UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TRABAJO DE FIN DE GRADO

APLICACIÓN QUE DETERMINE SI EXISTEN RESTRICCIONES DE ACCESIBILIDAD ENTRE DOS PUNTOS DENTRO DE UN EDIFICIO

APPLICATION THAT DETERMINES IF THERE ARE ACCESSIBILITY RESTRICTIONS BETWEEN TWO POINTS WITHIN A BUILDING

Directora: Sonia Estévez Martín

Alejandro Moreno Palma

Javier De La Oliva Sánchez-Valladares

Grado en Ingeniería Informática

Curso académico 2019 - 2020

Agradecimientos

A nuestras familias por el apoyo incondicional.

También queremos agradecer a nuestra tutora, Sonia Estévez, por haber participado muy activamente tanto en tareas de investigación como la excelente atención que nos ha brindado a lo largo del desarrollo de todo el proyecto.

Índice

Índice de figuras

Resumen:

La idea de este proyecto viene motivada principalmente por los impedimentos que personas con diversidad funcional, ya sea física o cognitiva sufren diariamente al acceder a ciertas infraestructuras sobre las cuales no tienen ninguna referencia previa. Nuestro objetivo es proporcionar una aplicación que permita a estas personas conocer con anticipación si para una infraestructura existe un camino entre dos puntos de su interior, libre de impedimentos físicos como escaleras. Por ese motivo es necesario identificar elementos como rampas de acceso o ascensores que les permitan conocer el nivel de accesibilidad de la propia infraestructura.

La aplicación, que ha sido enfocada sobre el edificio de la facultad de informática de la Universidad Complutense de Madrid, además de resolver el problema mencionado, también es capaz de calcular recorridos mínimos entre dos puntos cualesquiera de su interior. De este modo, cualquier otra persona sin impedimentos funcionales también podría hacer uso de la misma.

El proyecto está fundamentado sobre los sistemas de información geográfica o SIG [3] y se ha desarrollado a través de las siguientes etapas:

1. Geocodificación [4] de todas las áreas del edificio mediante la herramienta de edición de OpenStreetMap [5].
2. Implementación de un algoritmo de vuelta atrás [6] para el cálculo de recorridos mínimos junto con la herramienta OpenTripPlanner [7] para generación de grafos de rutas.
3. Desarrollo web con JavaScript y la librería Leaflet [8] para mostrar al usuario el mapa interactivo de la facultad sobre el que se ejecuta nuestra aplicación.

`Palabras clave`: Accesibilidad, Sistemas de información geográfica (SIG), Geocodificación, OpenStreetMap, OpenTripPlanner, Cartografía, Algoritmo de Vuelta Atrás, Leaflet, Aplicación Web, Mapa Interactivo.

Abstract:

The project has been motivated by the idea of restrictions that people with disabilities, either physical or cognitive, experience on a daily basis by accessing to different infrastructures on which they don't have any previous information about.

Our purpose is to provide them an application that makes possible to know in advance if there is a free barrier way, such as stairs, between two points from the inside of a building. In order to do that, its necessary to identify elements like ramps or elevators which lets users to know the accessibility level of the infrastructure.

The application, which has been focused on the Computer Science building of the Universidad Complutense de Madrid, in addition to solving the previous problem, it's also able to find shortest paths between any 2 points from the indoor. This way, any other person without functional disabilities could also use the application.

The project its based on the geographical Information Systems (GIS) [3] and it has been developed through the following steps:

1. Geocoding [4] with OpenStreetMap [5] editing tool of the whole areas of the building.
2. Deployment of the backtracking algorithm [6] for finding shortest paths along with OpenTripPlanner [7] tool for the routing graphs generation.
3. JavaScript web development together with Leaflet [8] library for displaying users the interactive map of the building.

`Keywords`: Accesibility, Geographical Information Systems(GIS), Geocoding, OpenStreetMap, OpenTripPlanner, Mapping, Backtracking, Leaflet, Web Application, Interactive Map.

1. Introducción

1.1. Motivación

Según el Informe Olivenza [9] realizado por el Observatorio estatal de la Discapacidad, cuyo fin es aportar a la sociedad datos obtenidos mediante estudios e investigaciones sobre personas con discapacidades, cerca de un 20 % de la población española sufre algún tipo de limitación en su actividad diaria debido a problemas de salud. Por ello, es necesario proporcionar los medios adecuados para que estas personas puedan visitar lugares o acceder a servicios con independencia de su diversidad funcional, garantizando así una buena accesibilidad universal.

Proyectos como Risewise [10] tratan de identificar las necesidades y las posibles acciones a realizar para la integración y la mejora de la calidad de vida de las personas con alguna discapacidad. Nuestro trabajo también es una aportación al proyecto Risewise.

Otros proyectos como Missing Maps [11] de Médicos Sin Fronteras, tienen como objetivo la utilización de nuevas tecnologías para mapear aquellas partes del mundo más vulnerables ante epidemias, crisis humanitarias o desastres naturales. Al igual que Missing Maps, promover el voluntariado digital es también otro de nuestros objetivos [12].

La mayor parte de los mapas que podemos ver a día de hoy en la red únicamente información de elementos que se encuentra en el exterior de los edificios. Por ello, también es necesario eliminar las barreras relativas a la falta de accesibilidad en ciertos servicios o infraestructuras como pueden ser teatros, cines, estaciones de transporte o edificios públicos mediante su identificación previa.

Esto se puede conseguir mediante la asignación de coordenadas a los elementos del interior de la infraestructura, también conocido como “indoor mapping” [13].

Mediante el conocimiento compartido y distribuido, las personas podrían desarrollar nuevas aplicaciones. Por ejemplo, algunas de ellas podrían proporcionar información por adelantado de una infraestructura determinada. De esta manera, cualquier usuario puede tener una idea sobre las garantías de accesibilidad de la instalación sin necesidad de acudir a la misma.

Gracias a esta información, las personas también podrían solicitar ayuda o buscar otras maneras de concluir sus actividades exitosamente.

1.2. Objetivos

Este proyecto tiene tres objetivos principales:

1. La creación de una aplicación que permita conocer a todo el mundo con antelación la ruta mínima entre dos puntos del edificio. A pesar de estar dirigida principalmente a personas con diversidad funcional, también puede ser utilizada por otros usuarios para el desarrollo de otras actividades como por ejemplo, el transporte de material pesado dentro del edificio o para encontrar la salida más próxima en caso de emergencia.

2. Ayudar a que personas con movilidad reducida puedan evitar obstáculos como escaleras en su itinerario, siempre y cuando sea posible. La aplicación es capaz de elegir aquellos caminos con ascensores para que estos usuarios puedan llegar a su destino con las mayores facilidades.

3. Fomentar el voluntariado digital pues, mediante este proyecto, aquellas personas con ideas análogas pueden tener otro punto de referencia a la hora de iniciar nuevas implementaciones sobre otras infraestructuras.

1.3. Plan de trabajo

El siguiente plan de trabajo pretende indicar cuales han sido las tareas principales que han contribuido a obtener el resultado final.

El plan de trabajo se divide en cinco etapas:

1. Investigación:
   - *a*) Recopilación de trabajos de investigación previos con ideas similares.
   - *b*) Recopilación de información acerca de las herramientas software utilizadas en los trabajos de investigación mencionados anteriormente.
   - *c*) Puesta en común de todas las herramientas software encontradas.
   - *d*) Valoración y elección de las herramientas que proporcionan las funcionalidades deseadas.
   - *e*) Obtención de los planos de las plantas 0, 1 y 2 de la Facultad de Informática.
2. Diseño:
   - *a*) Definición de las funcionalidades principales que la aplicación debe proporcionar.
   - *b*) Definición del uso de tecnologías web como las más apropiadas para alcanzar el resultado final.
3. Implementación:
   - *a*) Formación en el uso del editor Java para OpenStreetMap (JOSM).
   - *b*) Mapeado de todos los elementos de la misma planta del edificio.
   - *c*) Obtención del fichero con la información topográfica generada por JOSM.
   - *d*) Formación y despliegue de la herramienta OpenTripPlanner.
   - *e*) Generación del grafo de rutas para la planta del edificio de la facultad.
   - *f*) Estudio de los resultados generados por OpenTripPlanner.
   - *g*) Implementación de la aplicación web con Leaflet para visualizar los resultados en el mapa.
   - *h*) Desarrollo del código JavaScript que permite generar caminos entre dos puntos de una misma planta.
4. Pruebas:
   - *a*) Realización de pruebas y comprobación de los resultados esperados obtenidos al aplicar las pautas definidas en el apartado de implementación.

Tras las pruebas y la comprobación del correcto funcionamiento de rutas en la misma planta de la facultad es necesario realizar el proceso de mapeado y generar los grafos de las plantas restantes, como se refleja en la Figura 1:

Posteriormente se obtienen los ficheros con la información topográfica para cada planta por separado.

Y, finalmente, se procede al diseño y posterior desarrollo del algoritmo de backtracking que permite establecer una comunicación entre las diferentes plantas mediante los elementos de conexión vertical como escaleras y ascensores.

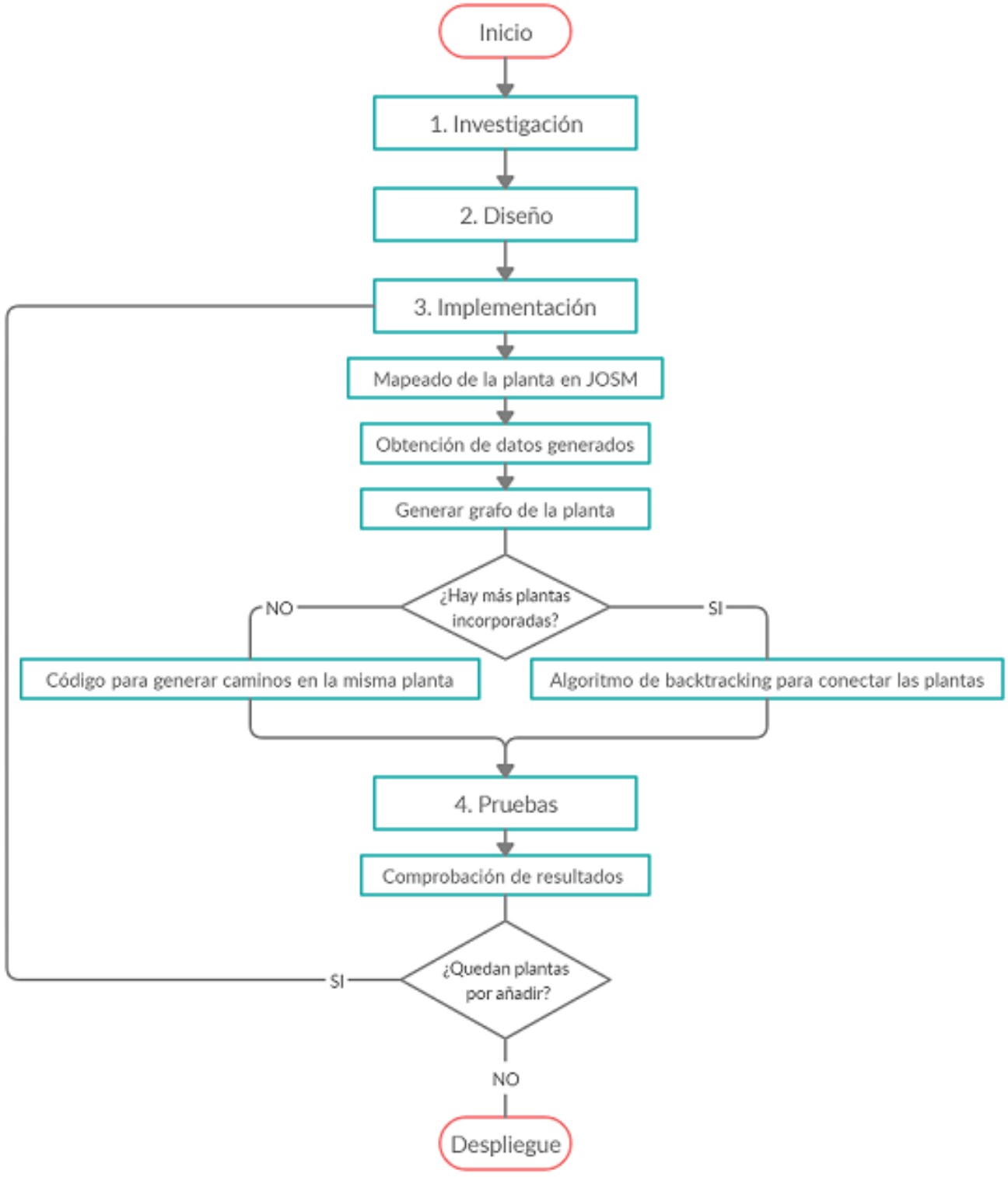


Figura 1: Esquema Plan de Trabajo

2. Introduction

2.1. Motivation

In our society, around 18 % [9] of the Spanish population faces restrictions at their daily activity due to health problems. Because of that, itś necessary to provide appropriate mechanisms for visiting some places or accessing to some services independently of their functional diversity by guaranteeing them a suitable universal accessibility.

Projects like Risewise [10] aims to identify the needs and potential actions to be taken for the integration and for the life quality enhancement of people with some disabilities.Our work its also a contribution to the Risewiseś project.

Another projects like Missing Maps [11] from Medicos Sin Fronteras, aims to use new IT technologies for mapping those parts of the world that are more vulnerable against epidemics, humanitarian crisis or natural disasters. Like Missing Maps, promoting the digital volunteering is another of our objectives [12].

Most of the maps that we can find nowadays on the internet contains only elements that are on the outside of buildings. For that, itś also necessary to remove some restrictions related to the indoors as the lack of accessibility at certain services or infrastructures like theaters, cinemas, transport stations or public buildings. This can be done by assigning coordinates to elements within the infrastructure, it is also known as indoor mapping [13].

With the shared and distributed knowledge people could make possible the deployment of new applications. For example, some of these applications could provide any information beforehand of a determined infrastructure that allows the users to have an idea about the accessibility guarantees without the necessity of moving to the same.

Thanks this information, they could also ask for help or look for another way to perform successfully their activities.

2.2. Objectives

This project has 3 main goals:

1. The development of an application that lets the users to know in advance the shortest path between to points within the building. Despite of being directed to people with disabilities, It also can also be used by other users to other activities development like transportation of heavy equipment inside the building or finding the closest exit in an emergency situation.

2. Helping people with reduced mobility can avoid some barriers like stairs within itineraries, as long as it can be possible. The application is able to choose those itineraries with elevators in order to make possible for this users to reach their destination by the simplest way.

3. Promoting the digital volunteering, thanks to this project those people who have similar ideas can have another reference point in order to start new deployments over other infrastructures.

2.3. Workplan

The work plan described below aims to show the main tasks that have contributed to get the final result.

Its divided into 5 steps:

1. Research:
   a) Gathering of previous research papers with similar ideas.
   b) Information gathering about the software tools that have been used in the research projects above mentioned.
   c) Sharing of the software tools that have been identified.
   d) Assessment and selection of software tools that provides the desired functionalities.
   e) Acquisition of zero, first and second drawings of the Computer Science Faculty building.
2. Design:
   a) Identification of the main functionalities that the application should provide.
   b) Identification of the most appropriate web technologies to achieve the final result.
3. Development:
   a) Training on the OpenStreetMap Java Editor (JOSM).
   b) Mapping of the points within the same floor.
   c) File generation that contains the topographical information generated by JOSM.
   d) Training and deployment of the OpenTripPlanner software tool.
   e) Routing graph generation for building floor of the faculty.
   f) Analysis of the results generated by OpenTripPlanner.
   g) Deployment of the web application together with Leaflet to display the results on the map.
   h) JavaScript code development that makes possible to generate paths between two points at the same level.
4. Tests:
   a) Testing and checking the results that have been generated as a result of the implementation guidelines application.

   After checking that the routing at the same level performs well, Itś necessary to repeat the mapping and graph generation for the remaining floors, as reflected in the Figure 2:

   Later, the files with the topographical information of each floor are obtained independently.

   This is followed by the design of the backtracking algorithm that enables the connection between plants through vertical connection elements like stairs or elevators.

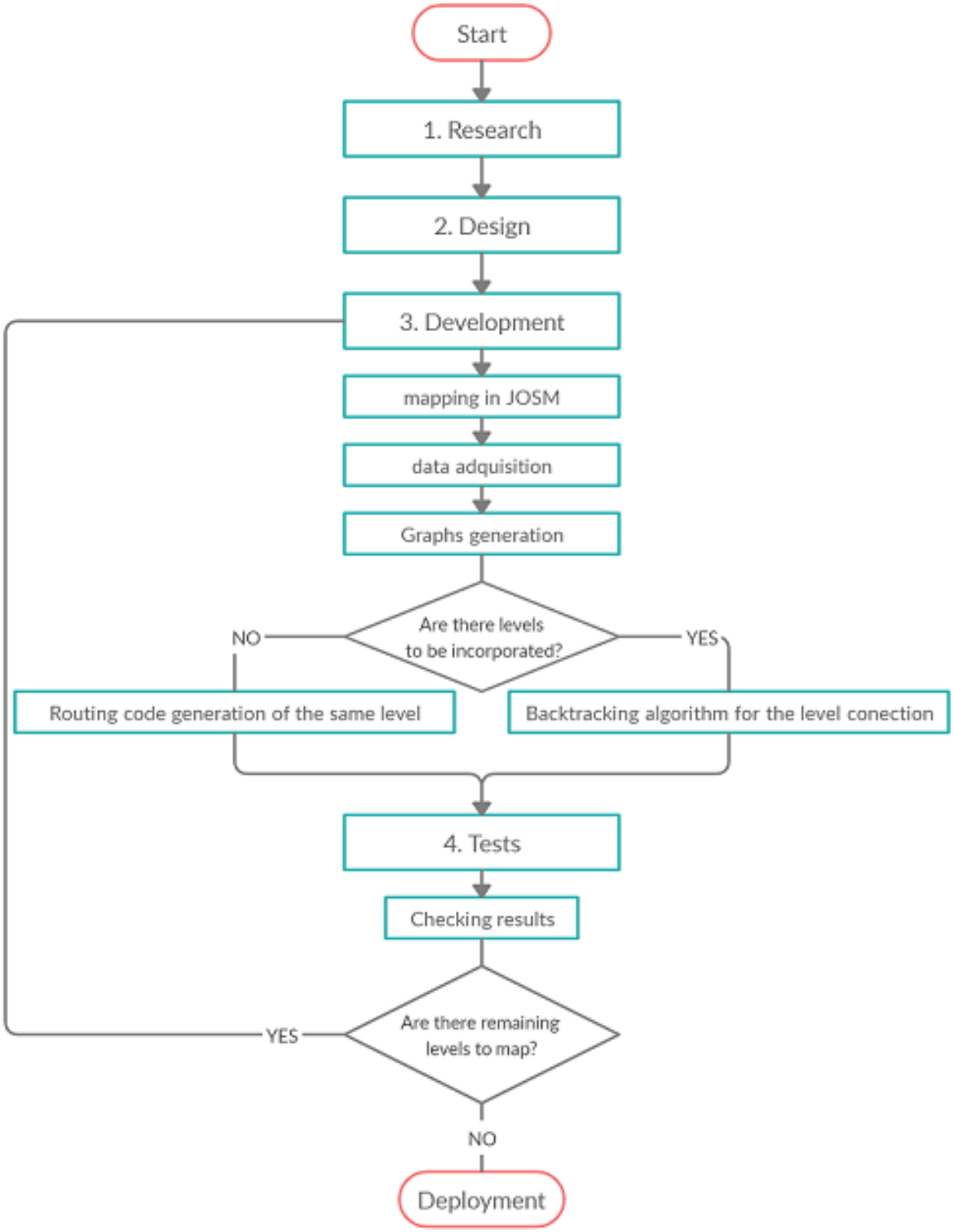


Figura 2: Outline of the Workplan

3. Aplicaciones

3.1. OpenStreetMap (OSM)

Un Sistema de Información Geográfica, GIS, es un conjunto de herramientas que permiten organizar, analizar y manipular datos. Se basa en la información geográfica, una ubicación, y la información atributiva, datos como nombres de calles o tipos de vías. Un GIS funciona como una base de datos de información geográfica que permite realizar consultas, editar datos y realizar presentaciones de los mismos.

Figura 3: OpenStreetMap

Sitios web frecuentes como Google Maps, del gobierno como el catastro y Sigpac [14], o como la plataforma Arcgis de ESRI [15], son algunos ejemplos de este tipo de tecnologías.

OSM almacena en su base de datos elementos de los siguientes tipos:

- Nodos: compuestos por un par de coordenadas que almacenan una posición concreta.
- Vías: formadas por un conjunto de nodos.
- Relaciones: contenedores lógicos para la agrupación de nodos y/o vías que compartan una o varias propiedades.
- Etiquetas: pares de tipo clave-valor que aportan información sobre los elementos anteriores.

Los datos se presentan en una extensión del formato XML conocida como OSM XML que almacena todos los elementos de una manera estructurada, jerárquica y legible.

La elección de OpenStreetMap frente a sus alternativas ha estado en gran medida influenciada por su servicio de datos de libre acceso pues se distribuye bajo una licencia Open Database License que permite la edición o adaptación de los mapas bajo el reconocimiento a la organización y sus colaboradores.

Además, dado que OpenStreetMap cuenta con numerosos colaboradores repartidos a lo largo de la geografía garantiza que se trata de un sistema estable, rápido y fiable gracias a la distribución de sus datos.

Pero también hemos encontrado ciertos inconvenientes a la hora de realizar el mapeado indoor pues a pesar de disponer de documentación en la wiki, algunas de ellas no están actualizadas o no son muy claras en cuanto a los procedimientos a seguir.

3.2. Java OpenStreetMap editor (JOSM)

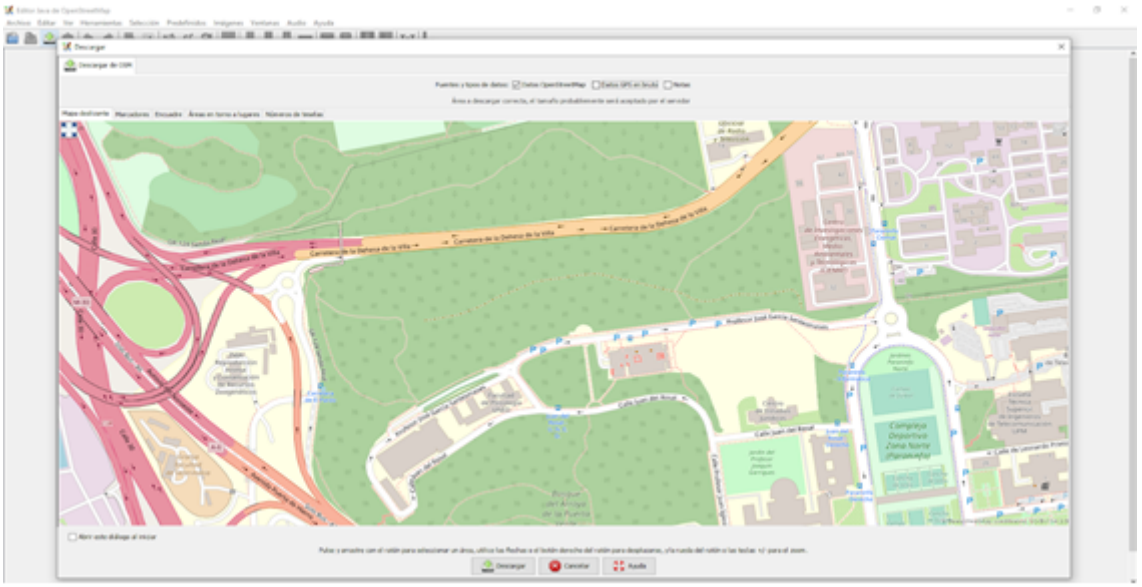


Figura 4: JOSM

JOSM es el editor open-source Java para la plataforma OpenStreetMap. Gracias a ello cualquier usuario puede obtener el código fuente y colaborar con nuevas contribuciones como actualizaciones o mejoras que proporcionen nuevas funcionalidades. También permite la creación de nuevos elementos de OSM como nodos, vías o relaciones mediante el proceso de geocodificación. Este proceso permite asociar una o varias coordenadas geográficas a un punto del mapa en función del elemento en que consista.

Mediante el etiquetado o uso de tags, cada elemento puede tener asociada información como el nombre, identificador o el tipo de elemento que representa. Gracias a ello, permiten realizar una clasificación por categorías en función de sus características o atributos.

Otra de las ventajas de ser open-source es la existencia de numerosos plugins que permiten extender su funcionalidad, entre ellos destacamos el uso de PicLayer [16]. Este plugin, permite añadir imágenes de planos en forma de capas para poder establecer una base sobre la que empezar la geocodificación de los distintos elementosv(edificios, calles, escaleras, mobiliario urbano, jardines...).

Por último, destacar que también posee una comunidad muy activa de usuarios que dan soporte a la herramienta resolviendo dudas a través del propio foro de JOSM o mediante el sitio de preguntas y respuestas de OpenStreetMap.

3.3. OpenTripPlanner (OTP)

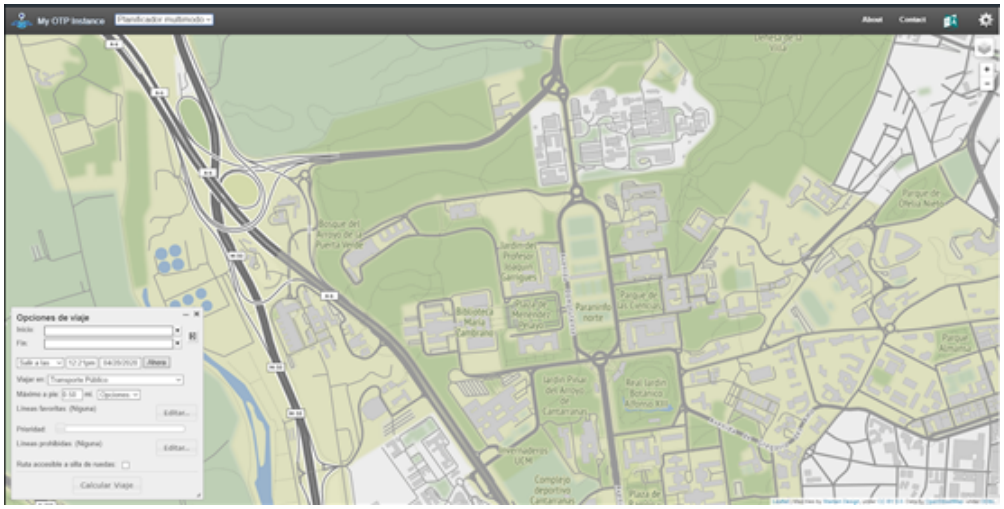


Figura 5: OpenTripPlanner

OpenTripPlanner u OTP es un proyecto de código abierto publicado bajo la licencia GNU LGPL que garantiza que el software distribuido bajo esta licencia se pueda modificar y compartir libremente.

El programa, disponible para Windows, Linux o MacOs se encuentra disponible en su propia web opentripplanner.org.

En cuanto al diseño, OTP cuenta con tres núcleos principales:

1. Graph builder: este componente se encarga de construir el grafo de rutas a través del algoritmo A* (A-star) y contracción de jerarquías.

2. Routing engine:escrito en java y que puede ser utilizado desde cualquier otra aplicación externa gracias a la integración de una API Rest.

3. User interface: cuyo componente, a diferencia de los dos anteriores está desarrollado en JavaScript y permite la visualización de mapas y rutas. Este componente es útil para la visualización de información no indoor (calles, parques o cualquier elemento externo a un edificio) pero actualmente está muy limitado en cuanto a la visualización de información indoor de un edificio o instalación.

Además, el núcleo de enrutamiento o Routing es capaz de recibir ciertos parámetros o criterios para la búsqueda de la solución óptima como por ejemplo si para la solución deben tenerse en cuenta únicamente nodos que garanticen la accesibilidad mediante silla de ruedas, si se debe escoger la solución con el menor número de transferencias o enlaces de comunicación (transporte público principalmente) o si la solución óptima sólo ha de ser aquella que se realice en el menor tiempo posible independientemente de los demás factores.

Dado que para ciertos criterios mencionados anteriormente OTP utiliza los tags de OSM es imprescindible que todos estos tags se hayan definido correctamente mediante la herramienta JOSM de la que hemos hablado con anterioridad.

3.4. Leaflet

Leaflet es una biblioteca de Javascript de código abierto para la creación de aplicaciones web de mapas interactivos. Junto con OpenLayers, o Google Maps es una de las librerías más utilizadas por diversos motivos, entre ellos su ligereza, pues supone un 30 % del total de código de OpenLayers [17].

Entre los motivos de la elección de Leaflet frente a OpenLayers [18] o Google Maps API, también destaca su facilidad de uso junto con la gran cantidad de plugins que se encuentran disponibles y su amplia documentación.

El uso de Leaflet se fundamenta principalmente en el uso de capas de tipo ráster o vectorial, entre otras.

Las capas ráster permiten representar la información de una entidad mediante una celda, ofrecen menos precisión en cuanto a la localización pero su procesamiento es más rápido que las capas vectoriales, las capas ráster suelen utilizarse en modelos matemáticos o analíticos.

Las capas vectoriales son mucho más precisas topográficamente que las de tipo ráster pues cada entidad (puntos, líneas o polígonos) está compuesta por una o varias coordenadas almacenadas en vectores. Por el contrario su procesamiento en ciertos tipos de modelos como los mencionados anteriormente es mucho más costoso debido a su tratamiento.

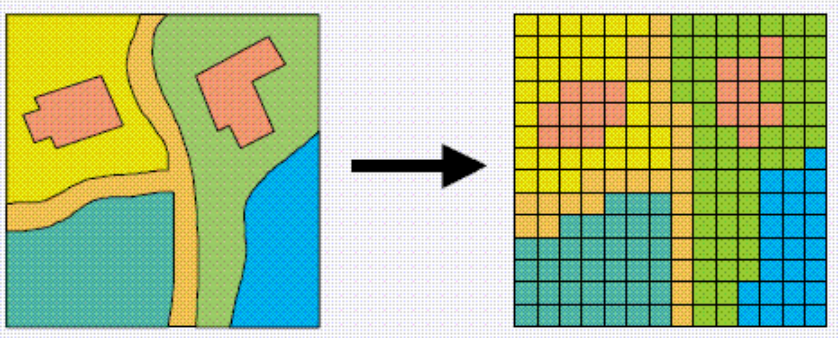


Figura 6: Capas raster, Leaflet [1]

Otra de las ventajas de Leaflet es su facilidad para el manejo de datos en formatos GIS estándar como geoJSON [19].

Además, Leaflet soporta el uso de formatos estándar GIS como el geoJSON utilizado en este proyecto para alimentar el modelo de datos de la aplicación web en JavaScript.

4. Métodos

4.1. Geocodificación con JOSM

4.1.1. Descarga de datos y calibración de imagen

En este primer paso se explica cómo obtener los datos ya existentes en la plataforma OSM para una zona determinada, estos datos que a simple vista vemos en el mapa de la aplicación no son más que una serie objetos en formato XML con sus atributos y etiquetas correspondientes que la aplicación interpreta para su visualización.

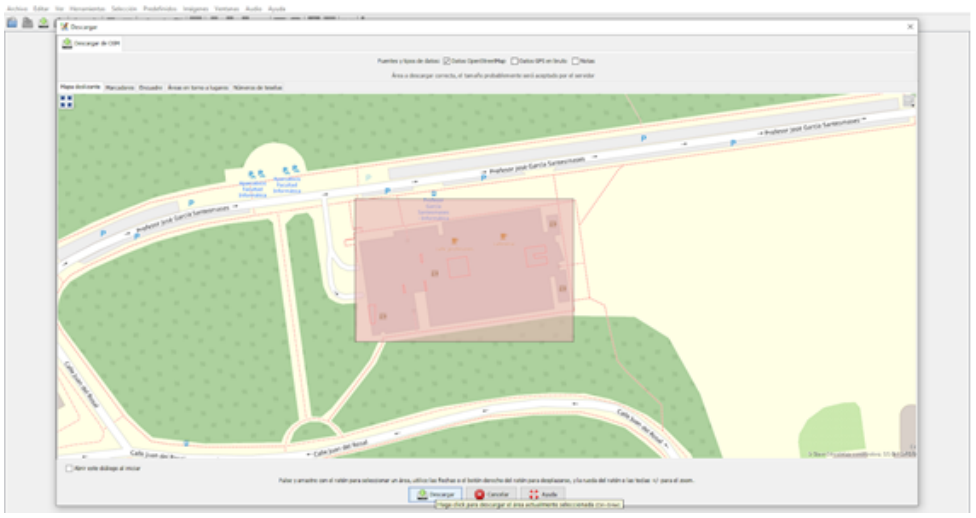


Figura 7: Descarga de datos, JOSM

Una vez seleccionado el área sobre la que trabajar es necesario añadir el plano sobre el que se quiere trabajar, en este caso se muestra el plano de la planta 0 de la facultad de informática.

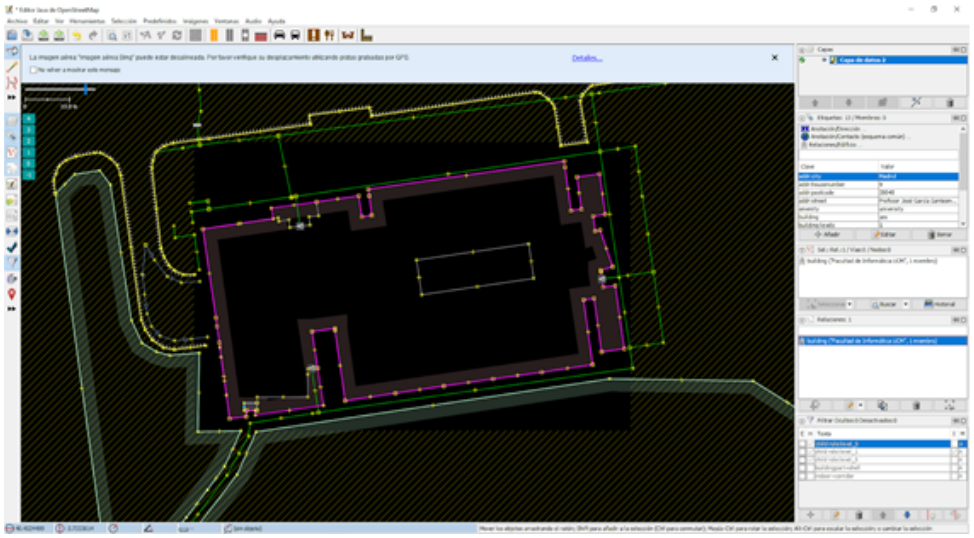


Figura 8: Area de trabajo, JOSM

Para añadir la imagen de la planta como capa es aconsejable el uso del plugin PicLayer, mencionado anteriormente.

Este plugin permite redimensionar, ajustar y calibrar la imagen de manera que el perfil exterior del edificio coincida con la imagen que se encuentra en OSM para así poder comenzar con la geocodificación de los elementos.

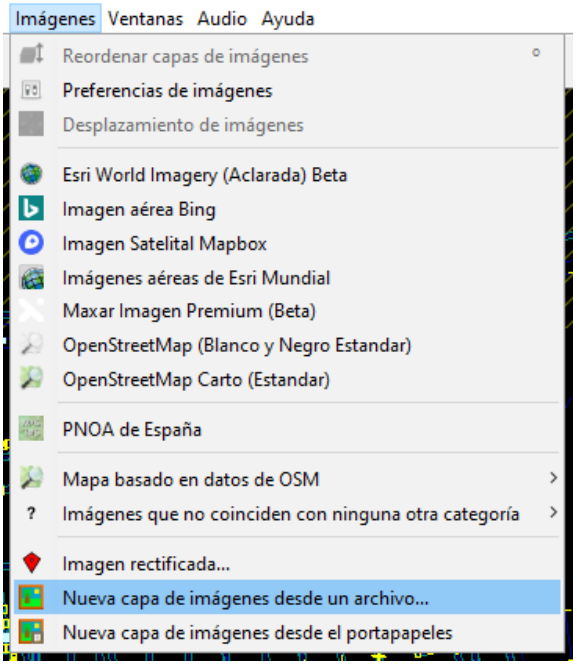


Figura 9: Añadir plano, JOSM

Figura 10: Plano calibrado, JOSM

Una vez ajustada la imagen y calibrada, podemos guardar esta configuración seleccionando la capa de imagen con el botón derecho "Guardar calibración de imagen".

En el siguiente punto se indica cómo añadir todos los elementos (aulas, aseos, ascensores, etc) que pertenecen a cada planta y cómo realizar un etiquetado correcto siguiendo las guías que proporciona OSM.

4.1.2. Mapeo de elementos y tags

Una vez establecida la base sobre la que realizar el mapeado es necesario identificar las cuatro entidades principales.

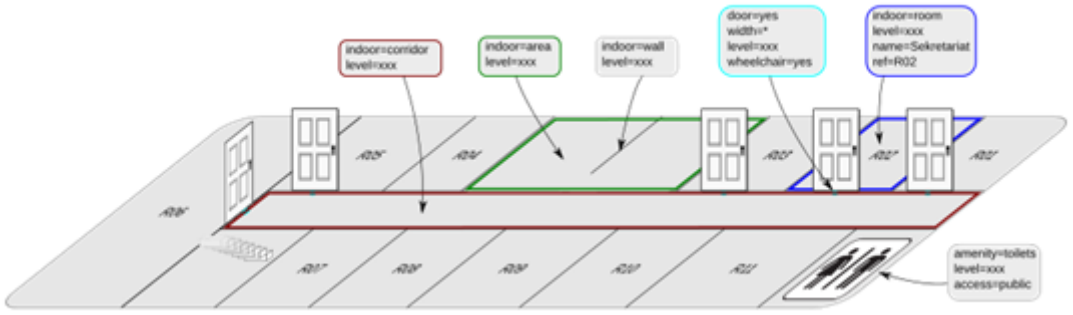


Figura 11: Ejemplo de elementos, JOSM [2]

- Room: áreas cerradas como aulas o laboratorios que pueden contener puertas para su acceso.

  Además, para ciertos espacios como aseos, cafeterías o la biblioteca se puede hacer uso de los elementos rápidos que permiten añadir información adicional como el horario de apertura, o el sexo en el caso de los aseos.

- Area: a diferencia de la entidad "room" se considera un área todo aquel espacio que no se encuentra delimitado mediante puertas o cualquier elemento de acceso.

- Corridor: típicamente representados mediante pasillos, permiten conectar diferentes entidades.

- Elevator y steps: en castellano ascensores y escaleras, permiten realizar conexiones verticalmente entre las diferentes plantas, en cuanto a su etiquetado se deben definir todos aquellos niveles en los que se encuentra presente la entidad.

Para el etiquetado de entidades tipo *room*, *area* o *corridor* se han utilizado los siguientes pares clave-valor;

```
indoor = <room / area / corridor>
name = <nombre del elemento>
level = <nivel en el que se encuentra>
```

En el etiquetado de entidades tipo *elevator* o *stairs* se ha utilizado la siguiente combinación;

```
highway = <elevator / steps>
name = <nombre del elemento>
repeat_on = <nivel inicial - nivel final> //solo si el elemento se
    encuentra en la misma posicion en todas las plantas que comunica.
```

Por ejemplo, el siguiente código representa un ascensor, que comunica la planta 0, 1 y 2 del mismo edificio.

```
highway = "elevator"
name = "Ascensor de la biblioteca"
repeat_on = "0-2"
```

Una vez se conocen cuales son las principales entidades y cómo han de representarse mediante etiquetas, el siguiente paso consiste en el mapeado.

Para ello es recomendable seguir un orden, primero aquellos elementos que pertenezcan a la entidad *room*:

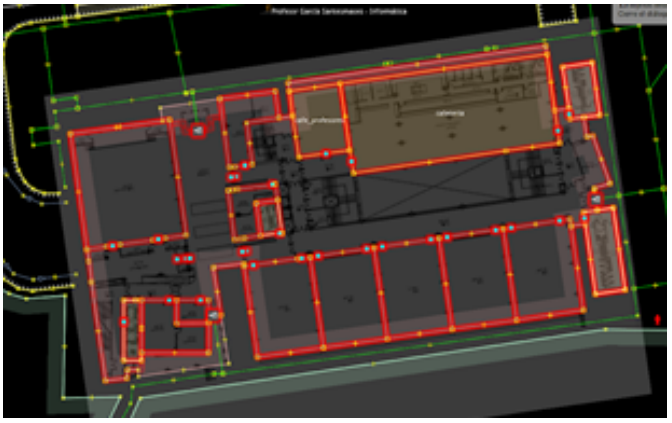


Figura 12: *Room* Planta 0, JOSM

Segundo los que pertenezcan a la entidad *corridor*:

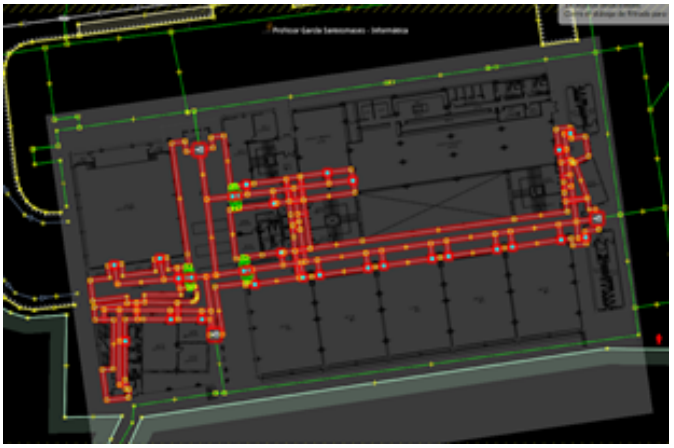


Figura 13: *Corridor* Planta 0, JOSM

Es muy importante que en el caso de los elementos de tipo *corridor* se defina una vía central que debe comunicarse con todos aquellos elementos con los que lo hace en la vida real como por ejemplo el caso de las puertas.

Esto se debe a que durante la generación del grafo de rutas, cada elemento que tenga conexiones ha de representarse como un vértice o nodo de enlace con otros nodos.

Finalmente se añaden aquellos elementos que pertenecen a las entidades *elevator* y *steps*

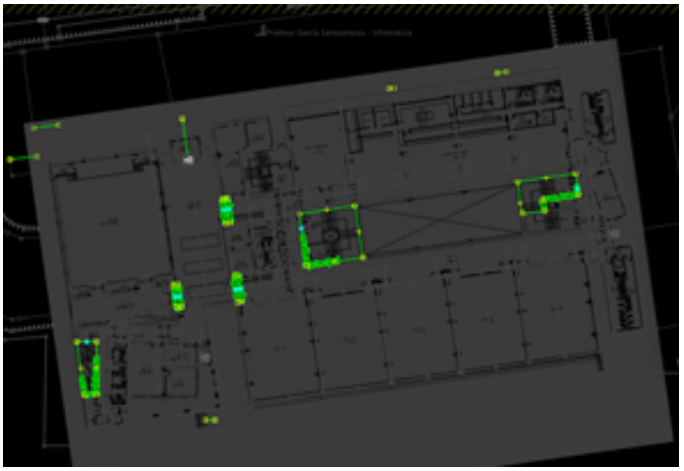


Figura 14: *Steps* Planta 0, JOSM

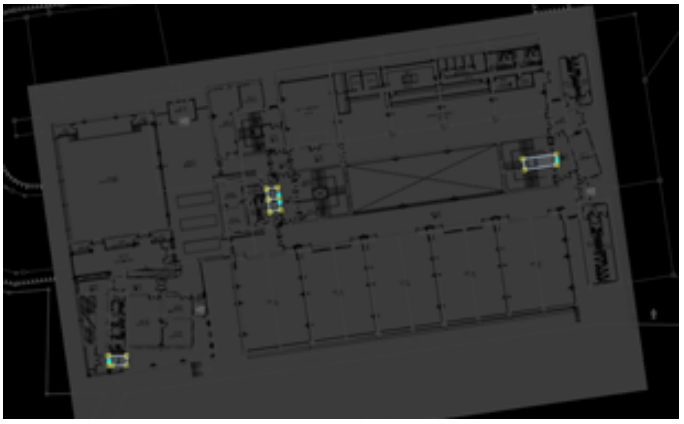


Figura 15: *Elevator* Planta 0, JOSM

Una vez se han añadido todos los elementos que pertenecen a la misma planta, se recomienda realizar una agrupación de los mismos mediante la creación de relaciones que veremos en el siguiente punto y que nos permitirán continuar el mapeado en las plantas posteriores de la manera más cómoda posible.

4.1.3. Relaciones y reglas de filtrado

Una relación es una agrupación de elementos de entidades iguales o diferentes que comparten alguna característica en común, en este caso la pertenencia de todas ellas al mismo nivel.

Las relaciones tienen carácter jerárquico, es decir, la relación de la planta baja contiene todos los elementos que se encuentran presentes en la vida real en la misma, lo mismo sucede con las plantas restantes.

A su vez, todas las relaciones que representan agrupaciones en forma de planta pertenecen a otra mayor, la relación del edificio.

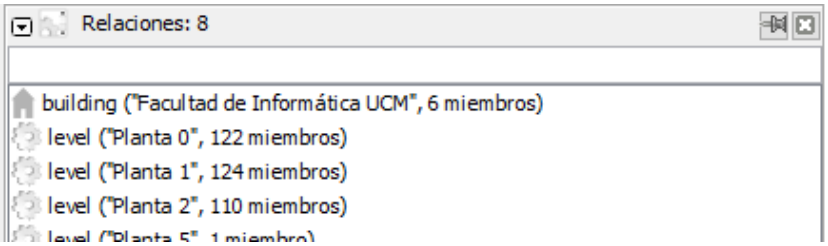


Figura 16: Relaciones, JOSM

El uso de relaciones permite trabajar eficientemente mediante reglas de filtrado que soporta el editor JOSM, mediante estas reglas se pueden mostrar u ocultar elementos que cumplan una condición como por ejemplo:

```
child role:level_0
```

Lo que mostraría u ocultaría, en función de la acción, todos aquellos elementos que tienen en su etiqueta level el valor 0.

El uso de reglas de filtrado es imprescindible cuando queremos comenzar con el mapeado de la siguiente planta pues permite ocultar todos los elementos que pertenecen a plantas anteriores.

Figura 17: Filtros, JOSM

4.1.4. Obtención de todos los datos generados

JOSM traduce toda la información acerca de los elementos que hemos añadido en los pasos anteriores a un fichero de tipo XML conocido como OSMXML.

El contenido de estos ficheros es muy extenso, sin embargo gracias a la notación XML es muy sencillo entender qué datos se están manejando o cómo se han representado.

El su contenido podemos encontrar elementos de los siguientes tipos:

→ Nodo, que contiene las etiquetas que se le han asociado:

```
1
2   <node id='6877809119' action='modify' timestamp='2019-10-14T12:12:46Z
       ' uid='10378375' user='orion200' visible='true' version='2'
       changeset='75663178' lat='40.45280577738' lon='-3.73361089154'>
3   <tag k='door' v='yes' />
4   <tag k='level' v='0;1;2' />
5   <tag k='repeat_on' v='0-2' />
6   </node>
```

→ Vías, que contienen las referencias de sus nodos para cada vía y los etiquetados asociados a esa vía en concreto.

```
1
2   <way id='-198908' action='modify' visible='true'>
3   <nd ref='6877809136' />
4   <nd ref='6878022452' />
5   <tag k='highway' v='corridor' />
6   <tag k='indoor' v='corridor' />
7   <tag k='level' v='0' />
8   </way>
```

→ Bloques de relaciones donde podemos encontrar todos los miembros que pertenecen a ella y las etiquetas asociadas a la relación.

```
1
2   <relation id='10154370' action='modify' timestamp='2019-10-14T11
       :04:43Z' uid='10378375' user='orion200' visible='true' version='1
       ' changeset='75658853'>
3   <member type='way' ref='-199080' role='' />
4   <member type='way' ref='-199067' role='' />
5   <member type='way' ref='-199030' role='' />
6   ...
7   <member type='node' ref='-170065' role='' />
8   <member type='node' ref='6878022435' role='' />
9   ...
10  <member type='way' ref='734425665' role='shell' />
11  <tag k='level' v='0' />
12  <tag k='name' v='Planta 0' />
13  <tag k='type' v='level' />
14  </relation>
```

Sin embargo, la herramienta Leaflet no es capaz de manejar ficheros XML pero sí ficheros en formato JSON o en este caso GeoJSON que utilizan los GIS. Por tanto, es necesario traducir el contenido del fichero a formato GeoJSON utilizando herramientas como osmtogeojson que además permiten realizar la traducción mediante una llamada a su API o directamente mediante su interfaz web.

4.2. OpenTripPlanner

4.2.1. Arquitectura

Una vez se haya finalizado el mapeado del edificio al completo y se han extraído en un fichero todos los datos generados, es el momento de la creación del grafo de rutas mediante OTP.

Como OpenTripPlanner sólo es capaz de generar recorridos en 2D, ha sido necesario tratar cada planta del edificio por separado, de manera que para cada una de ellas se genere su propio grafo de rutas.

Dado que cada grafo es independiente y no comparte nodos en común que permitan realizar recorridos verticalmente entre plantas, ha sido necesario el desarrollo de un algoritmo de backtracking que veremos más adelante. Este algoritmo además permite añadir en cierto modo, nuevos nodos de conexión vertical entre plantas como ascensores o escaleras para que el cálculo de rutas entre varias plantas sea posible.

OTP define como *router* el servicio que maneja una única región.En nuestro caso será necesario disponer de tres routers distintos pues cada planta representa una región independiente.Además, OTP gestiona cada router mediante directorios donde cada uno almacena el grafo (.obj) de la región que representa.

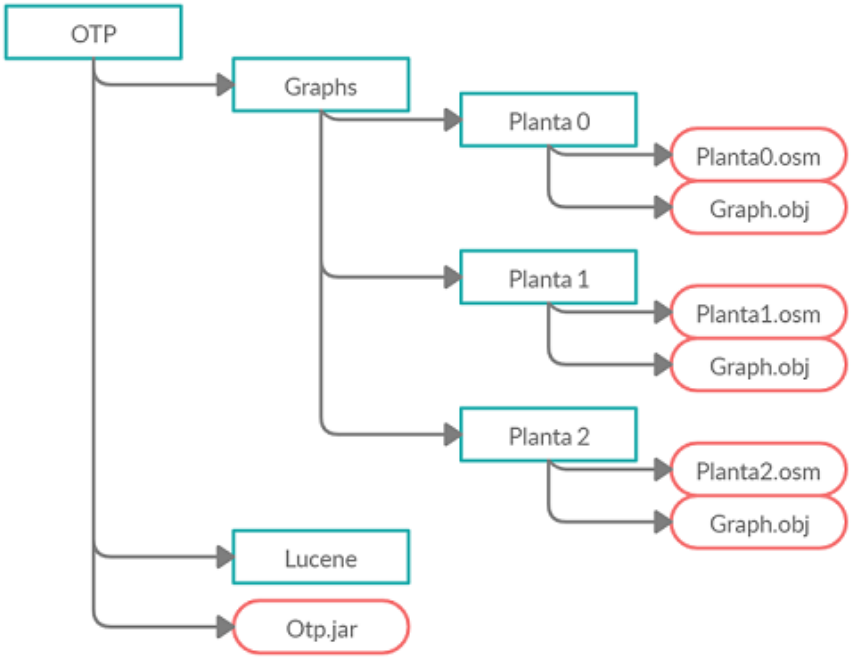


Figura 18: Sistema de carpetas, OTP

Dado que OTP se ejecuta sobre java, el comando para generar los grafos será de la forma:

```
java -Xmx2G -jar otp.jar --build graphs
```

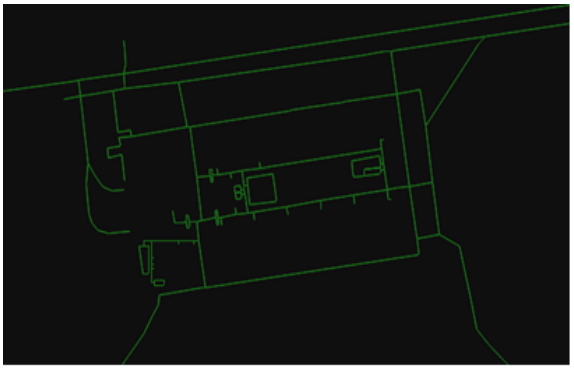
Donde el parámetro *-Xmx* indica qué cantidad máxima de memoria puede consumir para la ejecución y donde el parámetro *–build* indica la ruta donde se alojan los ficheros de entrada (.osm).

Para la visualización de los grafos utilizaremos la opción *–visualize*.

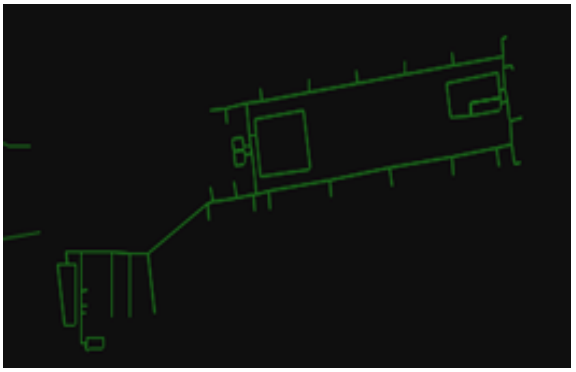
```
java -Xmx1G -jar otp.jar --visualize --router Planta\_0 --graphs graphs
```

Mediante el parámetro *–router* indicamos el grafo perteneciente al router que deseamos visualizar y con *–graphs* indicamos el directorio raíz donde se encuentran todos los routers [20].

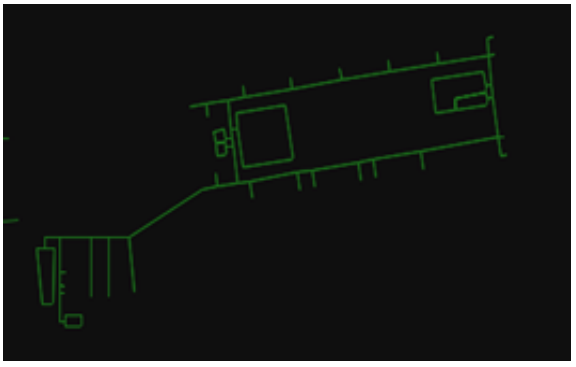
La aplicación abre una ventana que permite visualizar los caminos por los que podríamos acceder a las distintas áreas del edificio o lo que es lo mismo, aquellas conexiones que habíamos definido previamente en JOSM mediante los tags.



(a) Planta 0



(b) Planta 1



(c) Planta 2

Figura 19: Grafos de caminos

Finalmente, tras haber generado los grafos y comprender cómo funciona OTP, el siguiente paso es ejecutar la aplicación mediante la opción *–server* que, hará que la aplicación ejecute un servicio local sobre los puertos 8080 y 8081 para poder procesar las solicitudes que se realizan sobre su API REST.

```
java -Xmx1G -jar otp.jar --router <list_routers> --graphs graphs --server
```

4.2.2. Funcionamiento e incorporación a la aplicación

Gracias al núcleo *Routing engine* sobre el que está diseñado OTP es posible realizar peticiones de tipo HTTP GET sobre su API con los parámetros que necesitamos para el cálculo de la ruta en un nivel.

La información sobre los parámetros se puede consultar en el apidoc de OTP.

En este proyecto hemos utilizado 4 parámetros básicos:

- fromPlace: debe recibir un par de coordenadas que indique el punto de origen.
- toPlace: al igual que fromPlace, también necesita un par de coordenadas para indicar el destino.
- mode: para indicar qué tipo de transporte queremos utilizar, en este caso el único disponible es a pie.
- wheelchair: para que únicamente tenga en cuenta aquellos caminos accesibles mediante silla de ruedas.

Gracias al objeto XMLHTTPRequest de JavaScript, nuestra aplicación es capaz de realizar peticiones de tipo GET sobre la API de OTP que se está ejecutando en el puerto 8080 y obtener una respuesta en formato XML.

La estructura de estas peticiones es la siguiente:

```
xhr.open('GET', 'http://localhost:8080/otp/routers/pl0/plan?
     fromplace=' + coord_x + ',' + 'coord_y' +
     '&toPlace=' + coord_z + ',' + 'coord_w' +
     '&mode=WALK&wheelchair=' + wheelchair_selected, !true);
```

De este modo el servidor recibirá una petición con los puntos de origen y de destino, el modo de trayecto, en este caso a pie y si se ha seleccionado la opción de sólo calcular caminos para sillas de ruedas o no.

La respuesta tendrá el siguiente formato:

```
<Response>
<requestParameters>
 <mode>WALK</mode>
 <wheelchair>yes</wheelchair>
 <fromPlace>40.45293415757, -3.73316629732</fromPlace>
 <toPlace>40.45280577738, -3.73361089154</toPlace>
</requestParameters>
<plan>
<date>1588606202000</date>
<from>
 <name>Origin</name>
 ...
</from>
<to>
 <name>Destination</name>
 ...
</to>
```

```
<itineraries>
 <itineraries>
  <duration>46</duration>
  <startTime>1588606202000</startTime>
  <endTime>1588606248000</endTime>
  <walkdistance>46.46230074</walkdistance>
  ...
  <legs>
   <steps>
    <steps>
     ...
    <relativeDirection>DEPART</relativeDirection>
    <absoluteDirection>WEST</absoluteDirection>
    <lon>-3.7331631332199926</lon>
    <lat>40.45291803870797</lat>
    ...
    </steps>
    <steps>
    ...
    <relativeDirection>CONTINUE</relativeDirection>
    <absoluteDirection>WEST</absoluteDirection>
    <lon>-3.733537432</lon>
    <lat>40.45287631766</lat>
    ...
    </steps>
    ...
   </steps>
  </legs>
 </itineraries>
</itineraries>
</plan>
</Response>
```

Del cuerpo de la respuesta podemos destacar los siguientes elementos:

- requestParameters: aquellos parámetros que hemos proporcionado en la petición HTTP GET.
- itineraries: es una agrupación de recorridos independientes, por ejemplo, en el caso de que hubiese que realizar una parte a pie y otra en transporte público aparecían varios itinerarios anidados.
- walkdistance:pertenece a itineraries, indica la cantidad de recorrido en metros que se realiza a pie, este dato es muy importante pues de él, depende que mediante el algoritmo de backtracking obtengamos siempre el recorrido mínimo.
- legs: que nos indica que ese itinerario se está realizando a pie, tal y como indicamos con el mode = WALK
- steps: es el conjunto de instrucciones que la aplicación proporciona al usuario para la ejecución del recorrido, de ellas es importante destacar la dirección relativa, que nos indica qué debemos hacer (girar, continuar...) y la dirección absoluta, que indica en qué sentido cardinal debemos hacerlo.

4.3. Aplicación web (JavaScript + Leaflet + Backtracking)

4.3.1. Arquitectura de Leaflet

Para el desarrollo de la aplicación web, ha sido necesario conocer cúal es la arquitectura, sobre la que se fundamenta la librería y cómo se pueden añadir nuevas funcionalidades que no se proporcionan por defecto.

La existencia de una gama tan amplia de plugins está basada en la facilidad que tiene la propia librería para extender de tres clases hijas principales que proporcionan distintas funcionalidades

Estas clases son las siguientes:

1. L.Layer: es la clase que leaflet utiliza para la representación de capas, estas capas interactúan mediante eventos de zoom y puede haber una o varias que estén contenidas en el mapa para representar distintos tipos de información.

2. L.Handler: los objetos de esta clase son principalmente manejadores de eventos que responden a acciones del navegador como drag, tap, double click, scroll... y que no tienen una representación visual.

3. L.Control: que permiten instanciar elementos fijos de la interfaz que permiten seleccionar qué datos se visualizan o cómo se visualizan.

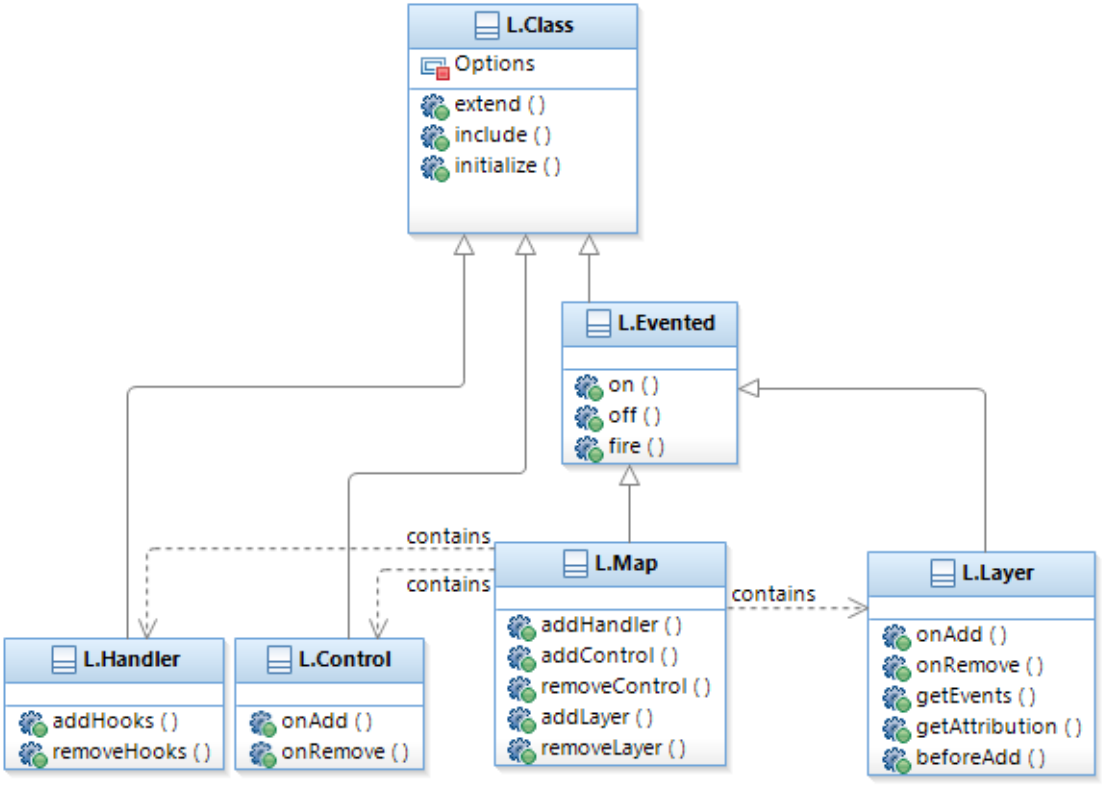


Figura 20: Diagrama de Clases, Leaflet

Todas estas clases, a su vez extienden de la clase padre o L.Class que además, permite cambiar partes de una clase ya definida añadiendo o modificando funcionalidades sobre la misma mediante el método L.Class.include().

El conocimiento de cómo se construye leaflet a nivel de código ha sido imprescindible para el desarrollo de nuestro proyecto pues, ha sido necesario adaptar ciertas clases del plugin L.Indoor que permiten crear mapas internos de edificios.

Este plugin transforma los elementos almacenados en el fichero OSMXML obtenido como resultado del mapeado en JOSM en un conjunto de capas vectoriales que representan puntos, líneas o polígonos en función del tipo de elemento leído.

4.3.2. Desarrollo y funcionamiento de la aplicación

El funcionamiento de la aplicación se podría resumir principalmente en tres etapas que explicaremos más detenidamente a continuación.

- En el primer paso, la aplicación se encarga de leer el fichero que contiene todos los datos geográficos de las aulas o elementos del edificio y asignarlos a la capa correspondiente.
- El segundo paso consiste en el cálculo de la solución para los datos de entrada que proporciona el usuario, en este caso siempre será un punto del mapa.
- El tercer y último paso consiste en la representación visual de la solución que se ha calculado mediante el algoritmo anterior.

4.3.2.1 Clasificación de datos y capas en Leaflet

Este punto permite obtener un conjunto de datos ordenados según ciertos criterios para que la aplicación pueda interpretarlos correctamente.

La lectura de los datos de entrada se realiza sobre el fichero que contiene todos los elementos en notación JSON. Debido a su gran tamaño esta lectura se realiza asíncronamente.

Gracias a esta notación y a la flexibilidad que proporciona JavaScript en cuanto al tipo de variables podemos trabajar con un único objeto “feature” que es capaz de proporcionarnos todos los atributos asociados al mismo.

```
    "type": "Feature",
    "id": "way/-213088",
    "properties": {
     "amenity": "toilets",
     "female": "yes",
     "indoor": "room",
     "level": "1",
     "male": "yes",
     "name": "Aseos traseros Planta 1",
     "wheelchair": "yes",
     "id": "way/-213088"
    },
    "geometry": {
     "type": "Polygon",
     "coordinates": [
                          ...
      ]
     ]
    }
```

Para cada elemento, se procesa la información de su nivel y se añade a la capa vectorial que le corresponde en función del mismo.

Además, todos los elementos accesibles (aulas, aseos, laboratorios, etc) tienen asociados una o varias puertas gracias a las cuales, el elemento puede ser indexado en un array o colección de puertas que permitirán representar los puntos de origen y de destino mediante marcadores de Leaflet.

Cada marcador que representa una puerta de un área se visualiza únicamente en el nivel al que pertenece y tiene asociado un evento.

El manejador del evento de cada marcador es el encargado de desplegar un menú lateral o sidebar que sirve de espacio para mostrar al usuario toda la información relativa al área que ha seleccionado y permite establecer el origen hacia un cierto destino.

4.3.2.2 Estructura del algoritmo para el cálculo de recorridos mínimos

Para resolver el problema del cálculo de recorridos mínimos en la facultad de informática hemos decidido dividirlo en dos subproblemas más pequeños pues, cada parte o "edificio" consta de sus propios elementos verticales de conexión en cada una de las 3 plantas disponibles.

De este modo, la idea consiste en tener dos edificios por separado de manera que ambos estén unidos mediante un nodo o conexión situado en la planta 0, en este caso la entrada a la biblioteca.

Por tanto, una solución que comprenda un recorrido desde un área de la biblioteca hasta otro área del edificio principal o viceversa estará compuesta por dos soluciones individuales que son:

1. El recorrido mínimo encontrado desde el área de la biblioteca hasta el punto de conexión de la planta 0.

2. El recorrido mínimo encontrado desde el área del edificio principal hasta el punto de conexión de la planta 0.

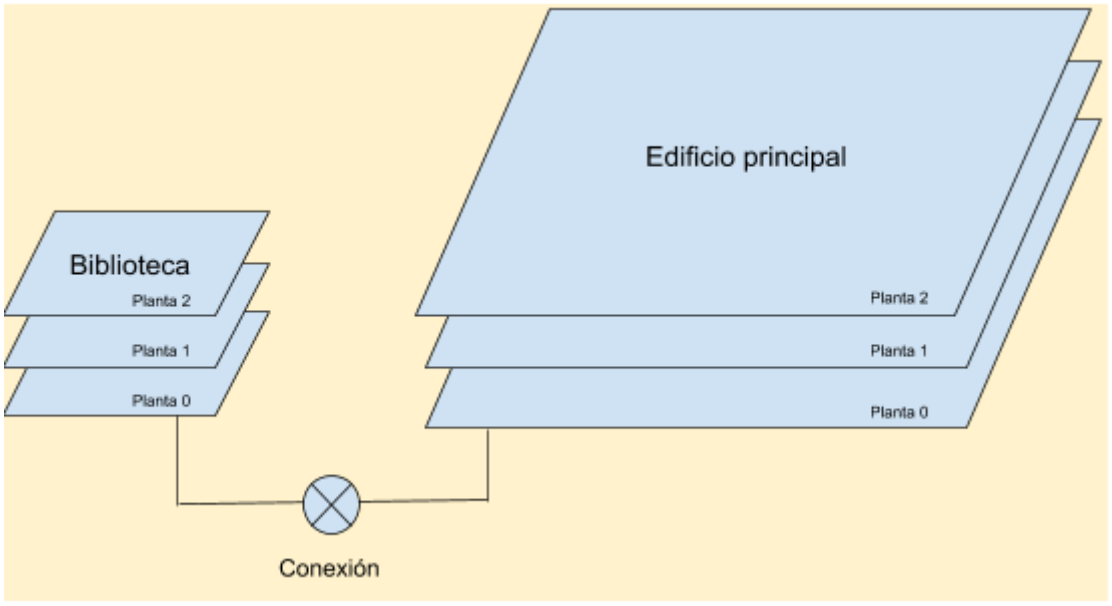


Figura 21: Esquema de edificios

Una posible aproximación sería la siguiente:

```
if (origen != destino){
 if (distintosEdificios(origen, destino)){
 intermedio = entrada_biblioteca;
 solucion = backtracking(origen, intermedio, ... ) +
    backtracking (destino, intermedio, ... );
 }
 else{
 solucion = backtracking (origen, destino, ...)
 }
}
```

Como vimos anteriormente, OpenTripPlanner solo es capaz de generar recorridos en 2D, es decir, que el origen y el destino sean puntos que se encuentren en la misma planta, pues no es posible establecer conexiones verticales con las demás plantas.

Para poder construir una solución 3D o que permita la transición entre niveles, es necesario establecer un nodo intermedio en cada nivel que, corresponderá con el ascensor o escaleras tal que de un punto hasta él, sea el recorrido mínimo posible.

De esta manera, se van construyendo soluciones parciales intercambiando en cada caso el punto de origen por el punto intermedio que se haya encontrado y pasando al siguiente nivel para volver a construir otra solución parcial.

Finalmente, el conjunto de estas soluciones parciales constituirá una solución final cuya distancia total, sea la mínima de las sumas que se han ido obteniendo durante el proceso.

El esquema del algoritmo es el siguiente:

```
backtracking (punto_A, punto_B, distanciaActual, distanciaMaxima,
   solucionParcial, solucionTotal, nivelActual, wheelchair, ){

 //CASO BASE: MISMO NIVEL
if (mismoNivel(punto_A, punto_B)){

  otpResponse = otp_server(punto_A, punto_B, nivelActual,
     wheelchair);
  //ES UNA SOLUCION
  if (distanciaActual + otpResponse.distancia < distanciaMaxima){

   actualizarDistanciaMaxima(...)
   solucionTotal.push(solucionParcial)
  }
}

//CASO RECURSIVO: DISTINTO NIVEL, HAY QUE MOVERSE VERTICALMENTE
//ENTRE PLANTAS
else{

 //Recorre la lista de elementos de conexion vertical
 //(ascensores, escaleras)
 for (elemento in lista_conexiones){

  otpResponse = otp_server(punto_A, elemento, nivelActual,
     wheelchair);

  if (distanciaActual + otpResponse.distancia < distanciaMaxima){

   actualizarDistanciaActual(...)
   solucionParcial.push(otpResponse)

   backtracking(elemento, punto_B,
      distanciaActual,
      solucionParcial, solucionTotal,
      nivelActual - 1,
      wheelchair);
  }//FIN IF

 }//FIN FOR

}//FIN ELSE

}//FIN BACKTRACKING
```

El parámetro wheelchair determinará si se han de tener en cuenta las escaleras como elementos de conexión o si por el contrario, sólo serán válidos aquellos caminos cuya transición entre plantas se realice mediante ascensores.

Una vez calculado el recorrido para el origen y destino que ha proporcionado el usuario y teniendo en cuenta si ha seleccionado el uso de silla de ruedas o no, el siguiente paso consiste en la representación gráfica del resultado obtenido.

4.3.2.3 Representación de la solución

Mediante una tercera clase que proporciona el plugin indoor de Leaflet, concretamente la clase L.Control.Level que extiende de la clase L.Control, hemos podido adaptarla a nuestro proyecto para representar la solución mediante conjuntos de líneas.

La clase L.Control permite gestionar qué datos se muestran en el mapa en función de las acciones que se realicen sobre el mismo.

La clase L.Control.Level que extiende de la anterior, permite mostrar u ocultar los elementos de la facultad que se representan mediante puntos, líneas o polígonos en función del nivel que se selecciona en un spinner lateral.

La solución total que se ha calculado en el apartado anterior, se estructura en un array mediante niveles, es decir, cada posición o nivel indexa en el array la solución calculada para ese nivel, por ejemplo:

- solucion_total [nivel], almacena el recorrido que se realiza en el nivel 0, 1 o 2, si existe solución posible.

La estructuración de la solución por niveles permite ir mostrando las soluciones parciales en cada nivel en función del evento que gestiona el cambio de nivel.

Cuando éste realiza un cambio, comprueba si previamente se estaba visualizando una solución para ese nivel, si es así, se elimina del mapa. A su vez, cuando se realiza la representación del nuevo nivel, se vuelve a comprobar si para el nuevo nivel existe una solución calculada y en caso afirmativo la añade al mapa mediante un objeto polylines o una colección de líneas.

5. Resultados

El resultado del trabajo de investigación realizado a lo largo de este proyecto y de los conocimientos adquiridos durante el mismo, nos ha permitido desarrollar nuestra propia aplicación web, sencilla en cuanto apariencia, útil en cuanto a funcionalidad e intuitiva en cuanto a la información que representa.

Mediante el uso de OpenStreepMap y su editor en Java hemos podido crear el modelo de datos para la facultad de informática de la UCM.Con OpenTripPlanner y el desarrollo de nuestro algoritmo hemos sido capaces de encontrar la ruta óptima para el usuario. Y gracias a JavaScript y la librería Leaflet hemos sido capaces de representar visualmente el resultado sobre un mapa interactivo.

La aplicación se presenta al usuario de la siguiente manera:

Figura 22: Página inicial

Cada elemento alcanzable de la facultad se ha representado con marcadores que están situados en el mismo sitio que se sitúan las puertas.

En orden a reducir la sobrecarga de marcadores, y mejorar la vista sobre el mapa, se ha optado por representar una sola puerta en aquellas aulas que disponen de dos puertas.

En el momento que un usuario selecciona uno de estos marcadores, la aplicación despliega un menú lateral que muestra la información en función del propio elemento.

Además, también muestra una lista de todos los posibles destinos que pueden ser alcanzados desde el punto que representa el marcador seleccionado.

La idea tras este método es que, un usuario que no haya estado nunca en las instalaciones y no conozca dónde se encuentra cada área, pueda seleccionar mediante una lista de nombres el destino, evitando así tener que recorrer todos los marcadores buscando el mismo.

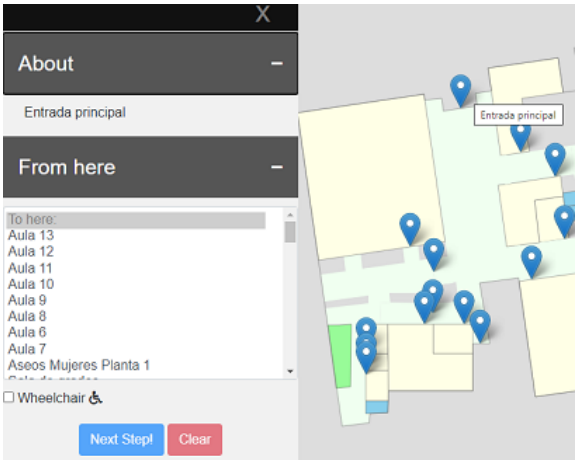


Figura 23: Menú para elegir destino

A continuación, se muestran algunos casos de prueba en los que se puede apreciar cómo se visualizan los recorridos y cómo se comporta la aplicación.

5.1. Caso primero: Navegación en el mismo edificio (Ed. aulas)

Para las siguientes pruebas, se ha tomado el edificio de aulas de la facultad de informática excluyendo la biblioteca pues, posteriormente veremos el comportamiento de la aplicación para la instalación al completo (edificio de aulas + biblioteca).

5.1.1. Navegación en la misma planta

Para el siguiente caso de prueba, se selecciona como origen la entrada principal del edificio y como destino el aula 4 y se pulsa el botón de la Figura 24.

Figura 24: Botón de inicio

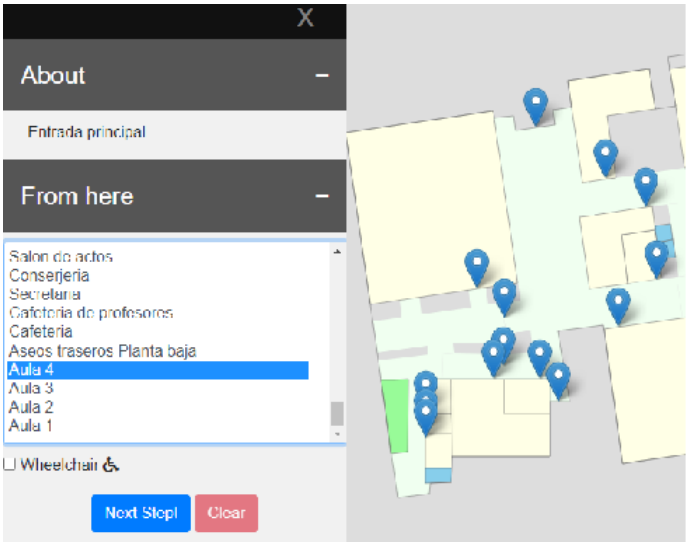


Figura 25: Destino elegido

La aplicación muestra para ese nivel, la solución obtenida.

Figura 26: Ruta dibujada

Además, se permite que una vez dibujada una ruta se pueda borrar del mapa mediante el botón *Clear* que se activa si hay al menos una solución dibujada sobre el mismo.

Figura 27: Botón de reinicio

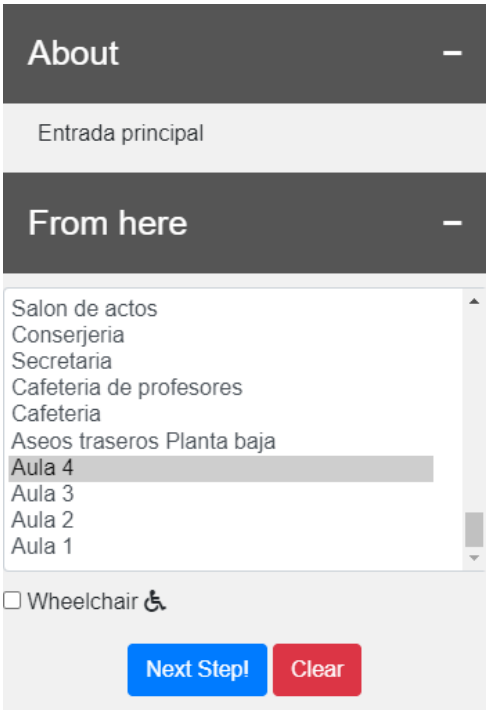


Figura 28: Botón activo

Al pulsar sobre el botón, el mapa vuelve al estado inicial donde no existía ninguna solución dibujada sobre él.

5.1.2. Navegación entre plantas

En el siguiente caso, la aplicación calculará el recorrido desde el salón de actos situado en la planta 0 hasta el laboratorio 10 situado en la planta 2.

Figura 29: Recorrido en Planta 0

La imagen muestra la parte de recorrido que se realiza en la planta 0, en este caso nos indica que debemos continuar por las escaleras principales.

Cuando avanzamos a la planta 1 el resultado es el siguiente:

Figura 30: Recorrido en Planta 1

Lo cual nos indica que en esa planta no hay recorrido y que debemos avanzar hasta la siguiente planta.

Al alcanzar la planta 2 el resultado que se visualiza es el recorrido desde las escaleras por las que nos ha mandado subir en la planta 0 hasta el laboratorio indicado:

Figura 31: Recorrido en Planta 2

5.2. Navegación entre edificios (Ed. aulas y biblioteca)

En los siguientes apartados se muestra cómo se relacionan ambos edificios mediante el nodo de conexión que existe en la planta 0 o lo que es lo mismo, mediante la entrada a la biblioteca.

5.2.1. Navegación en la misma planta

Para el siguiente paso nos situamos en la planta 0 de la facultad, concretamente en la entrada al edificio y seleccionamos los aseos de minusválidos situados en el edificio de la biblioteca.

Figura 32: Ruta dibujada en Biblioteca

Podemos comprobar que el resultado es el que se esperaba.

5.2.2. Navegación entre plantas

Para el siguiente caso, partimos del destino anterior, los aseos de minusválidos de la biblioteca en la planta 0 y seleccionamos como destino la asociación Ascii, además seleccionamos la opción de silla de ruedas para el cálculo de la ruta.

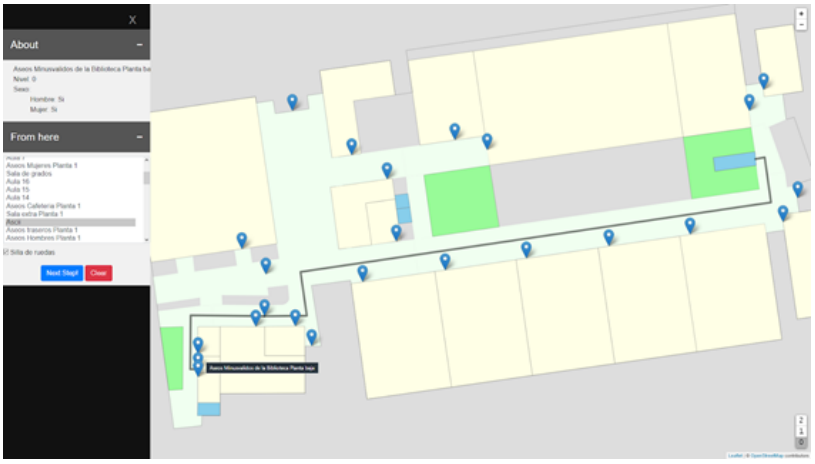


Figura 33: Recorrido de Biblioteca en Planta 0

La aplicación nos indica que debemos de tomar los ascensores traseros para subir a la planta 1.

Figura 34: Recorrido de Biblioteca en Planta 1

En la planta 1, podemos ver el recorrido desde la salida de los ascensores hasta el espacio dedicado a la asociación.

Se puede visualizar una presentación de la aplicación en los siguientes enlaces.

Video largo (4:25):
`https://drive.google.com/file/d/1Lp2lzNFuInKlvOUeH_2jEQl5S6ZzRYM9/view?usp=sharing`

Video corto (2:45):
`https://drive.google.com/file/d/1Z58i13CV5ukngI8SLS-wIVF5yt5WUoKM/view?usp=sharing`

6. Conclusiones y Trabajo futuro

Como el título del proyecto indica, el propósito consistía en encontrar un modelo que permitiese identificar con anterioridad las posibles barreras arquitectónicas entre dos puntos distintos del edificio de facultad de informática de la Universidad Complutense de Madrid (UCM). A pesar de ser un edificio totalmente accesible por cualquier persona con problemas de movilidad, también se podría adaptar a otros edificios, permitiendo así que estas personas pudieran evitar cualquier barrera siempre que fuese posible.

En este proyecto se ha utilizado un modelo de mapas en 2D para la representación de las soluciones. Frente al modelo de programación con restricciones planteado inicialmente, el modelo actual permite mostrar la solución de una manera más visual e interactiva, mejorando la accesibilidad y usabilidad web.

Dado que ni OpenStreetMap (OSM) ni OpenTripPlanner (OTP) son capaces por sí mismas de resolver el problema, pero cada una resuelve una parte del mismo, ha sido necesario establecer una conexión entre ellas, mediante la programación y el uso de un algoritmo de backtracking para conseguir una solución completa.

Nuestra aplicación, que ha tenido como referencia el proyecto análogo sobre la Universidad de Heidelberg, es capaz de mostrar al usuario una representación gráfica del edificio de la facultad de informática y todos aquellos caminos calculados para dos puntos cualesquiera de la misma.

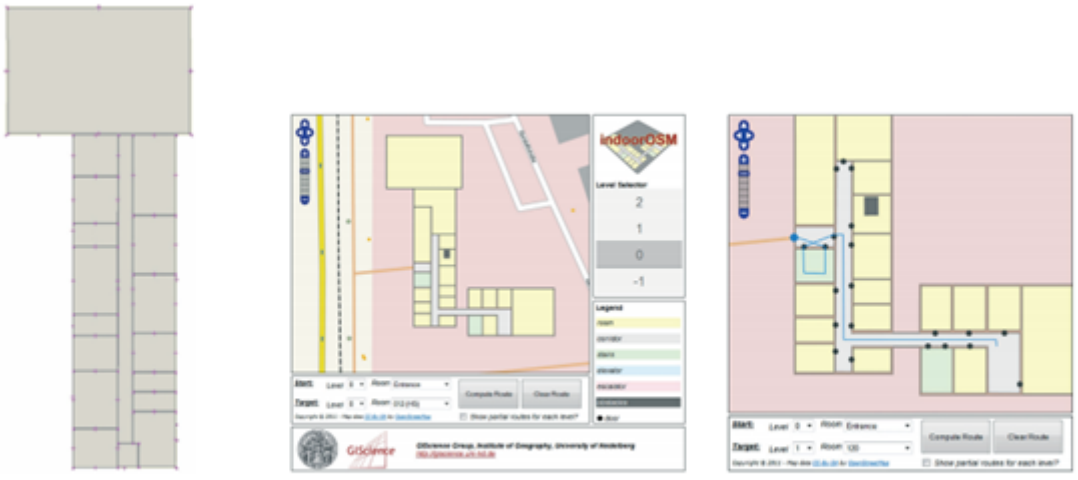


Figura 35: Universidad de Heidelberg, IndoorOSM

A pesar de haber conseguido nuestro objetivo principal, consideramos que su desarrollo aún se encuentra en una fase temprana, pues, todavía hay muchas mejoras que podrían añadir más valor al trabajo ya realizado.

Por ejemplo, en lo referente a funcionalidad, la incorporación de un modelo 3D en lugar del 2D actual permitiría al usuario una visualización mucho más detallada de todos los elementos del edificio.

Durante su desarrollo hemos podido aplicar conocimientos ya obtenidos durante nuestra formación en la universidad. Algunos, como por ejemplo una programación eficiente y clara junto con la capacidad de adaptación a los cambios que han ido surgiendo a lo largo del ciclo de vida han resultado ser muy útiles. También hemos adquirido nuevos conocimientos relativos a los Sistemas de Información Geográfica o SIG en los que se incluyen las herramientas sobre las que hemos trabajado. Conocer el funcionamiento de los SIG y cómo manejan los datos sobre los que trabajan ha sido necesario para concluir con éxito el trabajo.

Nuestro deseo es que todas aquellas personas que durante el desarrollo de su actividad cotidiana se ven afectadas por problemas de movilidad, sin importar cual sea el origen de la misma, puedan ver mejorada su calidad de vida gracias al uso de este tipo de aplicaciones.

Finalmente, esperamos que al igual que la universidad de Heidelberg nos sirvió como ejemplo y motivación, aquellas personas interesadas puedan utilizar los conocimientos y metodologías aquí empleadas para replicar el proyecto en otros edificios como centros medicos, museos, estaciones de transporte, etc.

7. Conclusions and future work

As the project title suggest, the purpose was to find an approach that could make possible to identify in advance potential accessibility restrictions between two different points within the Computer Science building of the Universidad Complutense de Madrid (UCM). Despite of being a fully accessible building by any person with mobility difficulties, it could also be adapted to another buildings, thus allowing that they could avoid any restriction as long as it was possible.

In this project a 2D map model has been used for displaying the solutions. Compared to the restrictive programming model that was initially proposed, the current one allows to display a more interactive solution to the user while improving the web accessibility and usability.

Since neither OpenStreetMap(OSM) nor OpenTripPlanner(OTP) are able to resolve the problem, but each one resolves a part, It has been necessary to establish a connection between them by programming and the use of a backtracking algorithm to achieve a complete solution.

Our application, whose reference has been the analogous project of the Heidelberg University, Its able to display a representation of the Computer Science Faculty building and display the paths that have been calculateds as a solution to this problem.

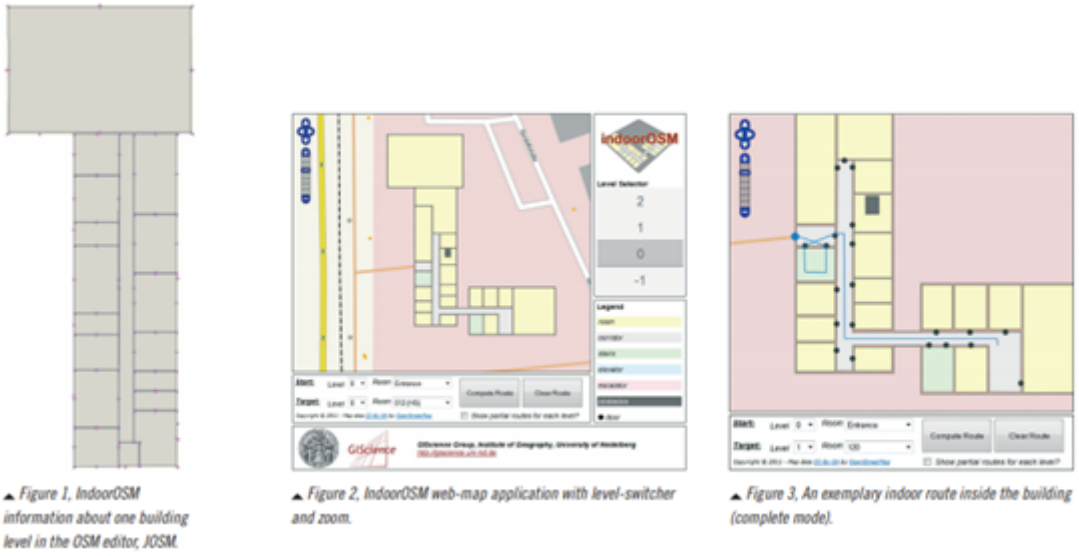


Figura 36: Heidelberg University, IndoorOSM

Although the main goal has been reached, we consider that the development is still on an early phase since there are many improves that could add more value to the work already done.

For example, regarding to functionality, a 3D model instead of the current 2D could be also incorporated to the project, this way the displaying of the whole elements of the building would be much more detailed.

During its development we have been able to apply learnings obtained during our training at the university. Some of them, such as efficient programming along with the capacity to adapting to change during the life cycle have prove to be very useful. We also have learned new skills about the geographic information system or GIS that includes the software tools on which we have been working on. Knowing the behaviour of he GIS and how they handle the data has been necessary to conclude the work successfully.

We hope that all those persons who during their daily activities are affected by mobility problems, no matter the origin of it, can see their quality of life improved thanks to the use of this kind of applications.

With that in mind, we also hope that same as Heidelberg University served us as an example and motivation, those persons who are interested on replicating the same project on other infrastructures, e.g., health centres, museums, transport stations, could also use the information here described to make it possible.

Referencias

[1] Capas Raster. https://desktop.arcgis.com/es/arcmap/10.3/manage-data/geodatabases/raster-basics.htm.

[2] JOSM. https://wiki.openstreetmap.org/wiki/Simple_Indoor_Tagging.

[3] GIS. https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n_geogr%C3%A1fica.

[4] Wikipedia. https://es.wikipedia.org/wiki/Geocodificaci%C3%B3n.

[5] OpenStreetMap. https://wiki.openstreetmap.org/wiki/About_OpenStreetMap.

[6] Backtracking. https://es.wikipedia.org/wiki/Vuelta_atr%C3%A1s.

[7] OpenTripPlanner. https://www.opentripplanner.org/.

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

[9] Observatorio Estatal de la Discapacidad. Informe olivenza 2018, sobre la situación general de la discapacidad en españa, 2018.

[10] 2017 Risewise. Rise women with disabilities in social engagement - "this project has received funding from the european union's horizon 2020 research and innovation programme under the marie skłodowska-curie grant agreement no 690874". http://www.risewiseproject.eu/.

[11] Missing Maps. https://www.missingmaps.org/es/.

[12] Medicos Sin Fronteras. https://www.msf.es/mapaton-la-cartografia-herramienta-humanitaria.

[13] ArcGis. https://www.esri.com/arcgis-blog/products/arcgis-indoors/mapping/what-is-indoor-mapping/.

[14] Sigpac. http://sigpac.mapama.gob.es/fega/visor/.

[15] ESRI. https://www.esri.es/arcgis/.

[16] PicLayer. https://wiki.openstreetmap.org/wiki/JOSM/Plugins/PicLayer.

[17] Comparación Leaflet. https://en.wikipedia.org/wiki/Leaflet_(software).

[18] OpenLayers. https://openlayers.org/.

[19] GeoJSON. https://geojson.org/.

[20] Marcus Young. Opentripplanner-creating and querying your own multi-modal route planner. *https://github.com/marcusyoung/otp-tutorial/raw/master/introotp.pdf*, 2018.

[21] Nair Isabel Braga Simões Alves. *Uma solução para navegação indoor*. PhD thesis, 2012.

[22] Guillermo Amat, Javier Fernandez, Alvaro Arranz, and Angel Ramos. Using open street maps data and tools for indoor mapping in a smart city scenario. *http://repositori.uji.es/xmlui/bitstream/handle/10234/98703/03agile2014_82.pdf*, 2014.

[23] Zhiyong Wang and Alexander Zipf. Using openstreetmap data to generate building models with their inner structures for 3d maps. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2017.

[24] Zhiyong Wang and Lei Niu. A data model for using openstreetmap to integrate indoor and outdoor route planning. *Sensors*, 18(7):2100, 2018.

[25] Jorge Rocha and Nair Alves. Osm indoor: moving forward. In *OGRS2012-Symposium proceedings*, pages 261–167, 2012.

[26] Mohammad Forghani and Mahmoud Reza Delavar. A quality study of the openstreetmap dataset for tehran. *ISPRS International Journal of Geo-Information*, 3(2):750–763, 2014.