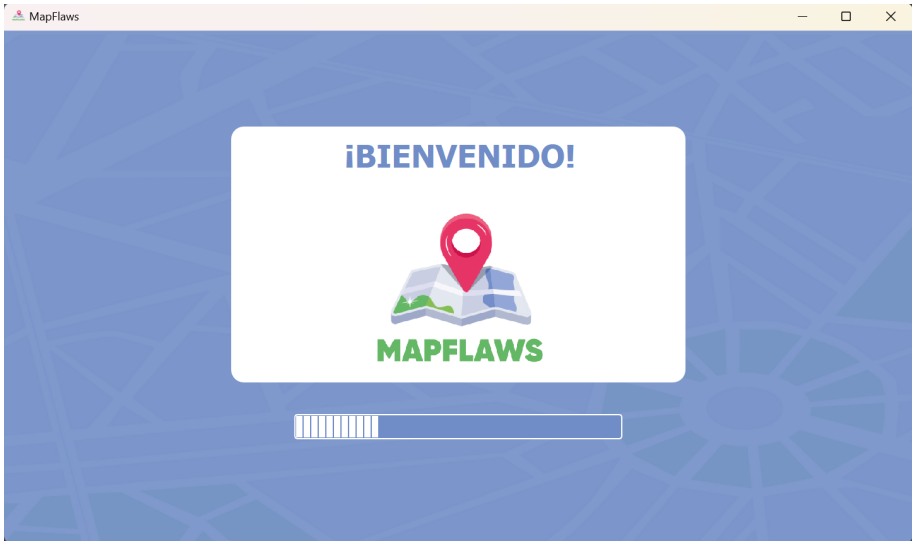


Figura 4.3: Ventana de presentación

Lo siguiente que aparece es la ventana de la figura 4.4 cuyo diseño se corresponde exactamente con el mostrado en la figura 4.2b. La brand se mantiene en la imagen de fondo, la tipografía, el logotipo y la gama de colores utilizada. En cuanto a la funcionalidad, esta ventana representa el menú de la aplicación al que se podrá volver en cualquier momento y desde cualquier punto. El objetivo es mostrar al usuario las distintas vías por las cuáles puede cargar el mapa OSM interactivo: utilizando una posición por defecto que se escoge de forma aleatoria entre los mapas de Madrid, Londres y São Paolo; introduciendo una posición concreta de cualquier parte del mundo; o seleccionando un mapa preexistente que guarde el usuario entre sus archivos. En función del botón pulsado, el camino será más o menos directo, pero el resultado siempre terminará en una ventana como la que aparece en la figura 4.6. Pensando siempre en la interacción con el usuario, y aunque en las imágenes no se pueda apreciar, cuando la flecha del ratón pasa por encima de cualquier botón e incluso cuando se pulsa, éste cambia su fondo blanco por uno azul claro y la flecha se convierte en una mano. De esta forma es más sencillo detectar los botones y las acciones que se están realizando sobre ellos.

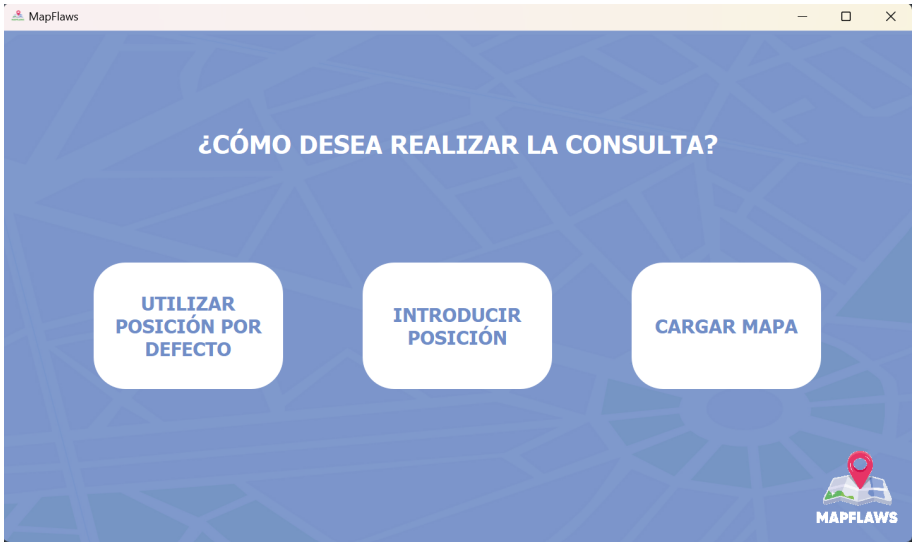


Figura 4.4: Ventana de menú

Antes de analizar la ventana de selección de filtros debemos detenernos en el comportamiento del botón “Introducir posición”. Al pulsarlo, aparece una ventana emergente con un cuadro de diálogo que solicita la dirección exacta en la que el usuario desea centrar la búsqueda. De nuevo, hemos añadido una serie de elementos que ayudan al usuario a entender qué debe hacer en caso de estar perdido. En este caso lo más importante es que los datos de la dirección sean correctos por lo que decidimos insertar un texto explicativo en color gris que sirviese de guía respecto al formato. Cuando el usuario pulsa sobre el cuadro de texto, el comentario gris desaparece y el rectángulo blanco es enfatizado con un borde negro, dejando clara la posición en la que se encuentra el cursor. Los botones de abajo tienen el mismo diseño que los del menú pero con un tamaño menor y sirven para continuar con el proceso, o cancelar y volver a la ventana anterior.

Figura 4.5: Ventana emergente para introducir una posición

Una vez escogida la forma de cargar el mapa, el usuario aparecerá en la ventana de selección de filtros que muestra la figura 4.6a. Es la primera vez que aparece el mapa interactivo y, teniendo en cuenta que el proyecto gira en torno a él, era importante otorgarle visibilidad. Por esta razón el mapa ocupa 3/5 partes de la ventana y el resto se reserva para los filtros tanto de errores como de áreas de búsqueda. Los filtros son de selección múltiple y permiten que el usuario ejecute varias pruebas metamórficas (MT) a la vez, obteniendo simultáneamente varios tipos de errores. Como interacción para el usuario añadimos, en el listado de filtros, un pequeño texto informativo sobre el tipo de errores que busca cada uno, de forma que aparezca cuando el ratón pasa por encima. Respecto a los filtros del área de búsqueda solamente se puede seleccionar una de las opciones ya que son excluyentes. Su objetivo es delimitar la zona de consulta proponiendo dos opciones: restringirla exclusivamente a la posición original o aumentar el radio en 100 metros desde esta misma posición original. Por defecto se marcará la segunda opción por lo que será el usuario el que deba marcar la otra casilla si lo desea.

(a) Ninguna selección (b) Una selección

Figura 4.6: Ventana de selección de filtros para la búsqueda

Un pequeño detalle añadido para evitar que se ejecute una búsqueda sin haber seleccionado previamente un filtro es la capacidad de que el botón "Ejecutar" esté habilitado o deshabilitado para ser pulsado. En la figura 4.6a se ve claramente que el botón tiene un color gris y, si pasáramos el ratón por encima, se mantendría en forma de flecha dando a entender que no se puede usar. En cambio, en la figura 4.6a, como se ha seleccionado el primer filtro, el botón ha recuperado su estado original. En cualquier caso, a su lado siempre aparece un botón de "Volver a menú" cuya función es precisamente volver a la ventana del menú para, si fuera el caso, poder cargar otro mapa.

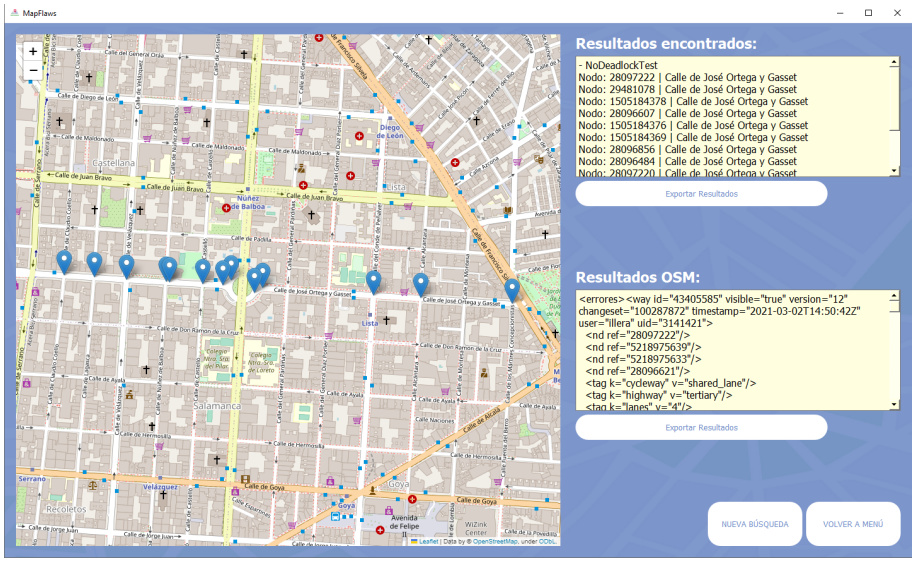


Figura 4.7: Ventana de resultados

La figura 4.7 muestra la ventana a la que será redirigido el usuario tras pulsar el botón de "Ejecutar". Si todo ha salido bien y se han encontrado errores, aparecerán marcados en el mapa y enumerados en los cuadros de texto del lateral. De lo contrario el mapa permanecerá igual que en la ventana anterior y los cuadros de texto aparecerán vacíos. El usuario tiene en total tres formas de ver los resultados: gráficamente sobre el mapa, como un listado de nodos, o en formato XML, estos dos últimos con posibilidad de ser exportados a PDF. Los botones mantienen el diseño establecido desde el principio y su interacción con el usuario. En este caso, además de los botones de exportación, tenemos dos botones en la esquina inferior derecha que nos informan de que posibilidad tenemos. El de "Nueva búsqueda" regresa a la ventana de selección de filtros manteniendo el mismo mapa OSM y la misma posición central, mientras que "Volver a menú" deja claro que se vuelve a la casilla de salida, pudiendo escoger un nuevo mapa interactivo.

4.3. Problemas encontrados

La mayoría de los problemas que nos hemos ido encontrando durante la realización de la aplicación ya han sido comentados. Por lo tanto vamos a tratar de

recuperar los más importantes y detenernos sobre aquellos que no han sido especificados previamente. Por suerte, no se han dado problemas personales dentro del grupo. Cada uno ha sabido encontrar su lugar y ha utilizado sus habilidades a favor del trabajo. Siempre surgen pequeñas desavenencias a la hora de tomar decisiones, pero nunca han sido tan grandes como para generar conflictos internos. Por lo tanto, vamos a centrarnos en analizar lo sucedido a nivel técnico en el desarrollo del código.

Tal y como aparece en los capítulos 2 y 3, la elección del lenguaje de programación no resultó nada fácil puesto que había que tener en cuenta muchos factores de vital importancia para el correcto desarrollo de la aplicación. Entre *JavaScript* y Python nos decidimos por este último ya que disponía de bibliotecas que nos facilitarían la comunicación con la base de datos y la interacción con el mapa OSM. Respecto a los mapas, el único problema realmente importante con el que nos hemos encontrado y para el que no existe solución es que no es posible cargar mapas grandes. BaseX tiene una capacidad de almacenamiento de datos finita y los mapas OSM demasiado grandes poseen un número muy elevado de nodos imposible de procesar.

En cuanto a la base de datos, al ser necesario enlazar la implementación heredada del trabajo Almendros-Jiménez et al. (2023) con la nuestra, tuvimos que realizar varias funciones auxiliares. Algunas de ellas nos dieron pequeños problemas, pero una que nos mantuvo parados durante varios días fue la encargada del acceso a una query concreta de la base de datos desde Python. Esta comunicación era fundamental para poder obtener los resultados y plasmarlos sobre el mapa. Para realizar este proceso utilizamos la función *session.query()* pero por alguna razón desconocida no lograba ejecutarse con éxito. Nos pusimos a investigar sobre cómo se realizaba una consulta sobre una base de datos desde Python y finalmente nos dimos cuenta de que estábamos cometiendo dos errores. Por un lado, la sintaxis del nombre de la consulta que estábamos transmitiéndole a la base de datos no era correcta y, por otro lado, sin ejecutar posteriormente el comando *session.execute()* el trabajo posterior no servía de nada. En relación con los problemas que nos ha ocasionado la base de datos, es importante mencionar un inconveniente que todavía a día de hoy sigue sin estar resuelto. Algunos de los miembros del grupo no logran crear una sesión en la base de datos por, supuestamente, no tener acceso. Por suerte uno de nosotros es capaz de acceder sin problemas pero, a pesar de haber verificado los parámetros y consultado la documentación una infinidad de veces, el problema persiste.

Si nos centramos ahora en los problemas que han ido surgiendo durante el desarrollo de la interfaz gráfica, el capítulo 3 comenta que hubo que elegir entre la librería *Tkinter* y *PyQt5*. El principal inconveniente de *Tkinter* fue la imposibilidad de introducir un mapa en una de sus ventanas, algo que era imprescindible para poder llevar a cabo la idea que habíamos planteado. La única solución factible fue cambiar a *PyQt5* y diseñar el resto de la aplicación con esta herramienta, obteniendo un resultado mucho más profesional.

4.4. Resultados obtenidos

La parte más esperada de cualquier proyecto informático es el análisis de los resultados obtenidos. En ese momento se decide si el tiempo invertido ha merecido la pena o no. En este caso, el trabajo realizado solo tenía sentido si la aplicación conseguía mostrar gráficamente cualquier tipo de error en cualquier parte del mundo. Las tablas 4.1, 4.2 y 4.3 muestran precisamente los resultados obtenidos gráficamente tras una amplia serie de pruebas en distintos mapas OSM cargados de distintas formas. Aparentemente todo sucede según lo esperado pues en cada caso se muestran errores distintos lo que indica que las búsquedas se están realizando correctamente. La primera tabla se centra en los errores encontrados en rotondas, es decir de tipo **EntRAbout** y **ExitRAbout**, en las ciudades de Madrid y Nairobi. En la segunda aparecen errores de tipo **EntranceWay**, **ExitWay** y **NoIsolatedWay**, todos situados en Londres y São Paolo. Por último, la tercera tabla muestra errores de tipo **NoDeadlock** y **Connected** encontrados en Madrid, São Paolo y Londres.

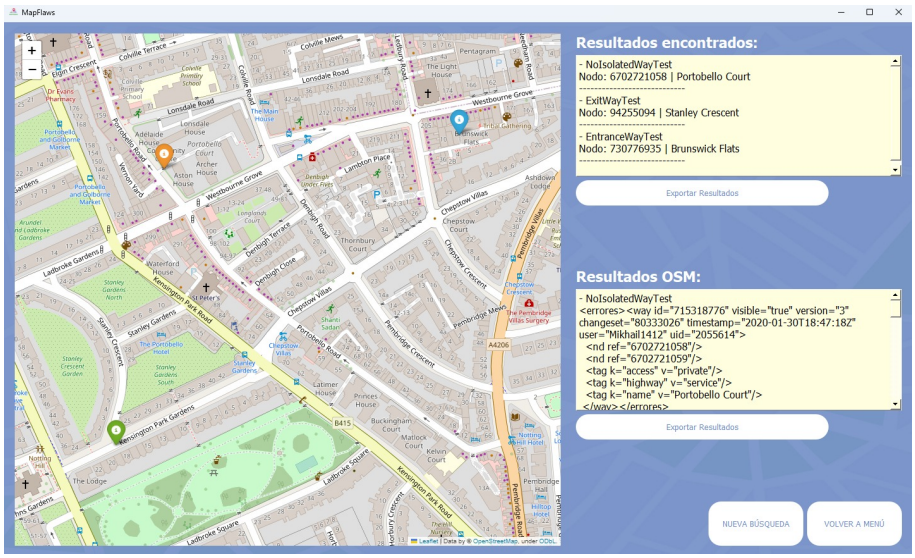


Figura 4.8: Ventana de resultados

Además de los resultados que aparecen en las tablas, cabe destacar la figura 4.8, que es la consecuencia de una selección múltiple de filtros. En efecto, los marcadores muestran errores de tipo **NoIsolatedWay** en color naranja, **ExitWay** en color verde y **EntranceWay** en color azul. El hecho de que cada tipo de error estuviera marcado con un color diferente fue un aspecto que quisimos resaltar especialmente para seguir con la idea de interfaz intuitiva e interactiva. Instintivamente el usuario entenderá que cada color está asociado a un tipo de error diferentes, además de aparecer de forma explícita en los cuadros de resultados situados a la derecha. Los colores siempre son útiles a la hora de diferenciar elementos de forma rápida y visual.

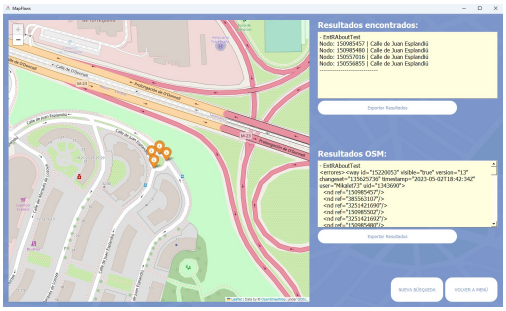
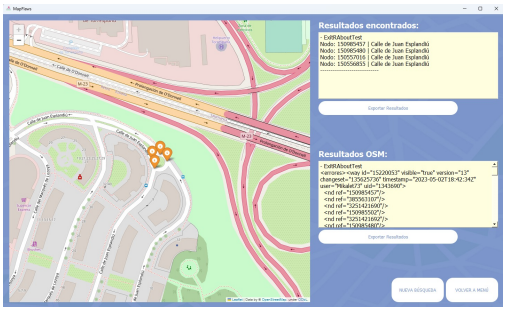
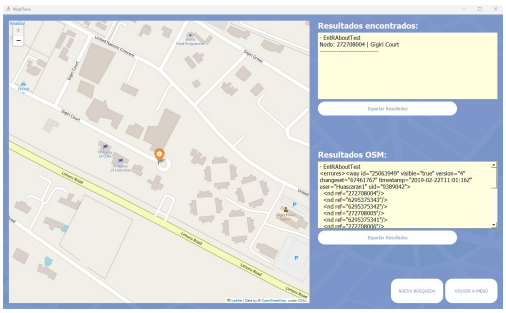
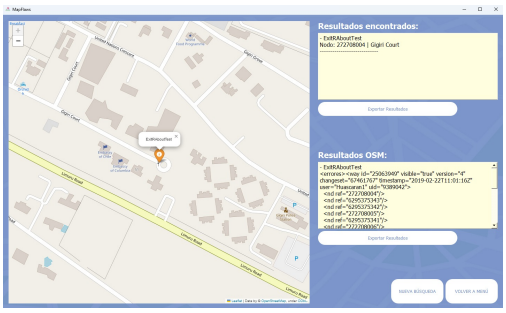
	EntRAbout	**ExitRAbout**
Madrid		
Nairobi		

Tabla 4.1: Errores del tipo **EntRAbout** y **ExitRAbout**

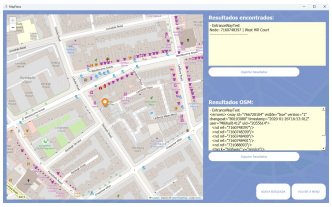
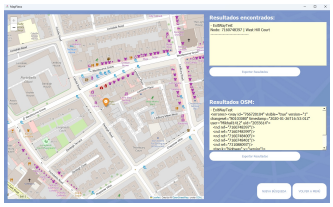
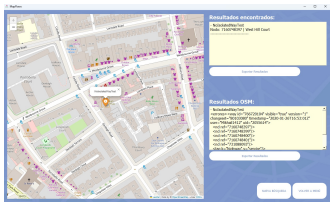
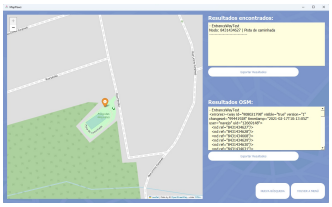
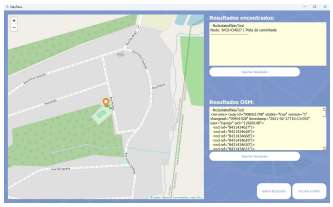
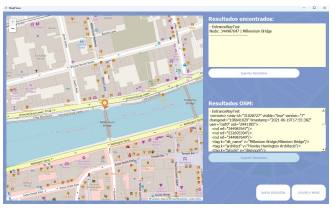
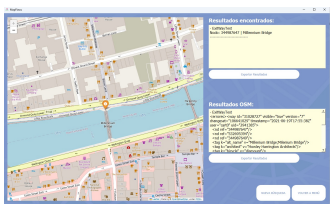
	EntranceWay	**ExitWay**	**NoIsolatedWay**
Londres			
São Paolo			
Londres			

Tabla 4.2: Errores del tipo **EntranceWay**, **ExitWay** y **NoIsolatedWay**

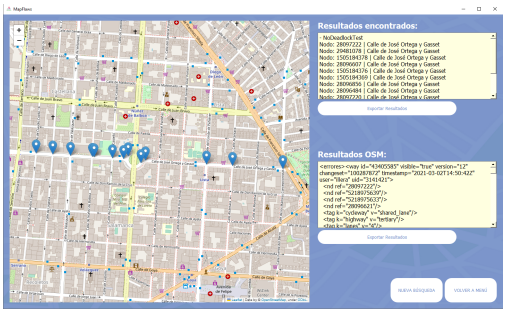
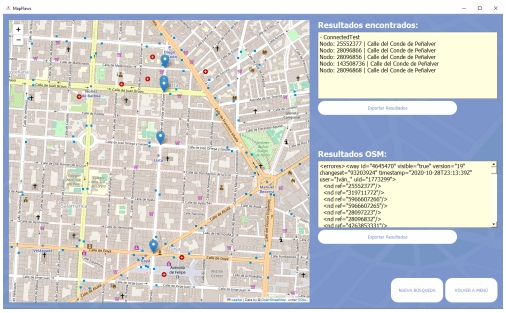
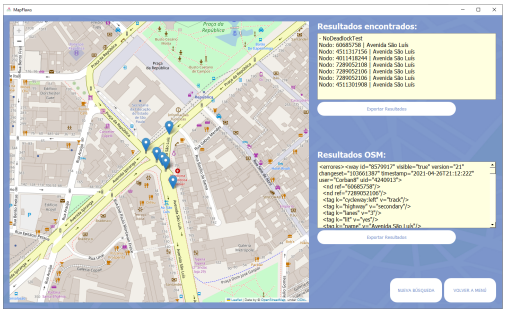
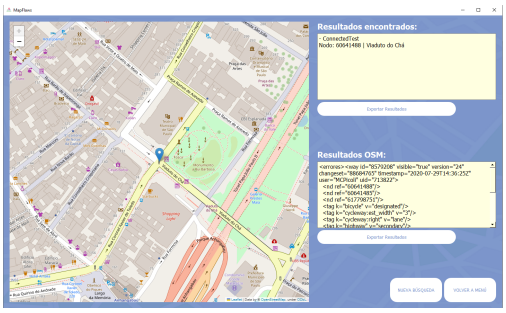
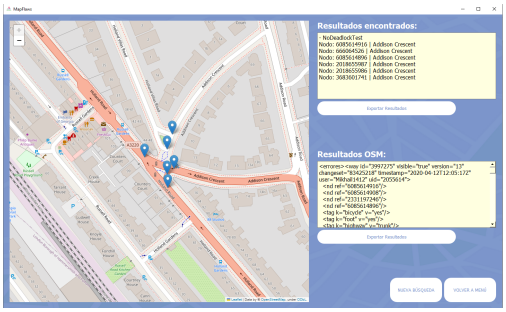
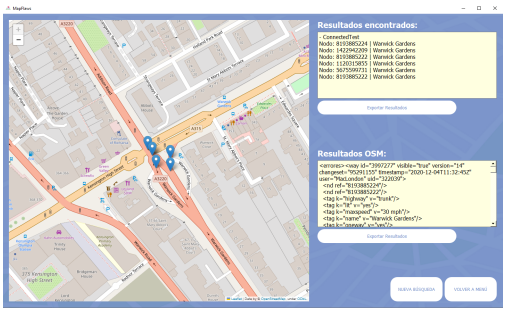
	NoDeadlock	Connected
Madrid		
São Paolo		
Londres		

Tabla 4.3: Errores del tipo **NoDeadlock** y **Connected**

De entre todos los errores encontrados y mostrados en las tablas 4.1, 4.2 y 4.3 faltan dos de los estudiados en la investigación previa: **AreaNoInt** y **NoOverlap**. En efecto, tuvimos muchos problemas a la hora de ejecutar este tipo de errores ya que parecían no acabar nunca y dejaban el programa totalmente colgado. Este contratiempo apareció de forma inesperada y sin poder preverlo pues dimos por hecho que los mapas utilizados en el artículo Almendros-Jiménez et al. (2023) habían sido verificados y se ejecutaban correctamente en cualquier mapa OSM. Resultó bastante frustrante no poder ver la aplicación funcionando a pleno rendimiento pero, dado que algunas de las XQuery si se estaban realizando, el problema no podía estar relacionado con la comunicación entre ambas partes. De hecho, si la ejecución de la MT no lograba finalizar, quería decir que la consulta seguía en curso, por lo que seguramente el conflicto lo estuviese causando el tamaño del mapa, demasiado grande para ser evaluado.

Para resolver el problema, decidimos realizar una serie de pruebas de escalabilidad con el objetivo de cerciorarnos del origen y saber cuál era su alcance exacto. Las tablas 4.4 y 4.5 muestran el tiempo de ejecución de las pruebas metamórficas (MT) **AreaNoInt** y **NoOverlap** respectivamente en función del tamaño del conjunto de datos introducido. El tamaño se mide en KB mientras que el tiempo aparece en ms, aunque cuando empieza a ser demasiado largo también se introduce su correspondiente notación en horas, minutos y segundos. Se puede observar como hemos ido aumentando progresivamente el tamaño de los conjuntos en función de la capacidad que veíamos que podía soportar cada uno. En el caso de **AreaNoInt** empezamos en 111 KB y fuimos progresando de 100 en 100 aproximadamente, sin embargo para **NoOverlap** tuvimos que empezar 20 KB ya que no conseguíamos ejecutar rápidamente conjuntos cercanos a la centena. En ambos casos el aumento del tiempo de ejecución es bastante exponencial, pegando un salto muy brusco en un momento dado en el que pasa de tardar unos minutos a varias horas. En la tabla 4.5 observamos un extraño descenso del tiempo de ejecución en los mapas de 93 KB pero enseguida se recuperan los tiempos que superan la hora.

Estas tablas demuestran que el origen del problema se encuentra en el tamaño de los mapas considerados y de la propia *exponencialidad* subyacente a estas dos relaciones metamórficas (MR). Teniendo en cuenta los tamaños necesarios para que las consultas pueden ejecutarse de manera óptima, resulta inviable encontrar algún error de este tipo con los ordenadores que utilizan la mayoría de los usuarios. Seguramente utilizando uno más potente se puedan obtener mejores resultados pero la aplicación está destinada a un consumo genérico por lo que su utilidad debe medirse en dichos términos. El resto de pruebas metamórficas (MT) consiguen ejecutarse en ms sea cual sea el tamaño del mapa analizado, sin embargo estas dos últimas necesitarán una cantidad enorme de tiempo para poder ejecutarse en mapas de dimensiones relativamente normales. Cabe destacar que estos problemas de escalabilidad ya estaban presentes en el trabajo Almendros-Jiménez et al. (2023) en el cual se basa este TFG y que, teniendo en cuenta que decidimos reutilizar su implementación de las MRs, era inevitable que se trasladasen a la aplicación.

Tamaño del mapa en KB	**Tiempo de ejecución en ms**
111	1274.08
229	8575.65
317	21574.67
477	374902.17 (00:06:14)
532	516656.58 (00:08:36)
602	7295534.36 (02:01:35)

Tabla 4.4: Prueba de escalabilidad de **AreaNoInt**

Tamaño del mapa en KB	**Tiempo de ejecución en ms**
20	420.16
31	1831.97
41	2607.91
51	58128.83
62	177465.29 (00:02:57)
71	105300.91 (00:01:45)
82	5934074.53 (01:38:54)
93	284367.59 (00:04:44)
102	3741316.03 (01:02:21)
111	4366379.3 (01:12:46)
121	21474836.47 (05:57:54)

Tabla 4.5: Prueba de escalabilidad de **NoOverlap**

Capítulo 5

Contribuciones Personales

5.1. Tyson Mendes de Graça

Durante la realización de este trabajo de fin de grado titulado "Testeando Mapas", me surgieron diversos problemas; el primero de estos apareció a la hora de iniciar el proyecto, ya que nos dificultó un poco la complejidad matemática de este. Una vez conseguimos comprenderlo y solucionarlo, procedimos al reparto de tareas entre los componentes del equipo de trabajo, siendo estas, dos, la parte funcional y la parte visual.

Mi trabajo se centrará primordialmente en la implementación de la parte funcional, debido a que es la que más interesante me resulta tanto de cara al futuro de mi carrera, como para el desarrollo de las que considero mis habilidades; además he participado en la resolución de problemas de la parte visual cuando ha sido necesario. Hasta ahora, me he sentido cómodo realizando proyectos en *C++* y *Java* a lo largo de mi aprendizaje, por eso trabajar con Python ha supuesto un gran reto para mí, ya que he tenido una formación desde 0 y autodidacta; todo esto me ha permitido aumentar mis conocimientos y sacar adelante el trabajo desde un escenario desconocido.

Para llevar a cabo mi contribución, me centré primero en analizar la investigación que se nos había proporcionado y seguir los pasos de la misma para crear una base de datos y encontrar los errores; para conseguir esto, fue necesario indagar en el funcionamiento de la herramienta BaseX, ya que sería mi medio para realizar la parte funcional; cuando me sentí completamente familiarizado, busqué la manera de cargar un mapa en Python y mostrar la visualización mediante el mismo. Investigando las funcionalidades del mapa, descubrí la manera de marcar los puntos a elección propia, lo que nos permitirá marcar los errores y hacerlos visibles, en función de su latitud y su longitud. Durante el desempeño del que creía era el siguiente paso: conseguir la lista de errores para situarlos en el mapa, realicé numerosas pruebas para enlazar las consultas xquery con el mapa mostrado; esto me hizo concluir que necesitaría almacenar sus nodos (map.osm) en una base de datos y realizar la consulta sobre la misma, lo que es precisamente la clave del funcionamiento de esta aplicación.

Posteriormente, realicé diferentes pruebas que me condujeron a diversas posibilidades de introducción de mapas en las bases de datos, tanto mediante descargas, como de forma manual o a través de la API de openstreetmap. Las conclusiones que saqué tras dicha investigación fueron que los dos métodos más adecuados para llevar a cabo este proceso serían el uso de la API, bien para hacer búsquedas por zonas o proporcionando coordenadas más precisas para seleccionar un cuadrante concreto, y la carga de los mapas desde el explorador de archivos de mi PC. Otras de las pruebas que realicé con la herramienta (API) me permitieron descubrir nuevas utilidades acerca de esta, tales como la extracción de información acerca de los nodos: obtención de coordenadas usando un ID, especificar nombres de calles, y otras características del nodo. Una vez realizado todo lo anterior, me dispongo a enlazar el archivo MT Geometry con el código implementado previamente por mí para hallar la lista de errores y guardarla en una variable. Durante la realización de este paso, me surgieron ciertos problemas que me impidieron continuar el proceso; una vez resueltos comprobé que la respuesta de las consultas realizadas era correcta, y tras obtener este resultado, diseñé una función que extrajese los datos necesarios del xml para poder marcar los puntos, compactando el xml. Simultáneamente, he añadido otras funciones adicionales, tales como el posicionamiento de la visualización del mapa en la calle concreta, cuyo nombre es indicado.

El último procedimiento de la parte funcional, fue la ejecución de testeos para la búsqueda de funcionalidades que me permitieran al usuario una selección dinámica a su gusto, y que además de detectar los clics, devuelva una lista de puntos marcados en el mapa que forman un área, que se introducirá en la base de datos.

Otra contribución fue llevar a cabo la fusión de la parte visual con la funcional junto a mis compañeros, y no surgieron muchos problemas. Uno de los principales inconvenientes que aparecieron se debió a la dificultad a la hora de introducir un mapa en una ventana *Tkinter*, y la solución que hallamos fue reemplazar esta biblioteca por *PyQt5*, que si nos permitía hacerlo, además de facilitar un diseño más óptimo.

Los últimos avances que realicé fueron modificaciones del trabajo anteriormente realizado, como por ejemplo, la carga en la base de datos de secciones de mapa dando un punto concreto y el radio, devolver la lista de errores y el xml, y otros.

Por lo tanto, este proyecto ha sido de mucha utilidad para mi desarrollo profesional como programador, tanto por los imprevistos como por el aprendizaje y contacto con nuevas herramientas y lenguajes de programación.

Además, el trabajo cooperativo con mi equipo me ha aportado habilidades de trabajo en equipo, comunicación y escucha activa, y constancia con mi trabajo, destrezas que estoy poniendo en práctica en mi ámbito laboral; por tanto, esta tarea me parece un buen broche final para mi etapa universitaria.

5.2. Roberto Anguís Martínez

Para el desarrollo de nuestro trabajo de fin de grado titulado "Testeando Mapas", al ser un grupo nos dividimos el trabajo para investigar y realizar diferentes de manera simultánea, encontrándonos diferentes problemas a lo largo del desarrollo del mismo. El primer problema con el que nos encontramos, nada más comenzar el proyecto fueron las funciones matemáticas y relaciones metamórficas que se habían utilizado en el proyecto que teníamos como punto de partida, debido a su dificultad; con ayuda de nuestro tutor pudimos resolver dichas dudas y comenzar a repartirnos el trabajo, separando este en dos partes, la parte de la implementación de las funciones que extraerán los resultados de la base de datos y la parte visual; siendo en esta segunda en la cuál centre la mayoría de mi trabajo, apoyando en caso de ser necesario con cualquier problema en la otra parte.

El trabajo realizado se centró en la parte visual, ya que es la parte del proyecto que más me interesaba, puesto que el desarrollo de una interfaz de usuario de una aplicación es algo que me puede servir en un futuro profesional y hacia dónde quiero ir. Mis estudios durante la carrera me han permitido realizar las tareas, ya que pese a no haber trabajo durante la misma con el lenguaje que hemos utilizado,si habíamos trabajado con otros como pueden ser *Java* o *C++*, lo cuál me ha permitido extrapolar los conocimientos y técnicas a este lenguaje que no conocía, permitiendo ampliar los lenguajes conocidos.

Una vez repartidos los trabajos el primer paso que debía tomar era el seleccionar como queríamos desarrollar el programa y si íbamos a realizar una aplicación de escritorio, o una aplicación web. Para ello me puse a investigar diferentes opciones que teníamos, llegando a considerar tres ideas principales; la primera de ellas era realizar una aplicación web mediante un *GeoServer*, y las otras dos serían realizar una aplicación de escritorio utilizando diferentes lenguajes de programación siendo las opciones *Java* o Python. Por lo que teniendo estas tres posibilidades fue necesaria la investigación de los beneficios y problemas que acarreaban cada una de ellas.

La opción de los geoserver fue mi idea propuesta para realizar la aplicación como una aplicación web. Destacando la facilidad para utilizar mapas renderizados, su fácil uso en páginas web y la capacidad de utilizar e integrar diferente información geográfica, por otro lado esta idea traía consigo muchos problemas como la cantidad de conocimiento requerido, la necesidad de realizar el estilo de los mapas manualmente y un despliegue complicado. Por lo que al poner en conocimiento del equipo la idea la descartamos en favor de realizar una aplicación de escritorio, para lo que fue necesario investigar diferentes opciones siendo la propuesta final realizar el código en Python, debido a que no era un lenguaje que hubiéramos utilizado durante la carrera y nos permitiría ampliar nuestro conocimiento y llevar a un nuevo lenguaje lo aprendido durante la carrera, ya que en mi caso no había utilizado Python.

Una vez decidido cómo se iba a desarrollar la aplicación era necesaria una investigación sobre qué librerías iban a ser necesarias para realizar las diferentes opciones

y procesos de nuestra aplicación. Como mi trabajo consistía en realizar la parte visual, comencé investigando las librerías de *Tkinter* y *PyQt5* y realizando con ambas diferentes pruebas que me permitían crear diferentes ventanas, que mediante la interacción con el usuario nos permite navegar por la aplicación y sus diferentes funcionalidades. Las primeras pruebas con ambas librerías se centraron en la capacidad de ambas de modificar visualmente las diferentes ventanas y elementos que se incluían en ellos, además de cómo interaccionar entre ellos los diferentes elementos que vamos creando mientras desarrollamos el diseño.

Los tres principales problemas que me encontré durante las diferente pruebas realizadas fueron los siguientes:

1. Me encontré con que al tratarse de un nuevo lenguaje no tenía del todo claro como debería de organizarse cada una de las diferentes secciones que utilizamos durante la aplicación, por lo que trás investigar y haciendo uso del conocimiento adquirido programando en otros lenguajes, decidí realizar una separación en clases en función de las diferentes vistas de nuestra aplicación. Lo cuál nos dejó con cuatro clases, tres donde teníamos nuestras vistas principales y otra donde se encontraban funciones auxiliares.

2. Mientras desarrollaba y hacía pruebas con las capacidades de ambas librerías, para el diseño gráfico me encontré con que la librería de *Tkinter* me permitía menor capacidad de diseño y flexibilidad en el mismo, lo cuál junto al tercer punto me llevó a decantarme finalmente por el uso de *PyQt5*.

3. El principal problema que me encontré fue la implementación de los Widget que iban a almacenar los diferentes mapas. Fueron necesarias diferentes pruebas con diferentes librerías para comprobar cómo integrar los mapas, decantándonos finalmente por folium, la cuál se integraba perfectamente con *PyQt5*, pero me daba ciertos problemas con *Tkinter*, por lo que sumado al anterior problema nos hizo descartar su uso a la hora del diseño visual en favor de *PyQt5*.

Tras diferentes pruebas con ambas librerías y poner en consenso con el equipo los resultados nos decantamos por utilizar *PyQt5*. Mi siguiente paso una vez realizadas las pruebas y cerradas las herramientas a utilizar era realizar las diferentes ventanas y transiciones entre ellas, además de las diferentes funcionalidades que tendrían cada una de ellas. A continuación se explican algunas funcionalidades desarrolladas:

1. Me parecía interesante crear una primera ventana de saludo, como tienen la mayoría de las aplicaciones, la cuál nos daría paso al cabo de un tiempo a la aplicación en sí.

2. Una de las funcionalidades que más interesante me han parecido a la hora del desarrollo ha sido la interacción entre la parte visual y la parte funcional con respecto a la base de datos. En este caso era necesario recoger las diferentes opciones o posibilidades que quería darle al usuario y facilitarles al código

que interactuaba con la base de datos; para lo cúal he utilizado diferentes disposiciones gráficas como pueden ser selecciones múltiples, o únicas. Lo cuál trabajando con la librería de *PyQt5* se puede realizar mediante *ChechBoxes* o *RadioButton* respectivamente. De esta manera recogimos los datos y trás interpretarlos se le pasaban a las funciones que interaccionan con la base de datos para poder ejecutar las consultas sobre ella.

3. Tras realizar las consultas a la base de datos era necesario mostrar los resultados, además de permitir la capacidad de visualizarlos en diferentes formatos y almacenarlos, para posibles pruebas futuras. Por tanto se crearon varias vistas de los mismos resultados, con la capacidad de realizar una exportación de los mismos a PDF.

Durante el desarrollo del trabajo de fin de grado he podido poner a prueba lo aprendido durante la carrera de diferentes maneras, ya sea directamente gracias a los conocimientos adquiridos o extrapolando los mismo y los diferentes métodos y técnicas aprendidos a otro lenguaje. Además el haber trabajado en grupo anteriormente me ha facilitado el proceso de separar las tareas y dividir los diferentes procesos a realizar. Por otro lado, he aprendido cómo trabajar bajo las demandas de un proyecto, lo cuál me servirá en mi futuro profesional; de cara al cuál gracias al proyecto he potenciado tanto mis habilidades individuales sin perder la perspectiva global, la coordinación y la comunicación con el equipo.

5.3. Isabel Pérez Pereda

Partiendo de que se trata de un trabajo en grupo, hemos logrado coordinar nuestros conocimientos y esfuerzos para conseguir un resultado serio y satisfactorio. Cada uno aportaba una cualidad diferente y creo que en mi caso la labor global ha sido de coordinación y unificación. Desde el primer momento traté de organizar las tareas y aportar soluciones cuando nos encontrábamos con un problema.

Desde el punto de vista de estudios realizados, dado que pertenezco al Doble Grado de Matemáticas e Ingeniería Informática, podía aportar algo más en el ámbito matemático. Por ello, me ofrecí a analizar en detalle la investigación previa de la que debíamos partir y estudiar cada una de las relaciones metamórficas que se presentaban con el objetivo de tener una base sólida sobre la que trabajar. Esto nos ayudaría a comprender cuál era nuestro punto de partida y cómo funcionaba el mecanismo de búsqueda de errores. Estuve leyendo el artículo con detenimiento e investigando la anatomía de los nodos de los mapas OSM para poder posteriormente transmitírselo a mis compañeros.

Durante el desarrollo de la aplicación he ejercido de pilar de apoyo para mis compañeros. En efecto, Tyson se iba a encargar del back end y Roberto del front end, por lo que me ofrecí a ayudar en ambos sitios, teniendo además una visión global de lo que había tanto por detrás como por delante lo que me permitía analizar los problemas que iban surgiendo de forma integral. Sin embargo, conforme íbamos avanzando

me iba dando cuenta de que mi verdadero campo de apoyo estaba en la parte visual, cuyo avance iba bastante retrasado. Cuando nos rechazaron el primer diseño de la interfaz gráfica, decidí implicarme más activamente en esta parte del proyecto, en la que creía que realmente podía aportar una visión fresca, diferente y novedosa. Para ello, realicé una serie de bocetos en los que me centraba en la creación de una imagen de marca profesional pensando en un nombre para la aplicación, construyendo un logo y definiendo la tipografía y gama cromática que debía utilizarse en todas las ventanas. Se lo propuse a mis compañeros como alternativa a los diseños actuales y tuvieron una gran acogida por lo que en seguida comenzamos a modificar el código.

A partir de ahí, trabajé mano a mano con Roberto para desarrollar con todo lujo de detalles la interfaz gráfica. Rehice por completo la ventana del menú para que los elementos tuvieran un aspecto más profesional y un acabado más pulido. Tuve muchos problemas con los botones ya que el texto no se adaptaba a los márgenes del mismo y el diseño se perdía por completo. Tras documentarme en profundidad sobre los atributos de este tipo de widgets y consultar varios artículos, conseguí resolver los inconvenientes y crear una clase con las características que necesitaba en los botones. De esta forma podría construir todos los botones que quisiera manteniendo el diseño establecido. También tuve problemas con la disposición de los elementos en la ventana ya que nunca quedaban colocados como yo quería. Descubrí que este era uno de los problemas más comunes al utilizar la biblioteca *PyQt*, pero poco a poco fui consiguiente el aspecto que buscaba. Por último, tuve la iniciativa de color el logo de la aplicación en la esquina inferior izquierda de la ventana para contribuir a crear una imagen de marca y obtener un aspecto más profesional. El cuadro de diálogo en el que se solicita la dirección para situar el mapa era otro de los elementos que causaba problemas. Para conseguir el diseño que buscábamos, utilicé la estrategia de la creación de una clase en la que asigné a cada atributo el valor que le correspondía. Tras muchas pruebas me fui dando cuenta de que la mayoría de los problemas de diseño se podían solucionar utilizando el método *setStyleSheet*, el cuál poseían todos los widget y abarcaba una infinidad de opciones.

Por otro lado, aunque la interacción con el usuario siempre ha sido un aspecto importante en una aplicación, en nuestro caso era primordial ya que una de las premisas era que no fuese necesario disponer de conocimientos informáticos para poder utilizarla. Por ello, decidí encargarme de que la interfaz fuera lo más intuitiva y fácil de entender posible. Aspectos como el cambio de color y de flecha del ratón al pasar por encima de un botón, los cuadros informativos que aparecen al pasar por los filtros o el texto gris que sirve de plantilla a la hora de introducir la dirección, fueron pensados por mi para guiar al usuario. Pueden parecer detalles, pero resultan de gran ayuda. Otra de mis aportaciones fue la implantación de una barra de progreso en la ventana de presentación pues anteriormente se mantenía la imagen durante seis segundos y pasaba automáticamente al menú. A no ser que seas el programador, es imposible entender lo que está sucediendo y nada se mueve y nada es accesible. Se me ocurrió entonces que una barra de progreso podría simular un efecto de carga y mantendría al usuario a la espera.

Durante el proceso de enlace entre el front y back end, trabajé activamente junto con mis compañeros en la resolución de los problemas que iban surgiendo. El hecho de que cada uno tuviera una versión diferente del código nos generó muchos problemas, pero con paciencia logramos reconstruirlo todo. En esta parte del proceso la colaboración era esencial puesto que ninguno tenía conocimiento pleno del código. Fuimos avanzando conjuntamente de forma que no se perdiese ni olvidase ningún elemento y finalmente logramos que la aplicación funcionase correctamente.

Como tarea final me encargué de la redacción de la memoria del proyecto. Aunque conocía de primera mano todo el proceso por el que habíamos pasado, fui recopilando aquellos detalles técnicos de los que se había encargado cada uno. En primer lugar decidí la estructura global a seguir y poco a poco fui completando los huecos de forma esquemática con todas las ideas que quería introducir. Me ha resultado un proceso complejo en cuanto a que debía resultar comprensible para cualquier persona que lo leyera y debía permitir entender las pautas que habíamos seguido en cada etapa. Redactando la memoria empecé a ser consciente del trabajo realizado y de las posibilidades futuras que podría tener la aplicación. Me di cuenta de la viabilidad de este Trabajo de Fin de Grado y sus posibilidades comerciales pudiendo servir de gran ayuda a los ciudadanos, lo que me llevó a tratar de mejorar, en la medida de lo posible, su usabilidad y sobre todo su interfaz gráfica.

Este proyecto me ha resultado muy interesante para mi crecimiento en este campo. Me ha aportado nuevos conocimientos profesionales y me ha ayudado a mejorar mis habilidades de mediación, teniendo que ponernos de acuerdo en cada aspecto para elaborar una interfaz con la que todo el equipo se sintiese satisfecho. Además, las personas con las que he trabajado me han proporcionado nuevas visiones de la programación y la informática, tanto en el back end como en el front end. De hecho, he descubierto que este último aspecto me llama especialmente la atención y creo que puedo tener un gran desarrollo en el campo de la comunicación.

Ha sido una tarea muy enriquecedora para finalizar esta etapa universitaria y comenzar a encaminarme en mi trayectoria profesional.

Capítulo 6

Conclusiones y futuras aportaciones

6.1. Conclusiones

A pesar de las dificultades encontradas en el camino, pensamos que el objetivo principal del Trabajo de Fin de Grado ha sido completado con éxito, pues hemos conseguido crear una aplicación funcional e interactiva con utilidad real para la sociedad. Las necesidades básicas están cubiertas: el diseño gráfico es sencillo e intuitivo, la base de datos es accesible, las consultas XQuery se ejecutan correctamente y los resultados se muestran gráficamente sobre el mapa. Evidentemente esta no es más que una primera versión con mucha capacidad de mejora, pero la aplicación funciona y hace lo esperado.

A lo largo del proyecto han aparecido muchas dudas, problemas y dificultades, pero hemos conseguido solucionarlas o, en su defecto, plantear alternativas viables. El haber conseguido enlazar la base de datos BaseX con el mapa OSM interactivo para poder mostrar los errores visualmente, es uno de nuestros logros más destacados y sin el cual no se habría podido desarrollar el resto de la aplicación. De hecho, fue uno de los subobjetivos que nos propusimos cumplir al inicio del proyecto al descubrir que disponíamos de una implementación de las relaciones metamórficas completa y verificada. Sin embargo, también hay que comentar los aspectos negativos porque no todos estos subobjetivos han logrado llevarse a cabo. A nivel funcional, hay muchas opciones que, a pesar de haberse probado numerosas veces e investigado a fondo la documentación relacionada, no hemos conseguido integrar en el prototipo final, como por ejemplo la selección dinámicas de áreas de búsqueda. En cuanto a la parte gráfica, somos conscientes de que hay muchas posibilidades de mejora y de que no hemos alcanzado el nivel interactivo que nos hubiese gustado. La aplicación resulta bastante intuitiva para el usuario pero debería incorporar más facilidades en cuanto a la carga de mapas, la selección de zonas específicas o incluso la visualización de los resultados.

Por otro lado, debemos mencionar el problema ocurrido con las consultas a la base de datos. Fue algo totalmente inesperado que no habíamos ni si quiera planteado que pudiera pasar ya que en principio so la aplicación mostraba los errores

de un tipo, debería mostrarlos todos. Surgió en el último momento, justo cuando pensábamos que habíamos conseguido enlazar todas las funcionalidades con sus respectivos widgets y formado una unidad sólida y sin fallos técnicos. Aparentemente el fallo no emanaba de nuestro código, pero a nivel usuario daba la sensación de que la aplicación era lenta y no conseguía procesar grandes cantidades de datos. Esta incidencia nos ha enseñado que por mucho que tengas controlada la situación los problemas pueden aparecen en cualquier momento. De hecho, en realidad es imposible controlarlo todo. También nos ha permitido entender que, para que todo funcione correctamente, cada una de las partes activas del proyecto debe funcionar correctamente, así como mantener una comunicación dinámica, clara y eficaz entre sí. De esta forma pudimos encontrar el origen del problema, abordando un estudio en profundidad de la máxima escalabilidad del mapa OSM con la que se podían ejecutar las consultas.

A nivel conceptual, hemos proporcionado una forma bonita y sencilla de conseguir que los mapas OSM, fuente indiscutible de consulta y seguimiento, estén libres de errores que puedan llegar a producir situaciones de riesgo. En una sociedad totalmente mimetizada con el uso de la tecnología, no es de extrañar que se recurra diariamente a aplicaciones como esta para resolver problemas. Teniendo en cuenta además que OpenStreetMap es un proyecto colaborativo, a través de nuestra aplicación se podrían corregir fácilmente los fallos geométricos de cualquier parte del mundo, incitando a la participación ciudadana. Se trata por tanto de una herramienta sin límite geográfico que podría conseguir obtener un mapa actualizado de forma inmediata, lejos de las actualizaciones anuales a las que nos tienen acostumbrados.

6.2. Futuras líneas de trabajo

La aplicación realizada se encuentra todavía en su primer estadio de desarrollo y su margen de mejora es bastante amplio. Por ello, resulta sencillo pensar en futuras líneas de trabajo tanto a nivel de back end como de front end. Respecto a la carga de mapas OSM, sería interesante tener una especie de repositorio con zonas especialmente conflictivas al que pudiera acceder el usuario. También podrían incluirse las capas de las que dispone OpenStreetMap en las que se muestran rutas ciclistas o de transporte y poder realizar las consultas directamente sobre esta base de datos. Por último, una línea de trabajo que requeriría bastante tiempo de diseño e implementación pero que podría llegar a tener un gran impacto sería la posibilidad de crear perfiles de usuario. Cada uno podría tener búsquedas preguardadas a las que, por las razones que fueran, necesitase acceder rápidamente. También sería interesante poder comparar varias búsquedas entre sí y así poder analizar la corrección de errores a largo plazo.

A nivel visual las mejores son numerosas, pero solamente vamos a mencionar las que nos parecen más interesantes. De forma generalizada, se podría trabajar mucho más en el diseño intentando que sea más dinámico e interactivo para el usuario.

Por ejemplo, la selección de filtros podría convertirse en una sucesión de ventanas completamente guiadas en las que se vayan planteando las opciones y el usuario vaya escogiendo una a una. Pero para que esto tuviera realmente sentido habría que añadir bastantes filtros para concretar lo máximo posible la búsqueda. En este misma línea, se podría revisar el diseño de la ventana de selección de filtros para que el mapa ocupase la globalidad del espacio y el resto de opciones se situasen encima de él, que es básicamente lo que sucede en la mayoría de las aplicaciones que trabajan con mapas.

Por otro lado, dejamos como futura aportación una de las ideas que no conseguimos llevar a cabo durante nuestro proyecto. Se trata de la selección dinámica sobre el mapa cuyo objetivo era que los filtros seleccionados solamente fueran aplicados a las zonas resaltadas por el usuario, pudiendo ser una única zona o varias. Ya comentamos anteriormente que utilizamos el plugin *Draw* de la librería *folium* para agregar esta funcionalidad al mapa, pero no encontramos la manera de devolver el perímetro de las zonas resaltadas ni las coordenadas de los vértices. Al realizarse de forma dinámica sobre el html no tenemos acceso a los atributos de las figuras creadas y por consiguiente no podemos guardarlos en una variable en tiempo real. Sin embargo, la idea resulta muy interactiva y visual, por lo que en un futuro se podría volver a intentar.

Todas las líneas planteadas resultan interesantes, pero la más útil respecto al propósito inicial de este estudio es ampliar la base de datos con nuevas relaciones metamórficas (MR) que analicen otro tipo de propiedades de los mapas OSM. En concreto, podríamos añadir aquellas de la investigación previa que buscaban errores de *tags* y todas las que aparecen en el artículo Almendros-Jiménez et al. (2023). De hecho, se podrían incluir todas las búsquedas que se quisieran siempre y cuando se pudiese implementar una MR que las encontrase. También sería interesante ir ampliando y mejorando la información que abarca cada nodo, consiguiendo que los mapas fuesen cada vez más precisos e incluyesen información más contrastada.

Chapter 7

Introduction, conclusions and future work

7.1. Introduction

Introduction to the subject area, motivation and objectives of the work. This section contains the translation of Chapter 1.

The use of open source software is an increasingly expanded strategy and seems to be going through one of its best stages. Its particularity is based on being able to be used and modified by any user without any type of restriction, since it is freely accessible. And although at first sharing and exposing a code that had taken so much effort and dedication to generate seemed truly crazy, this philosophy has more followers every day. It is a practice that allows us to adapt to the specific needs of each user and, above all, it keeps the code alive by being constantly maintained and updated.

Some world-famous examples of open source software are the Linux operating system, the Apache OpenOffice office suite, the Moodle educational platform or the OpenStreetMap (OSM) digital mapping project. The latter represents one of the most important open source geospatial databases on the market. It is a completely open and collaborative platform that, thanks to the diversity and plurality of its users, provides a considerably accurate and updated map of the world. However, this extensive network of collaborators whose additions are not supervised can also lead to the appearance of errors that cause certain inconsistencies.

A study carried out in collaboration between the University of Almería and the Complutense University of Madrid, has managed to generate a theoretical framework that identifies the erroneous information of any OpenStreetMap, in addition to providing its respective free access implementation. In this project we intend to use this research to create an application that allows to find and spatially locate any type of error on the map. In this way, the identification of inconsistencies will be within the reach of any user, regardless of their computer knowledge. Considering that we already have an implementation of the theoretical framework, we will focus on how to visually show what the research did at the conceptual level.

7.1.1. Motivation

The 21st century is the century of modernity, technology and immediacy. Everything that is not computerized and published on the net does not exist, and the easier the access, the much more repercussions it will have. This paradigm shift has filled our lives with electronic devices, social networks, artificial intelligence and computer applications of all kinds. Therefore, there is no better way to bring an investigation to life than through an application that makes all the information discovered accessible.

In this case, the study carried out presents a quick and simple way to find the errors present in an OSM map. It is based on a more generic tool that is proving its usefulness when testing software in systems that do not have an oracle or whose execution is too computationally expensive. The development of an application that displays all this information in a visual way would make it possible to easily correct errors and expand the field of accessibility.

7.1.2. Objectives

The main objective of the work is to create a user-level application that is as simple and intuitive as possible. On a functional level, it should be able to find the faulty information in any OSM map and display it visually and accurately on the map itself. As for the interactive part, the most important thing is that it be easy to use so that it can be used by all types of audiences, regardless of age or knowledge. In order to make the project as complete and accurate as possible, we will also seek to meet the following specific objectives:

- To transfer the spirit of the previous research to the application, trying to make the user perceive it as a powerful tool to help in the search for erroneous information and therefore to evaluate the quality of the map in question.
- Generate a visual image as clear and clean as possible of the different errors found in a given OSM map showing also their respective position and attributes.
- Analyze the database provided by the previous research and study both the most convenient way to access it and the most appropriate way to integrate the existing implementation within the application.
- Design a graphical interface that is intuitive and interactive for the user, always prioritizing its comfort and the perfect understanding of the functions of each of its elements.

7.2. Conclusions

Conclusions of the entire work. This section contains the translation of section 6.1 of Chapter 6.

Despite the difficulties encountered along the way, we believe that the main objective of the Final Degree Project has been successfully completed, since we have managed to create a functional and interactive application with real utility for society. The basic needs are covered: the graphic design is simple and intuitive, the database is accessible, the XQuery queries are executed correctly and the results are displayed graphically on the map. Obviously this is only a first version with a lot of room for improvement, but the application works and does what is expected.

Throughout the project there have been many doubts, problems and difficulties, but we have managed to solve them or, failing that, to propose viable alternatives. Having managed to link the BaseX database with the interactive OSM map to be able to show the errors visually, is one of our most outstanding achievements and without which it would not have been possible to develop the rest of the application. In fact, it was one of the sub-goals we set out to accomplish at the beginning of the project when we discovered that we had a complete and verified implementation of the metamorphic relationships. However, we must also comment on the negative aspects because not all of these sub-objectives have been achieved. At the functional level, there are many options that, despite having been tested numerous times and thoroughly researched the related documentation, we have not managed to integrate in the final prototype, such as the dynamic selection of search areas. As for the graphical part, we are aware that there is a lot of room for improvement and that we have not reached the interactive level we would have liked. The application is quite intuitive for the user but it should incorporate more facilities in terms of loading maps, selecting specific areas or even visualizing the results.

On the other hand, we must mention the problem that occurred with the database queries. It was something totally unexpected that we had not even considered that it could happen because in principle if the application showed errors of one type, it should show all of them. It came up at the last moment, just when we thought we had managed to link all the functionalities with their respective widgets and formed a solid unit without technical glitches. Apparently the bug did not emanate from our code, but at user level it gave the feeling that the application was slow and failed to process large amounts of data. This incident has taught us that no matter how much you have the situation under control, problems can appear at any time. In fact, it is impossible to control everything. It has also allowed us to understand that, for everything to work correctly, each of the active parts of the project must work correctly, as well as maintain a dynamic, clear and effective communication between them. In this way we were able to find the source of the problem, addressing an in-depth study of the maximum scalability of the OSM map with which the queries could be executed.

At a conceptual level, we have provided a nice and simple way to make OSM maps, the undisputed source of consultation and monitoring, free of errors that could lead to risky situations. In a society that is totally immersed in the use of technology, it is not surprising that applications like this one are used on a daily basis to solve problems. Bearing in mind that OpenStreetMap is a collaborative project, through

our application it would be easy to correct geometric faults anywhere in the world, encouraging citizen participation. It is therefore a tool without geographical limits that could get an updated map immediately, far from the annual updates to which we are accustomed.

7.3. Future work

Future lines of work that we would like to develop. This section contains the translation of section 6.2 of Chapter 6.

The application is still in its first stage of development and its margin for improvement is quite wide. Therefore, it is easy to think of future lines of work both at back end and front end level. Regarding the loading of OSM maps, it would be interesting to have a kind of repository with particularly conflicting areas that could be accessed by the user. It would also be possible to include OpenStreetMap layers showing cycling or transport routes and be able to perform queries directly on this database. Finally, a line of work that would require a lot of design and implementation time but could have a great impact would be the possibility of creating user profiles. Each user could have pre-saved searches that, for whatever reason, they need to access quickly. It would also be interesting to be able to compare several searches with each other and thus be able to analyze the correction of errors in the long term.

At a visual level, the best ones are numerous, but we are only going to mention the ones we find most interesting. Generally speaking, much more work could be done on the design, trying to make it more dynamic and interactive for the user. For example, the selection of filters could become a succession of completely guided windows in which the options are presented and the user chooses one by one. But for this to really make sense, it would be necessary to add many filters to make the search as specific as possible. In this same line, the design of the filter selection window could be revised so that the map would occupy the whole space and the rest of the options would be placed above it, which is basically what happens in most applications that work with maps.

On the other hand, we leave as a future contribution one of the ideas that we did not manage to carry out during our project. It is about the dynamic selection on the map whose objective was that the selected filters were only applied to the zones highlighted by the user, being able to be a single zone or several zones. We have already commented previously that we used the plugin *Draw* of the library *folium* to add this functionality to the map, but we did not find the way to return the perimeter of the highlighted zones nor the coordinates of the vertices. As this is done dynamically on the html we do not have access to the attributes of the created figures and therefore we cannot save them in a variable in real time. However, the idea is very interactive and visual, so it could be tried again in the future.

All the proposed lines are interesting, but the most useful with respect to the initial purpose of this study is to extend the database with new metamorphic relations (MR) that analyze other types of properties of OSM maps. Specifically, we could add those from the previous research that looked for errors in *tags* and all those in the Almendros-Jiménez et al. (2023) article. In fact, we could include as many searches as we wanted as long as we could implement an RM that would find them. It would also be interesting to expand and improve the information covered by each node, making the maps more and more accurate and including more contrasted information.

Bibliografía

Almendros-Jiménez, J. M., Becerra-Terón, A., Merayo, M. G. y Núñez, M. Metamorphic testing of OpenStreetMap. *Information & Software Technology*, vol. 138, página 106631, 2021.

Almendros-Jiménez, J. M., Becerra-Terón, A., Merayo, M. G. y Núñez, M. Using metamorphic testing to improve the quality of tags in OpenStreetMap. *IEEE Transactions on Software Engineering*, vol. 49(2), páginas 549–563, 2023.

API. Api reference map. `https://python-visualization.github.io/folium/latest/reference.html`, s.f.

Bengreen. Basexclient 8.4.4. `https://pypi.org/project/BaseXClient/`, 2016.

Cassingena Navone, E. Python vs. Javascript – ¿Cuáles son las diferencias entre estos dos lenguajes de programación? `https://www.freecodecamp.org/espanol/news/python-vs-javascript-cuales-son-las-diferencias-entre-estos-dos-lenguajes-de-prog` 2022.

Chen, T. Y., Cheung, S. C. y Yiu, S. M. Metamorphic testing: a new approach for generating next test cases. Informe Técnico HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, 1998.

Chen, T. Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T. H. y Zhou, Z. Q. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys*, vol. 51(1), páginas 4:1–4:27, 2018.

Connors, L. Creando un mapa simple con folium y python. `https://towardsdatascience.com/creating-a-simple-map-with-folium-and-python-4c083abfff94`, 2021.

Dixon and Moe. Selector de color html. `https://htmlcolorcodes.com/es/selector-de-color/`, s.f.

Dominguez Minguez, T. *Desarrollo de interfaces gráficas en Python 3 con Tkinter*. Marcombo, 2021.

Fitzpatrick, M. Widgets de pyqt5 usando los widgets qpushbutton, qcheckbox, qcombobox, qlabel y qslider. https://www.pythonguis.com/tutorials/pyqt-basic-widgets/, 2023.

Fundación Geoespacial de Código Abierto. Geoserver servidor de código abierto para compartir datos geoespaciales. https://geoserver.org/, s.f.

Harwani, B. M. *QT5 Python Gui programming cookbook: Building responsive and powerful cross-platform applications with PyQt*. Packt Publishing, 2018.

Moore, A. D. *Python GUI Programming with Tkinter: Design and build functional and user-friendly GUI applications*. Packt Publishing, 2021.

Moreno, M. El primer medio de comunicación en español sobre internet, redes sociales y tecnología. https://www.trecebits.com/que-es-openstreetmap/, 2023.

Mozilla Foundation. border-top-left-radius. https://developer.mozilla.org/en-US/docs/Web/CSS/border-top-left-radius, 2023.

Python Software Foundation. tkinter — interface de python para tcl/tk. https://docs.python.org/es/3/library/tkinter.html, 2023.

Qt Group. Hojas de estilo qt. https://doc.qt.io/qt-6/stylesheet-examples.html, s.f.

QtSnippet. Qt snippet: Qpushbutton with word wrap feature. https://falsinsoft.blogspot.com/2015/11/qt-snippet-qpushbutton-with-word-wrap.html, 2015.

Segura, S., Towey, D., Zhou, Z. Q. y Chen, T. Y. Metamorphic testing: Testing the untestable. *IEEE Software*, vol. 37(3), páginas 46–53, 2020.

Workana LLC. ¿Qué es código abierto? https://i.workana.com/glosario/codigo-abierto/, s.f.

Apéndice A

Guía práctica para ejecutar la aplicación

Para poder ejecutar el programa necesitaremos preparar nuestro equipo instalando ciertos requisitos y aplicaciones necesarias para que todo se realice correctamente. Este proceso de preparación consiste principalmente en tres puntos:

1. Capacidad de ejecutar un programa en python
2. Instalación de la base de datos (BaseX)
3. Preparación del entorno de desarrollo agregando las librerías necesarias

1. Para poder ejecutar un programa desarrollado en Python descarga un entorno compatible con él. A continuación dejamos algunas opciones con una guía para instalarlas:

- Spyder: `https://docs.spyder-ide.org/3/installation.html`
- Jupyter: `http://programacion.espol.edu.ec/%23/gu%C3%ADas-de-instalaci%C3%B3n/instalaci%C3%B3n-de-jupyter-notebook/`

2. Para preparar la base de datos hay que realizar los siguientes pasos:

- Descargar el program BaseX Query de su respectiva página web: `https://basex.org/download/`
- Descargar el repositorio de Github con dirección `https://github.com/jalmenUAL/MT-OSM` y guardar la carpeta llamada "MT-OSM-master" con todo su contenido en el entorno del proyecto en el que estás ejecutando tu aplicación

3. Para la correcta instalación de las librerías necesarias para que se ejecute correctamente nuestra aplicación hay que seguir los siguientes pasos:

- Abrir un terminal o la cmd
- Ejecutar las instalaciones o updates de las siguientes librerías mediante los comandos *pip install "libreria"* o *pip install -U "libreria"*:

- *folium*
- *requests*
- *xml*
- *base64*
- *BaseX* (o *BaseXClient*)
- *webrowser*
- *PyQt5*
- *tkinter*
- *PIL*
- *time*
- *datetime*
- *threading*
- *reportlab*
- *res*
- *sys*
- *os*

Nota importante: Antes de ejecutar la aplicación debes abrir el servidor de la base de datos ejecutando la aplicación *BaseX HTTP Server (Start)* que has instalado junto con BaseX. Del mismo modo, al finalizar debes ejecutar *BaseX HTTP Server (Stop)* para cerrar el servidor. Si no realizar este paso no tendrás acceso a la base de datos.