Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

WALKABILITY ANALYZER

Alumno/Alumna: Caballero, García, Jessica

Director/Directora (1): Villamañe, Gironés, Mikel

Director/Directora (2): Azanza, Sesé, Maider

Curso: 2017-2018

Fecha: Bilbao, 23 de julio de 2018

Resumen

El conocido como "miedo al delito" o inseguridad afecta a la calidad de vida, provocando que los ciudadanos eviten ciertos espacios públicos, influenciando la elección del modo de transporte e interfiriendo en la toma de decisiones de las personas. Las ciudades pensadas para el peatón reducen el impacto en el medio ambiente, son más saludables y seguras. Para desarrollar una ciudad centrada en los ciudadanos, es necesario analizar las valoraciones que las personas le otorgan a las rutas a pie que realizan en las zonas urbanas.

Walkability Analyzer permite analizar las rutas realizadas por los ciudadanos ya almacenadas en él, y aplicar gran cantidad de filtros para poder visualizar las rutas en función de las características deseadas. De esta manera, al poder visualizar las rutas junto con la valoración de las mismas, se puede realizar diversos estudios para analizar, por ejemplo, la seguridad en una zona de la ciudad o lo apto que es un espacio para las personas con movilidad reducida. Este Trabajo Fin de Grado se ha desarrollado en colaboración con los investigadores del grupo CRIM-AP de la UPV/EHU.

Abstract

The "fear of crime" or insecurity affects the quality of life, causing citizens to avoid certain public spaces, influencing the choice of mode of transport and interfering in the decision-making of people. Cities designed for pedestrians reduce the impact on the environment, they are healthier and safer. To develop a city centered on citizens, it is necessary to analyze the ratings that people give to the walking routes that they carry out in urban areas.

Walkability Analyzer allows to analyze the routes carried out by the citizens already stored in the system, and to apply a large number of filters to be able to visualize the routes according to the desired characteristics. In this way, to be able to visualize the routes together with with their ratings, diverse studies can be realized to analyze, for example, the security in a zone of the city or how apt a space is for people with reduced mobility. This Final Degree Project has been developed in collaboration with the researchers of the CRIM-AP group of the UPV / EHU.

Laburpena

"Delituari beldurrak" edo segurtasun ezak bizitza kalitatean eragiten du, herritarrek espazio publiko jakin batzuk saihestea ekar dezakeelarik, edo garraio modua aukeratzeko eta pertsonen erabakiak hartzeko orduan eragina izan dezakeelarik. Oinezkoentzat diseinatutako hiriek ingurumenean duten eragina murrizten dute, osasuntsuagoak eta seguruagoak dira. Herritarrei zuzendutako hiri bat garatzeko, herritarrek oinez egiten dituzten ibilbideei ematen dizkieten balorazioak aztertu behar dira.

Walkability Analyzer-ek dagoeneko sisteman dauden herritarrek egindako ibilbideak aztertu eta iragazki ugari egiteko aukera eskaintzen du, ibilbideak nahi diren ezaugarrien arabera ikusteko. Horrela, ibilbideak eta berauen balorazioa ikusteko gai izanik, azterketak egin daitezke, esate baterako, hiriaren inguruko segurtasuna edo mugikortasun mugatua duten pertsonentzako espazio bat egokia den. Gradu Amaireako Lan hau UPV / EHUko CRIM-AP taldeko ikertzaileekin lankidetzan garatu da.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1. Introducción

El conocido como "miedo al delito" afecta a la calidad de vida, motivando conductas de auto-protección como la evitación de ciertos espacios públicos (San Juan, 2012), afectando a las decisiones de los padres sobre la seguridad de sus hijas e hijos (Vozmediano, 2017), e influenciando la elección del modo de transporte y con ello, la sostenibilidad urbana y la salud de la ciudadanía. Por ejemplo, al reducir el temor, la probabilidad de desplazarse andando o en bicicleta aumenta (N. C. McDonald) (C. Kelly, 2010).

El miedo al delito se ve afectado por experiencias que ocurren en un momento y lugar determinado (Sacco., 1989) (Pain., 2000), y fluctúa de modo importante cuando se está en lugares y situaciones diferentes (Solymosi, 2015). Por ello, es en este punto donde surge el problema, ya que las medidas de encuesta tradicionalmente utilizadas no son adecuadas para reflejar esta variabilidad en el espacio y el tiempo, y son muy limitadas si se pretende relacionar las percepciones con las rutas que llevan a cabo las personas en su vida diaria.

El concepto de Walkability o entornos caminables indica en qué medida un espacio concreto es amigable para el peatón. Es decir, describe aquellos espacios que invitan a ser transitados a pie, ofreciendo a los ciudadanos la oportunidad para incrementar su actividad física, mejorando su salud (Pucher et al., 2010), reducir su impacto ambiental y relacionarse. Se trata de un concepto de importancia creciente, que aúna los intereses de las áreas de salud pública y diseño sostenible (Ewing y Handy, 2009).

Las ciudades pensadas para el peatón reducen el impacto en el medio ambiente, son más saludables y seguras y ofrecen más oportunidades para la interacción y la cohesión social; lo que avala la idoneidad de incorporar estas ideas a los modelos de “Smart City” o ciudades inteligentes. Así, por ejemplo, el Libro Blanco Smart Cities indica que: *“el propósito final de una Smart City es alcanzar una gestión eficiente en todas las áreas de la ciudad (urbanismo, infraestructuras, transporte, servicios, educación, sanidad, seguridad pública, energía, etcétera), satisfaciendo a la vez las necesidades de la urbe y de sus ciudadanos”*.

Sin embargo, un diseño de ciudad que favorezca el tránsito a pie sólo es exitoso si impacta en el proceso de toma de decisiones de la ciudadanía, que opta por esta estrategia de movilidad y no una alternativa. En ocasiones, son los propios ciudadanos los que escogen un medio de transporte alternativo evitando desplazarse a pie, privándose así de los beneficios que esta actividad genera no solo en el organismo del ciudadano, sino también en su calidad de vida y en el desarrollo de la propia ciudad. La ciudadanía, a la hora de desplazarse a pie, escoge una ruta o la desecha en función de aspectos como la iluminación o la falta de ella, la sensación de inseguridad o la carencia en infraestructuras necesarias, por ejemplo, la falta de rampas o escaleras mecánicas en tramos de pendientes pronunciadas.

1.1. Origen y propósito del proyecto

Debido a la importancia de recabar información sobre los aspectos que determinan la movilidad a pie en la ciudad, sobre los objetivos de las rutas a pie y la valoración de las calles por las que se realizan estas rutas, se crearon los proyectos de Walkability Capturer y Walkability Analyzer. La aplicación móvil de Walkability Capturer, ya desarrollada en otro proyecto, recoge las rutas realizadas a pie por los usuarios, y posteriormente, almacena las opiniones de los mismos sobre dicha ruta. En cambio, este proyecto, denominado Walkability Analyzer, consiste en el desarrollo de una aplicación web que a partir de la información que aporten los usuarios de la aplicación móvil, permita su análisis y visualización.

Este Trabajo Fin de Grado se realiza en colaboración con el grupo CRIM-AP de la UPV/EHU, interesados en aumentar y promocionar la calidad de vida urbana, tratando de mejorar en la construcción de ciudades amigables, saludables y seguras para todos los ciudadanos y colectivos sociales. Es por ello por lo que la aplicación Walkability Analyzer deberá facilitar a dichos investigadores la visualización de las rutas que realiza la ciudadanía en función de una serie de criterios, (su objetivo, su valoración en un aspecto concreto, el número de viandantes que realizan esa ruta, etc.). Por ejemplo, visualizar las rutas realizadas por los hombres mayores de 50 años entre dos puntos concretos.

Para facilitar la labor de los miembros del grupo CRIM-AP, la aplicación web deberá permitir la descarga de todos los datos en un formato que facilite su posterior análisis con otras herramientas. De esta manera, el análisis y la obtención de estadísticas respecto al comportamiento de la ciudadanía podrá realizarse de una manera más exhaustiva. Por otra parte, los investigadores deberán tener la libertad de configurar el sistema, por ejemplo, modificar las encuestas que realiza la aplicación móvil de Walkability Capturer sobre las rutas que realizan los viandantes.

Walkability Analyzer, además de prestar servicio a los psicólogos e investigadores, podría llegar a prestar servicio a la ciudadanía en general, de forma que cada usuario pueda buscar y escoger la ruta que más se adecue a sus necesidades. Por ejemplo, la mejor ruta entre Termibús y el Guggenheim para ir de compras, o la más bonita entre esos dos mismos puntos. El usuario que utilice la aplicación web tendrá así la oportunidad de descubrir nuevos lugares, u orientarse en función de sus necesidades y preferencias, si no conoce la ciudad, y no sólo a través de las rutas más cortas, que es lo que hacen los sistemas de navegación actuales existentes en el mercado.

De esta forma, con la utilización tanto de Walkability Capturer como de Walkability Analyzer, se conseguirá la información necesaria para analizar las decisiones de los usuarios a la hora de desplazarse a pie por la ciudad. Con la utilización del sistema completo se dará la posibilidad a los usuarios de detectar las rutas más y menos utilizadas, y comparar dichos trayectos para extraer recomendaciones para el diseño urbano.

Este proyecto forma parte del proyecto *City4All (la ciudad para tod@s)*, del equipo CRIM-AP de la Universidad del País Vasco UPV/EHU. Esta investigación ha obtenido el informe favorable del Comité de Ética para las Investigaciones relacionadas con Seres Humanos de la UPV/EHU.

1.2. Razones de la elección del TFG

En primer lugar, y por ello la razón más importante para la elección de este Trabajo Fin de Grado, es la idea del mismo. Walkability Analyzer es un proyecto que se basa en la idea de conseguir que las ciudades sean más aptas para los ciudadanos. De este modo se aumenta la calidad de vida de las personas, mejorando el tránsito a pie y la seguridad en la ciudad. Por ello, este proyecto no sólo es un trabajo que una vez finalizado se olvide, sino que puede ayudar a los miembros del grupo CRIM-AP a obtener estadísticas que mejoren su labor.

En segundo lugar se encuentra la posibilidad de aprender tecnologías que no se han trabajado en el grado universitario pero que, sin embargo, son muy importantes en la actualidad para desarrollar aplicaciones web rápidas y robustas siguiendo arquitecturas modernas. En este caso, aprender a diseñar y a trabajar con una buena estructura, separando y diferenciando las partes de front-end, back-end (ver apartado 2.1) y Base de Datos.

En tercer lugar, y en relación a la segunda razón ya explicada, se encuentra la superación personal de realizar un proyecto de esta magnitud individualmente, aplicando los conocimientos aprendidos durante la carrera, incorporando nuevas tecnologías y tomando de manera personal las decisiones que afecten al proyecto. Además, el hecho de desarrollar una aplicación web nueva, permite decidir de manera libre tanto el aspecto gráfico de la misma como su funcionamiento, pudiendo seguir un camino sin restricciones marcadas.

2. Planteamiento inicial

En este apartado se abordarán algunos de los aspectos iniciales del proyecto. Por un lado, se tratarán desde los objetivos de la realización de Walkability Analyzer hasta cómo se pretende estructurar y desarrollar el proyecto. Por otro lado, se definirán las tareas que se van a desarrollar, la forma en la que estas se van a llevar a cabo, se estimarán los tiempos de realización de las diferentes partes del proyecto y se analizarán los riesgos y el coste de Walkability Analyzer.

2.1. Definiciones, acrónimos y abreviaturas

- **API**: Una API, abreviatura del término en inglés Application Programming Interfaces (Interfaces de programación de aplicaciones), es una especificación sobre cómo un módulo interactúa con otro. Una API consta de un conjunto de comandos, funciones y protocolos que permite al programador utilizar dichas funciones para interactuar con otro programa y así simplificar el proceso de desarrollo. (Periodico ABC, 2015)

- **REST**: REST, abreviatura del término en inglés Representational State Transfer (Transferencia de Estado Representacional), es cualquier interfaz entre sistemas que utilice HTTP para obtener datos o generar operaciones entre ellos. De esta manera, cada petición HTTP contiene la información necesaria para ejecutarla, simplificando así el desarrollo. En un sistema REST existen cuatro operaciones importantes relacionadas con los datos: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar). (Marqués, 2013)

- **API REST**: Un API REST es una API desarrollada según la arquitectura REST. (BBVA, 2016)

- **Front-end**: El front-end de una aplicación se considera toda la lógica ubicada en la parte cliente del servidor, es decir, en el navegador. Esta parte de la aplicación se ocupa de los componentes externos del sitio web, es la responsable de que el usuario pueda visualizar la aplicación de una manera correcta, maquetada y estilizada. El front-end de una página web es la parte "gráfica" de la misma, y, además, se encarga de realizar las peticiones correspondientes a la parte servidor de la aplicación, y posteriormente, mostrar el resultado de dichas consultas. (SERPROGRAMADOR, 2014)

- **Back-end**: El back-end se considera la parte de la aplicación que está situada en el lado del servidor. Esta parte de la aplicación, entre otras cosas, es la encargada de interactuar con las bases de datos, gestionar y manejar los datos. En esta parte de la aplicación reside la funcionalidad del sistema, siendo la encargada de responder a las peticiones realizadas por el front-end. Debido a ello, esta parte de la aplicación carece de interfaz gráfica. (SERPROGRAMADOR, 2014)

- **Framework**: Un Framework es un entorno de trabajo o un marco de desarrollo. En ocasiones puede estar constituido por herramientas que permiten estructurar de una manera más eficiente el código y reutilizar proyectos. (Lenguajes de programación, 2015)

- **JSON**: JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas y ordenadores también es simple interpretarlo y generarlo. (JSON , s.f.)

- **SPSS**: SPSS son las siglas de *Statistical Package for the Social Sciences*, que en su traducción al castellano quedaría como "Paquete Estadístico para las Ciencias Sociales". Se trata de un programa o software estadístico que se emplea muy a menudo para realizar análisis más exhaustivos de los datos, por ejemplo, en investigaciones. (QuestionPro, s.f.)

- **QGIS**: QGIS es un Sistema de Información Geográfica (SIG) de Código Abierto licenciado bajo GNU - General Public License. (QGIS, s.f.)

2.2. Objetivos

Los objetivos principales de la realización de este proyecto son:

- La creación de una aplicación web que permita a los investigadores el análisis de los datos recogidos permitiendo la visualización de las rutas en función de distintos criterios. Es decir, Walkability Analyzer debe permitir a los psicólogos, mediante el uso de una serie de filtros o criterios, visualizar en un mapa las diferentes rutas que siguen los transeúntes.

- La aplicación debe permitir la descarga de los datos almacenados en el sistema en un formato que permita su posterior análisis con herramientas estadísticas más específicas, como SPSS. Además, debe permitir la descarga de datos en formatos adaptados para trabajar con herramientas QGIS.

- Crear una aplicación web intuitiva, en la que no se requieran amplios conocimientos informáticos para poder utilizar el sistema de una manera fácil y óptima. Para evaluar este objetivo se realizarán encuestas a diferentes tipos de usuarios sobre la usabilidad de la aplicación web.

- Aprendizaje personal en la realización de páginas web mediante la utilización de frameworks de desarrollo utilizados actualmente. Aumentar los conocimientos de nuevos lenguajes de programación utilizados en el sector del desarrollo de plataformas web.

En función del grado de consecución de los objetivos principales y del estado del proyecto Restorative City que se desarrolla en paralelo a este, se han planteado los siguientes objetivos secundarios:

- Profundizar en el tipo de análisis que permita hacer la aplicación web a los investigadores. Es decir, permitir que la aplicación web pueda realizar análisis más exhaustivos de los datos almacenados en el sistema.

- Desarrollo de un buscador para los usuarios en el que, en base a los datos recogidos y almacenados en el sistema, proporciona a los transeúntes la mejor ruta entre dos puntos atendiendo a los criterios que el usuario decida.

- Integración de la visualización de rutas con la visualización de espacios restauradores del proyecto Restorative City.

2.3. Alcance

En este apartado se analizará la forma en la que se va a desempeñar el proyecto según el ciclo de vida seleccionado para su desarrollo. Además, se definirá la estructura en la que se va a descomponer el trabajo a realizar.

2.3.1. Ciclo de vida

Walkability Analyzer se desarrollará siguiendo un modelo de ciclo de vida incremental. Este tipo de modelo consiste, a su vez, en la iteración de varios ciclos de vida en cascada. Es decir, las actividades a realizar en el proyecto se repiten en fases o iteraciones, aumentando en cada una de las fases el nivel de desarrollo del producto. Así, en cada iteración el proyecto adquiere una mayor funcionalidad, y se obtienen prototipos más completos a medida que avanza el desarrollo del proyecto.

La razón principal por la que se ha escogido este modelo de ciclo de vida y no otro se debe a que el ciclo de vida incremental se basa en la obtención de prototipos. Debido a ello, al desarrollar el proyecto en fases se puede determinar de una manera más adecuada las necesidades de cada uno de los prototipos. En este caso, al tener que realizar tanto el front-end como el back-end de una página web que consta de diferentes funcionalidades, el hecho de dividir el trabajo en prototipos o partes más pequeñas resulta ventajoso a la hora de abordar el problema. Por ello, para la realización de Walkability Analyzer se realizarán seis prototipos:

- **Acceso de los usuarios:** En este prototipo se desarrollará la funcionalidad mediante la que los investigadores y administradores del sistema podrán acceder al mismo.

- **Aplicación de filtros y obtención de datos:** En la realización de este prototipo se desarrollarán los filtros aplicables a las rutas para su visualización. En este apartado se desarrollarán no solo los filtros básicos tales como género o edad del ciudadano que realiza la ruta, sino también filtros geográficos como punto de inicio, punto final o punto intermedio de la ruta.

- **Visualización de datos mediante mapa:** Una vez que se haya realizado el prototipo de filtrado de rutas, las rutas resultantes se mostrarán en un mapa que contará con diferentes opciones de visualización. Dentro de las rutas mostradas en el mapa se podrá elegir como visualizar dichas rutas, y entre las opciones de visualización se podrán mostrar las rutas por género, por horario en el que dicha ruta fue realizada o por movilidad del usuario entre otras alternativas.

- **Alta de usuarios:** En este prototipo se desarrollará la funcionalidad mediante la cual, los administradores de la aplicación podrán registrar en el sistema a nuevos investigadores, otorgarle permisos o eliminar otros usuarios.

- **Edición de cuestionarios:** Walkability Capturer, la aplicación móvil que precede a este proyecto, se basa en diversos cuestionarios, preguntas y configuraciones para obtener las valoraciones de los usuarios con respecto a las rutas realizadas. Walkability Analyzer permitirá a los investigadores, mediante esta funcionalidad, añadir, editar o eliminar tanto los propios cuestionarios como las preguntas que forman parte de ellos.

- **Descarga de datos:** En este prototipo se desarrollará la funcionalidad mediante la cual los usuarios de la aplicación podrán descargarse diferentes ficheros con los datos que sean necesarios para realizar un mayor análisis, por ejemplo, las

coordenadas de las rutas un formato compatible con sistemas QGIS o la información de las mismas en formatos compatibles con herramientas SPSS.

Se debe mencionar que cada uno de estos prototipos consta de dos subprototipos: desarrollo del apartado de back-end y de front-end. Walkability Analyzer se realizará con la mayor modularidad posible, por ello, el desarrollo de la aplicación se llevará acabo como dos subproyectos diferenciados. Este hecho, junto con haber elegido realizar el proyecto mediante un ciclo de vida incremental, conlleva una gestión más compleja del mismo y una mayor cantidad de trabajo. Esto se debe, principalmente, a que las labores de análisis y diseño, incluyendo la realización de diagramas, se deberán ejecutar para cada prototipo y subprototipo, y no sólo en el producto final, como sí ocurre en otros tipos de ciclos de vida.

Por otro lado, en caso de cometer un error, la subsanación del mismo es más sencilla, ya que sólo afectaría a la última iteración realizada, por lo que bastaría con descartar esta última parte sin que llegara a afectar a los prototipos ya realizados.

A pesar de la mayor cantidad de trabajo que requiere, este sistema se adecua en gran medida a la forma que se quiere trabajar, desarrollando el proyecto de una forma incremental gestionando los errores, por ello, y tras valorar tanto las ventajas como los inconvenientes, el ciclo de vida incremental es el escogido para el desarrollo de Walkability Analyzer.

2.3.2. Estructura de Descomposición del Trabajo

La distribución y descomposición del trabajo se ha organizado por módulos personalizados, los cuales están formados por paquetes de trabajo o tareas. Muchos de los proyectos y trabajos de este calibre se gestionan con una serie de módulos preestablecidos: organización, análisis, diseño, implementación, pruebas y documentación. En cambio, para este proyecto, los paquetes de trabajo se han agrupado en diferentes módulos según las características de los mismos. Así, en el diagrama correspondiente al EDT (Estructura de Descomposición del Trabajo), que corresponde a la *Ilustración 1*, se pueden visualizar los paquetes agrupados por características similares, por ejemplo, todos los paquetes de formación personal del proyecto están en un mismo módulo, el módulo de “Aprendizaje”.

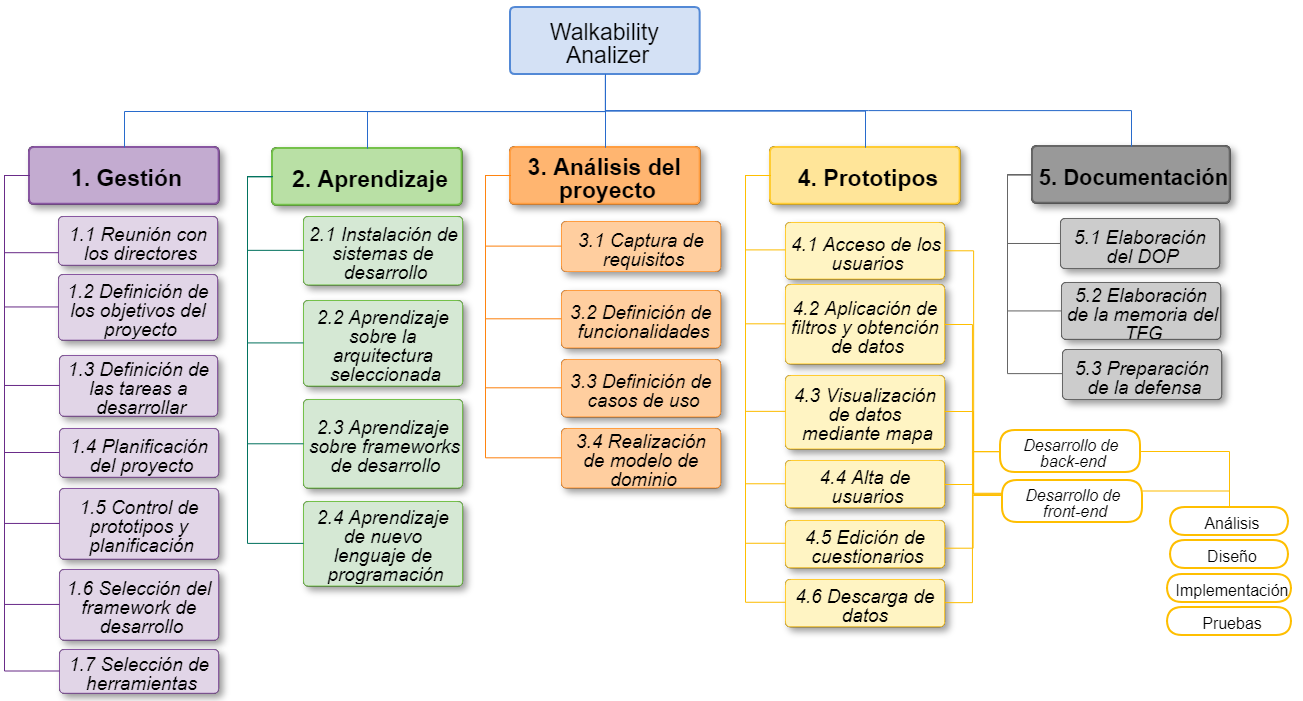


Ilustración 1 - Estructura de Descomposición del Trabajo (EDT)

Para cada paquete de trabajo o tarea se requerirá realizar una serie de tareas más o menos complejas. Por ejemplo, dentro del módulo "Aprendizaje", en el paquete de trabajo "Instalación de sistemas de desarrollo", habrá que instalar las herramientas, leer la documentación necesaria, configurar el sistema y realizar pruebas para comprobar su correcto funcionamiento. Además, existirá la posibilidad de desarrollar un paquete de trabajo antes que otro, aunque este segundo aparezca después en el diagrama EDT, siempre y cuando no exista dependencia alguna entre ellos o dicha dependencia ya se haya realizado. Por ejemplo, dentro del módulo "Prototipos", el paquete de trabajo "Alta de usuarios" se podrá realizar antes de "Visualización de datos mediante mapa".

Los módulos de los que se compone la Estructura de Descomposición del Trabajo son los siguientes:

- **Gestión:** Consistirá en definir, organizar y planificar el proyecto a realizar y sus diversas tareas. Esta fase es la primera que se realizará en el proyecto y algunos de sus paquetes de trabajo se desarrollarán durante todo el transcurso del mismo, por ejemplo, la tarea de "Control de prototipos y planificación". Además, se estudiarán los lenguajes y herramientas necesarios para el desarrollo del trabajo.

- **Aprendizaje:** Consistirá en la instalación, configuración y aprendizaje de todas las herramientas, lenguajes y programas necesarios que se utilizarán durante el transcurso del proyecto. Un proyecto de esta magnitud requiere la utilización de diversas herramientas o lenguajes desconocidos, por lo que es necesario que se dedique el tiempo suficiente al aprendizaje de los mismos.

- **Análisis del proyecto:** Consistirá en realizar las tareas necesarias para definir los requerimientos de la aplicación, sus características y las funcionalidades que poseerá cada usuario de la aplicación web. Este tipo de análisis se realizará de la aplicación en su totalidad, de esta manera se aclarará todo lo que se pretende conseguir y que características debe poseer la aplicación.

- **Prototipos:** Este módulo está compuesto por todos los prototipos de los que se

compone la aplicación. Cada prototipo tiene diferentes fases: análisis, diseño, implementación y pruebas. Es decir, por cada prototipo se ha de analizar los requerimientos que debe cumplir y las funcionalidades que debe tener, realizar su diseño junto con los diagramas correspondientes, implementar el prototipo y desarrollar las pruebas necesarias para comprobar su correcto funcionamiento.

- **Documentación:** Módulo formado por todos los paquetes de trabajo cuya finalidad es documentar los objetivos del proyecto, el desarrollo del mismo y su resultado final.

A continuación, se explicará cada paquete de trabajo presente en el diagrama EDT de una manera más detallada:

Módulo 1: Gestión

1. Gestión **Paquete de trabajo:** *1.1 Reunión con los directores*
Duración: 6 horas
Descripción: Reuniones con los directores que se sucederán durante el transcurso del proyecto. Dichas reuniones abarcan desde la reunión inicial hasta las reuniones de control o de resolución de posibles dudas que puedan surgir.
Salidas/Entregables: Documento que indique lo tratado en las reuniones y las decisiones tomadas o dudas resueltas.

Tabla 1 - Reunión con los directores

1. Gestión **Paquete de trabajo:** *1.2 Definición de los objetivos del proyecto*
Duración: 1 hora
Descripción: Establecer los objetivos del proyecto, tanto los que se consideran principales y como los que son considerados secundarios. Además se definirán las necesidades básicas del mismo.
Salidas/Entregables: Documento que indique qué objetivos se establecen como prioritarios y que objetivos son considerados secundarios.
Precedencias: Paquete de trabajo 1.1

Tabla 2 - Definición de los objetivos del proyecto

1. Gestión **Paquete de trabajo:** *1.3 Definición de las tareas a desarrollar*
Duración: 10 horas
Descripción: Definir las tareas y prototipos que son necesarios para la elaboración del proyecto. Descomponer el proyecto en subtareas para desarrollar el proyecto según el ciclo de vida seleccionado.
Salidas/Entregables: Tareas y prototipos del proyecto definidos y perfectamente documentados.
Precedencias: Paquete de trabajo 1.2

Tabla 3 - Definición de las tareas a desarrollar

1. Gestión **Paquete de trabajo:** *1.4 Planificación del proyecto*
Duración: 4 horas
Descripción: Organizar el proyecto estableciendo un orden en el desarrollo de los prototipos, un orden en la realización de las tareas y una estimación de tiempos para el desarrollo de cada una de ellas.
Salidas/Entregables: Diagrama que indique la planificación en la ejecución de las tareas.
Precedencias: Paquete de trabajo 1.2 y 1.3

Tabla 4 - Planificación del proyecto

1. Gestión **Paquete de trabajo:** *1.5 Control de prototipos y planificación*
Duración: 6 horas
Descripción: Durante el desarrollo se realizarán controles tanto de la elaboración de los prototipos como de la planificación temporal. En dichos controles se analizará si se está cumpliendo la estimación de tiempos de la planificación y las especificaciones de cada prototipo.
Salidas/Entregables: Documentación que recoja los ajustes en la planificación en caso de ser necesarios, el número de horas de retraso, o en otro caso, el número de horas de adelanto.
Precedencias: Paquete de trabajo 1.3 y 1.4

Tabla 5 - Control de prototipos y planificación

1. Gestión **Paquete de trabajo:** *1.6 Selección del framework de desarrollo*
Duración: 3 horas
Descripción: Búsqueda de información sobre las diferentes herramientas, lenguajes y entornos disponibles para el desarrollo de la aplicación web.
Salidas/Entregables: Framework seleccionado teniendo en cuenta qué herramientas y lenguajes son adecuados para el desarrollo del proyecto y cuáles no.
Precedencias: Paquete de trabajo 1.1 y 1.2

Tabla 6 - Selección del framework de desarrollo

1. Gestión **Paquete de trabajo:** *1.7 Selección de herramientas*
Duración: 2 horas
Descripción: Búsqueda de información sobre qué herramientas son necesarias para complementar el desarrollo del proyecto, por ejemplo, la documentación o la realización de diagramas.
Salidas/Entregables: Conocer las herramientas que son adecuadas para el proyecto y que pueden ser útiles en la realización del mismo.
Precedencias: Paquete de trabajo 1.1 y 1.2

Tabla 7 - Selección de herramientas

Módulo 2: Aprendizaje

2. Aprendizaje **Paquete de trabajo:** *2.1 Instalación de sistemas de desarrollo*
Duración: 5 horas
Descripción: Instalación y configuración de todas las herramientas y entornos para la elaboración del proyecto. Esta tarea comprende la descarga de las diferentes herramientas, configuración de las mismas y realización de pruebas de funcionamiento.
Salidas/Entregables: Software instalado y configurado de la manera necesaria.
Precedencias: Paquete de trabajo 1.6 y 1.7

Tabla 8 - Instalación de sistemas de desarrollo

2. Aprendizaje **Paquete de trabajo:** *2.2 Aprendizaje sobre la arquitectura seleccionada*
Duración: 20 horas
Descripción: Búsqueda de información, aprendizaje de los conceptos necesarios de la arquitectura elegida y realización de pequeñas pruebas en el framework de desarrollo seleccionado.
Salidas/Entregables: Información recabada en forma de enlaces, documentos, tutoriales y ejemplos prácticos.
Precedencias: Paquete de trabajo 1.6 y 2.1

Tabla 9 - Aprendizaje sobre la arquitectura seleccionada

2. Aprendizaje **Paquete de trabajo:** *2.3 Aprendizaje sobre frameworks de desarrollo*
Duración: 40 horas
Descripción: Búsqueda de información y aprendizaje de los conceptos necesarios para la utilización de los frameworks de desarrollo seleccionados. Esta tarea incluye la lectura de documentación, tutoriales y realización de pequeñas pruebas.
Salidas/Entregables: Información recabada en forma de enlaces y tutoriales.
Precedencias: Paquete de trabajo 1.6 y 2.1

Tabla 10 - Aprendizaje sobre frameworks de desarrollo

2. Aprendizaje **Paquete de trabajo:** *2.4 Aprendizaje de nuevo lenguaje de programación*
Duración: 20 horas
Descripción: Aprendizaje de los nuevos lenguajes de programación, realización de pequeñas pruebas y visualización de tutoriales sobre los conceptos principales de los mismos.
Salidas/Entregables: Información recabada en forma de enlaces, documentos, tutoriales y ejemplos prácticos.
Precedencias: Paquete de trabajo 1.6 y 2.1

Tabla 11 - Aprendizaje de nuevo lenguaje de programación

Módulo 3: Análisis del proyecto

3. Análisis del proyecto **Paquete de trabajo:** *3.1 Captura de requisitos*
Duración: 5 horas
Descripción: Analizar y capturar los requisitos que debe cumplir cada prototipo para satisfacer las necesidades del cliente.
Salidas/Entregables: Documentación en la que consten los requisitos que se han de cumplir para lograr satisfacer las necesidades del cliente.
Precedencias: Paquete de trabajo 1.1, 1.2 y 1.3

Tabla 12 - Captura de requisitos

3. Análisis del proyecto **Paquete de trabajo:** *3.2 Definición de funcionalidades*
Duración: 5 horas
Descripción: Analizar y concretar de qué manera se van a desarrollar las funcionalidades de las que va a constar la página web para lograr cumplir con la captura de requisitos desarrollada.
Salidas/Entregables: Documentación en la que consten las funcionalidades que va a poseer cada prototipo de la aplicación.
Precedencias: Paquete de trabajo 1.1, 1.2 y 3.1

Tabla 13 - Definición de funcionalidades

3. Análisis del proyecto **Paquete de trabajo:** *3.3 Definición de casos de uso*
Duración: 6 horas
Descripción: Realizar el diseño de casos de uso para visualizar las acciones que pueden realizar los diferentes actores de la página web.
Salidas/Entregables: Diagrama de casos de uso.
Precedencias: DOP (Documento de Objetivos del Proyecto)

Tabla 14 - Definición de casos de uso

3. Análisis del proyecto **Paquete de trabajo:** *3.4 Realización de modelo de dominio*
Duración: 8 horas
Descripción: Realizar el modelo de dominio de la base de datos para gestionar los aspectos de la aplicación web que no estén ya reflejados en la base de datos de *Walkability Capturer*.
Salidas/Entregables: Diagrama de modelo de dominio.
Precedencias: DOP (Documento de Objetivos del Proyecto)

Tabla 15 - Realización de modelo de dominio

Módulo 4: Prototipos

4. Prototipos **Paquete de trabajo:** *4.1 Acceso de los usuarios*
Duración: 40 horas
Descripción: Funcionalidad correspondiente al acceso de los usuarios a la aplicación, es decir, el sistema de *login* de la página. Incluye *Análisis, Diseño, Implementación y Pruebas* de cada uno de los subprototipos: desarrollo de back-end y de front-end. Además, abarca la documentación del paquete de software: diagrama de clases y diagramas de secuencia.
Salidas/Entregables: Módulo de software desarrollado.
Precedencias: Grupo de tareas 1, 2 y 3.

Tabla 16 - Acceso de los usuarios

4. Prototipos **Paquete de trabajo:** *4.2 Aplicación de filtros y obtención de datos*
Duración: 100 horas
Descripción: Funcionalidad correspondiente a la aplicación de filtros al sistema de visualización de rutas y obtención de los datos correspondientes de la base de datos. Incluye *Análisis, Diseño, Implementación y pruebas* de cada uno de los subprototipos: desarrollo de back-end y de front-end. Además, abarca la documentación del paquete de software: diagrama de clases y diagramas de secuencia.
Salidas/Entregables: Módulo de software desarrollado.
Precedencias: Grupo de tareas 1, 2 y 3.

Tabla 17 - Aplicación de filtros y obtención de datos

4. Prototipos **Paquete de trabajo:** *4.3 Visualización de datos mediante mapa*
Duración: 75 horas
Descripción: Funcionalidad correspondiente al tratamiento de datos obtenidos mediante los filtros y su visualización de manera correcta en el mapa con las diferentes opciones que se le ofertan al usuario. Incluye *Análisis, Diseño, Implementación y pruebas* de cada uno de los subprototipos: desarrollo de back-end y de front-end. Además, abarca la documentación del paquete de software: diagrama de clases y diagramas de secuencia.
Salidas/Entregables: Módulo de software desarrollado.
Precedencias: Grupo de tareas 1, 2 y 3. Paquete de trabajo 4.2.

Tabla 18 - Visualización de datos mediante mapa

4. Prototipos **Paquete de trabajo:** *4.4 Alta de usuarios*
Duración: 20 horas
Descripción: Funcionalidad correspondiente al registro de usuarios en el sistema por parte de los usuarios acreditados para ello, eliminación de usuarios y concesión de los permisos necesarios. Incluye *Análisis, Diseño, Implementación y pruebas* de cada uno de los subprototipos: desarrollo de back-end y de front-end. Además, abarca la documentación del paquete de software: diagrama de clases y diagramas de secuencia.
Salidas/Entregables: Módulo de software desarrollado.
Precedencias: Grupo de tareas 1, 2 y 3.

Tabla 19 - Alta de usuarios

4. Prototipos **Paquete de trabajo:** *4.5 Edición de cuestionarios*
Duración: 30 horas
Descripción: Funcionalidad correspondiente a la modificación de los cuestionarios, preguntas y configuraciones de la aplicación *Walkability Capturer* por parte de los usuarios acreditados para ello. Incluye *Análisis, Diseño, Implementación y pruebas* de cada uno de los subprototipos: desarrollo de back-end y de front-end. Además, abarca la documentación del paquete de software: diagrama de clases y diagramas de secuencia.
Salidas/Entregables: Módulo de software desarrollado.
Precedencias: Grupo de tareas 1, 2 y 3.

Tabla 20 - Edición de cuestionarios

4. Prototipos **Paquete de trabajo:** *4.6 Descarga de datos*
Duración: 30 horas
Descripción: Funcionalidad correspondiente a la descarga de datos de la aplicación por parte de los usuarios tanto en formato compatible con herramientas QGIS como con sistemas SPSS. Incluye *Análisis, Diseño, Implementación y pruebas* de cada uno de los subprototipos: desarrollo de back-end y de front-end. Además, abarca la documentación del paquete de software: diagrama de clases y diagramas de secuencia.
Salidas/Entregables: Módulo de software desarrollado.
Precedencias: Grupo de tareas 1, 2 y 3. Paquete de trabajo 4.2.

Tabla 21 - Descarga de datos

Módulo 5: Documentación

5. Documentación **Paquete de trabajo:** *5.1 Elaboración del DOP*
Duración: 25 horas
Descripción: Realizar el documento de objetivos del proyecto que incluye los objetivos y requerimientos del mismo, planificación y definición de tareas, análisis de riesgos y valoración económica.
Salidas/Entregables: DOP (Documento de Objetivos del Proyecto)
Precedencias: Grupo de tareas 1 y 2.

Tabla 22 - Elaboración del DOP

5. Documentación **Paquete de trabajo:** *5.2 Elaboración de la memoria del TFG*
Duración: 50 horas
Descripción: Realizar la memoria del proyecto que incluye toda la información del mismo durante todo su desarrollo. Tareas, diseños y diagramas realizados, pruebas realizadas, dificultades, conclusiones, valoraciones y experiencia personal.
Salidas/Entregables: Memoria final del TFG.
Precedencias: Grupo de tareas 1, 2, 3 y 4. Paquete de trabajo 5.1.

Tabla 23 - Elaboración de la memoria del TFG

5. Documentación **Paquete de trabajo:** *5.3 Preparación de la defensa*
Duración: 15 horas
Descripción: Creación del material de apoyo para la defensa, es decir, la presentación. Esta tarea incluye la preparación de la defensa del proyecto así como posibles ensayos de la misma.
Salidas/Entregables: Defensa del proyecto.
Precedencias: Grupo de tareas 1, 2, 3 y 4. Paquete de trabajo 5.1 y 5.2.

Tabla 24 - Preparación de la defensa

2.4. Planificación temporal

Una vez se conocen las tareas que se han de desarrollar para completar el proyecto, es necesario calcular en que momento y durante cuánto tiempo se va a desempeñar cada una, es decir, distribuirlas a lo largo del tiempo. De esta manera se puede realizar un control más exhaustivo durante el desarrollo y conocer el momento estimado en el que el proyecto estará finalizado. La planificación temporal, así como la representación del tiempo previsto para el desempeño de cada tarea, se puede visualizar mediante un diagrama Gantt en la *Ilustración 2*. En este caso se trata de un diagrama Gantt semanal.

Para realizar el proyecto se ha estimado una carga mínima diaria de 2 horas para los días laborables, y, por otro lado, para días no laborables se estima una carga de 5 horas diarias. Como resultado final se prevé una carga semanal de 20 horas. Se ha de tener en cuenta que al termino de ciertas tareas, en ocasiones, y siempre que la finalización de la tarea coincida con el final de la semana laboral, se estimará necesario algunas jornadas de descanso antes de comenzar el siguiente paquete de trabajo; por ello, la siguiente tarea se comenzará al inicio de la siguiente jornada laboral. Cabe destacar que el tiempo estimado para el desempeño de cada prototipo está dividido en dos fragmentos, uno para el desarrollo del back-end y otro para el desarrollo del front-end.

El diagrama Gantt se ha dividido según las tareas o paquetes de trabajo generados en la Estructura de Descomposición del Trabajo (EDT): Gestión, Aprendizaje, Análisis del proyecto, Prototipos y Documentación. Así, a la izquierda del diagrama se puede visualizar la lista con todas las tareas que componen estos módulos del Trabajo Fin de Grado. En la parte superior del diagrama, en cambio, se visualiza el periodo temporal donde se encuentra la acción que se estima realizar. Cada tarea está ubicada en el diagrama Gantt en su periodo de realización correspondiente y con el color que se le asignó a dicha tarea en el diagrama EDT.

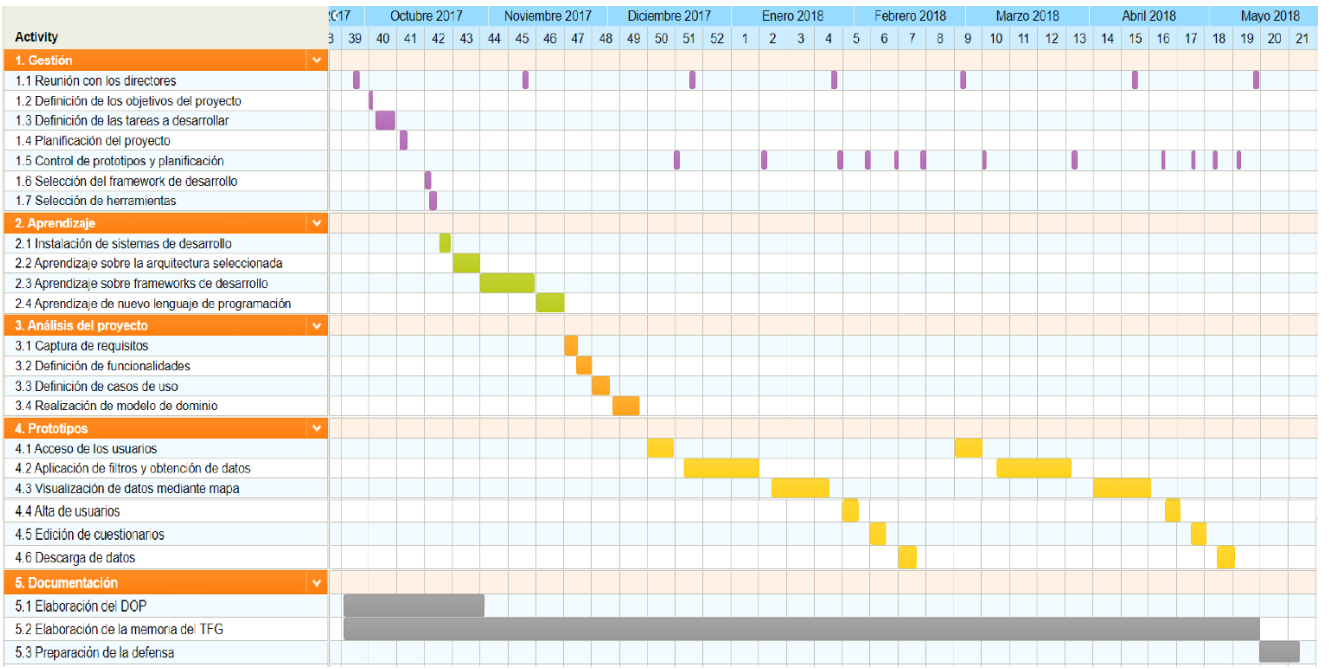


Ilustración 2 - Diagrama Gantt

Ciertos paquetes de trabajo, a pesar de que tienen una planificación de horas estimada, se realizarán durante todo el proyecto. En el caso de la tarea de "Control de prototipos y planificación", siempre que se termine de desarrollar un prototipo, se dedicarán una serie de horas a comprobar si el prototipo se ha desarrollado de acuerdo a los plazos establecidos, y se comprobará el número de horas dedicadas al mismo. Así, se podrán detectar fallos e incumplimientos de la planificación inicial. Por otro lado, las documentaciones se realizarán de forma paralela al proyecto, por ejemplo, dedicando ciertas horas diarias tanto a la implementación como a la documentación de dicho desarrollo.

A continuación, en la *Tabla 25* se especifica el día de inicio y finalización de cada tarea en particular:

Tareas	***Duración***	***Periodo de realización***
1. Gestión	***32 horas***	***28 de septiembre de 2017 - 13 de mayo de 2018***
1.1 Reunión con los directores	6 horas	*28 de septiembre de 2017 – 13 de mayo de 2018 (Tarea fragmentada)*
1.2 Definición de los objetivos del proyecto	1 hora	*2 de octubre de 2017*
1.3 Definición de las tareas a desarrollar	10 horas	*3 – 8 de octubre de 2017*
1.4 Planificación del proyecto	4 horas	*9 – 11 de octubre de 2017*
1.5 Control de prototipos y planificación	6 horas	*18 de diciembre de 2017 – 8 de mayo de 2018 (Tarea fragmentada)*
1.6 Selección del framework de desarrollo	3 horas	*16 de octubre de 2017*
1.7 Selección de herramientas	2 horas	*17 – 18 de octubre de 2017*

Tareas	*Duración*	*Periodo de realización*	
2. Aprendizaje	***85 horas***	***18 de octubre de 2017 – 19 de noviembre de 2017***	
2.1 Instalación de sistemas de desarrollo	5 horas	18 – 19 de octubre de 2017	
2.2 Aprendizaje sobre la arquitectura seleccionada	20 horas	23 – 29 de octubre de 2017	
2.3 Aprendizaje sobre frameworks de desarrollo	40 horas	30 de octubre de 2017 – 12 de noviembre de 2017	
2.4 Aprendizaje de nuevo lenguaje de programación	20 horas	13 – 19 de noviembre de 2017	
3. Análisis del proyecto	***24 horas***	***20 de noviembre de 2017 – 08 de diciembre de 2017***	
3.1 Captura de requisitos	5 horas	20 – 23 de noviembre de 2017	
3.2 Definición de funcionalidades	5 horas	23 – 26 de noviembre de 2017	
3.3 Definición de casos de uso	6 horas	27 de noviembre de 2017 – 1 de diciembre de 2017	
3.4 Realización de modelo de dominio	8 horas	2 – 8 de diciembre de 2017	
4. Prototipos	***295 horas***	***11 de diciembre de 2017 – 07 de mayo de 2018***	
4.1 Acceso de los usuarios	40 horas	Front-end	11 – 17 de diciembre de 2017
		Back-end	26 de febrero de 2018 – 4 de marzo de 2018
4.2 Aplicación de filtros y obtención de datos	100 horas	Front-end	20 de diciembre de 2017 – 7 de enero de 2018
		Back-end	8 – 27 de marzo de 2018
4.3 Visualización de datos mediante mapa	75 horas	Front-end	11 - 25 de enero de 2018
		Back-end	2 – 16 de abril de 2018
4.4 Alta de usuarios	20 horas	Front-end	29 de enero de 2018 - 1 de febrero de 2018
		Back-end	20 – 23 de abril de 2018
4.5 Edición de cuestionarios	30 horas	Front-end	5 - 8 de febrero de 2018
		Back-end	26 – 30 de abril de 2018
4.6 Descarga de datos	30 horas	Front-end	12 - 16 de febrero de 2018
		Back-end	3 – 7 de mayo de 2018
5. Documentación	***90 horas***	***28 de septiembre de 2017 – 23 de mayo de 2018***	
5.1 Elaboración del DOP	25 horas	28 de septiembre de 2017 – 2 de noviembre de 2017	
5.2 Elaboración de la memoria del TFG	50 horas	28 de septiembre de 2017 – 13 de mayo de 2018	
5.3 Preparación de la defensa	15 horas	14 – 23 de mayo de 2018	
Total	***526 horas***	***28 de septiembre de 2017 - 23 de mayo de 2018***	

Tabla 25 - Duración de las tareas del proyecto

2.5. Arquitectura

A la hora de elegir la arquitectura a seguir por Walkability Analyzer, una elección que marcará en gran medida el proyecto, se han analizado varios estilos de desarrollo o programación, en concreto, el modelo REST [1] y el estilo SOAP [2].

SOAP (Simple Object Access Protocol)

SOAP (Simple Object Access Protocol), es uno de los protocolos más utilizados para la elaboración de páginas web, y uno de los estándares que más capacidades y funcionalidades ofrece en la creación de servicios web. Los servicios SOAP, servicios que basan su comunicación en el protocolo SOAP, realizan las comunicaciones mediante XML, una de las diferencias principales frente al modelo REST. En la *Ilustración 3* se puede visualizar un ejemplo de comunicación mediante el protocolo SOAP. (Blancarte, 2017)

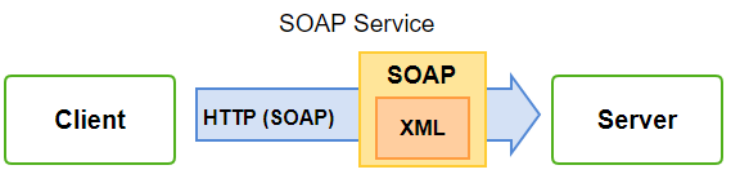


Ilustración 3 - Comunicación mediante el protocolo SOAP

SOAP es un protocolo de comunicación muy potente que puede ser utilizado para multitud de tareas, tanto tareas de una alta dificultad como tareas más sencillas. Esta misma característica, a pesar de ser su punto fuerte, también hace que se convierta en un protocolo más pesado tanto en tamaño como en procesamiento. Además, al tratarse de un protocolo con tantas capacidades, su aprendizaje se vuelve más complicado, haciendo que el tiempo de dedicación al aprendizaje del sistema de comunicación se pueda incrementar de manera considerable.

Representational State Transfer (REST)

El modelo de desarrollo REST (Representational State Transfer), o Transferencia de Estado Representacional, se vale del estándar HTTP para obtener los datos necesarios, tratar dichos datos y generar operaciones sobre estos mismos en diversos formatos, como, por ejemplo, XML o JSON. A pesar de que este modelo de arquitectura soporta la transferencia de datos en diferentes formatos, la mayoría de comunicaciones se realizan en formato JSON. Esto se debe a que este formato es interpretado de una manera natural por JavaScript, lo que facilita la utilización y aprovechamiento de frameworks conocidos como Angular o NodeJS (ver apartado 2.6).

Una de las características principales de REST es que sigue un protocolo cliente-servidor sin estado. Cada petición HTTP que se realiza contiene toda la información necesaria para ser ejecutada sin problema alguno, por ello, no se requiere que ni el cliente ni el servidor almacenen un estado previo para satisfacer la petición. La separación entre el cliente y el servidor posee multitud de ventajas, entre ellas, facilita la portabilidad del desarrollo a otro tipo de plataformas y aumenta la escalabilidad de los proyectos. En la *Ilustración 4* se puede ver una arquitectura basada en el modelo REST.

[1] *REST: Representational State Transfer*
[2] *SOAP: Simple Object Access Protocol*

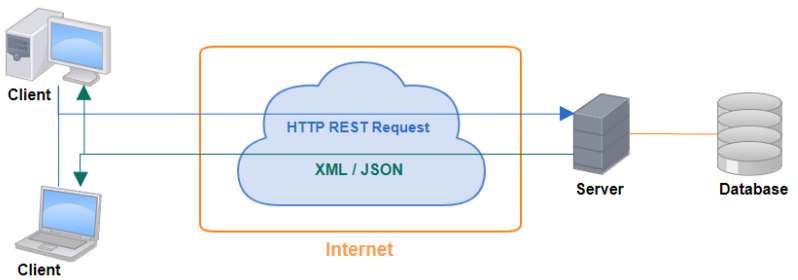


Ilustración 4 - Arquitectura basada en el modelo REST

Analizando tanto el modelo REST como el estilo SOAP, se ha llegado a la conclusión de que este último posee un mayor grado de complejidad para su desarrollo, por lo que aumentaría en gran medida el tiempo de creación de Walkability Analyzer. Por ello, para el desarrollo de este Trabajo Fin de Grado se utilizará una API REST, un estilo muy extendido para el desarrollo actual de páginas web.

Por otro lado, y al seleccionar REST como arquitectura a seguir, se han analizado que opciones de desarrollo existían tanto para la parte de back-end y front-end como para realizar la propia API REST. En este caso, para realizar la selección entre las alternativas existentes en el mercado se ha valorado lo que está en auge actualmente y las tecnologías más utilizadas por los desarrolladores web. Entre estas opciones destacan Angular como framework de desarrollo para la parte de front-end de la aplicación web, NodeJS como framework para la parte de back-end, y Express como ayuda para la realización de una API REST que comunique las dos partes, la parte cliente y la parte servidor. En el próximo apartado se proporcionan más detalles sobre las herramientas elegidas para el desarrollo de Walkability Analyzer, entre ellas Angular, NodeJS y Express.

2.6. Herramientas

En esta sección se detallan las herramientas que se utilizarán durante el transcurso y el desarrollo del proyecto, así como el uso que se le dará a cada una de ellas:

- **Tom's Planner:** Tom's Planner es una herramienta online que permite la creación de diagramas Gantt de manera sencilla. En concreto, este programa se utilizará para realizar los diagramas necesarios para la planificación y la gestión del proyecto.

- **Visual Studio Code:** Esta herramienta es un editor de código fuente que facilita en gran medida el desarrollo de páginas web ya que, entre otras cosas, incorpora potentes características de edición y un sistema de depuración de código. Mediante Visual Studio Code se desarrollará tanto la lógica de la aplicación como la parte de interfaz gráfica.

- **Oracle VM VirtualBox:** Oracle VM VirtualBox es un sistema de virtualización que permite instalar sistemas operativos adicionales o "sistemas invitados" dentro de otro sistema operativo, manteniendo cada uno con su propio ambiente y estructura. VirtualBox se utilizará para crear una máquina virtual que actúe como servidor dentro del propio equipo portátil.

- **Cacoo:** Esta herramienta en línea permite la creación de gráficos, mapas, ilustraciones y diagramas. Cacoo se utilizará para realizar algunos de los diagramas e ilustraciones para la elaboración de la documentación del proyecto.

- **Dropbox:** Dropbox es un servicio de almacenamiento de archivos y documentos en la nube. Permite a los usuarios almacenar, sincronizar, compartir archivos y carpetas entre diferentes dispositivos y con distintas personas. En este proyecto Dropbox se utilizará para realizar copias de seguridad tanto de la documentación del proyecto como de los archivos que incluyen la lógica de la aplicación.

- **Microsoft Office Word:** Microsoft Office Word es un programa de procesamiento de textos que permite crear y personalizar gran cantidad de documentos. Esta herramienta se utilizará para realizar la documentación del proyecto.

- **Visual Paradigm for UML:** Visual Paradigm es una herramienta que permite realizar diagramas UML; por ejemplo: diagramas de secuencia o diagramas de modelo de dominio. En concreto, este programa se utilizará para realizar los diagramas relacionados con la implementación de código durante el desarrollo.

- **Angular**: Angular es un framework de JavaScript de código abierto que se utiliza para crear y mantener aplicaciones web mediante una serie de patrones de diseño. En concreto está formado por un conjunto de librerías útiles para el desarrollo de aplicaciones en el lado del cliente. En concreto, para Walkability Analyzer será utilizado para desarrollar la parte de front-end de la página web.

- **NodeJS**: Node.js es un entorno de ejecución multiplataforma de código abierto para desarrollar aplicaciones web en la parte del lado del servidor. Está basado en el motor V8 de Google, utiliza JavaScript como lenguaje de programación y es una arquitectura asíncrona orientada a eventos. NodeJS será la plataforma encargada del funcionamiento de la lógica de la aplicación web en la parte del servidor.

- **Express**: Express en un módulo escrito en JavaScript para NodeJS. Este framework ofrece soporte que cubre las necesidades básicas de una aplicación web; por ejemplo, gestión de peticiones HTTP, gestión rápida de respuestas y creación de APIs REST de una forma más sencilla. Express facilita la construcción de aplicaciones web.

- **JavaScript**: JavaScript es un lenguaje de programación que se utiliza, en la mayoría de los casos, para realizar páginas web dinámicas. Entre otras muchas cosas, este lenguaje es orientado a objetos y basado en prototipos. En la realización de este proyecto JavaScript se utilizará como lenguaje principal de programación.

- **PhpMyAdmin**: PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web. En este caso esta herramienta se utilizará para gestionar la base de datos de la aplicación y poder así trabajar con ella de manera gráfica.

- **HTML**: HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de las siglas que corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto. En concreto, HTML se utiliza para realizar la parte grafica de la aplicación y es con este lenguaje con el que se va a realizar la interfaz de Walkability Analyzer.

- **CSS**: CSS es el acrónimo de Cascading Style Sheets, es decir, hojas de estilo en cascada. CSS es un lenguaje de estilo que define la presentación de los documentos HTML. En este proyecto se utilizará para aplicar estilos de diseño a la interfaz gráfica de la página web.

- **TypeScript**: TypeScript es un lenguaje de programación de código abierto que posee herramientas de programación orientada a objetos. Este lenguaje es utilizado principalmente en el proceso de desarrollo de páginas web, ya que los navegadores son capaces de interpretarlo como si de JavaScript se tratará. Entre otras cosas, esto es así porque TypeScript es muy similar a JavaScript.

2.7. Gestión de riesgos

Durante la realización de un proyecto surgen imprevistos de menor o mayor magnitud, por ello, generar un plan de gestión de riesgos adquiere una gran importancia. Realizando un análisis de riesgos se pueden identificar y prevenir los factores que pueden afectar de manera directa o indirecta al trabajo. Además, se podrá actuar de manera inmediata y mitigar así los efectos en caso de que dichos riesgos se lleguen a materializar. La probabilidad de que una amenaza se llegue a materializar es relativa y difícil de medir mediante porcentajes, por ello, se utilizará una tabla orientativa (*Tabla 26*).

Probabilidad	Porcentaje
Improbable	0 – 30 %
Probable	30 – 70 %
Muy probable	70 – 100 %

Tabla 26 - Probabilidad de los riesgos

Además de analizar la probabilidad de que una amenaza se llegue o no a materializar, para realizar una buena gestión de riesgos se ha de valorar la importancia o el impacto que tendría la ejecución de dicho riesgo. En función de la amenaza que se produzca, puede tener un impacto más o menos grave, retrasando así en mayor o menor medida el proyecto. Por ello, para estimar el retraso del trabajo en función de la amenaza, se utilizará la tabla orientativa correspondiente (*Tabla 27*).

Impacto	Retraso
Muy leve	Menos de un día
Leve	1 – 2 días
Medio	3 – 5 días
Alto	5 – 7 días
Muy alto	Más de 7 días

Tabla 27 - Impacto de los riesgos

Para cada amenaza no solo se han de definir la probabilidad y el impacto, sino que además se deben seguir tanto métodos de prevención, que se realizan para evitar dicho riesgo, como planes de contingencia, para que en caso de que el riesgo finalmente se llegue a producir, las consecuencias afecten en la menor medida posible al transcurso del proyecto. Además, en función de la probabilidad y el impacto de cada amenaza se deberán realizar revisiones periódicas del sistema de gestión de riesgos para comprobar que sus directrices se están cumpliendo correctamente.

Para analizar los riesgos de manera individual se han dividido en tres secciones: “Riesgos de diseño y planificación”, es decir, las contingencias que afectan a la planificación o al diseño del proyecto (*Tabla 28*); “Riesgos de índole personal del desarrollador”, como,

por ejemplo, enfermedad que impida la realización del trabajo o falta de tiempo para continuar con él proyecto (*Tabla 29*); y por último, "Riesgos de hardware y software" que puedan afectar al desarrollo de la aplicación web (*Tabla 30*).

Riesgos de diseño y planificación	
Planificación temporal incorrecta	
Descripción	El proyecto puede verse afectado por una planificación temporal incorrecta de manera que se demore en el tiempo más de lo previsto.
Probabilidad	Muy probable
Impacto	Medio
Prevención	Intentar prever de manera correcta los tiempos para realizar una planificación temporal precisa.
Plan de contingencia	Ajustar la planificación temporal para recuperar el tiempo de retraso en el proyecto.
Revisión del riesgo	Semanal
Variaciones en los requisitos o especificaciones del proyecto	
Descripción	El hecho de variar los requisitos y especificaciones puede hacer que haya partes del proyecto que haya que realizar de nuevo e incluso que haya que desechar. Esto puede ocasionar una demora en la planificación temporal.
Probabilidad	Improbable
Impacto	Muy alto
Prevención	Realizar un proyecto modular de manera que, en caso de tener que realizar cambios, se pueda reutilizar la mayor parte del trabajo.
Plan de contingencia	Plantear de nuevo las partes del proyecto afectadas.
Revisión del riesgo	Mensual
Problemas de análisis y diseño	
Descripción	Los problemas de análisis y diseño, como diagramas inconsistentes o erróneos pueden ocasionar retrasos en el proyecto y dificultades en el desarrollo.
Probabilidad	Probable
Impacto	Alto
Prevención	Dedicar el tiempo necesario para realizar el análisis y diseño.
Plan de contingencia	Plantear un nuevo diseño que solucione el problema existente y que sea viable para aprovechar al máximo las partes del proyecto realizadas.
Revisión del riesgo	Quincenal

Tabla 28 - Riesgos de diseño y planificación

Riesgos de índole personal del desarrollador	
Enfermedad	
Descripción	Una posible enfermedad durante el desarrollo puede ocasionar que el trabajo no se llegue a realizar en las fechas estimadas.
Probabilidad	Muy probable
Impacto	Leve
Prevención	Tomar precauciones para evitar exponerse a enfermedades.
Plan de contingencia	Tomar medidas para recuperarse en el menor tiempo posible y reorganizar la planificación temporal.
Revisión del riesgo	Mensual
Exámenes - Prácticas	
Descripción	El tiempo dedicado a otras actividades o responsabilidades, como el estudio para exámenes o la realización de prácticas puede retrasar el desarrollo del proyecto.
Probabilidad	Probable
Impacto	Medio
Prevención	Realizar una planificación temporal teniendo en cuenta periodos en los que el tiempo de dedicación al proyecto sea menor.
Plan de contingencia	Reorganizar la planificación temporal para recuperar el tiempo perdido.
Revisión del riesgo	Mensual

Tabla 29 - Riesgos de índole personal del desarrollador

Riesgos de Hardware y Software	
Perdida de datos	
Descripción	Perdida de datos, por ejemplo, al eliminar archivos importantes o borrar parte del código ya implementado.
Probabilidad	Probable
Impacto	Alto
Prevención	Realizar una copia de seguridad semanal del proyecto tanto en un dispositivo externo como en la plataforma Dropbox. De esta manera, en caso de pérdida, los daños producidos por la pérdida son menores.
Plan de contingencia	Restaurar la última copia de seguridad del proyecto y recuperar el trabajo perdido.
Revisión del riesgo	Semanal

Rotura del equipo de trabajo	
Descripción	Una rotura del equipo de trabajo puede ocasionar perdida de datos y paralización del desarrollo del proyecto hasta contar con un equipo de sustitución.
Probabilidad	Improbable
Impacto	Alto
Prevención	Realizar una copia de seguridad semanal del proyecto tanto en un dispositivo externo como en la plataforma Dropbox. No utilizar el equipo en situaciones o lugares potencialmente peligrosos.
Plan de contingencia	Restaurar la última copia de seguridad del proyecto en un segundo ordenador hasta que se subsane el problema y recuperar el trabajo perdido.
Revisión del riesgo	Quincenal
Infección de un virus	
Descripción	Una infección por virus en el equipo de trabajo puede dañar los archivos y la implementación desarrollada.
Probabilidad	Improbable
Impacto	Leve
Prevención	Realizar una copia de seguridad semanal del proyecto tanto en un dispositivo externo como en la plataforma Dropbox. Realizar análisis del sistema de manera periódica. Mantener tanto los antivirus como las aplicaciones del sistema actualizadas.
Plan de contingencia	Realizar un análisis completo del equipo, restaurar la última copia de seguridad del proyecto y recuperar el trabajo perdido.
Revisión del riesgo	Mensual
Problemas de instalación de software	
Descripción	Problemas de instalación de software, como problemas de instalación de herramientas o incompatibilidades.
Probabilidad	Probable
Impacto	Medio
Prevención	Documentarse correctamente sobre las especificaciones de los programas a instalar y los procesos de instalación correspondientes.
Plan de contingencia	Revisar el proceso de instalación para hallar el error y, en caso de no tener solución, buscar un programa que realice una función similar.
Revisión del riesgo	Quincenal

Cambios/Actualizaciones en las librerías utilizadas	
Descripción	Los cambios o actualizaciones de las librerías utilizadas pueden ocasionar cambios en el código.
Probabilidad	Improbable
Impacto	Medio
Prevención	Mantenerse informado sobre posibles actualizaciones y cambios que se produzcan en las librerías.
Plan de contingencia	En caso de actualizaciones revisar en qué consisten para adaptar el proyecto a los nuevos cambios introducidos.
Revisión del riesgo	Quincenal

Tabla 30 - Riesgos de Hardware y Software

2.8. Evaluación económica

Todo proyecto tiene un coste asociado. Por ello, una vez realizada la planificación temporal, y conociendo el tiempo estimado de duración del proyecto, se presenta la evaluación económica del mismo. El coste total es la suma de diferentes costes y gastos asociados a la realización del proyecto, por ejemplo, sueldo del programador, licencias de software o utilización del hardware necesario.

Según el Boletín Oficial del Estado (BOLETÍN OFICIAL DEL ESTADO, 2017), el suelo anual de un programador asciende a 16917,60€. En la planificación temporal se estima que el proyecto tendrá una duración de 9 meses. Debido a ello, teniendo en cuenta el suelo anual y la duración del proyecto, el coste de trabajo del mismo en cuanto a desarrollo se estima en 12688,2 (*Tabla 31*).

Descripción	Meses	Sueldo anual	Coste
Coste de desarrollo del proyecto: documentación, análisis, diseño, implementación y pruebas.	9	16917,60€	12688,2 €

Tabla 31 - Coste de desarrollo del proyecto

Por otro lado, algunos programas específicos tienen asociados costes en concepto de licencias de software. Para calcular dichos costes se debe tener en cuenta la vida útil del software a utilizar, por ello, se estima que un programa queda amortizado en una media de unos 3 años (*Tabla 32*).

Descripción	Coste	Amortización	Coste en 9 meses
Office Hogar y Estudiantes 2016 para PC	149,00 €	3 años	37,25 €
Visual Paradigm for UML Standard Edition	349 $ = 296 €		74 €
Coste total en software			111,25 €

Tabla 32 - Coste de software del proyecto

Para desarrollar Walkability Analyzer se empleará un ordenador portátil valorado en 699 euros. Dicho ordenador portátil incluye la licencia del sistema operativo de manera gratuita, por lo que esto no supondrá ningún coste añadido al proyecto. La vida útil asociada a un equipo portátil es de 5 años, por lo que el coste correspondiente se puede visualizar en la *Tabla 33*.

Descripción	Coste	Amortización	Coste en 9 meses
Asus A55V Intel Core i5 6 GB RAM	699 €	5 años	104,85 €

Tabla 33 - Coste del hardware del proyecto

Además de los gastos de desarrollo, software y hardware, se deben calcular los costes indirectos generados durante la elaboración del proyecto. Estos costes indirectos, por ejemplo, agua y electricidad durante los 9 meses de desarrollo de Walkability Analyzer, se presuponen un 1 % del coste total del proyecto. Así, el coste total del proyecto asciende a 16145,26 euros (*Tabla 34*).

Descripción	Coste
Licencias de software	111,25 €
Equipo informático	104,85 €
Desarrollo del proyecto	12688,2 €
Gastos indirectos	149,16 €
Coste total	13053,46 €

Tabla 34 - Coste total del proyecto

Como ya se ha mencionado anteriormente, la realización de este proyecto no está enfocada a la obtención de beneficios económicos. Walkability Analyzer pretende ayudar en la investigación de los psicólogos, profundizando en el conocimiento de la movilidad a pie en zonas urbanas, analizando la toma de decisiones de los ciudadanos a la hora de transitar a pie por la ciudad, y ayudando a construir entornos pensados para el peatón. De esta manera, mediante el desarrollo de este proyecto, se pretende realizar un bien social y ayudar a construir ciudades más seguras.

3. Antecedentes

Cuando se va a realizar un proyecto de esta magnitud, en el que el objetivo principal reside en desarrollar software, conviene investigar qué productos existen en el mercado con características y funciones similares al proyecto qué se va a realizar. En este caso, para Walkability Analyzer no existen antecedentes de aplicaciones web similares. Esto se debe, principalmente, a que el proyecto se basa en las necesidades específicas del grupo CRIM-AP de la UPV/EHU, por ello, se requiere la realización de un software con funciones que no existen en el mercado actualmente.

3.1. Situación actual

Walkability Analyzer forma parte de un sistema más completo, junto con la aplicación Walkability Capturer, encargada de recoger los datos con los que la página web pretende trabajar. Walkability Capturer, desarrollada como Trabajo Fin de Grado por David Puerto Caldero, es una aplicación móvil que monitoriza los movimientos del usuario de manera automática y permite la valoración de las rutas realizadas.

El funcionamiento de la aplicación es simple para el usuario. Una vez que Walkability Capturer es descargada e instalada se presenta un breve cuestionario que se ha de realizar. En concreto, se le pregunta al usuario su género, edad, provincia y localidad de procedencia, horario deseado para realizar las encuestas de la aplicación y si posee, o no, algún problema de movilidad que pueda afectar al desplazamiento. Así, la aplicación ya tiene todos los datos necesarios para crear el perfil del usuario, poder empezar a capturar los movimientos, siempre que el geolocalizador esté activado, y almacenarlos correctamente.

Walkability Capturer analiza de manera interna la ruta realizada para comprobar si cumple, o no, los requisitos establecidos por el grupo CRIM-AP. Por ejemplo, el usuario puede realizar una ruta en transporte público, pero dicha ruta, al no realizarse a pie, no se ajusta a lo que los investigadores definen como ruta, por lo que no se guardarán sus datos en el sistema. Por ello, para que una ruta realizada por el usuario sea considerada válida para su posterior estudio, se consideran aspectos como la velocidad de desplazamiento o el tiempo de duración de dicha ruta.

Cuando se ha realizado una ruta, a la hora seleccionada por el usuario, se le realizan las preguntas correspondientes. Estas preguntas, que se podrán modificar desde la página web de Walkability Analyzer, examinan el motivo por el que se ha realizado la ruta, la seguridad, el tráfico encontrado en esa zona e incluso si la ruta es o no visualmente atractiva para el peatón. De esta manera, los datos de la ruta realizada, así como las opiniones de la misma, quedan asociados al perfil del usuario, y posteriormente son almacenados en el sistema.

Es a partir de esos datos recogidos mediante la aplicación Walkability Capturer de donde parte este Trabajo Fin de Grado, es decir, es el punto de salida desde donde se desarrollará la página web de Walkability Analyzer. En la *Ilustración 5* se puede ver un diagrama explicativo del sistema formado por Walkability Capturer y Walkability Analyzer.

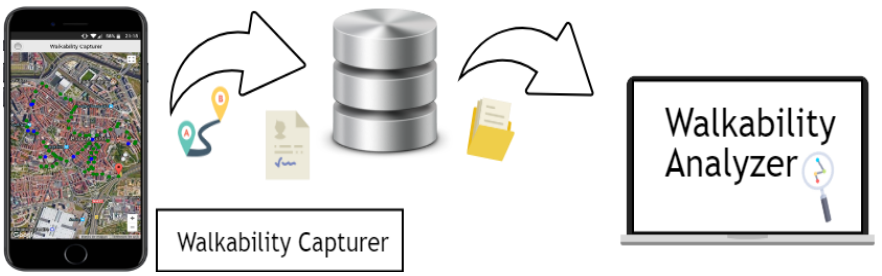


Ilustración 5 - Sistema formado por Walkability Capturer y Walkability Analyzer

3.2. Estudio de alternativas existentes

En la última década el desarrollo de software ha sufrido grandes avances, y con ello han evolucionado otros aspectos importantes en cuanto a la programación, por ejemplo, el número de módulos o APIs que están al alcance del programador. La evolución de estas mismas ha sido considerable, y por ello, han aparecido en el mercado diferentes alternativas que considerar a la hora de desarrollar software. En este caso, en Walkability Analyzer, al tener que mostrar las rutas sobre un mapa que permita interactuar con él, es necesario el uso de una API que posea dichas opciones. En este caso, a la hora de seleccionar qué alternativa escoger se valorarán dos opciones: Google Maps API y Open Street Maps.

Google Maps API

La API de Google Maps, propiedad de Google, consta de multitud de opciones al alcance del desarrollador. Entre dichas opciones destacan la multitud de direcciones que posee, la personalización en el diseño de los mapas o la posibilidad de interactuar con ellos definiendo rutas según ciertos puntos establecidos. En este caso, para el desarrollo de Walkability Analyzer, lo más destacable de la API de Google Maps es la posibilidad de visualizar en el mapa rutas creadas mediante una serie de puntos. Esta funcionalidad permitiría poder mostrar gran cantidad de rutas con diferentes aspectos, ya que el grosor o el color de las rutas las define el propio desarrollador. Además, su funcionamiento es sencillo, por lo que el tiempo de aprendizaje disminuye. (Google Maps API, s.f.)

Por otro lado, Google Maps API trabaja con otras funciones no incluidas en su propia librería, pero desarrolladas por el mismo fabricante, que aumenta aún más si cabe su funcionalidad. Este es el caso de Google Maps Geocoding API, que permite introducir una dirección y es la propia API la que devuelve sus coordenadas. De esta manera se podría centrar el mapa en el lugar que se desee únicamente introduciendo una ciudad o población, una funcionalidad que resulta muy útil para los investigadores ya que permite desplazarse por el mapa con una mayor rapidez.

Google Maps API no es completamente gratuita, ya que, en función de unos criterios establecidos, al superar un número de peticiones al servicio en un determinado margen de tiempo la utilización de sus funciones comenzará a tener coste. A pesar de ello, analizando el volumen de peticiones que necesitaría Walkability Analyzer, y teniendo en cuenta que los únicos usuarios en utilizar la aplicación serían los miembros del grupo CRIM-AP, se ha estimado que no sería necesario desembolsar importe alguno.

Open Street Maps

Open Street Maps (OSM) es un proyecto de código abierto mantenido por la Comunidad OpenStreetMap que proporciona mapas editables gratuitos de todo el mundo. Al ser una herramienta libre, esta cuenta con la colaboración de todos los usuarios que la utilizan, por ello, todos los tipos de mapas que posee son actualizados constantemente gracias a los datos que aportan los miembros de la comunidad.

A pesar de ser de código abierto, el problema principal reside en que utilizando únicamente esta API no se obtendrían todas las funciones necesarias para el desarrollo de Walkability Analyzer. Esta API cuenta con algunas de las funciones necesarias, como, por ejemplo, visualización del mapa e interacción con él por parte de los investigadores. En cambio, algunas funciones, como el servicio de dirección inversa, no están incluidas en la propia librería, sino que es necesario cumplimentar Open Street Maps con otras APIs de terceros para conseguir todas las funcionalidades necesarias. Esto aumenta el número de APIs a utilizar y dificulta el aprendizaje ya que, al ser módulos diferentes de varios fabricantes, la forma de interactuar con los servicios es distinta. En la *Ilustración 6* se puede visualizar un mapa generado con Open Street Maps API.

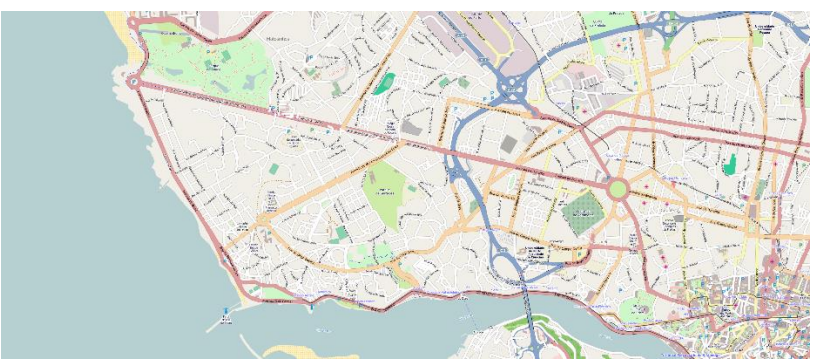


Ilustración 6 - Mapa de Open Street Maps API

El mayor punto fuerte que presenta Open Street Maps frente a Google Maps reside en que la primera es gratuita y de código abierto, mientras que la segunda únicamente permite utilizar sus servicios y realizar las peticiones necesarias a su API, sin llegar a ver el código interno del módulo a utilizar.

Una vez analizadas las alternativas existentes para la utilización de una API para la visualización de mapas, se ha determinado que la mejor opción es utilizar Google Maps API. Esta API no sólo es más sencilla en su utilización, sino que además en Internet están disponibles multitud de tutoriales y ejemplos que hacen que el aprendizaje sea más ágil y que, en caso de encontrar algún problema, la solución se encuentre de una manera más rápida.

4. Captura de requisitos

En este capítulo se va a presentar la captura de requisitos de Walkability Analyzer, y se definirá, entre otras cosas, qué tiene que cumplir el trabajo que se va a desarrollar, qué necesidades tiene que satisfacer o quiénes van a ser los usuarios finales. Realizar una correcta captura de requisitos es el primer paso para realizar una buena aplicación, por ello, este paso es un paso fundamental en el proceso de desarrollo.

4.1. Requisitos previos

En este apartado se expondrán los requisitos previos que debe poseer la aplicación web. Estos requisitos definen funciones del sistema o componentes que este debe de tener. Por lo tanto, analizando este aspecto del proyecto se ha de tener en cuenta lo siguiente:

- Walkability Analyzer será una aplicación creada para cubrir las necesidades de los miembros del grupo CRIM-AP de la UPV/EHU.

- Walkability Analyzer únicamente estará disponible para aquellos usuarios registrados en el sistema, por lo que para utilizar la aplicación web habrá que realizar un proceso previo de autentificación.

- El sistema final debe estar preparado para contener dos perfiles de usuarios diferenciados y con permisos distintos. Esto se debe a que no todos los investigadores del grupo CRIM-AP podrán realizar todas las acciones que poseerá la aplicación web.

- Walkability Analyzer debe permitir visualizar, aplicar filtros y descargar las rutas que los usuarios registran mediante la aplicación móvil ya existente para ello. Por otro lado, el sistema debe contar con una parte que permita gestionar los datos configurables de Walkability Capturer, el proyecto que precede a Walkability Analyzer, como preguntas, cuestionarios, configuraciones del sistema o categorías.

- Todos los usuarios identificados en el sistema podrán aplicar filtros sobre las rutas y datos existentes en la base de datos, podrán visualizar dicha información en la aplicación web y descargar el material necesario en el formato deseado. Además, todo usuario identificado podrá visualizar la información configurable perteneciente al proyecto Walkability Capturer. Esta información configurable, podrá ser visualizada, pero únicamente podrá ser creada, modificada o eliminada por los usuarios del sistema que tengan permisos para ello.

Finalmente, durante el transcurso del proyecto y debido a problemas encontrados en el funcionamiento de la aplicación móvil Walkability Capturer, los requisitos previos de la página web se vieron aumentados. A continuación, se detallan las especificaciones adicionales a tener en cuenta:

- Walkability Capturer, en ocasiones, obtiene de manera errónea el municipio de inicio y de fin de la ruta realizada. La obtención de los municipios es realizada por un servicio externo a la aplicación, en concreto, por un servicio de geolocalización de Google Maps, y este no siempre funciona con el nivel de precisión necesario. Por ejemplo, en ocasiones detecta como municipio "*Bilbao*", mientras que otras veces, almacena como municipio "*Blas de Otero, 4*". Debido a ello, para los investigadores es necesario tener una pantalla donde poder visualizar las rutas junto con el valor de los municipios guardado, así, comparando la ruta en el mapa con los valores

almacenados, podrán modificarlo en caso de detectar algún error.

- Otro de los errores encontrados reside en que, debido a la imprecisión de la geolocalización mediante GPS y a los problemas ocasionados por falta de cobertura, algunas de las rutas capturadas no son correctas, hay grandes desviaciones que provocan que estas no deban ser tenidas en cuenta. En este caso los investigadores necesitan tener la opción de poder eliminar dichas rutas, o, al menos, poder eliminar su valoración para que esta no influya en su investigación posterior.

4.2. Casos de uso

Tras haber realizado la captura de requisitos funcionales de la página web que se va a desarrollar, posteriormente, se detectarán y analizarán los casos de uso que guardan una relación directa con estos.

4.2.1. Jerarquía de actores

La jerarquía de actores para este sistema consta de tres únicos actores o roles. En la *Ilustración 7* se puede visualizar dichos actores.

- **Usuario no identificado:** Representa a todo usuario que llega a la página web y no está identificado.
- **Investigador:** Es el actor que ejecuta aquellas acciones que están disponibles en el sistema para el rol de: "*Investigador*".
- **Administrador:** Es el actor que ejecuta aquellas acciones que están disponibles en el sistema para el rol de: "*Administrador*". Este actor, además, heredará las acciones del *"Investigador"*.

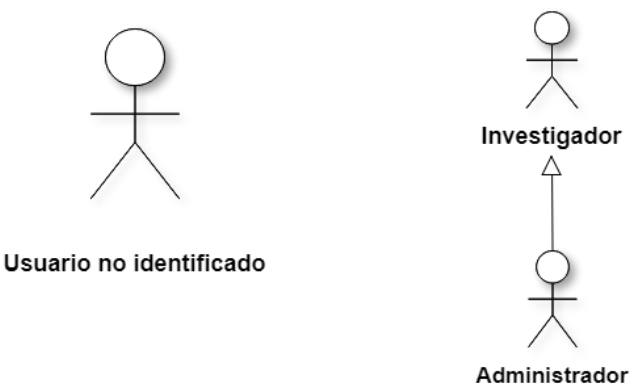


Ilustración 7 - Jerarquía de actores

4.2.2. Modelo de casos de uso

Los casos de uso son utilizados como una representación gráfica de las funcionalidades que posee el sistema, y muestra, además, como debería interactuar el sistema con cada uno de los actores para conseguir su objetivo. Por consiguiente, en la *Ilustración 8* se puede visualizar el diagrama de casos de uso de Walkability Analyzer. Estos casos de uso se explicarán más detalladamente en el Anexo I de esta documentación, donde se analizarán los casos de uso extendidos.

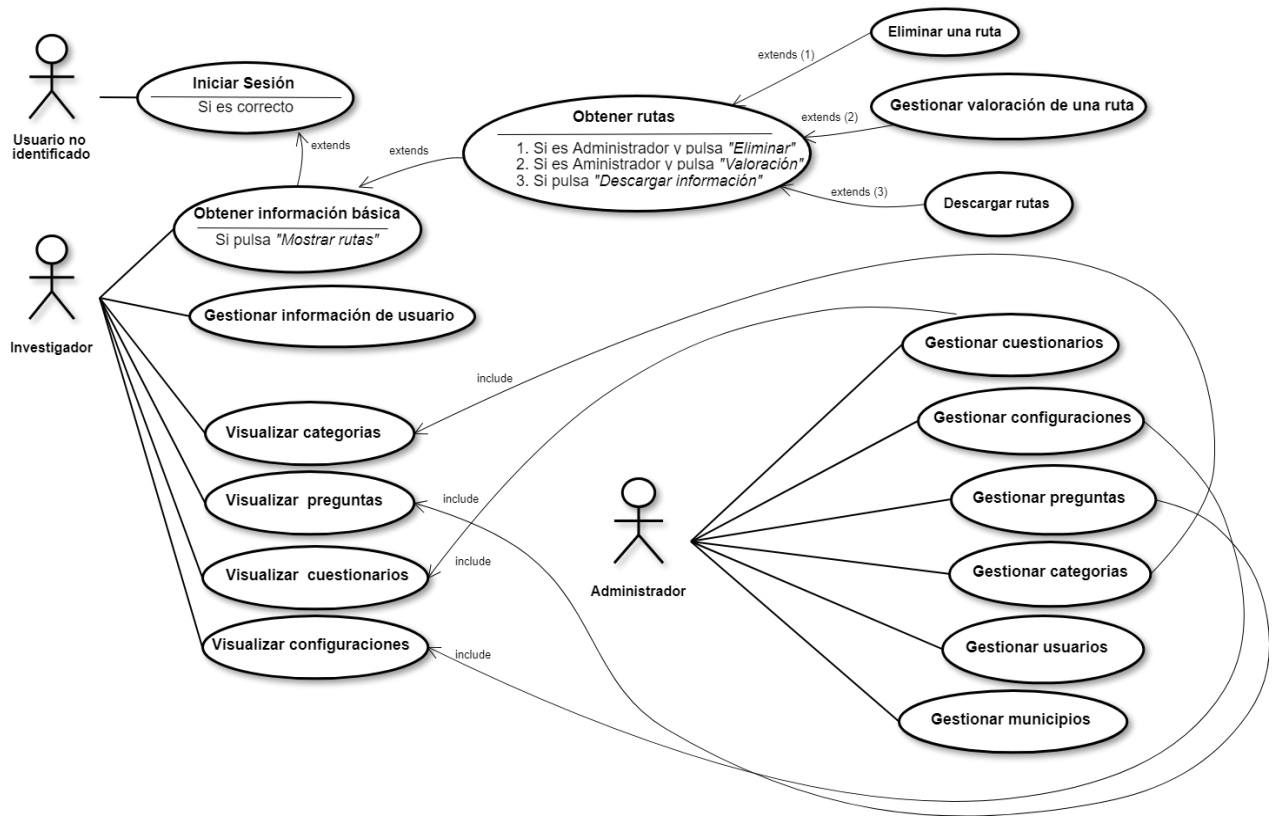


Ilustración 8 - Modelo de casos de uso

En primer lugar se analizará el caso de uso perteneciente al actor "*Usuario no identificado*":

- **Iniciar Sesión:** Permite a un usuario identificarse en el sistema con sus credenciales para obtener un *Token* válido, y poder así acceder a los recursos que tenga establecido su rol de usuario.

A continuación, se detallarán todos los casos de uso pertenecientes al actor "*Investigador*":

- **Obtener información básica:** Una vez que el usuario inicia sesión en el sistema, o cuando se accede a la pantalla principal de la aplicación, se cargarán algunos datos principales, como, por ejemplo, el texto de las categorías o los municipios de inicio y de fin de las rutas almacenadas en el sistema. Este recurso permite al usuario visualizar dicha información en la pantalla principal. En caso de que el usuario seleccione "*Mostrar rutas*", este tendrá disponible un sub-caso de uso que se explicará a continuación:
    - **Obtener rutas:** Este sub-caso de uso describe la funcionalidad de obtener las rutas del sistema, permitiendo al actor aplicar una serie de filtros o condiciones sobre las mismas. En este sub-caso de uso, según la acción que el usuario decida realizar se tendrán disponibles otros sub-casos de uso:
        - **Eliminar una ruta:** Al obtener y visualizar una ruta, si esta no cumple con lo establecido o es una ruta defectuosa, el actor podrá eliminarla del sistema. Esta acción solo puede ser realizada por el actor "*Administrador*".

  - **Gestionar valoración de una ruta:** Al visualizar las rutas del sistema, el usuario podrá obtener y visualizar la valoración de las rutas que estén valoradas previamente. Además, el usuario podrá eliminar dicha valoración. Esta acción solo puede ser realizada por el actor "*Administrador*".

  - **Descargar rutas:** Al obtener las rutas, el actor, si lo desea, puede descargar la información de dichas rutas en formato compatible con sistemas QGIS o en formato compatible con SPSS.

- **Obtener información de usuario:** Este recurso permite a todo usuario identificado en el sistema obtener su información básica, y posteriormente, en caso de que el usuario lo desee, podrá modificar dicha información.

- **Visualizar categorías:** Este recurso permite obtener y visualizar toda la información perteneciente a las categorías.

- **Visualizar preguntas:** El caso de uso de "*Visualizar preguntas*" permite obtener y visualizar en pantalla el catálogo de preguntas con toda su información asociada.

- **Visualizar cuestionarios:** Este recurso permite obtener el catálogo de cuestionarios presente en el sistema y visualizarlos en pantalla.

- **Visualizar configuraciones:** Este caso de uso permite visualizar en pantalla todas las configuraciones que definen los parámetros de las rutas capturadas por Walkability Capturer.

Finalmente, se detallarán todos los casos de uso del sistema pertenecientes al actor "*Administrador*":

- **Gestionar cuestionarios**: Este recurso permite, en primer lugar, obtener los cuestionarios y visualizarlos en pantalla con toda su información asociada. Posteriormente el actor *"Administrador"* podrá realizar otras acciones sobre los cuestionarios, como, por ejemplo, editar el mismo, eliminarlo, activarlo o desactivarlo. Las preguntas presentes en el cuestionario activo serán las que se pregunten a los usuarios en el momento de valorar una ruta en la aplicación móvil. Además, en la pantalla de visualización de cuestionarios se podrá añadir un cuestionario nuevo.

- **Gestionar configuraciones**: El caso de uso de *"Gestionar configuraciones"* permite obtener y visualizar las configuraciones presentes en el sistema. Una configuración está formada por el conjunto de parámetros que define una ruta. Por ejemplo, uno de los valores de una configuración es la distancia mínima de recorrido a partir del cual se considera que un trayecto es una ruta. Una vez las configuraciones se han visualizado, se podrán realizar varias acciones: eliminar una configuración, activar una configuración o desactivarla. Al activar una configuración, las rutas que se obtienen en la aplicación Walkability Capturer son en base dicha configuración activa. Además, en la pantalla de visualización de configuraciones se podrá añadir una configuración nueva.

- **Gestionar preguntas**: Este caso de uso permite obtener y visualizar las preguntas, así como la información de las mismas presentes en el sistema. Posteriormente se podrán realizar varias acciones sobre las mismas: eliminar la pregunta o editar su información. Además, después de visualizar las preguntas se puede, en caso de que el usuario lo desee, añadir una nueva pregunta al sistema.

- **Gestionar categorías**: Este recurso permite, en primer lugar, visualizar las categorías presentes en el sistema. A continuación, en caso de que el usuario lo desee, puede editar dichas categorías.

- **Gestionar usuarios**: El actor "*Administrador*" puede visualizar todos los usuarios registrados en el sistema, ya sea con el rol de "*Investigador*" como con el rol de "*Administrador*". A continuación, se permite al usuario realizar varias acciones: actualizar los permisos de los usuarios registrados o eliminarlos. Además, una vez los usuarios son visualizados, el actor podrá registrar nuevos usuarios en el sistema y asignarles un rol.

- **Gestionar municipios**: Para poder solucionar el problema de los municipios erróneamente obtenidos detectado en el sistema, se podrán obtener los municipios de inicio y de fin de las rutas almacenadas en el sistema. Una vez que el usuario visualiza los municipios de las rutas, en caso de detectar algún municipio que desee cambiar, podrá actualizar el valor del mismo en el sistema.

4.3. Modelo de dominio

En un diagrama de modelo de dominio se pueden visualizar de manera gráfica todos los datos que van a ser almacenados, así como las relaciones existentes entre ellos. En la *Ilustración 9* se muestra el modelo de dominio de Walkability Analyzer en su representación UML. Este diagrama está realizado a dos colores, con fondo blanco están las tablas que se crearon para almacenar la información de Walkability Capturer, y son dichas tablas de las que hay que obtener, crear, modificar o eliminar la información. En cambio, con fondo gris y rodeada con un borde amarillo se encuentra la tabla creada exclusivamente para Walkability Analyzer.

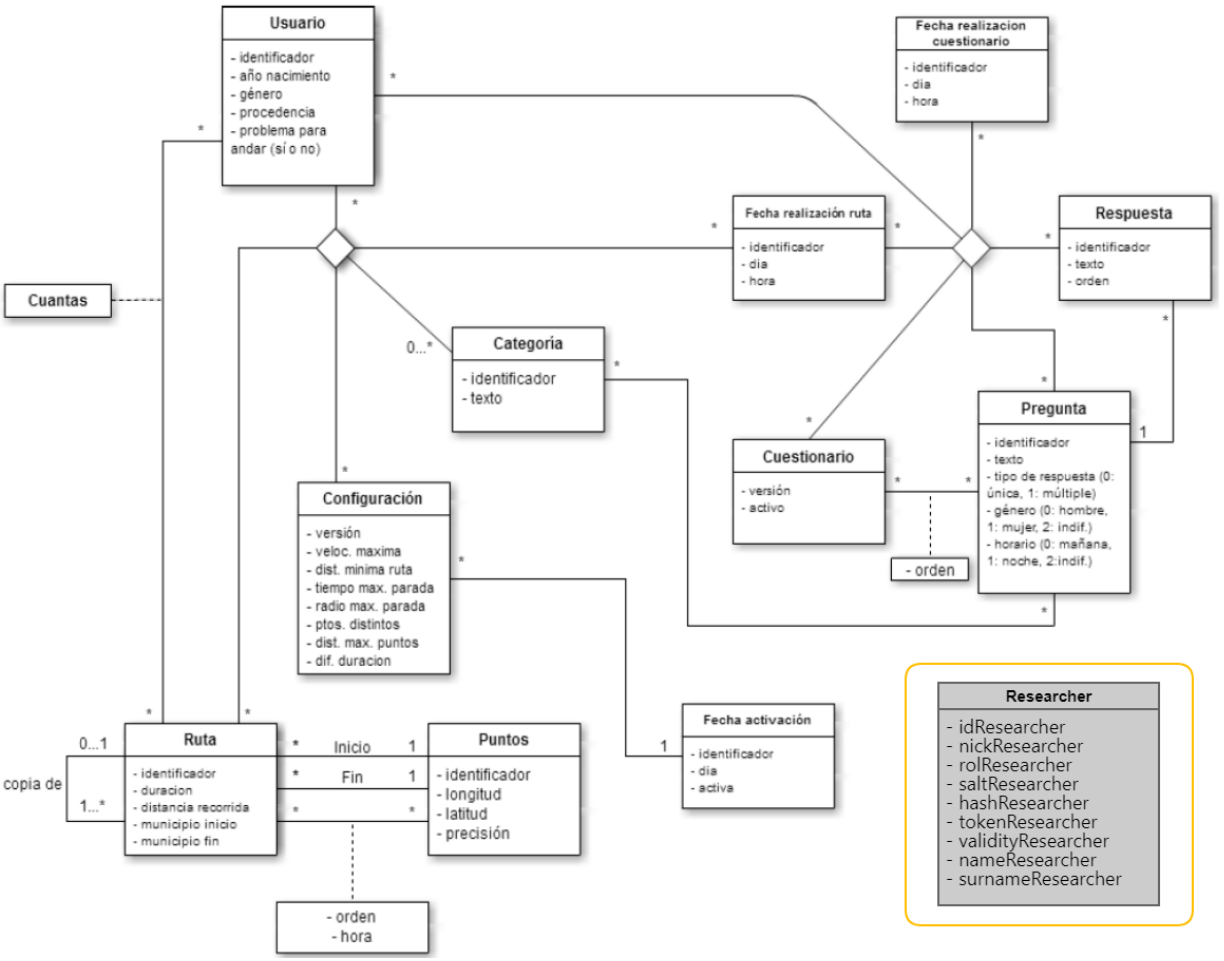


Ilustración 9 - Modelo de dominio

A continuación, se explicará de forma detallada el significado de cada una de las entidades y sus relaciones:

- **Usuario**: En la entidad **Usuario** se almacenan los datos de los usuarios de la aplicación Walkability Capturer.

- **Ruta**: En la entidad **Ruta** se almacenan todos los datos pertenecientes a una ruta, como puede ser la duración de la misma y la distancia recorrida. Para facilitar el análisis de los datos mediante Walkability Analyzer, el grupo CRIM-AP solicitó que en esta entidad se almacenasen también los municipios de inicio y fin, ya que así podría conocerse el origen y el fin de la ruta sin necesidad de pasar las coordenadas a un Sistema de Información Geográfico (SIG).

La entidad **Ruta** cuenta con una relación hacia sí misma. Esto es así ya que es necesario saber si una ruta es o no copia de otra y, en caso afirmativo, saber de qué ruta es copia. El atributo Cuantas, que se encuentra en la relación entre las entidades **Usuario** y **Ruta**, tiene como objetivo saber cuántas veces ha realizado un usuario una determinada ruta.

- **Configuración**: En la entidad **Configuración** se almacenan los valores, definidos por parte de los investigadores, que pueden variar a lo largo del estudio. Por ejemplo, la velocidad máxima de movimiento para que una ruta se considere que ha sido realizada a pie.

- **Fecha activación**: Esta entidad almacena en qué momento se ha activado una configuración concreta del sistema.

- **Puntos**: Cada ruta está formada por una serie de puntos, así como un punto de inicio y otro de fin. Debido a esto, la entidad **Puntos** está relacionada de manera directa con la entidad **Ruta**.

Entre las entidades **Punto** y **Ruta**, además de las relaciones de inicio y fin necesarias para saber dónde comienza y finaliza una ruta, se encuentra una tercera relación con los atributos Orden y Hora. Estos datos son necesarios para conocer la posición de cada punto dentro de cada ruta que ha sido realizada.

- **Cuestionario**: La entidad **Cuestionario** almacena las versiones de cuestionario existentes en el sistema.

- **Pregunta**: Esta entidad almacena los datos necesarios para formular las preguntas que componen los cuestionarios. Una pregunta puede pertenecer a varios cuestionarios simultáneamente.

El atributo Orden en la relación entre las entidades **Cuestionario** y **Pregunta** se debe a que las preguntas deben guardar un orden dentro del cuestionario ya que este hecho es relevante para los investigadores.

- **Respuesta**: La entidad **Respuesta** contiene el texto correspondiente a las respuestas de las preguntas que se le muestran al usuario. Cada respuesta únicamente pertenece a una pregunta.

- **Categoría**: La entidad **Categoría** almacena el tipo de ruta que realizará el usuario, es decir, la finalidad con la que ha sido hecha, como, por ejemplo, hacer recados. La categoría solo es asignada a las rutas que el usuario ha valorado una vez realizadas, por eso la relación entre **Ruta** y **Categoría** tiene cardinalidad *0-1*, ya que pueden tener o no una valoración. En la relación entre **Categoría** y **Pregunta** la cardinalidad, en cambio, es *n-m*, ya que una misma pregunta puede pertenecer a una o más categorías.

- **Fecha-realización-ruta**: En esta entidad se guarda qué día y a qué hora ha sido realizada una ruta concreta.

- **Fecha-realización-cuestionario**: Una ruta puede ser valorada más de una vez, por lo que se debe almacenar en que momento ha sido valorada. Como los cuestionarios varían a lo largo del estudio, también se almacena con qué versión ha sido valorada.

La relación entre **Usuario**, **Ruta** y **Fecha-realización-ruta** se debe a que es necesario saber qué usuario ha hecho qué ruta en qué momento. Un usuario puede hacer muchas rutas en distintos momentos. En esta relación también están presentes las entidades **Categoría** y **Configuración**, esto se debe a que, dependiendo del momento en el que una ruta sea realizada, la percepción de la misma puede cambiar, al igual que la configuración activa en ese momento.

La relación entre seis entidades aparece porque es necesario saber qué ha contestado un usuario, a qué pregunta de qué cuestionario realizado en un momento concreto, y sobre qué ruta realizada en un momento especifico ha realizado dicha valoración.

- **Researcher**: Esta entidad, creada únicamente para el desarrollo de la página web de Walkability Analyzer, almacenará los datos de los usuarios que tienen acceso a esta aplicación. La entidad **Researcher** almacenará las credenciales de acceso de los usuarios al sistema, el rol que poseen y algunos datos básicos personales.

5. Análisis y diseño

Una vez definidos los requisitos del sistema, la siguiente fase a realizar en un proyecto de desarrollo de software es la fase de análisis y diseño. En esta parte se profundizará en la arquitectura del proyecto definida en el planteamiento inicial, de esta manera, definiendo de una forma más precisa la estructura del proyecto se facilitará el proceso de construcción del sistema.

En concreto, se analizarán y diseñaran los aspectos imprescindibles para el desarrollo de la aplicación, especificando la estructura de trabajo que seguirá el back-end, el front-end y, por último, el diseño de bases de datos. Para una mejor comprensión de las funcionalidades del sistema, en el Anexo II de esta documentación se adjuntan los diagramas de secuencia más importantes relacionados con el diseño de Walkability Analyzer.

5.1. Estructura del back-end y API REST

Walkability Analyzer seguirá la arquitectura REST mediante el desarrollo de una API REST, la cual, se divide principalmente en tres fragmentos. Por una parte, se encuentra la parte servidora, en el otro extremo, en cambio, se encuentra la parte cliente, que consume los recursos ofrecidos por la API, y, entremedias de estos fragmentos, se encuentra un canal de comunicación mediante el cual se envían los datos.

Debido a que los datos necesarios para realizar las comunicaciones entre los extremos se enviarán utilizando redes al alcance de cualquiera, es importante tener en cuenta el aspecto de la seguridad. Por ello, la parte back-end de Walkability Analyzer se construirá siguiendo un esquema que permita implementar ciertos aspectos de seguridad, como, por ejemplo, la autentificación mediante *Tokens*, incorporación y comprobaciones de roles y protección contra ataques a la base de datos. En el apartado de “Seguridad en la parte servidor” se analizará más detalladamente el funcionamiento de la seguridad de la aplicación, incluido el proceso de generación de *Tokens*.

Para construir este sistema en la parte de back-end se realizará una separación en módulos o capas que no solo favorezca la seguridad, sino que permita desarrollar un sistema lo más modular posible. En la *Ilustración 10* se puede visualizar gráficamente el sistema de módulos que seguirá Walkability Analyzer.

En primer lugar, existirá un *Middleware* en la parte servidora que se encargará de filtrar y validar las peticiones entrantes. El Middleware, que se activará una vez que se produzca el inicio de sesión de un usuario, comprobará en cada petición que el *Token* que se encuentra en la cabecera de la petición no ha expirado y que es un *Token* válido.

Una vez que la primera capa de seguridad del *Middleware* es traspasada, ya que la petición es válida, esta será recibida por el *Módulo de comunicación*. El *Módulo de comunicación* tendrá la función de recibir las peticiones válidas entrantes al sistema y determinar que el recurso solicitado existe en la API. Una vez que esto se efectúa, validará los parámetros presentes en la cabecera o en el cuerpo de la petición para comprobar si cumplen con la estructura requerida. Además, una vez el sistema tenga una respuesta para la petición entrante, este módulo será el encargado de devolver el resultado de la petición junto con el status o estado de la misma.

El *Módulo de comunicación*, una vez que de por validada la petición, se encargará de enviar la solicitud al nivel inferior correspondiente. En este caso, este nivel inferior es el

Módulo de coordinación. El *Módulo* de *coordinación* será el encargado de tratar con los datos y realizar las llamadas necesarias al *Módulo de gestión* para poder así obtener la información solicitada. El *Módulo de gestión* se encargará de proporcionar esta información y atender las solicitudes del *Módulo de coordinación*, y para ello, accederá a la base de datos.

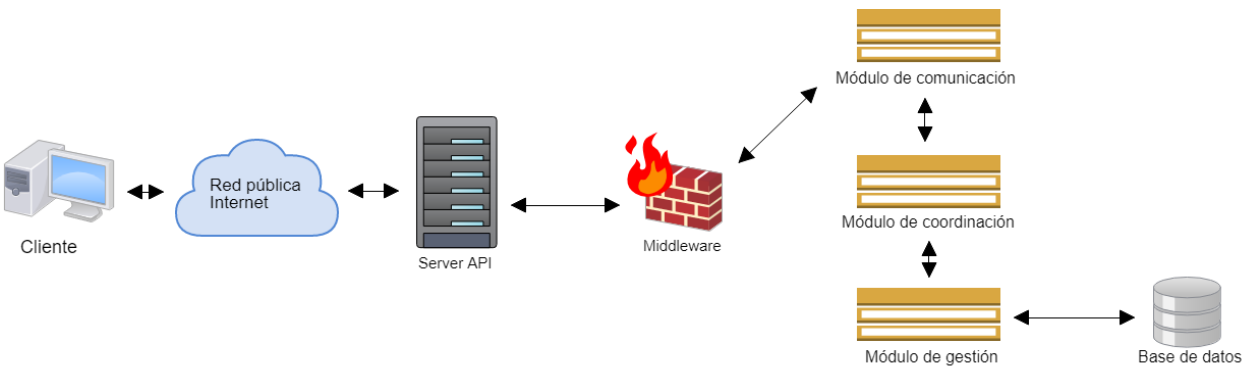


Ilustración 10 - Arquitectura de Walkability Analyzer API

Una vez analizados los módulos o capas en los que está dividido el back-end se detallará más en profundidad el proceso de comunicación entre dichas capas y los ficheros que las forman. En la *Ilustración 11* se muestra de manera gráfica como se realiza el proceso de comunicación entre los módulos y ficheros del back-end.

Cuando una petición accede a la parte back-end de la página web, y esta petición pertenece a un usuario que ha iniciado sesión y posee un Token valido, el *Middleware* la valida y la solicitud llega a la capa superior del sistema. Esta petición entrante posee una ruta o URL, y unos parámetros, por lo que, primeramente, la ruta se analiza en el fichero base de la aplicación para comprobar si existe o no realmente. Este fichero base, o fichero de configuración del sistema, posee una lista de todas las rutas entrantes junto con el nombre del fichero donde se tratan, por ello, al entrar una petición al sistema, este fichero analiza la ruta, y en caso de estar en el listado de rutas, redirige la petición al fichero correspondiente.

Si la ruta entrante en el sistema es correcta, la petición se envía al archivo que dicta el fichero base, y este archivo, situado en el *Módulo de comunicación*, comprueba que los parámetros que tiene establecidos existen realmente en la petición entrante. Por ejemplo, si el método del *Módulo de comunicación* correspondiente a "*Eliminar categoría*" tiene establecido que debe recibir un identificador en la cabecera, en caso de que este parámetro no esté en la solicitud, generará un error y la petición no se considerará válida.

Cuando el *Módulo de comunicación* finalmente acepta una petición, entonces esta, junto con sus parámetros, es redirigida a un archivo del *Módulo de coordinación* que se encargue de tratar el recurso solicitado. En concreto, la petición se envía al método que se encargue de realizar la acción solicitada, y eso se conseguirá gestionando los ficheros del *Módulo de gestión* que se necesiten. Por ejemplo, si una petición que llega al *Módulo de coordinación* necesita obtener tanto las preguntas del sistema como las categorías, el método al que llega la petición se encargará de llamar a los ficheros correspondientes del *Módulo de gestión*, en este caso, al archivo que se encarga de realizar las peticiones a la base de datos con acciones relacionadas con las preguntas, y al fichero que gestiona las categorías.

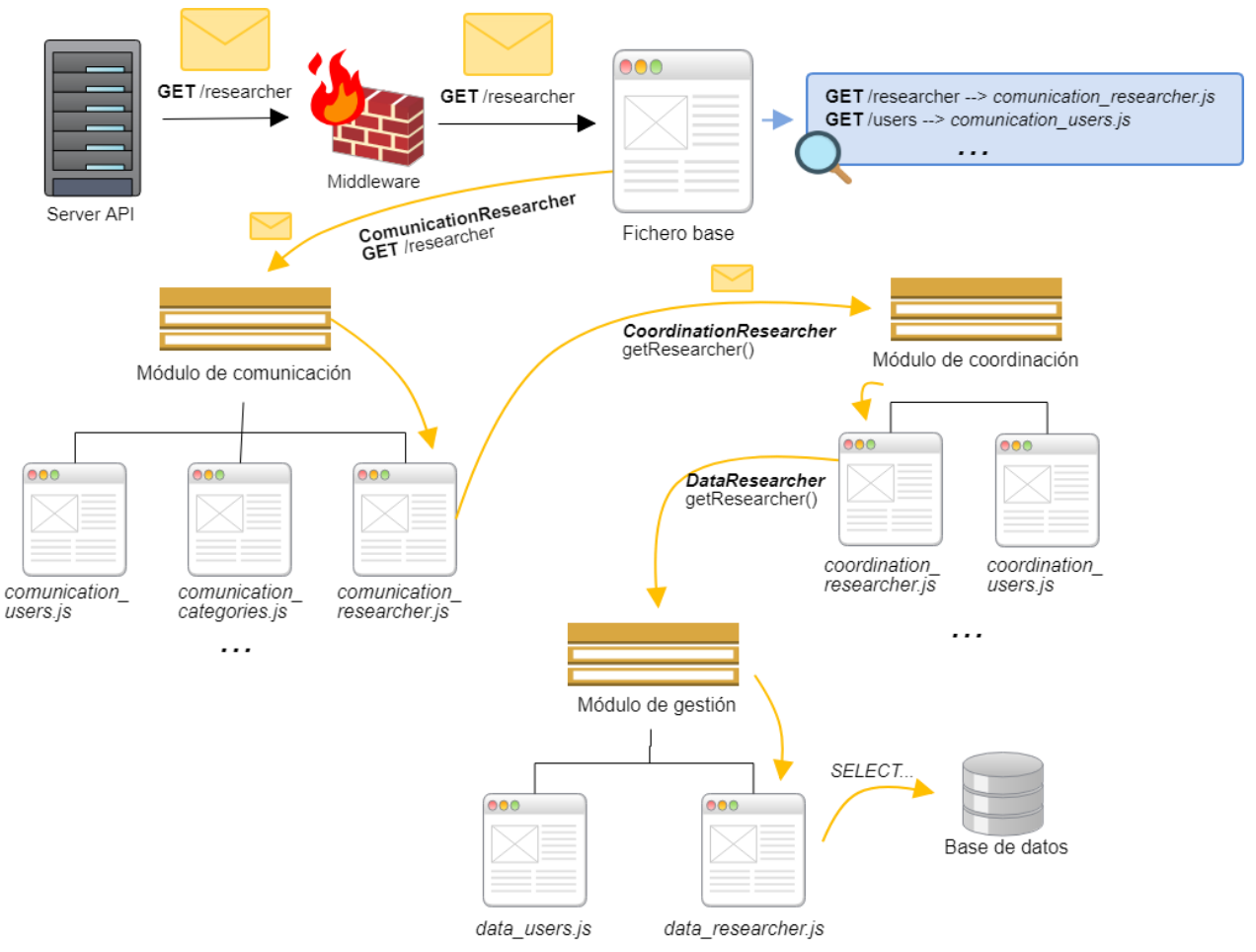


Ilustración 11 - Proceso de comunicación en el back-end

En la ilustración anterior se puede apreciar un ejemplo del proceso de comunicación que sigue la parte del servidor. En dicho ejemplo se realiza una solicitud al sistema con la ruta, o URL, */researcher*, y cuyo tipo de petición es un GET. Esta petición pasa por el *Middleware* del sistema y este valida la petición, por lo que la ruta es analizada posteriormente por el fichero base, o fichero de configuración. Este fichero compara internamente la ruta, y tras analizar la misma y comprobar que dicho recurso existe en el sistema, redirige la petición al *Módulo de comunicación*, en concreto, al fichero *comunication_researcher.js*.

Una vez en dicho fichero, se comprueba que los parámetros recibidos, en caso de necesitar parámetros dicha petición, estén correctos, y en caso de que esto sea así se redirige la solicitud al fichero correspondiente en el *Módulo de coordinación*. De esta manera, este último módulo puede comunicarse con el *Módulo de gestión* para realizar las solicitudes necesarias a la base de datos.

5.2. Estructura del front-end

Walkability Analyzer, tal y como se ha explicado en apartados anteriores, estará separado en back-end y front-end. Para desarrollar la parte de front-end, la parte de la aplicación visible para el cliente, se utilizará el framework Angular. Angular permite comenzar un proyecto de desarrollo con la estructura básica del código ya creada, de esta manera, resulta más sencillo construir una arquitectura que los navegadores interpreten de manera correcta y rápida.

Angular es un framework de desarrollo basado en módulos y componentes. Por un lado, un componente es una porción de código que es posible reutilizar en otros proyectos de Angular sin apenas esfuerzo, lo que permite un desarrollo de aplicaciones mucho más ágil. Un componente va a controlar un trozo de interfaz gráfica o de vista, tanto su estructura y su estilo como su funcionamiento. Por ejemplo, la barra de navegación de una página web, que forma parte de todas las pantallas de la misma, podría ser un componente.

Por otro lado, un módulo de Angular, es una agrupación de componentes. Cada módulo agrupa al menos un componente, y la estructura a elegir para el desarrollo es decisión del programador, pudiéndose seguir muchos criterios. Por ejemplo, se pueden agrupar componentes que se reutilizan en toda la aplicación, o, por el contrario, se pueden agrupar componentes en función de la pantalla a la que pertenecen.

Walkability Analyzer se desarrollará en dos módulos principales, los cuales agruparán los componentes necesarios para permitir realizar todas las funcionalidades necesarias de la página web. En la *Ilustración 12* se puede visualizar de manera gráfica la estructura del proyecto que se seguirá para el desarrollo del front-end.

El primer módulo de Walkability Analyzer será el módulo de *Login*, que poseerá el componente de *Login* que permitirá a los usuarios Iniciar Sesión en el sistema. El otro módulo, llamado *ManageRoutes*, englobará el resto de la aplicación y agrupará el resto de componentes específicos del sistema, en este caso, se utilizará un componente por cada pantalla necesaria en la página web. Por ejemplo, dentro de *ManageRoutes* habrá un componente llamado *Questionaries* que contendrá la pantalla de gestión de cuestionarios, otro componente llamado *Users* que permitirá gestionar los usuarios con acceso al sistema, y así sucesivamente. Los dos módulos principales de la aplicación se agruparán en un paquete llamado *Modules*.

Por otro lado, existen algunos componentes que no representarán pantallas de la aplicación, sino que tal y como especifica la definición de componente, serán reutilizados en toda la aplicación. Estos componentes se agruparán dentro de un paquete llamado *Components*, que englobará, entre otras cosas, el formato de las tablas a utilizar en la aplicación o los mensajes de confirmación o error que se le mostrarán al usuario. Por ejemplo, el componente *PageNotFound* se mostrará cuando alguna página solicitada no exista, y el componente *Confirmation* solicitará la confirmación del usuario antes de realizar una acción crítica como la de eliminar algún elemento.

Una vez analizada la estructura de módulos y componentes que se desarrollará para construir el front-end de la aplicación, se deben definir los servicios encargados de comunicar la interfaz con el back-end, es decir, de realizar las peticiones al back-end a través de la API. Para ello se creará un paquete llamado *Services* que contará con dos sub-paquetes: *GeneralServices* y *WebServices*:

- ***GeneralServices***: Este paquete poseerá los archivos que pueden ser útiles para el resto de ficheros de la aplicación, es decir, servicios comunes, por ejemplo, el atributo del puerto y la ruta base mediante los que poder conectarse al back-end.

- ***WebServices***: Este paquete contendrá todos los ficheros que realizan las peticiones al back-end. Dentro de él se creará un fichero por cada pantalla de la aplicación, es decir, un fichero para todas las llamadas al back-end relacionadas con las funcionalidades de los cuestionarios, otro fichero para las llamadas relacionadas con las rutas, y así sucesivamente.

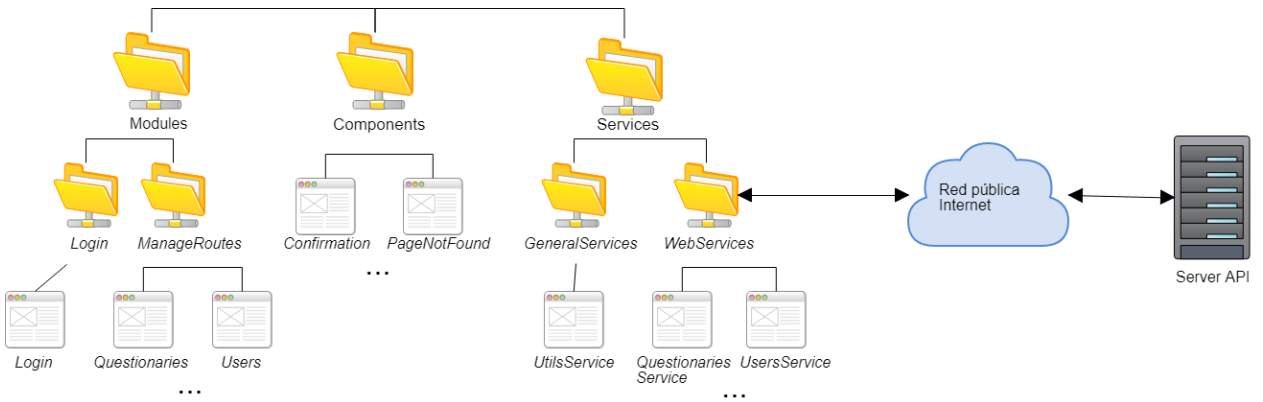


Ilustración 12 - Arquitectura de Walkability Analyzer front-end

5.3. Diagrama relacional de la Base de Datos

En el capítulo anterior se detalló el *Modelo de Dominio* de Walkability Analyzer. En concreto, sólo una única entidad pertenece exclusivamente a este trabajo, mientras que las demás fueron definidas en el proyecto de Walkability Capturer, y se gestionan en este proyecto para extraer y trabajar con la información almacenada en ellas. En la *Ilustración 13* se puede visualizar el diagrama relacional de la base de datos extraído a partir del *Modelo de Dominio* analizado previamente.

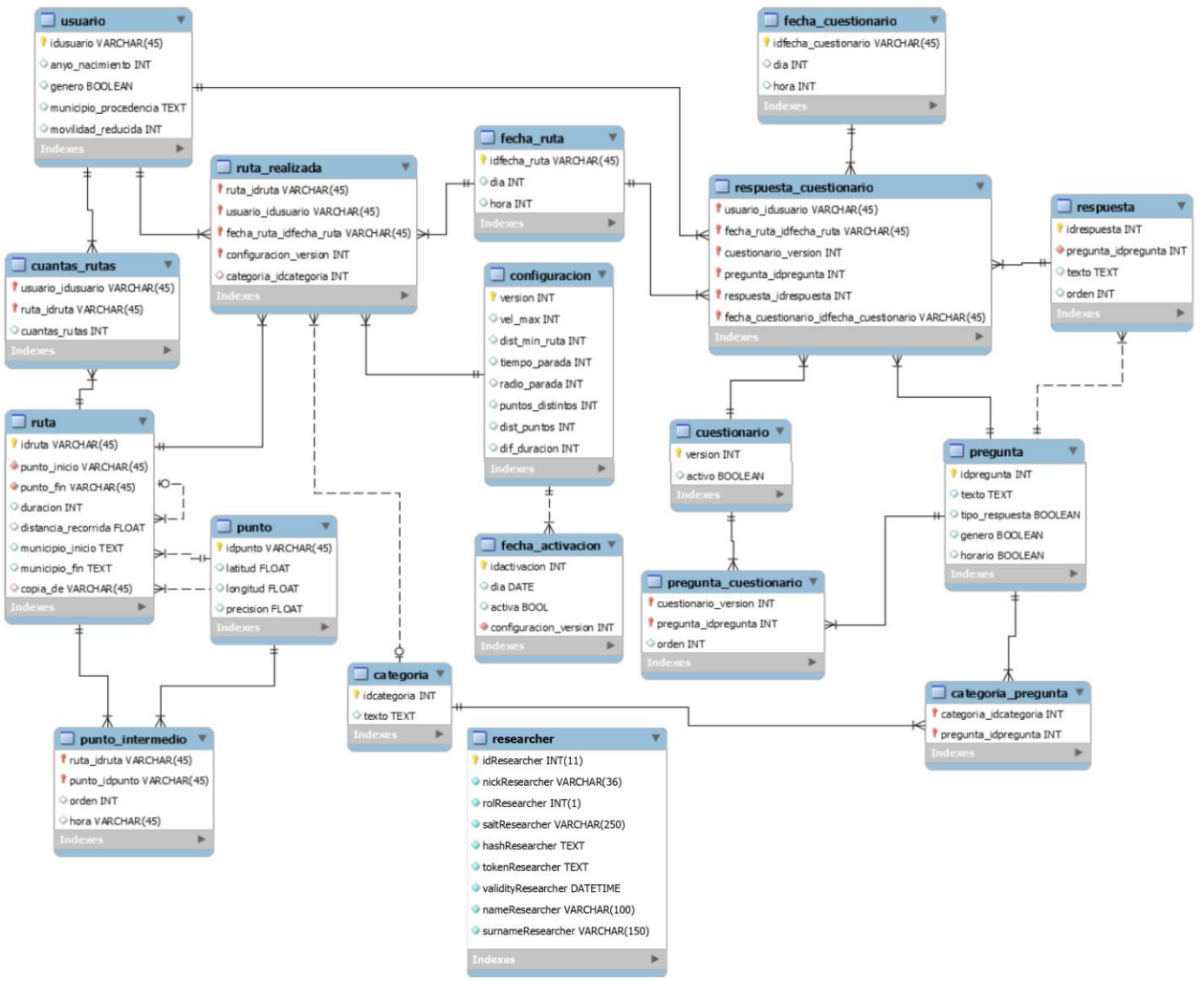


Ilustración 13 - DIagrama relacional de la Base de Datos

A continuación, se describirá la estructura de tablas y los atributos que las componen:

La tabla ***researcher*** almacena los datos de los miembros del grupo CRIM-AP que pueden acceder a la aplicación web de Walkability Analyzer. Esta tabla, que no posee ninguna relación, almacena el nombre del usuario en el sistema, su nombre real y sus apellidos. Por otro lado almacena otros datos relacionados con la parte de seguridad de la aplicación, como el *Token* del usuario, la fecha de validez de dicho *Token*, el rol que posee en la aplicación y dos valores utilizados para almacenar la contraseña encriptada: la *salt* y el *hash*.

La tabla ***usuario*** contiene los datos de los usuarios del sistema Walkability Capturer. Así, por cada usuario, se almacena el identificador, el año de nacimiento, el género, el municipio de procedencia y si sufre o no problemas de movilidad. Esta tabla está relacionada por tres partes. Por una parte, está directamente relacionada con la tabla *ruta*, aunque la relación tiene el atributo "*cuantas*" para saber cuántas veces ha realizado una misma ruta un usuario. Esto da lugar a la tabla ***cuantas_rutas*** en la que se almacena los identificadores de usuario, de ruta y el atributo *cuantas_rutas*.

Por otra parte, en el diagrama se encuentra la relación quíntuple entre las tablas ***usuario***, ***ruta***, ***fecha_ruta***, ***categoría*** y ***configuración***. Gracias a esta relación se permite saber qué usuario ha hecho qué ruta, en qué momento determinado, con qué configuración y

con qué finalidad. Para ello, se crea la tabla ***ruta_realizada***, que por cada registro almacena los identificadores *de usuario, ruta, fecha_ruta, categoría* y *configuración*. Como la finalidad con la que se realiza una ruta se conoce únicamente a la hora de valorarla, en la tabla *ruta_realizada,* el atributo *categoría_idcategoria* tiene el valor *null* en el caso de que la ruta no haya sido valorada, y en el caso de que sí lo haya sido, tiene el identificador correspondiente a la categoría seleccionada por el usuario.

La tabla ***ruta*** almacena todas las rutas realizadas por los usuarios. Cada registro de la tabla posee un identificador, la duración, la distancia recorrida, el identificador del punto de inicio, el municipio de inicio, el identificador del punto de fin y el municipio de fin. La tabla *ruta* tiene una relación consigo misma para conocer la relación entre una ruta original y su copia, que almacenará el identificador de la original, cuando la haya, en el atributo *copia_de*. Toda ruta tendrá un inicio y un fin, por eso la tabla está relacionada con la tabla ***punto*** y tiene como clave foránea *punto_inicio* y *punto_fin*, que son los identificadores de los puntos de inicio y fin de la ruta. Además, la tabla ***punto_intermedio*** almacena el identificador del punto, el de la ruta, la posición del punto en la ruta y la hora a la que fue recogido.

La tabla ***configuración***, contendrá todos los valores que pueden ser modificados por el equipo investigador según las necesidades que vayan surgiendo en el desarrollo del estudio. Esta tabla está relacionada con la tabla ***fecha_activación***, ya que es necesario registrar cuándo se activa cada configuración. Por cada registro se almacena un identificador, el día y si está o no activa la versión. En toda la tabla solo deberá haber una tupla con el atributo activo con valor true, ya que solo puede haber una configuración activa.

La tabla ***cuestionario*** contiene la versión actual del cuestionario que debe lanzar la aplicación al usuario para que valore una ruta. Por cada registro almacena la versión y si está activa o no. Al igual que en la tabla ***fecha_activación***, solo debe haber una tupla activa.

La tabla ***pregunta*** contiene todos los datos necesarios para realizar una pregunta en el cuestionario. Por cada registro, almacena un identificador, el texto de la pregunta y el tipo de respuesta que tiene la pregunta, ya que esta puede ser de respuesta única o múltiple. También se almacenará el género del usuario y la franja horaria en la que se ha realizado la ruta. Por defecto, se ha determinado que horario de mañana es desde las 07:00h hasta las 22:59h y el horario de noche desde las 23:00h hasta las 06:59h. Estos dos atributos, junto a la categoría, se tendrán en cuenta a la hora de elegir qué preguntas se le mostrarán al usuario en la aplicación móvil.

La tabla ***pregunta*** tiene varias relaciones. Por una parte, está relacionada con la tabla cuestionario con una cardinalidad *n:m*, por lo que se genera una tabla intermedia llamada ***pregunta_cuestionario***. En esta nueva tabla, por cada registro, se almacena los identificadores de las dos tablas. Por otro lado, la tabla ***pregunta*** se relaciona con la tabla ***respuesta***. Esta tabla, por cada registro, almacena un identificador, el texto de la respuesta y el orden en el de deberá ser mostrada.

Por otro lado, la tabla ***pregunta*** también estará relacionada con la tabla ***categoría*** ya que, como se ha mencionado, se tendrá en cuenta a la hora de mostrar el cuestionario. Como una pregunta puede pertenecer a más de una categoría, se crea la tabla ***categoría_pregunta*** que almacena los identificadores de las dos tablas relacionadas.

En la tabla ***categoría*** se almacenarán las categorías o finalidades que tiene una ruta para el usuario. Cada registro contiene un identificador y el texto que define la categoría. Inicialmente, el grupo CRIM-AP ha decidido que las categorías sean:

- Desplazamiento a y desde el trabajo o centro de estudios.
- Desplazamientos durante la jornada laboral, como parte del trabajo.
- Tareas familiares (por ejemplo, acompañar a familiares al colegio, al centro de salud...).
- Ocio, tiempo libre.
- Compras y recados.

Cuando una pregunta sea válida para cualquier categoría, se relacionará con todas las categorías. De esta manera a la hora de seleccionar las preguntas será suficiente indicar la categoría elegida.

Asimismo, también resulta necesario saber cuándo un usuario realiza una ruta y un cuestionario, al igual que ocurre con las fechas de activación de los cuestionarios y las configuraciones. Por ello se crean las tablas ***fecha_ruta*** y ***fecha_cuestionario*** que almacenan el identificador de la fecha o ruta, el día y la hora.

Por último, en este diagrama entidad relación se puede encontrar una relación a seis para almacenar las respuestas de los usuarios a los cuestionarios. En concreto, es necesario saber la siguiente información:

- Quién ha respondido (***usuario***).
- Qué ha respondido (***respuesta***).
- A qué ha respondido (***pregunta***).
- En qué versión del cuestionario (***cuestionario***).
- Cuándo lo ha respondido (***fecha_cuestionario***).
- Sobre qué lo ha respondido (***fecha_ruta***, que junto con ***usuario*** permite saber sobre qué ruta estaba respondiendo).

Debido a esta necesidad se crea la tabla ***respuesta_cuestionario*** que, por cada registro, almacena los identificadores de las distintas tablas relacionadas que forman la clave principal y además son claves foráneas.

6. Desarrollo

En este capítulo, llamado desarrollo, se explicará cómo se ha desarrollado el sistema de Walkability Analyzer y todos los pasos que se han ido realizando para la construcción del mismo. Debido a que la aplicación web está estructurada en dos partes, el apartado de desarrollo también estará dividido de la siguiente forma: desarrollo de back-end y desarrollo de front-end.

6.1. Desarrollo de back-end

En este apartado se analizará cómo crear la estructura del proyecto, la manera en la que se han creado los módulos que forman el back-end, la funcionalidad de los mismos, los problemas encontrados en el desarrollo de la parte servidor del sistema y las medidas de seguridad que se han aplicado.

6.1.1. Estructura de la aplicación y puesta en funcionamiento

Como ya se comentó anteriormente, concretamente en el apartado 2.5, para el desarrollo del back-end se van a utilizar NodeJS. Para ello, una vez que se ha instalado NodeJS en el sistema se instala Express mediante el siguiente comando ejecutado en la consola:

```
npm install -g express-generator
```

Una vez realizado este paso hay que construir la nueva aplicación e instalar en ella las dependencias necesarias para comenzar el desarrollo. Esto se consigue ejecutando los siguientes comandos:

```
express walkability_analyzer
cd walkability_analyzer
npm install
```

Una vez ejecutados todos los comandos, en la carpeta donde estaba situada la consola del sistema se habrá generado el esqueleto de un proyecto estándar. En concreto, en la *Ilustración 14* se puede visualizar la estructura básica de la que parte Walkability Analyzer.

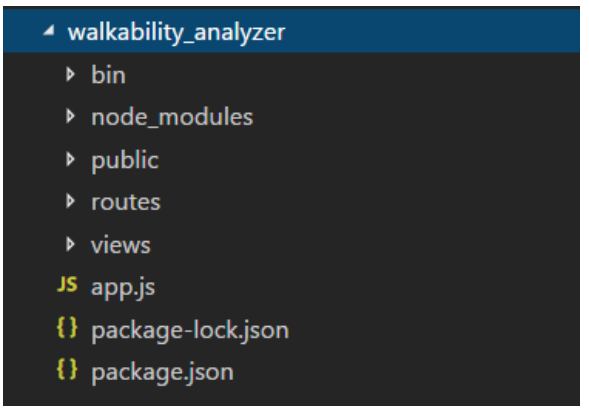


Ilustración 14 - Estructura básica del back-end (I)

Una vez realizados estos pasos únicamente queda probar si la aplicación se ha creado correctamente arrancando el servidor. Express, en su configuración particular indica que, por defecto, el servidor web de NodeJS se ejecuta mediante el puerto 3000, por lo que para ver la aplicación en ejecución habrá que dirigirse a la dirección: *http://localhost:3000*. El servidor se arranca mediante el siguiente comando ejecutado desde la carpeta raíz del proyecto:

```
npm start
```

Con el comando ejecutado, y dirigiéndose a la dirección anteriormente indicada se podrá visualizar una pantalla similar a la de la *Ilustración 15*. Esto indica que la estructura del proyecto está bien montada y la ejecución del servidor funciona de manera correcta.

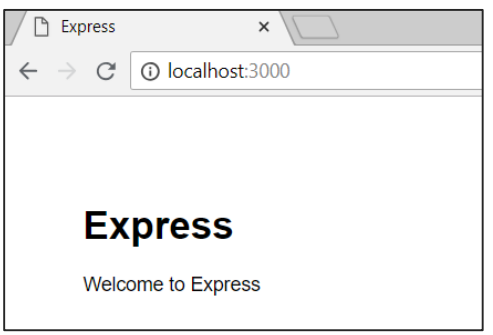


Ilustración 15 - Ejemplo básico de Express

Con todo lo realizado anteriormente ya se ha definido la estructura del proyecto y se ha comprobado que este funciona correctamente. Ahora bien, dentro de este proyecto hay carpetas que se pueden eliminar a elección del desarrollador, por ejemplo, el archivo *package-lock.json* es utilizado para establecer dependencias necesarias en proyectos realizados en equipo, por lo que aquí carece de sentido. La carpeta *views* contiene las vistas del sistema y un motor de representación de datos, por lo que, debido a que Walkability Analyzer posee un front-end independiente, esta carpeta no se utilizará. Debido a este motivo, la carpeta *public* también se puede eliminar, ya que ahí se definen y almacenan, entre otras cosas, las imágenes utilizadas por las vistas y las hojas de estilo.

La carpeta *routes*, donde se definen las rutas del sistema, también se elimina, ya que las rutas y los elementos de configuración del servidor y del API REST se definirán en un único archivo, el *app.js*. Por último, la carpeta *bin*, que también contiene configuraciones para lanzar y realizar pruebas con el servidor, al tratarse de configuraciones de la

aplicación, se incorporan al fichero *app.js*, anteriormente mencionado. Finalmente, la estructura mínima necesaria para el proyecto se visualiza en la *Ilustración 16*.

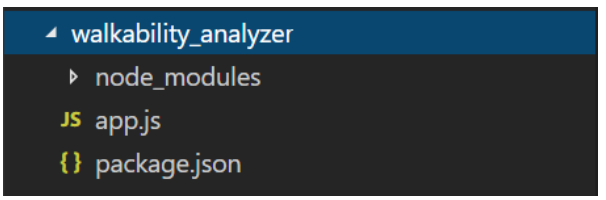


Ilustración 16 - Estructura básica del back-end (II)

En la estructura que queda una vez eliminadas las carpetas y archivos innecesarios para la realización de esta aplicación web, hay que analizar algunos elementos importantes. En primer lugar, el fichero de configuración *app.js* es donde se alojará toda la información referente al servidor del sistema. Entre esta información hay que destacar la definición de las rutas entrantes junto a qué fichero deberán ir redirigidas, es decir, este fichero es el archivo base que se explicaba en el capítulo anterior a la hora de definir la estructura del back-end. Por otro lado, el fichero *package.json* se encarga de gestionar las dependencias que utiliza el proyecto, por ejemplo, las dependencias necesarias con los módulos o librerías utilizadas. Estos módulos y librerías de terceros están almacenadas en la carpeta *node_modules* que se puede visualizar también en la estructura del sistema.

Con la estructura inicial ya creada, y todas las configuraciones indicadas en el archivo *app.js* es hora de comenzar a aplicar la estructura de capas o módulos analizada en el apartado "*5.1 Estructura del back-end y API REST*" del capítulo anterior. En concreto, se generaban tres módulos, "*Módulo de comunicación*", "*Módulo de coordinación*" y "*Módulo de gestión*".

Por otra parte, en Walkability Analyzer se tratan datos de diferentes tipos, desde todo lo relacionado con la configuración de la aplicación móvil Walkability Capturer, que engloba la gestión de configuraciones de ruta, cuestionarios, preguntas y categorías, hasta todo lo que tiene que ver con rutas y usuarios, tanto de la aplicación móvil como de la aplicación web. Por ello, dentro de los tres módulos, en los módulos de coordinación y gestión se ve necesario hacer una división para tratar cada tema en su apartado correspondiente. Así, la *Ilustración 17* muestra la estructura final de los módulos.

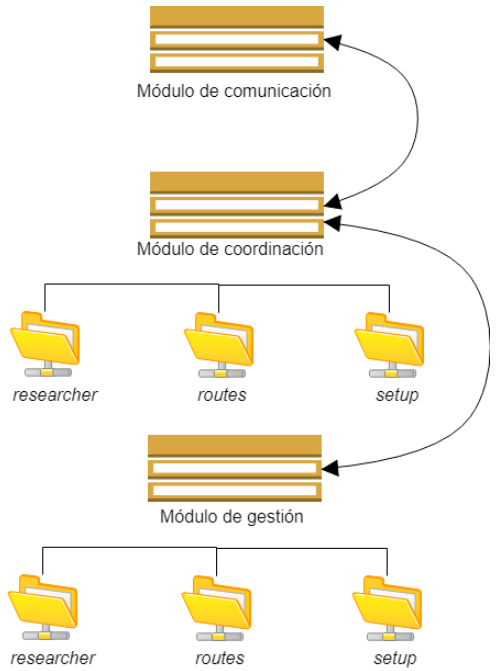


Ilustración 17 - División de los módulos del back-end

En concreto, una vez realizada esta división, tanto el "*Módulo de coordinación*" como en el "*Módulo de gestión*" se fraccionan en estos módulos:

- ***researcher***: En este apartado se almacenan todos los ficheros que se encargan de tratar y gestionar desde los propios usuarios hasta la información de los mismos, ya sea de la aplicación móvil Walkability Capturer o de la aplicación web que se está desarrollando.

- ***routes***: El sub-módulo *routes* almacena los ficheros que tratan o gestionan solicitudes en relación a las rutas realizadas por los usuarios de la aplicación móvil, por ejemplo, obtener todas las rutas o eliminar una ruta determinada.

- ***setup***: En este apartado se almacenan todos los ficheros relacionados con la configuración de la aplicación móvil Walkability Capturer. Es decir, todos los archivos que traten temas relacionados con cuestionarios, preguntas, categorías y la configuración de las rutas realizadas por los usuarios.

Con la estructura de los módulos clara, y sabiendo qué funciones realiza cada uno de ellos, es necesario seguir estructurando la aplicación de manera que esta sea lo más modular posible. Por cada módulo que contenga archivos se crea una carpeta, con lo que existe una única carpeta para el "*Módulo de comunicación*", ya que este no tiene división alguna, y tanto para el "*Módulo de coordinación*" como para el "*Módulo de gestión*" se generan tres carpetas, una por cada una de las divisiones. En concreto, las tres carpetas relacionadas con el módulo de gestión, al ser las que solicitan los datos a la base de datos, se llamarán "*data*".

Al igual que todos los módulos o librerías de terceros están almacenadas en *node_modules*, para englobar las siete carpetas o módulos propios generados, se crea una carpeta llamada *walkability_modules*. En la *Ilustración 18* se puede visualizar la estructura de la carpeta *walkability_modules*.

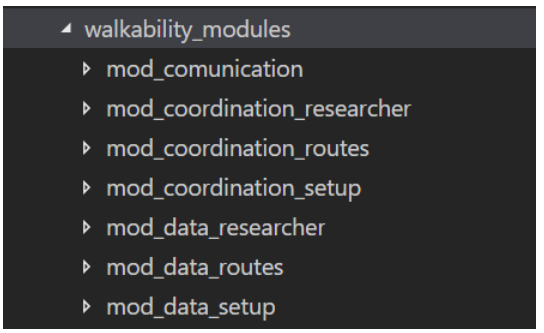


Ilustración 18 - Carpetas para módulos de Walkability Analyzer

Una vez establecida la estructura de carpetas para los módulos propios del sistema, también es necesario crear otros elementos antes de empezar a desarrollar. Por un lado, se crea, a la altura del archivo *app.js*, un fichero llamado *config.js*, en el cual se almacena, mediante un objeto en formato JSON, información estática del proyecto que puede ser accesible desde cualquier fichero del sistema. Por ejemplo, en este fichero se almacena la información necesaria para realizar las conexiones a la base de datos o la frase mediante la cual se encriptan las contraseñas de los usuarios que se registran en el sistema.

Por otro lado, se crea una carpeta, a la altura de *walkability_modules,* denominada *work_tools* que almacenará los ficheros con funcionalidades o métodos comunes para todas las partes del proyecto. Por ejemplo, una función que compruebe si una variable es un número entero puede resultar útil para diferentes métodos del proyecto, por ello se almacenará en un script dentro de esta carpeta. Esta carpeta permite en gran medida la reutilización de código. En la *Ilustración 19* se puede visualizar la estructura completa de Walkability Analyzer.

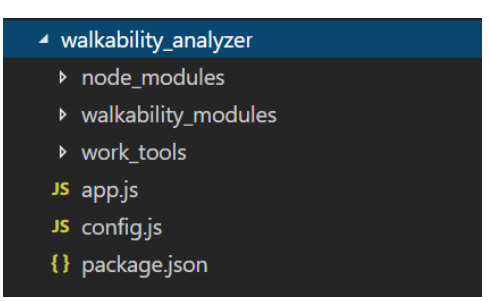


Ilustración 19 - Estructura completa de Walkability Analyzer

Para el desarrollo de la parte back-end de la página web solo es necesaria la instalación de dos librerías adicionales a las que ya vienen incluidas en la carpeta *node_modules* por defecto. La primera librería es *node-mysql*, necesaria ya que Walkability Analyzer posee una base de datos a la que hay que conectarse mediante NodeJS, por lo que su librería debe de estar instalada en el sistema. Para ello, desde la consola se ejecuta el siguiente comando:

```
npm install mysql
```

Además, Walkability Analyzer, como ya se ha comentado anteriormente, se vale de *Tokens* para controlar la autentificación de los usuarios en el sistema. La construcción de estos *Tokens* puede realizarse de varias maneras, desde concatenando diferentes valores o generando números aleatorios de mayor o menor longitud, hasta utilizando librerías externas. En este caso, para construir el *Token* se utiliza la librería *jsonwebtoken*. Para instalarla, se ejecuta en la consola el siguiente comando:

```
npm install jsonwebtoken
```

6.1.2. Rutas y proceso de comunicación

Dentro del back-end, como ya se ha explicado anteriormente, existe un proceso de comunicación entre los ficheros de manera que se pueda gestionar las peticiones entrantes de la manera más eficiente posible. Para que este proceso se pueda llevar acabo es necesario, en primer lugar, definir las rutas del sistema en un fichero de la aplicación al que se acceda en los primeros instantes, es decir, al primer fichero que se ejecuta en el momento en que entra una petición al sistema. Esto es así ya que, si las rutas se configuran en otro archivo, las peticiones entrantes no serán redirigidas al fichero que se encarga de tratarlas, por lo que se producirá un error en la ejecución del proceso. En el caso de Walkability Analyzer este fichero es *app.js*. En el siguiente fragmento se pueden visualizar las rutas declaradas en el fichero *app.js* de la aplicación.

```
// Módulos de comunicación
var comunication_auth = require('path/comunication_auth');
var comunication_api = require('path/comunication_api');
var comunication_category = require('path comunication_category');
var comunication_route = require('path/comunication_route');
//Rutas
app.use('/auth', comunication_auth);
app.use('/api', comunication_api);
app.use('/api/category', comunication_category);
app.use('/api/route', comunication_route);
```

Como se puede apreciar en el fragmento anterior, en la parte inferior se encuentran definidas las rutas al back-end, las cuales, al igual que ocurre con los módulos, están agrupadas por funcionalidad del sistema. Por ejemplo, todas las peticiones que estén relacionadas con las categorías poseerán, como inicio de ruta, */api/category*, o, por el contrario, todas las solicitudes relacionadas con las rutas, comenzarán por */api/route*.

Por otro lado, cada definición de ruta va acompañada por otro parámetro, el cual, apunta a una variable creada en la parte superior de la ilustración. Esta variable actúa como controlador y es la que se encarga de acceder a los recursos del sistema. Esta segunda variable tiene como contenido un *path,* o ruta, que apunta al fichero donde se tratan ese tipo de peticiones. En este caso, y debido a la forma de dividir los módulos, se puede apreciar que cada uno de los *paths* apunta a un fichero situado en el módulo de comunicación, la primera capa del sistema en la que se tratan las solicitudes entrantes. Por ejemplo, *app.use('/auth, comunication_auth)* establece que, al entrar una ruta cuyo comienzo sea */auth*, esta se redireccionará al *path* que está almacenado en la variable *comunication_auth*.

Analizando más profundamente el fragmento se puede apreciar que todas las rutas del sistema, salvo una de ellas, poseen una parte en común, la primera, la parte correspondiente a */api*. Además, existe una ruta que únicamente tiene este elemento como ruta, y es justamente en esta donde se establece el *Middleware* de la aplicación, es decir, la ruta donde se encuentra el fichero que actúa como *Middleware*.

Middleware

Un *Middleware* es un software que se sitúa entre un sistema y las partes que se ejecutan en él, en este caso, los módulos. Básicamente funciona como una capa oculta que permite, entre otras cosas, la administración y comprobación de datos. En este caso, el *Middleware* de la aplicación, situado en el fichero *comunication_api*, es la parte encargada de comprobar que el *Token* que es recibido mediante la cabecera de la petición entrante, corresponde realmente al usuario que ha iniciado sesión en el sistema, y en el caso de que esto sea cierto, comprueba que la validez de ese *Token* no haya expirado. Si esto se cumple, el propio *Middleware* actualiza la validez del mismo y cede el control a la siguiente parte de la ruta. En cambio, la ruta */auth* no pasa por el *Middleware* ya que corresponde a la solicitud de inicio de sesión en el sistema, por lo que en ese momento no se envía ningún *Token* en la cabecera de la petición. Por otro lado, si el Middleware dicta que la petición no es válida, el sistema responde directamente indicando que la solicitud no es correcta, por lo que no se tiene acceso a los recursos del sistema.

Por ejemplo, en el caso de la ruta */api/researcher*, una vez que se ha realizado el control por el fichero que se especifica en el *path* correspondiente a la ruta */api*, y la petición es válida, la solicitud pasa al *path* de la ruta completa */api/researcher,* es decir, al archivo *comunication_researcher*. Cuando el proceso de control se produce, y finaliza correctamente, la petición se redirecciona a la primera capa del sistema, el "*Módulo de comunicación*".

Módulo de comunicación

En cada módulo del sistema las funciones o métodos se construyen de distinta manera. Esto se debe a que la capa superior interactúa directamente con las solicitudes que se envían mediante la red, por lo que estas funciones deben declararse de manera que se pueda trabajar con las rutas de las peticiones entrantes. Por ello, en el "*Módulo de comunicación*", en primer lugar, las funciones son métodos anónimos, ya que carecen de nombre alguno. En concreto, los métodos son denominados métodos de ruta ya que se deriva de uno de los métodos HTTP y se adjunta a una instancia de la clase *express*. Para acceder a ellos se utiliza la ruta con la que llegan al sistema y el tipo de petición de que se trata, teniendo en cuenta que no puede existir dos peticiones similares a la misma ruta con el mismo tipo de petición, es decir, no pueden existir dos peticiones GET a la misma ruta con el mismo número de parámetros. En el siguiente fragmento de código se puede visualizar un método de ruta presente en el "*Módulo de comunicación*".

En el método de ruta se usa la clase *Router* de la librería *express,* que permite los redireccionamientos de ruta, junto con el tipo de petición que se trata en esa función, en este caso GET. Posteriormente, entre comillas se aplica la ruta, que es independiente de la ruta que se analiza en el fichero *app.js*. Por ejemplo, en *app.js*, se define que para llegar a este fichero la ruta debe empezar por */api/question*, pero si la ruta de la solicitud entrante es */api/question/example*, en el método únicamente habría que poner la parte de la ruta todavía no analizada, es decir */example,* o en caso de no quedar partes de la ruta sin analizar, únicamente se pondría una barra *'/'*.

```
//Obtener todas las preguntas
comunication_question.get('/', function (req, res, next) {
        coordination_question.GetAllQuestions()
            .then(function (data) { })
            .catch(function (error) { }); }
);
```

Uno de los problemas que se han encontrado en la realización del back-end reside en que la forma de comunicación de NodeJS es asíncrona, por lo que esto dificulta el hecho de tratar con datos. En funciones básicas no existe problema alguno, pero si lo que se busca es obtener unos datos, para posteriormente realizar acciones con ellos, resulta complicado. Esto es así porque al realizar la petición el código no espera a obtener una respuesta, sino que continúa con su ejecución, por lo que las acciones se empiezan a realizar sin dato alguno y se generan errores. Para solucionar este problema se han barajado dos posibles soluciones: utilizar "*Promises*" o utilizar el método "*async-await*".

En definición las dos alternativas funcionan de la misma manera, la función a la que se llama, la cual tiene que devolver los datos solicitados, devuelve una promesa (*return new Promise*). Para devolver los datos que se solicitan dentro de la promesa se realiza *resolve(datos),* en el caso de que no se haya producido ningún error, o *reject(datos)*, en el caso de que este sí que se produzca. A partir de este momento es donde se produce la diferencia entre los métodos citados anteriormente.

Por un lado, para recoger los datos que se devuelven con la promesa, el método conocido como "*Promises*" utiliza un *then* y un *catch,* que funcionan exactamente igual que las excepciones. Si la promesa se ha resuelto con un *resolve* irá por el *then* del método que la llama, en cambio, si ha ido mal, esta entrará por el *catch*. La consecuencia de esto es que, en el caso en el que se tengan que solicitar datos en diferentes funciones, se tienen que anidar las promesas generando gran cantidad de código. Este método para solucionar la asincronía se puede ver aplicado en el fragmento de código anterior. En ese caso, sería la función *GetAllQuestions()* la que en su interior devolvería la promesa junto con los datos.

Dado que en el "*Módulo de comunicación*" únicamente se hace la llamada a una función del módulo inferior, el código generado con el método "*Promises*" no resulta tan grande. Además, el control de errores de este método aporta gran estabilidad al sistema, ya que las peticiones que se han resuelto correctamente se diferencian de las que poseen algún error, de manera que, al entrar por el *then* o por el *catch*, se permite construir la respuesta a la petición de distinta manera en función de si la solicitud es o no correcta. Por ello, la solución al problema de asincronía en el "*Módulo de comunicación*" se resolverá con el método "*Promises*".

Por otro lado, el método "*async-await*" soluciona el problema de la cantidad de código, ya que, para recoger la promesa que contiene los datos no es necesario utilizar una estructura determinada si no que, únicamente es necesario, antes de llamar al método que devuelve la promesa, poner la palabra *await*, y en la función principal, en vez de definir la función como una *function,* se crea una *async function.* Esta forma de solucionar el problema se podrá visualizar al analizar el "*Módulo de coordinación*".

Módulo de coordinación

En el "*Módulo de coordinación*" se gestiona la solicitud que ha sido recibida en el "*Módulo de comunicación*", y se obtienen los datos necesarios o se realizan las acciones que sean requeridas. Para ello, en ocasiones, es necesario tratar con distintas divisiones del "*Módulo de gestión*". Por ejemplo, puede ser posible que para resolver una petición haya que obtener los cuestionarios, para posteriormente obtener las preguntas de cada uno. Todo ello se gestionará dentro del mismo método del "*Módulo de coordinación*" utilizando varias de las divisiones del módulo inferior.

Por ello, al tratar y manejar tanto datos como diferentes acciones con los mismos, en este módulo también se presenta el problema de la asincronía, a excepción de que la solución adoptada no ha sido la misma que en el apartado anterior. Debido a que en este módulo, al tratar con datos, sí se origina más código, utilizar el método "*Promises*" suponía aumentar en ocasiones en gran medida las líneas de código, generando además código más difícil de entender. Por ello, se ha optado por utilizar el método "*async-await*" que se puede visualizar en el siguiente fragmento de código.

```
//Obtener todas las categorias
exports.GetAllCategories = function () {
        return new Promise(async function (resolve, reject) {
                try {
                        let categories = await CategoryModel.getAllCategories();
                        resolve(categories);
                } catch (error) { reject(5001); }
        });
};
```

Módulo de gestión

El "*Módulo de gestión*" es el nivel más bajo en el proceso de comunicación, y se encarga de realizar las llamadas a la base de datos para obtener la información, o realizar las acciones, que el "*Módulo de coordinación*" necesita para gestionar la petición entrante. Una vez se han realizado las acciones oportunas, el "*Módulo de gestión*" devuelve el resultado al módulo superior, de manera que su labor ha concluido, al menos, hasta que el "*Módulo de coordinación*" solicite realizar otra acción contra la base de datos.

En un principio, la implementación de esta capa parecía sencilla, ya que únicamente consiste en realizar las peticiones correspondientes a la base de datos, y posteriormente, devolver el resultado de dichas acciones a la capa superior. En cambio, ha surgido un error inesperado que ha supuesto un problema, en concreto, con la conexión a la base de datos.

En Walkability Analyzer, en un principio, la conexión a la base de datos se creaba una única vez, de manera que al lanzar el servidor esta conexión era creada y todas las peticiones utilizaban la misma variable. Esta solución, aunque funciona, posee un problema a medio plazo, y es que dichas conexiones acaban muriendo, y a consecuencia, el sistema entero de la aplicación web deja de funcionar.

Las conexiones de MySQL tienen una caducidad de 8 horas, por ello, si el sistema está ese tiempo sin realizar ninguna petición, la variable caduca y la conexión finaliza, por lo que para poder utilizar el sistema nuevamente es necesario reiniciar el servidor.

Mantener un sistema de esta manera no es viable, por lo que se decidió crear una nueva conexión por cada solicitud a la base de datos y cuando la solicitud se resuelva cerrar dicha conexión. Esta solución, finalmente, no sólo no es correcta, sino que no funciona, ya que al almacenar la conexión en la misma instancia sigue ocurriendo el mismo problema.

Finalmente, la solución a este problema reside en generar un *pool* de conexiones MySQL. Un *pool* de conexiones es una herramienta que tiene abiertas varias conexiones a la base de datos, de manera que, en vez de abrirla directamente, se le pide al *pool* que gestione dicha labor. Cuando se va a realizar una petición a la base de datos, el *pool* coge una de las conexiones que ya tiene abierta, la marca como ocupada y la devuelve al archivo correspondiente del "*Módulo de gestión*" para que pueda realizar la conexión. Cuando finaliza la solicitud, el *pool* no se encarga de cerrarla, sino que la marca nuevamente como libre. Así, el propio *pool* de MySQL se encarga de gestionar las conexiones de manera que el error que ocurría se soluciona correctamente.

Respuestas y códigos de error

Una vez que el proceso de comunicación entre las diferentes capas ha finalizado y los datos necesarios para la respuesta ya se encuentran en el "*Módulo de comunicación*", es necesario devolver la respuesta a la petición entrante. En este punto se ha encontrado un problema respecto a cómo devolver las respuestas, tanto las de las solicitudes que han finalizado correctamente como las que han generado algún error. En concreto, el problema reside en que, debido a la multitud de solicitudes diferentes que se pueden hacer, es inviable que cada respuesta adopte una forma distinta. La solución a este problema es estandarizar el formato de las respuestas, tanto para las que devuelven datos y son correctas como para las incorrectas que devuelven algún tipo de error.

Para desarrollar la función de enviar una respuesta de una forma modular, de manera que pueda ser usada por todos los métodos de ruta, se ha creado un fichero dentro de *work_tools*, el directorio que se utiliza para herramientas comunes y reutilización de código. Este fichero, llamado *response.js* se encarga de tratar todas las respuestas y códigos de error para darles el formato adecuado.

En el "*Módulo de comunicación*", tal y como se puede visualizar en el siguiente fragmento de código, cuando la solicitud se ha gestionado de manera correcta, se realiza una llamada a un método del fichero *response* enviando dos parámetros, un código y los datos obtenidos. En cambio, cuando la petición ha ido mal, únicamente se envía el código de error, es decir, el primer parámetro. Esto se debe a que, este método de la clase *response,* analiza el número de parámetros que le llegan para saber si hay algún dato que tiene que formatear y mandar en el *body* de la respuesta.

```
.then(function (data) {
      let respond = response.getResponse(2001, data);
      res.status(respond.status).send(respond);
})
.catch(function (error) {
      let respond = response.getResponse(error);
      res.status(respond.status).send(respond);
});
```

Una vez que se ha desarrollado la parte del método que se encarga de comprobar cuándo es necesario enviar datos en la respuesta, para poder darles así el formato adecuado para su envío, es necesario saber cómo gestionar los códigos de error y el *status,* o código numérico*,* de las respuestas.

Para estandarizar los códigos de error internos de la aplicación, así como los códigos de error conocidos de las respuestas, se ha construido un objeto JSON que actúa de plantilla para cada mensaje que se genere. De esta manera, cuando en la aplicación se genere un código de error interno, se puede buscar exactamente a qué status o código general de respuesta corresponde y cuál es el mensaje de texto que lo acompaña, y así, se puede completar el proceso de generar la respuesta. La plantilla JSON, que se puede visualizar en el fragmento de código siguiente, se compone de las siguientes partes:

- ***status***: El status o código genérico corresponde a los códigos conocidos como respuestas del servidor. En concreto, para el desarrollo de este proyecto se ha trabajado únicamente con seis de estos códigos genéricos:

- ***code***: El *code* corresponde al código interno de la aplicación, que pertenecerá a un único *status*. El código interno especifica qué es lo que ha fallado.

- ***msg***: El mensaje que se enviará en la respuesta junto con el código interno generado y el *status* o código genérico al que corresponde.

```
{"status": generateStatus, "code": generateCode, "msg": generateMsg, "body": data}
{status:200, code:2001, msg:'"Correcto"'},
{status:400, code:4001, msg:'"Error: Parámetros recibidos incorrectos"'}
```

Por otro lado, a la hora de utilizar los códigos de error internos de la aplicación se ha tenido en cuenta el *status* al que pertenece dicho error. Analizando el fragmento anterior se puede comprobar que para el *status* 200, el primer código de error es el 2001, con lo que el siguiente mensaje perteneciente a ese *status* será el 2002 y así sucesivamente. Para desarrollar las respuestas de Walkability Analyzer se ha trabajado con los siguientes *status*:

- ***200 (OK)*:** Indica que todo se ha realizado de manera correcta y el servidor responde con un OK.

- ***400 (Bad Request)***: Los datos enviados por el cliente no son correctos, en concreto, el JSON de la solicitud no está correcto o faltan datos.

- ***401 (Unauthorized)***: Fallos de autorización relacionados con el *Token*, validez o caducidad del mismo, inicio de sesión, registro o petición de datos.

- ***403 (Forbidden)***: Error a la hora de realizar una acción sin tener los permios para ello. El servidor reconoce el método, pero el usuario no tiene acceso para ejecutarlo.

- ***404 (Not Found)***: La URL no es correcta o no se encuentra el método al llamarlo.

- ***500 (Internal Server Error)***: Error interno del servidor.

Finalmente, una vez que la respuesta se ha generado, esta se envía de nuevo al cliente de manera que el proceso de comunicación se cierra hasta que se produzca una nueva solicitud.

6.1.3. Gestión de rutas geográficas

Gran parte de la dificultad de desarrollar el back-end reside en gestionar la multitud de filtros que se pueden aplicar a las rutas geográficas almacenadas en la base de datos, así como obtener toda la información almacenada de dichas rutas. En concreto, y para poder explicar cómo se ha desarrollado esta funcionalidad, se comenzará por detallar los filtros que se pueden aplicar a dichas rutas:

- **Tipo de ruta:** Dentro del tipo de ruta existen dos opciones, "Rutas valoradas", por el usuario de la aplicación móvil que ha realizado la ruta, y "Rutas sin valorar". Si el usuario de la aplicación web desea filtrar por rutas valoradas, podrá seleccionar la categoría de dicha ruta, es decir, el motivo por el que se ha realizado dicha ruta, y la versión del cuestionario con la que se realizó dicha valoración. Por ejemplo, "Rutas valoradas" de la categoría "Ir de compras o hacer recados" valorada con el "Cuestionario 2".

- **Horario de la ruta:** Dentro del horario de la ruta se permite filtrar por rutas realizadas en horario diurno o rutas realizadas en horario nocturno.

- **Fecha de inicio y fecha de fin:** Mediante este filtro se puede acotar las fechas para que se muestren únicamente las rutas realizadas entre dichas fechas.

- **Velocidad de la ruta:** Al aplicar este filtro se permite al usuario acotar la velocidad de desplazamiento media a la que se ha movido el usuario que ha realizado la ruta. Gracias a este filtro se permite distinguir, por ejemplo, un paseo relajado de una ruta a marcha ligera.

- **Género**: Este filtro permite distinguir entre rutas realizadas por hombres y por mujeres.

- **Edad**: Mediante este filtro se puede acotar la edad del usuario que ha realizado la ruta. Este filtro es útil para los miembros del grupo CRIM-AP ya que permite ver, por ejemplo, qué rutas han sido realizadas por la gente joven en horario nocturno, para poder así detectar posibles zonas inseguras por donde no pase nadie.

- **Usuarios con movilidad reducida**: Uno de las finalidades de esta aplicación es detectar las zonas que no están adaptadas para todos los usuarios y dificultan la movilidad. Debido a esto es necesario aplicar un filtro para detectar qué zonas de la ciudad son las más utilizadas por la gente que presenta problemas de movilidad.

- **Municipio de inicio y municipio de fin**: Mediante este filtro se puede concretar entre qué municipios, pudiendo ser el mismo, se desean visualizar las rutas de los usuarios.

- **Punto de inicio y punto de fin**: En concreto, mediante la latitud, la longitud y el radio de circunferencia de cada uno de los puntos, se permite obtener las rutas que inician o finalizan dentro de esas zonas acotadas.

- **Punto de ruta**: Mediante la latitud, la longitud, y el radio de un punto, se filtra por las rutas que han pasado por algún punto dentro de ese sector.

Una vez definidos todos los filtros que se pueden aplicar para obtener las rutas, se procede a explicar la problemática de los mismos y como se han solucionado los problemas encontrados.

Filtros aplicados a rutas geográficas

En primer lugar, debido al gran número de filtros disponibles, y la complejidad que supone aplicar algunos de ellos, se ha visto necesario realizar este proceso de manera modular, ya que realizarlo en una misma función requería gran cantidad de código. Esto genera que el código sea difícil de entender, o peor aún, difícil de corregir en caso de que se tenga que cambiar o mejorar alguno de los filtros. Por ello, una de las primeras decisiones en cuando al desarrollo de los filtros es tratar cada uno por separado, de manera que se vaya así generando la condición de la consulta que se ha de aplicar contra la base de datos.

Hay filtros que pueden estar aplicados o no, por ejemplo, el usuario igual desea obtener únicamente las rutas realizadas en horario diurno, pero puede ser también que no desee filtrar por horario. Por ello, dentro de cada una de las funciones donde se analizan los filtros es necesario tener en cuenta este hecho, ya que esta característica no debería formar parte de la condición de la consulta si el usuario no desea realizar ningún filtro sobre ella. Para ello, dentro de los valores de la petición se indican qué filtros se desean aplicar, se especifica un valor por filtro que indica si este está o no aplicado, y en caso de que lo esté, qué valor toma.

Por otro lado, otro de los mayores problemas reside en que, no sólo hay que aplicar dichas características a la condición de la consulta, sino que, al estar los datos distribuidos en diferentes tablas, también hay que realizar las uniones necesarias según los filtros seleccionados para tener acceso a los datos. Por ejemplo, si no se aplica ningún filtro relacionado con el usuario, como su género o su edad, no es necesaria realizar una unión con dicha tabla, ya que no se aplicará ninguna condición a la consulta que incluya esas características. Por ello, en cada función donde se analiza un filtro específico no sólo se construye la condición que se aplica a la consulta, sino que también se construye la propia consulta, para indicar así a la base de datos en qué tablas debe realizar las búsquedas y mediante qué atributos se unen dichas tablas.

Debido a cómo están los datos almacenados en la base de datos, hay un último problema que se ha generado en cuanto a la obtención de los datos en base a filtros, y es que, algunos de los datos no se encuentran en la base de datos. Por ejemplo, en la base de datos no está disponible un campo de velocidad de la ruta, sino que este dato se puede calcular mediante el tiempo de ruta y la distancia recorrida. Por ello, esta característica no se analiza en la propia consulta contra la base de datos, sino que las rutas obtenidas son posteriormente analizadas para ver cuáles cumplen realmente con la velocidad establecida en el filtro, y sólo las que sean válidas serán enviadas al usuario en la respuesta.

Esta solución no sólo se aplica con la velocidad, sino que también es necesario aplicarla con las rutas filtradas por puntos, tanto punto de inicio y punto de fin como para el filtro de punto de ruta. En el caso de la velocidad, cuando se poseen todas las rutas que cumplen los demás filtros, se recorren dichas rutas analizando la velocidad de cada una, para almacenar y devolver únicamente las que se ajusten a los parámetros. En cambio, con los puntos de ruta ocurre algo más complejo, y es que, medir la distancia entre dos puntos geográficos es algo más difícil que calcular la velocidad de ruta.

Para solventar este problema es necesario analizar todos los puntos de todas las rutas que han pasado la primera selección, o al menos todos los puntos hasta encontrar un punto válido. Para calcular si los puntos de una ruta pertenecen o no a la zona del mapa marcada por el usuario es necesario conocer la distancia entre los dos puntos, el punto marcado como centro de la circunferencia y el punto de la ruta que se está analizando. Para calcular esto se ha creado una función que, valiéndose del radio de la Tierra y de unas fórmulas matemáticas, calculando el seno y el coseno de la latitud y longitud del punto a analizar, permite concretar si ese punto es válido o no, es decir, si pertenece o no a la zona

marcada. En caso de ser así esa ruta se da por correcta para los filtros marcados, con lo que es enviada junto con la respuesta del servidor.

Una vez que el primer problema respecto a las rutas se ha solucionado, y ya se puede aplicar todo el catálogo de filtros de manera modular para extraer las rutas que sí se ajustan a las características solicitadas, es necesario analizar el segundo problema: la descarga de todos los datos asociados a una ruta en un formato compatible con software SPSS, por ejemplo *csv*, y un formato compatible con sistemas QGIS.

Descarga de rutas en formato compatible con SPSS

Una ruta posee mucha información, desde todos los puntos que la conforman hasta información asociada con el usuario que la ha realizado. Además, también se considera información importante si la ruta ha sido o no valorada, y en caso de que sí haya sido valorada, sobre qué cuestionario se ha realizado la encuesta, qué preguntas lo componen, qué ha respondido a cada pregunta, y demás información que los miembros del grupo CRIM-AP consideran necesaria a la hora de extraer información.

El problema que reside en descargar esta información es que todas las rutas se tienen que adaptar al mismo formato, donde todas las líneas del fichero descargado deben tener el mismo número de columnas. Pero esto, en la base de datos no es así, ya que las rutas que no han sido valoradas no poseen información acerca de ningún cuestionario, ni poseen respuestas a las preguntas, por ello, se necesita crear una estructura común que se adecúe a todo tipo de rutas.

Para realizar esto y solventar este problema, se ha decidido fragmentar la obtención de datos en diferentes pasos. En primer lugar, se extrae toda la información común almacenada de cada ruta, por ejemplo, toda ruta ha sido realizada por un usuario, por lo que esa información siempre existirá en la base de datos. Los datos comunes de los que se ha extraído información están formados por el identificador de la ruta, el identificador del usuario, la fecha de realización de la ruta, el género y municipio de procedencia del usuario, si posee o no movilidad reducida, la distancia y la duración de la ruta realizada, el municipio de inicio y de fin de la ruta y otros valores que detallan más en profundidad cuándo se realizó dicha ruta.

Una vez se ha obtenido toda la información común, se obtienen todos los cuestionarios y preguntas que han realizado los usuarios sobre esas rutas, en caso de estar valoradas. Es decir, todas las posibilidades de cuestionarios y preguntas que se encuentran en el sistema. De esta manera se puede completar la cabecera del fichero. Hay que destacar que tanto las preguntas como los cuestionarios se pondrán en la cabecera mediante identificadores, ya que, si las preguntas poseen ciertos caracteres, como por ejemplo tildes, pueden generar problemas a los investigadores a la hora de realizar los estudios con herramientas específicas. Para ello, los identificadores se formarán de la siguiente manera:

- En caso de que la pregunta del cuestionario sea una pregunta con respuesta única, el código será: *C + "Identificador del cuestionario" + P + "Identificador de pregunta"*. En el campo correspondiente del fichero se almacenará, en caso de que se trate de una ruta valorada, el identificador de la respuesta del usuario a esa pregunta de ese cuestionario en concreto.

- Por otro lado, si la pregunta permite la respuesta múltiple, por cada posible respuesta múltiple se seguirá el siguiente código: *C + "Identificador del cuestionario" + P + "Identificador de la pregunta" + R + "Identificador de la posible respuesta múltiple"*. En este caso, dado que al ser respuestas múltiples lo que se está analizando, y no preguntas en sí, únicamente pueden estar marcadas o no, por lo que

en los campos correspondientes se almacenará en el fichero, en caso de que esté marcada esa respuesta, una X.

Cuando ya se posee la estructura completa del fichero a rellenar, únicamente se analiza cada ruta para construir el objeto completo que la compone, es decir, toda su información. En el caso de que tenga respondido ese cuestionario presente en la cabecera, en cada pregunta que lo compone se pone la respuesta del usuario, o si se trata de una pregunta con respuesta múltiple, se marca mediante un signo; en cambio, si no se encuentra respuesta alguna ese campo queda vacío. En la *Ilustración 20* se muestra parte de la estructura del fichero, donde, en la parte derecha de la cabecera se pueden apreciar los códigos seguidos para enumerar los cuestionarios y sus preguntas.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Ruta	Usuario	Fecha nacimi	Genero	Municipio pro	Movilidad red	Duracio	Distancia recc	Municipio inic	Municipio fin	Copia	Configuracion	Categ	Dia	Hora	Cuantas rutas	Cuestionario	C1P1	C1P2	C1P3	C1P4	C1P5
2	1747c7!	1747c755	1984	1	Bilbao	0	31	566.248	Bilbao	Bilbao		1		####	###	0						
3	1747c7!	1747c755	1984	1	Bilbao	0	18	696.949	Arenal / San N	Pedro Martinez	Artol	1		####	###	0						
4	1747c7!	1747c755	1984	1	Bilbao	0	26	682.135	Pedro Martin	Autonomia 2		1		####	###	0						
5	1747c7!	1747c755	1984	1	Bilbao	0	18	609.894	Amezola	Autonomia 1		1		####	###	0						
6	1747c7!	1747c755	1984	1	Bilbao	0	375	1077.14	Derio	Zamudio		1		####	###	0						
7	469b0c	469b0c24	1962	1	Errenteria	0	23	631.062	Meipi	Donostia-San Sebasti		1		####	###	0						
8	469b0c	469b0c24	1962	1	Errenteria	0	45	565.422	Donostia-San	Lezo		1		####	###	0						
9	541b2c	541b2c3f	1996	1	Pradoluengo	0	9	686.586	San Sebastian	Zumalakarregi 10		1		####	###	0						
10	541b2c	541b2c3f	1996	1	Pradoluengo	0	14	584.283	San Sebastian	San Sebastian		1		####	###	0						
11	5d1657	5d16574(	1996	1	Herce	0	6	505.247				1		####	###	0						
12	5d1657	5d16574(	1996	1	Herce	0	23	517.008				1		####	###	0						
13	5d1657	5d16574(	1996	1	Herce	0	11	501.485				1		####	###	0						
14	5d1657	5d16574(	1996	1	Herce	0	14	1108.04	San Sebastian	San Sebastian		1		####	###	0						
15	5d1657	5d16574(	1996	1	Herce	0	10	1147.92				1		####	###	0						
16	5d1657	5d16574(	1996	1	Herce	0	15	913.981				1		####	###	0						
17	5d1657	5d16574(	1996	1	Herce	0	35	564.893				1		####	###	0						
18	701791	7017918	1996	0	Errenteria	0	44	549.078	Renteria	Errenteria II		1		####	###	0						
19	701791	7017918	1996	0	Errenteria	0	25	595.091				1		####	###	0						
20	701791	7017918	1996	0	Errenteria	0	15	609.946				1		####	###	0						
21	95a99b	95a99bf5	1988	0	Errenteria	0	16	607.825	San Sebastian	Mirakontxa		1		####	###	0						
22	95a99b	95a99bf5	1988	0	Errenteria	0	59	554.323	San Sebastian	San Sebastian		1	2	####	###	0	1	1	6	12	16	
23	ae98ac	ae98acc	1996	1	Tolosa	0	271	537.823				1		####	###	0						

Ilustración 20 - Información de rutas en formato compatible con SPSS

El fichero de la información de las rutas en formato *csv*, compatible con herramientas SPSS, al descargarse, va acompañado de otro fichero en formato de texto con toda la información necesaria para que los investigadores puedan, mediante los códigos identificadores, obtener el texto de la pregunta realizada al usuario, así como de la respuesta.

Descarga de rutas en formato compatible con QGIS

Las herramientas QGIS permiten reconstruir una ruta en el mapa mediante los puntos geográficos que ha seguido dicha ruta. Por ello, los miembros del grupo CRIM-AP desean poder obtener estos puntos para analizar las rutas en herramientas más específicas de análisis. Para poder descargar las coordenadas de todas las rutas se ha debido de ordenar los puntos de cada ruta seguida, y posteriormente, construir un objeto idéntico para todas de manera que a la hora de plasmar dicha información en un fichero quede de forma similar. Para ello, en el objeto se almacenan cuatro campos, el identificador de la ruta, el orden del punto, y su latitud y longitud. Debido a esto una misma línea ocupará tantas líneas en el fichero como puntos posea, y en cada una de ellas se especificará su coordenada geográfica. En la *Ilustración 21* se puede visualizar un fichero en formato compatible con software QGIS.

	A	B	C	D
1	Identificador	Orden	Latitud	Longitud
2	1747c755756	1	432.637	-295.093
3	1747c755756	2	432.637	-295.096
4	1747c755756	3	43.263	-294.772
5	1747c755756	4	432.626	-294.742
6	1747c755756	5	432.626	-294.621
7	1747c755756	6	43.262	-294.471
8	1747c755756	7	432.618	-294.404
9	1747c755756	8	432.614	-294.235
10	1747c755756	9	432.614	-294.235

Ilustración 21 - Información de rutas en formato compatible con QGIS

6.1.4. Seguridad en la parte servidor

Toda aplicación web debería poseer un mínimo de seguridad de manera que se pueda evitar, en la manera de lo posible, todo ataque que pueda recibir el sistema o la base de datos. Por ello, Walkability Analyzer posee diferentes herramientas que aportan seguridad al sistema, por ejemplo, la utilización de *Tokens*, la validación de datos en el "*Módulo de comunicación*", la presencia de roles en el sistema y la seguridad en las contraseñas.

Utilización de *Tokens*

El primer mecanismo de seguridad que se utiliza en el sistema es la utilización de *Tokens*. Mediante el uso de estos, cuando le llega una petición al sistema, este puede comprobar, entre otras cosas, si el usuario posee los permisos necesarios para acceder a ese recurso o si la sesión del usuario está o no activa.

Para controlar el uso de *Tokens*, desde su creación hasta comprobar su validez o actualizar los valores del mismo, se ha creado un fichero en la carpeta *work_tools*, la carpeta de herramientas o funciones compartidas del back-end. En concreto, este fichero, llamado *token.js*, se encarga, en primer lugar, de generar un *Token* en el momento que un usuario inicia sesión en el sistema. El *Token* de seguridad se almacena en la base de datos junto con una fecha y hora de validez. El tiempo de validez de un *Token* en Walkability Analyzer se ha estipulado en 15 minutos, por lo que, si un usuario está esa cantidad de tiempo sin interactuar con el sistema, el *Token* caduca, y a partir de ese momento no se permite realizar ninguna acción más hasta que se haya iniciado sesión nuevamente.

Como se ha explicado anteriormente en el apartado 6.1.2, para construir un *Token* se utilizará la librería *jsonwebtoken*, que permite encriptar y desencriptar la información del mismo. En este sistema, el *Token* está formado por un código o palabra única, el identificador del usuario que ha iniciado sesión en el sistema, el nombre de usuario del mismo, su rol y la fecha y hora del momento en que se genera el *Token*. Así, mediante la encriptación se realiza un intercambio de datos más seguro a través de la red, y se hace más complicado que, en un ataque informático, se pueda adivinar el *Token* de un usuario para acceder al sistema.

En el archivo *token.js*, se construye el *Token,* y, además, se comprueba la validez del qué viaja en la cabecera de las peticiones entrantes en el sistema. La función para comprobar la validez es solicitada por el *Middleware*, la capa que se encarga de validar las peticiones, por lo que, si el *Token* de la petición ha expirado o no es válido, la solicitud no accede al sistema, evitando así posibles ataques. En el caso en el que el Token sí sea válido, se actualiza su momento de expiración, de manera que los 15 minutos de caducidad vuelven a estar disponibles.

Seguridad en el "*Módulo de comunicación*"

En primer lugar, una de las funciones principales del "*Módulo de comunicación*" es comprobar que el usuario tiene acceso al recurso solicitado. Para ello, cada función tiene una lista de roles permitidos, así, cuando una solicitud llega hasta este punto del sistema, se comprueba el rol que viaja en el *Token*, y si este coincide con alguno de los roles permitidos, la petición continua en el sistema. Por el contrario, si el usuario carece de los permisos necesarios, la respuesta se devuelve indicando el error concreto. En Walkability Analyzer únicamente hay dos roles denominados "*Investigador*" y "*Administrador*".

Por otro lado, los datos que se envían en una petición viajan en la cabecera o en el cuerpo de la misma, por lo que, si la petición es válida y llega al "*Módulo de comunicación*", otra de las funciones del mismo es revisar que dichos datos cumplen la estructura correcta y son del tipo requerido. Por ejemplo, si una petición tiene que recibir tres parámetros con un nombre concreto y un tipo concreto, este módulo se encarga de que esto sea así, y si no sucede, la petición es rechazada mediante un error.

Para realizar este control de seguridad, lo primero que se analiza es si los argumentos requeridos están presentes en la petición, y en caso de que sea así, se comprueba que sean del tipo necesario. Es decir, si el parámetro "*nombre*" tiene que ser una cadena de texto, se comprueba que realmente lo sea, y si por el contrario es un número, la petición es denegada. De esta manera se evita que lleguen caracteres no controlados al sistema o que se pueda acceder a la base de datos con parámetros no requeridos.

Gestión de las contraseñas de usuario

En la base de datos no se debería de almacenar información sensible, como, por ejemplo, contraseñas, sin protección alguna sobre ellas. Por ello, Walkability Analyzer incorpora varias herramientas de seguridad que permiten almacenar las contraseñas en la base de datos de una forma segura. Para tratar la seguridad en las contraseñas se ha creado un fichero llamado *securityControl.js* en el directorio *work_tools*.

Para tratar las contraseñas, en primer lugar, al registrar un usuario con su contraseña específica, esta se encripta mediante el algoritmo *MD5*, y posteriormente, en este mismo fichero, se genera una cadena única que se le añade a la contraseña. Esta cadena única se denomina *Salt*, y es un fragmento que se añade a la contraseña del usuario de manera que, en un posible ataque, para poder adivinar la contraseña de un usuario, es necesario adivinar las dos partes de la misma, y no únicamente el fragmento o palabra que es conocida por el usuario. En la base de datos se guarda tanto el conjunto completo encriptado, como la Salt, para así poder comprobar dichos valores en el momento en el que un usuario trate de iniciar sesión.

Cuando un usuario inicia sesión mediante su palabra de seguridad, esta se encripta en *MD5* y se le añade la Salt del usuario presente en la base de datos, si la clave completa coincide con la contraseña total almacenada, entonces el inicio de sesión es correcto y se permite que el usuario acceda al sistema.

6.2. Desarrollo de front-end

En este apartado se analizará cómo crear la estructura del proyecto, la manera en la que se han creado los módulos y componentes que forman el front-end, la funcionalidad de los mismos y los problemas encontrados en el desarrollo de la parte cliente del sistema.

6.2.1. Estructura de la aplicación y puesta en funcionamiento

Para el desarrollo de la parte front-end de la aplicación web, como ya se explicó en el apartado 0, se utiliza Angular, un framework de TypeScript. La puesta en funcionamiento de un proyecto en Angular es sencilla, ya que para ello únicamente hay que ejecutar los siguientes comandos en la consola del sistema. El primero de ellos instala el framework Angular en el equipo de trabajo, y el segundo y último de ellos genera el proyecto del front-end a partir del cual se creará la estructura deseada para este proyecto.

```
npm install -g @angular/cli

ng new walkabilityFront
```

Como resultado de ejecutar los dos comandos anteriores se ha creado el esqueleto base desde donde se empieza a estructurar la parte del front-end de Walkability Analyzer. En concreto, el esqueleto base se puede visualizar en la *Ilustración 22*.

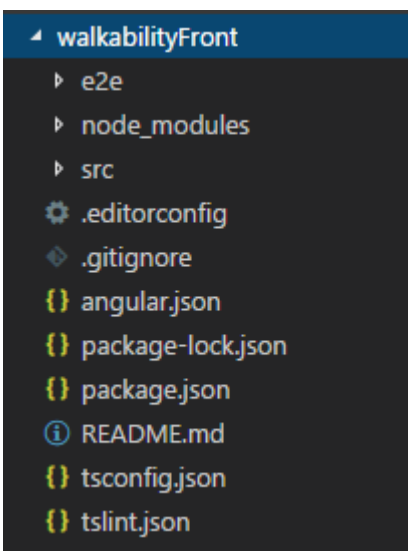


Ilustración 22 - Esqueleto de una aplicación Angular

Los archivos que aparecen en la ilustración anterior, a excepción del archivo *README.md,* son archivos propios de la configuración de Angular, que, aunque para el desarrollo de la aplicación no son importantes, ya que no se desarrolla ninguna funcionalidad nueva en ellos, estos no se pueden eliminar. Esto se debe a que estos contribuyen en el proceso de montar y compilar la aplicación para que pueda ser interpretada por el navegador, por lo que son esenciales para el desarrollo de la interfaz gráfica de Walkability Analyzer.

Por otro lado, en la ilustración aparece la carpeta *node_modules*. Esta carpeta es similar a la carpeta *node_modules* explicada en el back-end, donde se almacenan todos los

módulos externos o librerías que utiliza la aplicación.

El apartado donde se desarrolla la aplicación, donde se crean los módulos propios y componentes de Angular, es en el directorio *src*, en concreto, en el directorio *app* dentro del mismo. El interior del directorio *src* está formado por tres carpetas y diversos archivos, pero para el desarrollo de este proyecto únicamente se usará la carpeta *app*, que es donde se estructura la aplicación, y la carpeta *assets*, donde se almacenan los recursos que utiliza la página web, como, por ejemplo, imágenes. En la *Ilustración 23* se puede ver la estructura básica de la carpeta *app*.

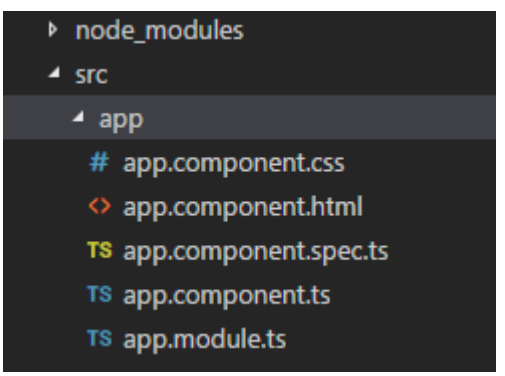


Ilustración 23 - Estructura del básica del directorio App

En la ilustración anterior aparecen todos los ficheros pertenecientes al componente principal. Por ejemplo el fichero *app.module.ts*, uno de los ficheros más importantes de la aplicación web, ya que todo componente que se cree en la aplicación debe estar declarado aquí, de manera que pueda ser encontrado por el navegador donde se ejecute la aplicación.

Un componente de Angular consta de cuatro ficheros que se pueden visualizar en la ilustración anterior:

- **Fichero *component.ts***: Este fichero es el corazón del componente, un fichero en formato TypeScript donde se situará la lógica del componente. Aquí se definen las acciones que se ejecutan al presionar botones, se realizan las llamadas a los servicios o se declaran los textos que se visualizan dentro del componente.

- **Fichero *css***: Este fichero permite colocar estilos gráficos al contenido de ese componente. Los estilos definidos en cada componente únicamente afectaran al diseño gráfico del mismo y no a toda la aplicación.

- **Fichero *html***: En este fichero se define la estructura equivalente a lo que se conoce como "vista" en la creación de páginas web, es decir, la pantalla que visualizan los clientes de la aplicación. Aquí se crea la estructura del componente, por ejemplo, si el componente es una tabla, se crea la estructura de la misma con sus columnas y su cabecera.

- **Fichero *spec.ts***: Un archivo TypeScript que está destinado a realizar pruebas con los componentes. En este caso este fichero no se va a utilizar por lo que no estará presente en ningún componente del sistema.

Por otro lado, un módulo en Angular es uno de los elementos principales mediante el que se puede organizar el código de una aplicación. Un módulo está compuesto al menos por un componente, y al igual que estos, los módulos también deben ser declarados en el fichero *app.module.ts*. Los módulos son utilizados para estructurar el código en fragmentos, evitando así crear los componentes sin orden alguno. Al igual que la aplicación en su

totalidad posee el fichero *app.module.ts,* cada módulo también posee el fichero *modulenombre.module.ts* donde se declararán algunas librerías importantes o constantes que utilizarán todos los componentes de los que consta dicho módulo.

Como se explicó en el apartado 5.2, el front-end del sistema está dividido en tres carpetas principales: *components*, *services* y *modules*. Esta última, concretamente, posee los dos módulos de la aplicación web, *Login* y *ManageRoutes*. Esta estructura de carpetas se puede visualizar en la *Ilustración 24*.

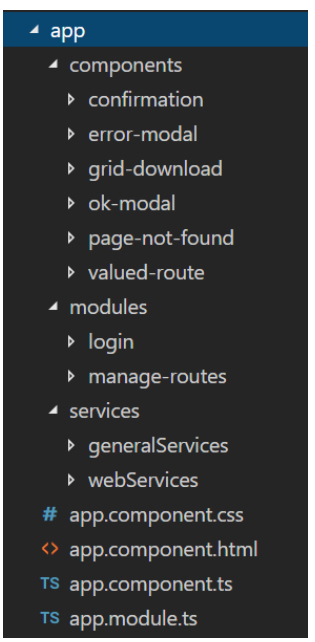


Ilustración 24 - Estructura del front-end en Angular

La estructura utilizada para Walkability Analyzer permite generar una aplicación de forma modular, mediante la separación de componentes y evitando en gran medida la repetición de código.

6.2.2. Funcionamiento y proceso de comunicación

En este apartado, para poder analizar más profundamente el funcionamiento del framework Angular, se procede a detallar algunas partes destacables de cómo se ha desarrollado la parte front-end de Walkability Analyzer, desde cómo se ha gestionado el inicio de sesión hasta como se realizan las peticiones al back-end.

Inicio de sesión en el front-end

Para poder mostrar el formulario de inicio sesión al ejecutar la aplicación web, Angular utiliza un sistema denominado *Routes*, que permite acceder hasta el componente donde se define dicha pantalla del sistema. Para ello, en primer lugar, en el fichero *app.module.ts* es necesario crear un objeto JSON de esta clase que conste de dos variables, el *path* y el componente al que hay que redirigir el sistema cuando aparece dicho *path*. Este objeto, además, se ha de crear en cada fichero *module.ts*, es decir, en cada fichero *module* que se crea al generar un módulo nuevo. Por ejemplo, si un módulo consta de cuatro componentes, el objeto debe tener cada *path* con el que acceder a cada componente. En el fragmento de código siguiente se muestra un breve ejemplo de este objeto:

```
const routes: Routes = [{path: '', component: LoginComponent}]
```

En la aplicación web, al pulsar el botón que permite mostrar en el sistema la pantalla de Gestión de Usuarios, por ejemplo, la ruta de la URL del sistema se redirige a */app-users*, con lo que en el fichero *module.ts* del módulo donde se encuentre dicho componente, debe estar presente la ruta de la URL junto con el componente que gestiona dicha pantalla. En cambio, al ejecutar por primera vez la aplicación todavía no se ha presionado botón alguno, por ello, para ir al componente *LoginComponent*, tal y como se muestra en el fragmento de código anterior, no es necesario ningún *path* especifico en la URL, con lo que al arrancar el sistema el componente de *login* se ejecutará inmediatamente.

Una vez se ha mostrado la pantalla de inicio de sesión mediante la utilización de la clase *Routes* de Angular, el usuario puede iniciar sesión en el sistema introduciendo sus credenciales en el formulario indicado para ello. Y posteriormente, cuando pulsa el botón de "*Iniciar sesión*" el fichero que se encarga de la lógica del componente, *login.component.ts*, recoge los valores que ha introducido en los campos y los envía al servicio *auth.service.ts*, dentro de la carpeta *webServices*, para que envié dichos datos al back-end mediante una petición y comprobar si el inicio de sesión es o no correcto.

Para enviar los datos a un servicio y esperar así la respuesta a la petición enviada es necesario subscribirse a ese servicio, y no llamar a una función como si de un método normal se tratara. Esto se debe, al igual que pasa en el back-end, a la asincronía. Cuando se ejecuta un servicio este llama al back-end, y es necesario que el front-end sepa qué se debe ejecutar una vez se hayan recibido los datos, por ello, se lanza una subscripción al servicio encargado de realizar la petición. Cuando llegue la respuesta el servicio redirigirá la respuesta al método subscrito a dicho servicio, y este método analizará la respuesta en función del código o status que está tenga. En la *Ilustración 25* se puede visualizar el proceso de comunicación con un servicio en Angular.

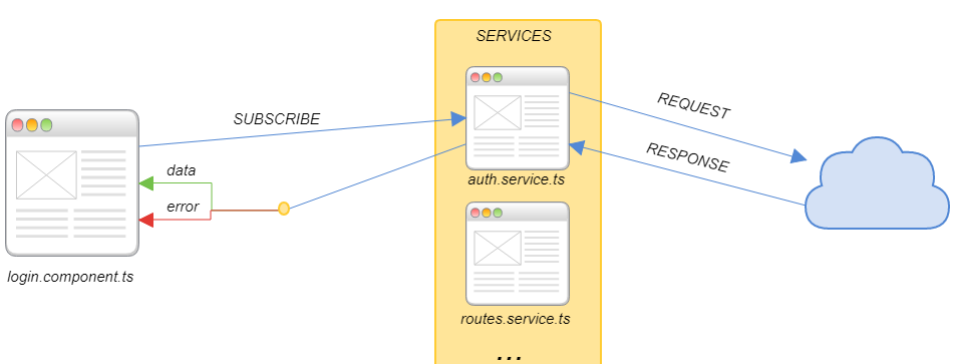


Ilustración 25 - Servicios en Angular

Por defecto, al subscribirse a un servicio, igual que ocurre con el método "*Promises*", se debe crear un *data* y *error*, de manera que las respuestas que poseen un código correcto, por ejemplo, 200, se analizan en el *data*, mientras que las que poseen algún error se redirigen al *error*.

En este caso, al intentar iniciar sesión y subscribirse al servicio, a este se le envían los datos que ha introducido el usuario. El propio servicio se encarga de construir la petición con el formato necesario para que sea aceptada por el back-end, indicar el tipo de solicitud del que se trata, por ejemplo, un POST o un PUT, y de crear el objeto con los datos y enviarlos en el *body* o en la cabecera de la solicitud.

De esta manera, una vez que se la subscripción se ha gestionado, si la respuesta es

correcta, se obtienen los datos que vienen junto con la respuesta. Y estos datos, que incluyen el rol del usuario que ha iniciado sesión, o el *Token* del mismo, son almacenados en la sesión del usuario, la *sessionStorage*. La sesión del usuario es un contexto que está presente en toda aplicación Angular, de manera que cada componente y cada servicio de la aplicación web pueden acceder a las variables almacenadas en dicho contexto. Así, almacenando ahí valores como el *Token*, este se podrá enviar en todas las cabeceras de las peticiones del resto de servicios que se ejecuten, y el *Middleware* del back-end podrá comprobar la validez del mismo. Por otro lado, en el momento en el que el usuario cierre la sesión en el sistema, las variables del *sessionStorage* serán eliminadas para favorecer la seguridad del sistema.

Si el inicio de sesión ha sido correcto se almacena la información del usuario y el sistema se redirige a la pantalla principal de la aplicación web, en este caso la página de gestión de rutas, que es la que se ejecuta una vez se accede al sistema. Para ello se redirige la URL al *path* correspondiente a ese componente presente en el JSON de *Routes*. Por el contrario, si la sesión no se ha iniciado correctamente al usuario se le muestra un mensaje de error explicando el motivo por el cual el proceso es incorrecto. Los mensajes de error de Walkability Analyzer son ejecutados con Bootstrap.

Utilización de Bootstrap

Bootstrap [3] es una librería externa que facilita en gran medida la utilización de Angular, desde mejorar el aspecto de diseño de una aplicación web hasta incluir funcionalidades en la misma. Por ello, en Walkability Analyzer se ha utilizado este módulo en diversos aspectos, desde los mensajes de error que se le muestran al usuario hasta la creación de mensajes pop-up o modales para indicar mensajes de confirmación o error más específicos.

En primer lugar, en el desarrollo de la aplicación web se muestran dos tipos de mensajes de error. Hay mensajes de error que no requieren la subscripción a ningún servicio ni la realización de ningún proceso para que se produzcan, son mensajes de error referentes a información que el usuario no ha introducido correctamente, como, por ejemplo, intentar iniciar sesión sin indicar las credenciales. Estos mensajes de error o de alerta son creados mediante alertas de Bootstrap. En *Ilustración 26* se puede visualizar una Bootstrap Alert.

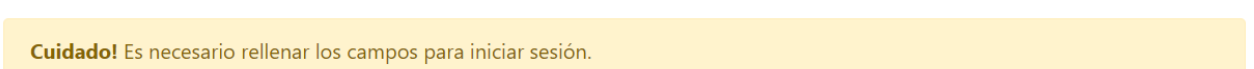


Ilustración 26 - Bootstrap Alert

Estos mensajes de error son activados o desactivados mediante la presencia de condiciones en el fichero *html* del componente, y gracias, además, a la utilización del *Binding*. El *Binding* no es más que conectar la información de las variables que están en el fichero de lógica del componente, el archivo *component.ts*, con el fichero *html* del componente. Así, desde el fichero de la vista, el archivo *html*, se pueden visualizar los valores que toman las variables en el momento en que estas cambian de valor. Una de las grandes ventajas del uso del *Binding* en Angular es la limpieza en el código resultante, ya que no es necesario implementar gran cantidad de eventos ni enviar datos de un sitio a otro.

Los otros mensajes de error presentes en el sistema son generados por las respuestas a las peticiones que llegan con un código de error. Estos mensajes están formados por una ventana pop-up, y pueden ser cerrados por el propio usuario, es decir, constan de botones.

[3] Bootstrap: *https://getbootstrap.com/*

Debido a ello estos mensajes no son únicamente utilizados en el sistema para mostrar mensajes de error, si no que se utilizan para pedir confirmación al usuario. Por ejemplo, cuando un usuario desea eliminar un cuestionario se le muestra un mensaje pop-up preguntándole si realmente desea realizar esta acción, y, mediante los botones, el usuario puede aceptar o cancelar la acción. Estos mensajes, también son realizados usando Bootstrap.

Las ventanas pop-up que se abren también son llamadas *Modals*, o modales. Estas ventanas se llaman desde el fichero donde se sitúa la lógica del componente, y al igual que ocurre con los servicios, también es necesario subscribirse a un modal. Esto es necesario ya que la decisión que tome el usuario en ese modal afectará a la acción que se debe realizar, por ello, es necesario que el método que lanza el modal capture la acción que el usuario ha seleccionado. Por ejemplo, si pulsa "Aceptar" es necesario realizar la acción, pero si cierra el modal no hay que realizar nada.

Los modales se crean de la misma manera que los componentes, por ello, disponen de los mismos ficheros. En la vista del modal se define su estructura y los botones de los que consta, mientras que en la lógica se crean los métodos asociados a los botones que permitan devolver la respuesta del usuario a la función que invoca dicho modal. En Walkability Analyzer se utilizan desde modales sencillos, para comunicar errores y pedir confirmación al usuario, hasta modales más elaborados que a su vez están subscritos a servicios para obtener información del back-end. En la *Ilustración 27* se puede visualizar un modal que está subscrito a un servicio para obtener la valoración de una ruta, y permitir así al usuario eliminar dicha valoración.

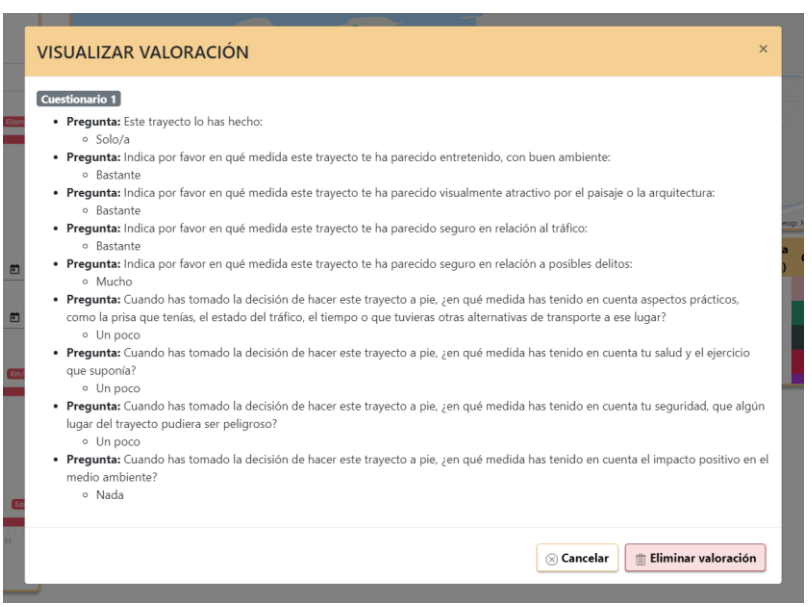


Ilustración 27 - Utilización de Modals en Walkability Analyzer

Además de utilizar Bootstrap para mostrar mensajes de error y modales, también se ha utilizado para dar estilo a diferentes elementos del sistema, por ejemplo, los desplegables presentes en la aplicación web o las tablas realizadas con agGrid [4].

[4] agGrid: *https://www.ag-grid.com/*

Tablas mediante agGrid

En Walkability Analyzer es necesaria la utilización de gran cantidad de tablas. Por ejemplo, las tablas son necesarias para mostrar los cuestionarios, las preguntas, las posibles respuestas a una pregunta, los usuarios, las categorías, las configuraciones o la información de las rutas. Por ello, debido a la gran cantidad de tablas que están presentes en la aplicación web, y a la multitud de diferentes acciones que se han de realizar con dichas tablas, era conveniente utilizar una librería externa que facilite la interacción con dichas tablas. Una de las librerías más conocidas en Angular para la gestión de tablas es agGrid, por ello, es la seleccionada para el desarrollo de las tablas de esta aplicación web.

Las tablas al uso de *html*, las definidas como *table,* que se construyen mediante etiquetas *th* y *td*, no facilitan la interacción del usuario con la tabla. Por ejemplo, para que se ejecute alguna acción al presionar una fila es necesario crear un evento, y, además, no permiten ordenar ni filtrar las filas de la tabla de manera natural. Todas estas características sí son propias de agGrid, y facilitan tanto la construcción de la tabla como la interacción con la misma.

Para la utilización de esta librería externa es necesario definir en el fichero donde está presente la lógica del componente un objeto de la clase *GridOptions*, una clase creada por la propia librería. A este objeto es necesario indicarle tanto las columnas como las filas de la tabla, así como algunos valores propios de agGrid que permiten filtrar, ejecutar acciones al pulsar sobre filas o ajustar las columnas al texto que contiene en su interior. Estas variables son configurables a elección del propio desarrollador, y se seleccionan en función de la complejidad que se le quiera dar a la tabla. Este objeto, mediante la utilización del Binding, es leído por la definición de la tabla en el *html*. En el siguiente fragmento de código se puede ver la etiqueta de la tabla, así como la elección de estilo para las tablas de Walkability Analyzer. El estilo es definido por la opción *class*.

```
<ag-grid-angular class="ag-fresh" [gridOptions]="gridOptions">
</ag-grid-angular>
```

En este caso, las tablas de Walkability Analyzer permiten el filtrado y la ordenación de las filas en función de la columna que se desee, permiten realizar acciones al pulsar sobre una fila en concreto y permiten eliminar o añadir filas a la tabla de manera dinámica por parte del usuario. Por ejemplo, si un usuario añade una pregunta a un cuestionario, esta pregunta aparece en la tabla de manera dinámica, sin necesidad de recargar la página. En la *Ilustración 28* se puede visualizar una tabla realizada con agGrid.

Nº Pregunta	Texto	Horario	Genero	Tipo de respuesta
Pregunta 1	Este trayecto lo has hecho:	Todo el día	Ambos	Única
Pregunta 2	Indica por favor en qué medida este trayecto te ha parecido entretenido, con buen amb…	Todo el día	Ambos	Única
Pregunta 3	Indica por favor en qué medida este trayecto te ha parecido visualmente atractivo por e…	Todo el día	Ambos	Única
Pregunta 4	Indica por favor en qué medida este trayecto te ha parecido seguro en relación al tráfico:	Todo el día	Ambos	Única
Pregunta 5	Indica por favor en qué medida este trayecto te ha parecido seguro en relación a posibl…	Todo el día	Ambos	Única

1 to 5 of 12 First Previous Page 1 of 3 Next Last

Ilustración 28 - Tabla de preguntas con agGrid

Aplicación web en diferentes idiomas

A pesar de que esto no es un requisito, toda buena aplicación web debe poder soportar diferentes idiomas, de manera que sea el propio usuario quien seleccione en que idioma desea utilizar la aplicación. En este caso, durante el desarrollo de Walkability Analyzer se valoró hacerla multi-idioma, y tras analizar qué suponía implementar esta

característica en cuanto a la planificación de tiempo estimada, se decidió desarrollarla.

Para desarrollar una aplicación que soporte diferentes idiomas, en un framework como Angular, lo más sencillo es utilizar algún módulo o librería externa que facilite el proceso de traducción. En este caso se ha utilizado la librería ng2-translate[5]. Mediante esta librería se declaran en un fichero todos los textos y palabras que se usan en la parte front-end de la aplicación, y se traducen en ese fichero a los idiomas deseados.

Posteriormente, en una parte de la aplicación, a seleccionar por el desarrollador, se implementa un selector con el que el usuario pueda seleccionar el idioma que desea utilizar, y gracias a la librería y a los textos que se han traducido manualmente, la aplicación se visualiza en el idioma requerido por el usuario.

6.2.3. Gestión de rutas geográficas

Al igual que ha ocurrido en el apartado de back-end, las principales funcionalidades que posee la parte de front-end de la aplicación web están relacionadas con el filtrado de rutas, su visualización, la descarga de datos y la corrección de municipios.

Gestión de filtros

Una vez se tuvieron definidos todos los filtros era necesario presentarlos en la pantalla de manera que trabajar con los diferentes filtros fuera sencillo para el usuario. Así, por ejemplo, una vez que se ha decidido filtrar por "Rutas valoradas", se le muestran al usuario los filtros de "Categorías" y "Versión de cuestionario" con el que una ruta ha sido valorada, para que pueda filtrar también por ellos. Por el contrario, si no ha seleccionado la opción de filtrar por "Rutas valoradas", este bloque de filtros no está visible, de manera que no genere confusión en el usuario.

Por otro lado, para aplicar los filtros que incluyen rangos de datos, como "Distancia recorrida", "Velocidad de la ruta" o "Rango de edad", así como para especificar la distancia de la circunferencia de la zona, al seleccionar un punto sobre el mapa en los filtros geográficos de "Punto de inicio y punto de fin" o "Punto de ruta", se han utilizado barras deslizables. Estas barras han sido aplicadas mediante una librería externa que permite desplazar los dos extremos del rango, visualizar los valores que van tomando los extremos, y, enviar dichos valores al fichero *component.ts* de manera inmediata, sin necesidad de ninguna función, mediante el uso de *Binding*.

Para utilizar esta librería o módulo externo, llamada *ng2-ion-range-slider*[6], es necesario añadir la etiqueta correspondiente al fichero *html* del componente donde se utiliza, y agregar las diferentes opciones que se quieran aplicar para aplicar mayor o mejor complejidad a la barra deslizable. En la *Ilustración 29* se puede visualizar una barra deslizable en los dos extremos realizada con esta librería externa.

Ilustración 29 - Barra deslizable con ng2-ion-range-slider

[5] ng2-translate: *https://www.npmjs.com/package/ng2-translate*
[6] ng2-ion-range-slider: *https://www.npmjs.com/package/ng2-ion-range-slider*

Por otro lado, para aplicar el filtro de "Municipio de inicio y municipio de fin", se obtienen, en el momento en que se carga la pantalla, todos los municipios. Para facilitar la tarea del usuario en la aplicación de dicho filtro, cuando se elige un municipio de inicio en el desplegable correspondiente, automáticamente, en el desplegable que contiene los municipios de fin únicamente se muestran los municipios de que tienen alguna ruta con ese punto de partida. Por ejemplo, si en el primer desplegable se selecciona Bilbao, únicamente se mostrarán en el segundo desplegable todos los municipios de fin en los que finalice una ruta con comienzo en Bilbao.

Para hacer esto posible, al obtener los municipios se obtienen tanto los municipios, como los pares de municipio de inicio y de fin presentes en las rutas almacenadas en el sistema. De esta manera, cuando el usuario selecciona el primer municipio, se realiza una búsqueda entre los pares y únicamente se muestran los municipios de fin coincidentes.

Otro filtro para el que se ha necesitado la utilización de una librería externa es la selección de "Fecha de la ruta". Mediante este filtro se puede acotar las rutas realizadas entre dos fechas, con lo que es necesaria la utilización de un calendario donde el usuario pueda marcar la fecha requerida, de esta manera, se garantiza que la fecha seleccionada sea válida. Para ello, el módulo externo a utilizar es *Angular Material DatePicker*[7]. Para su utilización, únicamente hay que indicar en el archivo *html* la etiqueta correspondiente a este seleccionador de fecha, y mediante el uso de *Binding*, cuando el usuario seleccione una fecha el archivo de lógica del componente almacenará la fecha elegida.

A pesar de que esta librería es ampliamente utilizada, posee un problema que ha sido complicado de solucionar. En concreto, al seleccionar una fecha, se muestra en un formato distinto al que es comúnmente utilizado en España. En concreto, si la fecha es 15/02/2018 *Angular Material DatePicker* lo muestra y lo almacena como 2018/02/15. La solución sencilla podría haber sido reconvertir ese valor antes de enviarlo en la petición de rutas al back-end, pero de esta manera, se seguiría mostrando en la aplicación web de manera errónea. Por ello se decidió realizar un proceso intermediario que se ejecuta entre el momento en el que el usuario selecciona una fecha y el instante en que esta fecha se empieza a visualizar en la pantalla. Así, el intermediario, que actúa como adaptador y hereda de la clase del propio *DatePicker*, formatea la fecha antes de mostrarla y almacenarla. En la *Ilustración 30* se puede visualizar el seleccionador de fecha utilizado con una fecha ya formateada.

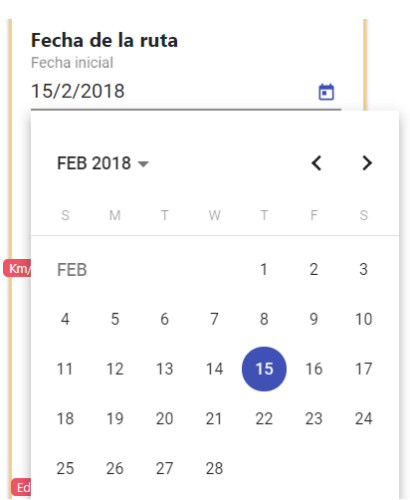


Ilustración 30 - Angular Material DatePicker

Por último, para aplicar los filtros que incluyen marcar puntos en el mapa, como el

[7] Angular Material DatePicker: *https://material.angular.io/components/datepicker*

filtro de “Punto de inicio y punto de fin”, se controla el número de veces que el usuario hace *click* en el mapa. Esta cuenta se realiza para que no pueda insertar más puntos de los necesarios, únicamente los solicitados. Esto mismo se realiza para el filtro de “Punto de ruta”. Una vez que el usuario ha marcado los puntos, se le muestra la barra deslizable para que configure el radio de la circunferencia de cada punto marcado, así, podrá delimitar las zonas de la manera que desee.

El resto de filtros para los que no se ha detallado su creación hasta este punto, son filtros en los que únicamente el usuario debe marcar si quiere aplicarlo o no. Para estos filtros se ha utilizado casillas seleccionables, o *checkboxes*, de manera que el usuario pueda seleccionar las características que desee de manera rápida y sencilla.

Visualización de las rutas

Cuando el usuario aplica los filtros que desea y pulsa el botón de “Mostrar rutas”, el método que controla dicho botón analiza qué filtros están marcados y envía dichos datos al servicio que se encarga de realizar la llamada al back-end para obtener las rutas solicitadas. Cuando la subscripción al servicio finaliza, el componente encargado de la lógica de la pantalla de rutas recibe todas las rutas que se ajustan a esos filtros. La información que trae consigo la respuesta incluye todos los puntos de las rutas, el género del usuario que realizó la ruta, el horario en que fue realizada y otras características que se detallarán en el apartado siguiente, donde se explicarán los criterios posteriores que se aplican a las rutas ya mostradas.

El mapa que se visualiza en la pantalla está realizado con la API de Google Maps, tal y como se comentó en el apartado 3.2 Estudio de alternativas existentes. Para incorporar el mapa a la aplicación web lo primero que se ha de hacer es solicitar una clave a Google mediante la cual se permite utilizar los servicios de Google Maps a la aplicación Walkability Analyzer. Una vez que esta clave está puesta en el proyecto, la aplicación ya puede utilizar Google Maps, y en este caso, se incorpora el mapa de Google al fichero *html* del componente, y se inicializa el objeto de Google Maps con los valores por defecto.

Una vez que se ha obtenido el objeto con todas las rutas, y con el mapa ya en funcionamiento, estas se analizan una a una y por defecto, se pintan en el mapa cada una de un color. Para visualizar las rutas en el mapa se han utilizado los objetos *PolyLine* propios de la API de Google Maps, objetos en los que, estableciendo puntos mediante coordenadas, los une para formar así una ruta. Además, a este objeto se le puede indicar también tanto color como grosor de línea, por lo que en este caso es muy útil para visualizar las rutas en el mapa. De esta misma forma, se visualizarán las rutas en los mapas correspondientes a la corrección de municipios. En la *Ilustración 31* se puede visualizar el mapa junto con las rutas pintadas mediante objetos *PolyLine*.

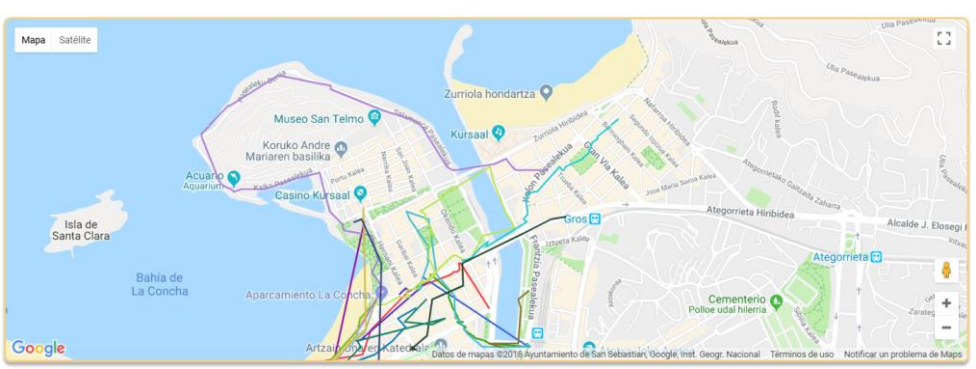


Ilustración 31 - Visualización de rutas mediante PolyLines

Cuando las rutas ya se visualizan en el mapa, la información de cada una de las rutas obtenidas se muestra en una tabla en la parte inferior del mapa a modo de leyenda. En dicha tabla se muestra el color de cada ruta, el municipio de inicio y de fin de la misma, así como la distancia recorrida. Una funcionalidad importante para los miembros del grupo CRIM-AP, y que ayuda en el estudio de cada ruta de manera individual, es poder ocultar del mapa ciertas rutas. Por ello, en la tabla leyenda se ha decidido incorporar unos *checkboxes*, de manera, que, si el *checkbox* de una ruta no está seleccionado, esta ruta se deja de visualizar en el mapa. En la *Ilustración 32* se puede visualizar la tabla leyenda perteneciente a la visualización de todas las rutas.

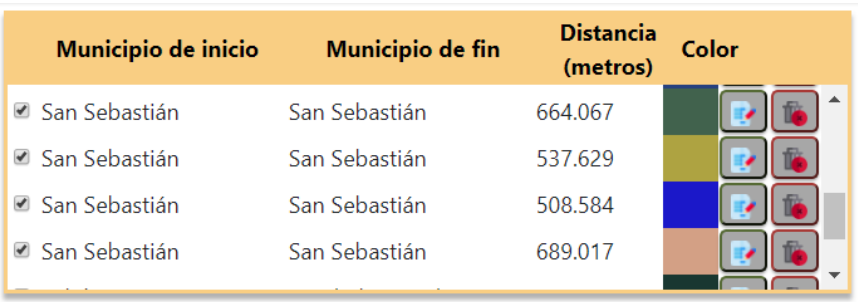


Ilustración 32 - Tabla leyenda para la visualización de rutas

Esta funcionalidad, sencilla a simple vista, ha generado gran cantidad de problemas, ya que una línea es un *PolyLine* que no posee nada característico que lo diferencie de los demás. Por ello, se ha incorporado, a nivel de objeto, un identificador a la hora de crear cada uno de los *PolyLine*, de manera que, cuando uno se deselecciona, se busca el identificador correspondiente a dicha ruta entre todos los objetos, y cuando este es encontrado, se deja de hacer visible.

Por otro lado, como ya se ha explicado anteriormente, debido a ciertos fallos presentes en el proceso de geolocalización en la aplicación móvil de Walkability Capturer, algunas rutas se obtenían de manera imprecisa. Por ello, en la propia tabla leyenda se han incorporado los botones correspondientes para eliminar tanto una ruta, como para visualizar, y posteriormente poder eliminar, la valoración de la misma. Estas acciones únicamente están presentes para el usuario "*Administrador*". Esta última opción, la que permite al usuario visualizar la valoración de una ruta, es la que abre, como ya se ha indicado, el elemento modal que se puede visualizar en la *Ilustración 27*.

Criterios para la visualización de rutas

Las rutas, al ser obtenidas, se visualizan de forma que cada ruta posee un único color. En cambio, puede ser útil para los miembros del grupo CRIM-AP visualizar las rutas en función de otros criterios, como, por ejemplo, visualizar las rutas por género, por horario en el que fueron realizadas, si se trata de una ruta valorada o sin valorar, o en función de la movilidad del usuario. Para ello, a la hora de obtener las rutas del back-end, estas, además de traer consigo la información geográfica, traen la información necesaria para poder aplicar estos criterios posteriores a la visualización.

Para realizar esta funcionalidad se eliminan todos los objetos *PolyLine* ya existentes de las rutas que ya están pintadas en el mapa, y se construyen nuevos objetos en función de lo que el usuario ha seleccionado. Por ejemplo, si el usuario selecciona visualizar las rutas en función del género, al recorrer las rutas para generar de nuevo los objetos, el color al *PolyLine* se le asigna en función de la información del usuario que viene con la ruta que se está pintando en ese momento. Así, también se genera una nueva leyenda en la que únicamente aparecen los géneros con el color de cada uno, y en la que también, al igual que ocurre al visualizar cada ruta de un color, se permite dejar de visualizar las rutas. En la *Ilustración 33* se puede visualizar las rutas pintadas por género, así como la leyenda correspondiente y el desplegable donde se seleccionan las opciones de visualización de las rutas.

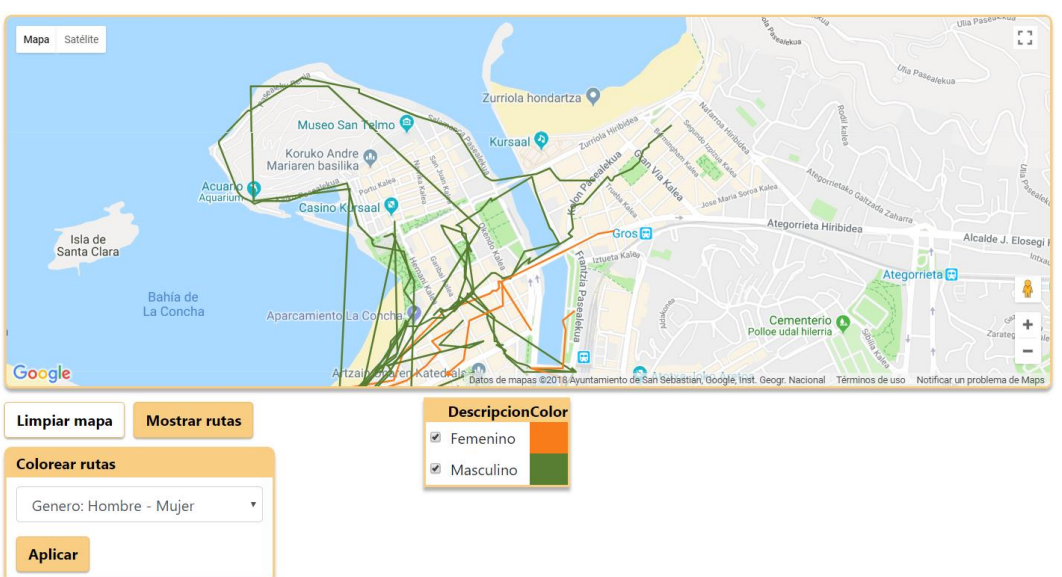


Ilustración 33 - Visualización de las rutas por genero del usuario

Descarga de datos

En el apartado donde se explicó la gestión de rutas en el desarrollo del back-end, se especificó cómo se gestionaba la obtención de la información a descargar. Una vez se generaba el objeto con la información, este era enviado mediante la respuesta al front-end para que se realizara la descarga de los datos. Por ello, cuando el usuario selecciona en qué formato desea descargar los datos, se realiza la subscripción al servicio correspondiente que se encarga de obtener dicha información, y cuando la subscripción se resuelve de manera correcta, comienza la gestión de la descarga de datos.

Para poder descargar el fichero se transforma todo el objeto en una única cadena de texto que contenga el formato que se desee obtener, por ejemplo, saltos de línea o comas

para separar los diferentes elementos. Una vez con la variable ya creada, se transforma en un *Blob*, (Binary Large Object), un objeto que permite almacenar un volumen variable grande de datos. Así, mediante este elemento *Blob* se genera el fichero y se descarga el archivo de manera automática, sin que sea necesario que el usuario introduzca ningún tipo de ruta de almacenaje en el equipo.

Corrección de municipios

El servicio de geolocalización de la librería externa de Walkability Capturer posee problemas de imprecisión en la obtención de los municipios de inicio y de fin de las rutas realizadas. Por ello, es necesario un apartado donde poder gestionar estas incidencias en la aplicación web de Walkability Analyzer. Durante el desarrollo de este sistema de corrección surgieron algunas dudas sobre cómo exactamente tendría que ser el proceso para visualizar los municipios incorrectos, por lo que en la aplicación web se permite seleccionar entre dos opciones de visualización.

- El usuario, al entrar en el apartado correspondiente, puede visualizar un mapa donde, tras hacer *click* en el punto seleccionado, puede definir un radio para acotar una zona. Tras realizar esto se obtendrá la información de todas las rutas que pasen por dicho punto, y en una tabla se mostraran todos los municipios de las rutas obtenidas. Pinchando cada línea de la tabla en la que está presente una ruta, se puede visualizar en el mapa esa ruta concreta, de manera que se puede ver si los municipios están bien almacenados o no, y en el caso de que no sea así, actualizar los erróneos.

- El usuario, al entrar, ya posee la tabla con todos los municipios de todas las rutas presentes en el sistema, así como un mapa sin ninguna ruta pintada. Seleccionando las líneas del mapa, las rutas correspondientes pueden visualizarse, de manera que se puede comprobar si los municipios son correctos o erróneos, y corregir los posibles errores.

Al igual que ocurre con la visualización de rutas en el apartado de filtros, el mapa ha sido creado mediante la API de Google Maps y las rutas se pintan en dicho mapa utilizando los objetos *PolyLines*. Estas dos formas de gestionar esta parte del sistema se han creado ya que, no todos los miembros del grupo CRIM-AP quieren gestionar la corrección de municipios de la misma manera.

6.2.4. Seguridad en la parte cliente

A pesar de que en la parte back-end de la aplicación web ya se han implementado mecanismos de seguridad para poder evitar posibles ataques malintencionados, en el apartado de front-end también se han desarrollado otras buenas prácticas relacionadas con la seguridad.

En primer lugar, y como se ha comentado en el apartado 6.2.2, al iniciar sesión en el sistema, el *Token* correspondiente al usuario, así como su rol, se almacenan en la *sessionStorage,* donde se puede obtener dicho *Token* cada vez que se vaya a construir una cabecera para enviar la petición al back-end.

Por otro lado, aunque el back-end no permite acceder a los recursos si el usuario no tiene permiso para ello, otra buena práctica es no dejar al usuario realizar acciones que no puede hacer. De esta manera no solo se refuerza el sistema de seguridad implementado en la parte servidor, si no que se desarrolla un sistema más intuitivo no mostrando al usuario acciones que no puede realizar. Para ello se utiliza la variable que contiene el rol del usuario

que también está almacenada en el *sessionStorage*. Por ejemplo, si inicia sesión un usuario con el rol de "Investigador", no se le muestran acciones como "Gestionar municipios" o la posibilidad de eliminar o añadir un cuestionario, ya que como se ha indicado en el apartado 4.2.2 Modelo de casos de uso, no tiene permisos para ello.

7. Verificación y evaluación

En este capítulo se explicarán y analizaran las pruebas realizadas sobre las diferentes partes que componen este proyecto, el contexto en la que estas fueron realizadas y los resultados obtenidos de las mismas. En concreto, se diferenciarán las pruebas realizadas a la parte back-end del proyecto, a la parte front-end y las pruebas de usabilidad realizadas al sistema completo.

7.1. Pruebas de back-end

Para realizar las pruebas de la API y el back-end de Walkability Analyzer se ha utilizado la herramienta Postman. Postman es una herramienta, propiedad de Google, que permite automatizar las pruebas de una API REST realizando solicitudes HTTP, de manera que, a la hora de modificar código, facilita la realización de las pruebas para comprobar que el sistema sigue siendo estable y sigue funcionando correctamente a pesar de los cambios realizados. Para automatizar las pruebas de este proyecto, debido a la gran cantidad de pruebas que se pueden realizar, se han definido baterías o colecciones de pruebas unitarias que permiten comprobar de una manera rápida si el sistema funciona correctamente.

Las colecciones están divididas según las funcionalidades de la aplicación. Así, se han creado colecciones para: rutas, cuestionarios, preguntas, categorías, configuraciones de Walkability Capturer, corrección de municipios y gestión de usuarios. De esta manera, si se realizan cambios en el código referente a los cuestionarios, por ejemplo, se puede ejecutar en un instante todas las pruebas pertenecientes a esa colección, para comprobar si estas siguen devolviendo el resultado que cabría esperar.

Para realizar las pruebas mediante Postman se configura cada una de ellas mediante la URL a la que hay que enviar la petición, en este caso, la URL correspondiente al back-end para el recurso solicitado. Posteriormente, mediante el *body* de la petición o la cabecera de la misma, se envían los parámetros necesarios para la prueba que se está implementando, por ejemplo, el *Token* en la cabecera o un objeto JSON en el cuerpo de la solicitud. Finalmente, se define el resultado esperado de la prueba para poder comparar lo que se esperaba con la respuesta real de la prueba.

Una vez la prueba está implementada, esta es ejecutada por el Postman, y tal y como ocurre con una petición realizada por un servicio cliente, esta herramienta muestra la respuesta que envía el back-end a esa petición concreta. Para documentar las pruebas realizadas, así como los resultados obtenidos, se analizará cada batería de pruebas por separado.

7.1.1. Cuestionarios

Para las pruebas realizadas a los cuestionarios se han tenido en cuenta las diferentes acciones que se pueden realizar sobre los mismos: obtener la información acerca de ellos, eliminar los cuestionarios no activos, editar dichos cuestionarios, añadir algunos nuevos e incluso activar uno de ellos. En las siguientes tablas se pueden visualizar las pruebas realizadas para cada funcionalidad presente en los cuestionarios.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
1. Obtener y visualizar uno o varios cuestionarios con parámetros correctos (*Token* e identificador del cuestionario correctos)	1A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	1B. Usuario autorizado	Obtención correcta de datos.	**Prueba 1:** Obtención incorrecta de datos.	Prueba incorrecta: los datos no tienen el formato adecuado. Fallo al tratar con los datos para devolver la petición.
			Prueba 2: Obtención correcta de datos.	Prueba correcta.
2. Obtener y visualizar uno o varios cuestionarios con parámetros incorrectos (*Token* o identificador del cuestionario incorrectos)	Cualquiera	Aviso indicando el error.	Todo OK. Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 35 - Pruebas de la obtención de cuestionarios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
3. Insertar un cuestionario con parámetros correctos (*Token* y objeto con los datos correctos)	3A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	3B. Usuario autorizado	Cuestionario añadido correctamente.	**Prueba 1:** El cuestionario no se añade correctamente.	Prueba incorrecta: El cuestionario no se añade debido a un fallo con la petición a la base de datos.
			Prueba 2: El cuestionario no se añade correctamente.	Prueba incorrecta: El orden de las preguntas a añadir no es el correcto.
			Prueba 3: Mensaje indicando que el cuestionario se ha añadido.	Prueba correcta.
4. Insertar un cuestionario con parámetros incorrectos (*Token* u objeto con los datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 36 - Pruebas para añadir cuestionarios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
5. Editar un cuestionario con parámetros correctos (*Token* y objeto con los datos correctos)	5A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	5B. Usuario autorizado	Si el cuestionario está activo: Al intentar editar el cuestionario activo se debe crear uno nuevo.	**Prueba 1:** Error, el cuestionario no se ha añadido.	Prueba incorrecta: El cuestionario no se ha añadido debido a un error en el tratamiento de los datos.
			Prueba 2: Mensaje indicando que un nuevo cuestionario se ha añadido.	Prueba correcta.
		Si el cuestionario no está activo: Cuestionario actualizado correctamente.	Mensaje indicando que el cuestionario se ha actualizado.	Prueba correcta.
6. Editar un cuestionario con parámetros incorrectos (*Token* u objeto con los datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 37 - Pruebas de actualización de cuestionarios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
7. Eliminar un cuestionario con parámetros correctos (*Token* e identificador del cuestionario correctos)	7A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	7B. Usuario autorizado	Si el cuestionario está activo: No se puede eliminar el cuestionario activo.	**Prueba 1:** El cuestionario activo se eliminaba del sistema.	Prueba incorrecta: El cuestionario se eliminaba por estar mal la condición.
			Prueba 2: Aviso indicando el error. No se puede eliminar un cuestionario activo.	Prueba correcta.
		Si el cuestionario no está activo: Cuestionario eliminado correctamente.	Mensaje indicando que el cuestionario se ha eliminado.	Prueba correcta.
8. Eliminar un cuestionario con parámetros incorrectos (*Token* u identificador del cuestionario incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 38 - Pruebas de eliminación de cuestionarios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
9. Activar un cuestionario con parámetros correctos (*Token* e identificador del cuestionario correctos)	9A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	9B. Usuario autorizado	Activación correcta.	Mensaje indicando la activación correcta.	Prueba correcta.
10.Activar un cuestionario con parámetros incorrectos (*Token* o identificador del cuestionario incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 39 - Pruebas de activación de cuestionarios en back-end

7.1.2. Preguntas

Dentro de la batería de pruebas relacionadas con las preguntas, estas se han separado para así comprobar cada acción que puede realizar los usuarios: obtener y visualizar las preguntas, eliminar las preguntas no deseadas, añadir nuevas preguntas o actualizar las ya existentes. En las siguientes tablas se pueden visualizar las pruebas realizadas para cada posible acción existente en el sistema.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
11.Obtener y visualizar una o varias preguntas con parámetros correctos (*Token* e identificador de la pregunta correctos)	11A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	11B.Usuario autorizado	Obtención correcta de datos.	**Prueba 1:** Las preguntas se obtienen sin sus posibles respuestas.	Prueba incorrecta: Las respuestas no se mostraban por un fallo en la sentencia a la base de datos.
			Prueba 2: Las preguntas se obtienen correctamente.	Prueba correcta.
12.Obtener y visualizar una o varias preguntas con parámetros incorrectos (*Datos* incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 40 - Pruebas de la obtención de preguntas en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
13.Insertar una pregunta con parámetros correctos (*Token* y objeto con los datos correctos)	13A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	13B.Usuario autorizado	Pregunta añadida correctamente.	**Prueba 1**: Las posibles respuestas de la pregunta no se han añadido.	Prueba incorrecta: Los datos de la petición no se estaban obteniendo correctamente.
			Prueba 2: Mensaje indicando que la pregunta se ha añadido.	Prueba correcta.
14.Insertar una pregunta con parámetros incorrectos (*Token* u objeto con los datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 41 - Pruebas para añadir preguntas en back-en

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
15.Eliminar una pregunta con parámetros correctos (*Token* e identificador de la pregunta correctos)	15A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	15B.Usuario autorizado	Pregunta perteneciente a un cuestionario activo: No se puede eliminar una pregunta de un cuestionario activo.	**Prueba 1:** La pregunta se eliminaba del sistema.	Prueba incorrecta: La pregunta se eliminaba del sistema por un error en la condición.
			Prueba 2: Aviso indicando el error. No se puede eliminar una pregunta de un cuestionario activo.	Prueba correcta.
		Pregunta perteneciente a un cuestionario no activo: Pregunta eliminada correctamente.	Mensaje indicando que la pregunta se ha eliminado.	Prueba correcta.
16.Eliminar una pregunta con parámetros incorrectos (*Token* u identificador de la pregunta incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 42 - Pruebas de eliminación de preguntas en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
17.Editar una pregunta con parámetros correctos (*Token* y objeto con los datos correctos)	17A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	17B. Usuario autorizado	Pregunta activa: Al intentar editar una pregunta del cuestionario activo se debe crear una pregunta nueva.	Mensaje indicando que una nueva pregunta se ha añadido.	Prueba correcta.
		Pregunta no activa: Pregunta actualizada correctamente.	Mensaje indicando que la pregunta se ha actualizado.	Prueba correcta.
18.Editar una pregunta con parámetros incorrectos (*Token* u objeto con los datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 43 - Pruebas de actualización de preguntas en back-end

7.1.3. Categorías

Para las categorías únicamente hay dos acciones que los usuarios pueden realizar: obtener y visualizarlas, y actualizar la categoría deseada. En las siguientes tablas de pueden visualizar las pruebas realizadas para probar estas dos acciones.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
19. Obtener y visualizar una o varias categorías con parámetros correctos (*Token* e identificador de la categoría correctos)	19A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	19B.Usuario autorizado	Obtención correcta de datos.	Obtención correcta de datos.	Prueba correcta.
20. Obtener y visualizar una o varias categorías con parámetros incorrectos (*Token* o identificador de la categoría incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 44 - Pruebas de la obtención de categorías en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
21. Editar una categoría con parámetros correctos (*Token* e objeto con datos correctos)	21A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	21B.Usuario autorizado	Actualización correcta de la categoría.	Mensaje indicando que la categoría se ha actualizado.	Prueba correcta.
22. Editar una categoría con con parámetros incorrectos (*Datos incorrectos*)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 45 - Pruebas de actualización de categorías en back-end

7.1.4. Configuraciones

Para las pruebas realizadas a las configuraciones se han tenido en cuenta las diferentes acciones que se pueden realizar sobre las mismas: obtener la información acerca de ellas, eliminar las configuraciones que no están activadas, añadir algunas nuevas e incluso activar una de ellas. En las siguientes tablas se pueden visualizar las pruebas realizadas para cada funcionalidad presente en las configuraciones.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
23. Obtener y visualizar una o varias configuraciones con parámetros correctos (*Token* e identificador de la configuración correctos)	23A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	23B. Usuario autorizado	Obtención correcta de datos.	Obtención correcta de datos.	Prueba correcta.
24. Obtener y visualizar una o varias configuraciones con parámetros incorrectos (*Token* o identificador de la configuración incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 46 - Pruebas de la obtención de configuraciones en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
25. Eliminar una configuración con parámetros correctos (*Token* e identificador de la configuración correctos)	25A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	25B. Usuario autorizado	Configuración activa: No se puede eliminar la configuración activa.	**Prueba 1:** La configuración activa no se eliminaba. Aparece un error.	Prueba incorrecta: La obtención del identificador de la configuración a eliminar se hacía mal y generaba un error.
			Prueba 2: La configuración activa se eliminaba del sistema.	Prueba incorrecta: La configuración se eliminaba por no existir esa condición en la consulta.
			Prueba 3: Aviso indicando el error. No se puede eliminar una configuración activa.	Prueba correcta.
		Configuración no activa: Configuración eliminada correctamente.	Mensaje indicando que la configuración se ha eliminado.	Prueba correcta.
26. Eliminar una configuración con parámetros incorrectos (*Token* u identificador de la configuración incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 47 - Pruebas de eliminación de configuraciones en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
27. Insertar una configuración con parámetros correctos (*Token* y objeto con los datos correctos)	27A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	27B. Usuario autorizado	Configuración añadida correctamente.	Mensaje indicando que la configuración se ha añadido.	Prueba correcta.
28. Insertar una configuración con parámetros incorrectos (*Token* u objeto con los datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 48 - Pruebas para añadir configuraciones en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
29. Activar una configuración con parámetros correctos (*Token* e identificador del cuestionario correctos)	29A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	29B. Usuario autorizado	Activación correcta.	Mensaje indicando la activación correcta.	Prueba correcta.
30. Activar una configuración con parámetros incorrectos (*Token* o identificador del cuestionario incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 49 - Pruebas de activación de preguntas en back-end

7.1.5. Municipios

La corrección de municipios es una funcionalidad adicional cuya única finalidad es solventar los errores que presentan los datos de la base de datos. Por ello, las únicas acciones disponibles sobre dichos municipios son, la obtención y visualización de los mismos, así como la actualización para corregir los errores. En las siguientes tablas se pueden visualizar las pruebas relacionadas con los municipios.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
31.Obtener los municipios de las rutas con parámetros correctos (*Token* e identificador de las rutas correctos)	31A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	31B.Usuario autorizado	Obtención correcta de datos.	Obtención correcta de datos.	Prueba correcta.
32.Obtener los municipios de las rutas con parámetros incorrectos (*Token* o identificador de las rutas incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 50 - Pruebas de la obtención de municipios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
33.Editar los municipios con parámetros correctos (*Token* e objeto con datos correctos)	33A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	33B.Usuario autorizado	Actualización correcta de los municipios.	Mensaje indicando que los municipios se han actualizado.	Prueba correcta.
34.Editar los municipios con parámetros incorrectos (*Token* u objeto con datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 51 - Pruebas de actualización de municipios en back-end

7.1.6. Usuarios

Las pruebas que se han realizado para los usuarios se han dividido en: obtener y visualizar los usuarios, insertar un nuevo usuario, eliminar usuarios existentes y actualizarlos. Actualizar un usuario engloba actualizar los permisos del mismo. En las siguientes tablas se puede visualizar las pruebas realizadas.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
35. Obtener y visualizar uno o varios usuarios con parámetros correctos (*Token* e identificador del usuario correctos)	35A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	35B. Usuario autorizado	Obtención correcta de datos.	Obtención correcta de datos.	Prueba correcta.
36. Obtener y visualizar uno o varios usuarios con parámetros incorrectos (*Token* o identificador del usuario incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 52 - Pruebas de obtención de usuarios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
37. Insertar un usuario con parámetros correctos (*Token* y objeto con los datos correctos)	37A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	37B. Usuario autorizado	Usuario añadido correctamente.	Mensaje indicando que el usuario se ha añadido.	Prueba correcta.
38. Insertar un usuario con parámetros incorrectos (*Token* u objeto con los datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 53 - Pruebas para añadir usuarios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
39. Eliminar un usuario con parámetros correctos (*Token* e identificador del usuario correctos)	39A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	39B. Usuario autorizado	Queda más de un usuario: Usuario eliminado correctamente.	Mensaje indicando que el usuario se ha eliminado.	Prueba correcta.
		Queda un único usuario: Aviso indicando el error.	Aviso indicando el error. No es posible eliminar el último usuario.	Prueba correcta.
40. Eliminar un usuario con parámetros incorrectos (*Token* e identificador del usuario incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 54 - Pruebas de eliminación de usuarios en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
41. Actualizar un usuario con parámetros correctos (*Token* y objeto con los datos correctos)	41A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	41B. Usuario autorizado	Usuario actualizado correctamente.	Mensaje indicando que el usuario se ha actualizado.	Prueba correcta.
42. Actualizar un usuario con parámetros incorrectos (*Token* u objeto con los datos incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 55 - Pruebas de actualización de usuarios en back-end

7.1.7. Rutas

Respecto a las rutas, se permite a los usuarios obtener y visualizar las rutas, así como la información de cada una de ellas, eliminar las rutas o la valoración de las mismas. Las siguientes tablas muestran las pruebas que se han realizado para cada acción disponible.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
43. Obtener y visualizar una o varias rutas según los filtros con parámetros correctos (*Token* y parámetros de los filtros correctos)	43A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	43B. Usuario autorizado	Obtención correcta de datos.	Obtención correcta de datos.	Prueba correcta.
44. Obtener y visualizar una o varias rutas según los filtros con parámetros incorrectos (*Token* o parámetros de los filtros incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 56 - Pruebas de obtención de rutas en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
45. Obtener la información de las rutas con parámetros correctos (*Token* e identificadores de las rutas correctos)	45A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	45B. Usuario autorizado	Obtención correcta de datos.	Obtención correcta de datos.	Prueba correcta.
46. Obtener la información de las rutas con parámetros incorrectos (*Token* e identificadores de las rutas incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 57 - Pruebas de obtención de la información de las rutas en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
47. Eliminar una ruta con parámetros correctos (*Token* e identificador de la ruta correctos)	47A. Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	47B. Usuario autorizado	Eliminación correcta de la ruta.	**Prueba 1**: La ruta no se ha eliminado del sistema.	Prueba incorrecta: La petición era incorrecta a la base de datos.
			Prueba 2: Mensaje indicando la eliminación correcta de la ruta.	Prueba correcta.
48. Eliminar una ruta con parámetros incorrectos (*Token* e identificador de la ruta incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 58 - Pruebas de eliminación de rutas en back-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
49. Eliminar la valoración de una ruta con parámetros correctos (*Token* e identificador de la ruta correctos)	49A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	49B.Usuario autorizado	Eliminación correcta de la valoración de la ruta.	Mensaje indicando la eliminación correcta de la valoración de la ruta.	Prueba correcta.
50. Eliminar la valoración de una ruta con parámetros incorrectos (*Token* e identificador de la ruta incorrectos)	Cualquiera	Aviso indicando el error.	Aviso indicando el error. Petición incorrecta.	Prueba correcta.

Tabla 59 - Pruebas de eliminación de las valoraciones de rutas en back-end

7.1.8. Inicio y cierre de sesión

Para acceder al sistema es necesario iniciar sesión, ya que Walkability Analyzer no dispone de funcionalidades de carácter público. Por ello, a continuación, se detallan las pruebas de back-end realizadas para comprobar el buen funcionamiento de esta funcionalidad.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
51. Iniciar sesión con credenciales correctas	Usuario no identificado	Se devuelve un objeto con los parámetros del usuario (*Token y rol*)	**Prueba 1:** No se inicia sesión y no se devuelve ningún objeto	Prueba incorrecta: No se estaba comprobando bien si la contraseña del usuario era correcta.
			Prueba 2: Se devuelve el objeto necesario con los parámetros correctos.	Prueba correcta.
52. Iniciar sesión con credenciales incorrectas	Usuario no identificado	Aviso indicando el error.	Aviso indicando el error. Valores incorrectos.	Prueba correcta.
53. Realizar alguna acción pasado el tiempo de validez del *Token*	Cualquier usuario que haya iniciado sesión en el sistema	Mensaje indicando que la sesión ha expirado y el *Token* no es válido.	Mensaje indicando que la sesión ha expirado y el *Token* no es válido.	Prueba correcta.
54. Cerrar sesión en el sistema	Cualquier usuario que haya iniciado sesión en el sistema	El *Token* del usuario se elimina.	El *Token* del usuario se ha eliminado y no se puede acceder hasta volver a iniciar sesión.	Prueba correcta.

Tabla 60 - Pruebas de inicio y cierre de sesión en back-end

7.2. Pruebas de front-end

Una vez se ha comprobado el funcionamiento del back-end, el front-end debe realizar las llamadas al servidor de manera correcta para que las acciones se realicen de manera exitosa. En este aspecto, el funcionamiento de la aplicación web se ha comprobado que es exitoso ya que, ejecutando diferentes pruebas se ha comprobado que las llamadas y las respuestas realizadas al back-end se ajustan al funcionamiento esperado. En los siguientes apartados se detallarán las pruebas realizadas divididas por funcionalidad.

7.2.1. Cuestionarios

Para las pruebas realizadas a los cuestionarios se han tenido en cuenta las diferentes acciones que se pueden realizar sobre los mismos. En las siguientes tablas se pueden visualizar las pruebas realizadas para cada funcionalidad presente en los cuestionarios.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
1. El usuario pulsa "Cuestionarios" en el menú de navegación	Independiente	Obtención correcta de datos.	**Prueba 1:** Los cuestionarios no se visualizan.	Prueba incorrecta: La tabla de cuestionarios no estaba recibiendo los datos.
			Prueba 2: Los cuestionarios se visualizan.	Prueba correcta.

Tabla 61 - Pruebas de visualización de cuestionarios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
2. El usuario pulsa "Añadir cuestionario"	2A. Usuario no autorizado	No le debe aparecer tal botón.	El botón de "Añadir cuestionario" no le aparece.	Prueba correcta.
	2B. Usuario autorizado	Mensaje indicando que el cuestionario se ha añadido correctamente.	**Prueba 1:** El cuestionario no se ha añadido y aparece un mensaje de error.	Prueba incorrecta: Los datos no se estaban mandado de la forma correcta al back-end.
			Prueba 2: El cuestionario no se añade correctamente.	Prueba correcta.

Tabla 62 - Pruebas para añadir cuestionarios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
3. Si el usuario pulsa "Guardar" después del editar un cuestionario.	3A. Usuario no autorizado	El usuario no debe visualizar ese botón.	El botón no aparece para el usuario.	Prueba correcta.
	3B. Usuario autorizado	Si el cuestionario está activo: Al intentar editar el cuestionario activo debe aparecer un mensaje indicando que se ha añadido un cuestionario nuevo.	Mensaje indicando que se ha añadido un cuestionario nuevo.	Prueba correcta.
		Si el cuestionario no está activo: Debe aparecer un mensaje indicando que se ha actualizado correctamente.	Mensaje indicando que el cuestionario se ha actualizado.	Prueba correcta.

Tabla 63 - Pruebas para actualizar cuestionarios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
4. Si el usuario pulsa el botón de "Eliminar" un cuestionario.	4A. Usuario no autorizado	El usuario no debe visualizar ese botón.	El usuario no visualiza el botón.	Prueba correcta.
	4B. Usuario autorizado	Si el cuestionario está activo: Mensaje indicando que no se puede eliminar el cuestionario activo.	Mensaje indicando que el cuestionario no se puede eliminar.	Prueba correcta.
		Si el cuestionario no está activo: Mensaje indicando que el cuestionario se ha eliminado.	Mensaje indicando que el cuestionario se ha eliminado.	Prueba correcta.

Tabla 64 . Pruebas para eliminar cuestionarios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
5. Si el usuario pulsa "Activar" sobre un cuestionario.	5A.Usuario no autorizado	El usuario no debe visualizar ese botón.	El usuario no visualiza el botón.	Prueba correcta.
	5B. Usuario autorizado	Mensaje indicando la activación correcta.	Mensaje indicando la activación correcta.	Prueba correcta.

Tabla 65 - Pruebas para activar cuestionarios en front-end

7.2.2. Preguntas

Para las pruebas realizadas relacionadas con la parte front-end de las preguntas, estas se han separado para así comprobar cada acción que puede realizar los usuarios: obtener y visualizar las preguntas, eliminar las preguntas no deseadas, añadir nuevas preguntas o actualizar las ya existentes. En las siguientes tablas se pueden visualizar las pruebas realizadas para cada posible acción existente en el sistema.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
6.El usuario pulsa "Preguntas" en el menú de navegación	Independiente	Las preguntas se muestran en la tabla con la información de cada una.	Las preguntas se muestran en la tabla con la información de cada una.	Prueba correcta.

Tabla 66 - Pruebas para visualizar preguntas en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
7.El usuario pulsa "Añadir pregunta"	7A.Usuario no autorizado	Ese botón no debe visualizarlo ese usuario.	El botón no se visualiza en la pantalla.	Prueba correcta.
	7B.Usuario autorizado	Un mensaje indicando que la pregunta se ha añadido correctamente.	**Prueba 1**: Mensaje indicando un error.	Prueba incorrecta: La lista de respuestas no se estaba mandando correctamente al back-end.
			Prueba 2: Mensaje indicando que la pregunta se ha añadido.	Prueba correcta.

Tabla 67 - Pruebas para añadir preguntas en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
8.El usuario pulsa "Eliminar" sobre una pregunta de la pantalla	8A.Usuario no autorizado	Ese botón no debe visualizarlo ese usuario.	El botón no se visualiza en la pantalla.	Prueba correcta.
	8B.Usuario autorizado	Pregunta perteneciente a un cuestionario activo: Mensaje indicando que no se puede eliminar una pregunta de un cuestionario activo.	Mensaje indicando que la pregunta no se puede eliminar.	Prueba correcta.
		Pregunta perteneciente a un cuestionario no activo: Mensaje indicando que la pregunta ha sido eliminada correctamente.	Mensaje indicando que la pregunta se ha eliminado.	Prueba correcta.

Tabla 68 - Pruebas para eliminar preguntas en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
9.El usuario pulsa "Guardar" después de haber actualizado una pregunta.	9A.Usuario no autorizado	Aviso indicando el error.	Aviso indicando el error. No posee los permisos.	Prueba correcta.
	9B. Usuario autorizado	Pregunta activa: Al intentar editar una pregunta del cuestionario activo se debe mostrar un mensaje indicando que se ha añadido una nueva pregunta.	Mensaje indicando que una nueva pregunta se ha añadido.	Prueba correcta.
		Pregunta no activa: Mensaje indicando que la pregunta se ha actualizado correctamente.	Mensaje indicando que la pregunta se ha actualizado.	Prueba correcta.

Tabla 69 - Pruebas para actualizar preguntas en front-end

7.2.3. Categorías

Para las categorías únicamente hay dos acciones que los usuarios pueden realizar: visualizarlas y actualizar la categoría deseada. En las siguientes tablas de pueden visualizar las pruebas realizadas en la parte front-end del sistema para probar estas dos acciones.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
10. El usuario pulsa "Categorías" en el menú de navegación	Independiente	Se muestran las categorías en una tabla.	Las categorías se visualizan de forma correcta.	Prueba correcta.

Tabla 70 - Pruebas para visualizar categorías en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
11. Si el usuario pulsa "Guardar" después de modificar una categoría	11A.Usuario no autorizado	El usuario no ha podido actualizar una categoría ni se le visualiza el botón correspondiente.	El usuario no puede modificar la categoría.	Prueba correcta.
	11B.Usuario autorizado	Mensaje indicando la actualización correcta de la categoría.	Mensaje indicando que la categoría se ha actualizado.	Prueba correcta.

Tabla 71 - Pruebas para actualizar categorías en front-end

7.2.4. Configuraciones

Para las pruebas realizadas a las configuraciones en la parte front-end del sistema, se han tenido en cuenta las diferentes acciones que se pueden realizar sobre las mismas. En las siguientes tablas se pueden visualizar las pruebas realizadas para cada funcionalidad presente en las configuraciones.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
12. El usuario pulsa "Configuración" en el menú de navegación	Independiente	Las configuraciones se muestran en una tabla	**Prueba 1:** No todos los valores de las configuraciones se estaban mostrando en la tabla.	Prueba incorrecta: Los valores de la respuesta no se estaban obteniendo bien.
			Prueba 2: Las configuraciones se mostraban correctamente en la tabla.	Prueba correcta.

Tabla 72 - Pruebas para visualizar configuraciones en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
13. El usuario pulsa "Eliminar" sobre una configuración	13A.Usuario no autorizado	El usuario no puede visualizar ese botón en la pantalla.	El botón no se muestra en la pantalla.	Prueba correcta.
	13B. Usuario autorizado	Configuración activa: Mensaje indicando que no se puede eliminar la configuración activa.	Se visualiza el mensaje indicando que no se puede eliminar.	Prueba correcta.
		Configuración no activa: Mensaje indicando que la configuración se ha eliminado.	Mensaje indicando que la configuración se ha eliminado.	Prueba correcta.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
14. El usuario pulsa "Añadir configuración"	14A.Usuario no autorizado	El usuario no debe visualizar ese botón en pantalla.	El botón no se visualiza.	Prueba correcta.
	14B. Usuario autorizado	Mensaje indicando que la configuración se ha añadido.	Mensaje indicando que la configuración se ha añadido.	Prueba correcta.

Tabla 73 - Pruebas para añadir configuraciones en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
15. El usuario pulsa "Activar" sobre una configuración.	15A.Usuario no autorizado	El usuario no debe visualizar ese botón en pantalla.	El botón no se visualiza.	Prueba correcta.
	15B. Usuario autorizado	Mensaje indicando la activación correcta de la configuración.	Mensaje indicando la activación correcta.	Prueba correcta.

Tabla 74 - Pruebas para activar configuraciones en front-end

7.2.5. Municipios

Tal y como se ha comentado al analizar las pruebas de back-end, las únicas acciones disponibles sobre los municipios son, la obtención y visualización de los mismos, así como la actualización para corregir los errores. En las siguientes tablas se pueden visualizar las pruebas relacionadas con los municipios en la parte front-end del sistema.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
16.El usuario pulsa sobre las incidencias de municipios en el menú de navegación	16A.Usuario no autorizado	El usuario no debe poder acceder a esta parte del sistema.	Los botones relacionados con municipios no se le muestran al usuario.	Prueba correcta.
	16B.Usuario autorizado	Los municipios se visualizan en la pantalla.	Los municipios se visualizan, y al pinchar sobre ellos, se muestra la ruta asociada.	Prueba correcta.

Tabla 75 - Pruebas para visualizar municipios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
17.El usuario pulsa "Guardar" después de actualizar los municipios	Usuario autorizado	Mensaje indicando la actualización correcta de los municipios.	Mensaje indicando que los municipios se han actualizado.	Prueba correcta.

Tabla 76 - Pruebas para actualizar municipios en front-end

7.2.6. Usuarios

Las pruebas que se han realizado para los usuarios se han dividido en: obtener y visualizar los usuarios, insertar un nuevo usuario, eliminar usuarios existentes y actualizarlos. En las siguientes tablas se puede visualizar las pruebas realizadas a parte front-end de la aplicación.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
18. El usuario pulsa "Gestión de usuarios" en el menú de navegación	18A. Usuario no autorizado	El usuario no debe poder acceder a esta parte del sistema.	Esa parte del sistema es invisible para el usuario.	Prueba correcta.
	18B. Usuario autorizado	Los usuarios se visualizan en la tabla correspondiente.	Los usuarios se visualizan correctamente.	Prueba correcta.

Tabla 77 - Pruebas para visualizar usuarios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
19. El usuario pulsa "Insertar usuario"	Usuario autorizado	Mensaje indicando que el usuario se ha añadido correctamente.	Mensaje indicando que el usuario se ha añadido.	Prueba correcta.

Tabla 78 - Pruebas para añadir usuarios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
20. El usuario pulsa "Eliminar" sobre un usuario.	Usuario autorizado	Queda más de un usuario: Mensaje indicando que el usuario se ha eliminado.	Mensaje indicando que el usuario se ha eliminado.	Prueba correcta.
		Queda un único usuario: Mensaje indicando que no se puede eliminar.	Mensaje indicando que no se puede eliminar.	Prueba correcta.

Tabla 79 - Pruebas para eliminar usuarios en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
21. El usuario pulsa "Actualizar usuario"	Usuario autorizado	Mensaje indicando que el usuario se ha actualizado correctamente.	Mensaje indicando que el usuario se ha actualizado.	Prueba correcta.

Tabla 80 - Pruebas para actualizar usuarios en front-end

7.2.7. Rutas

Respecto a las rutas en la parte de front-end, se permite a los usuarios obtener y visualizar las rutas, así como la información de cada una de ellas, eliminar las rutas o la valoración de las mismas. Las siguientes tablas muestran las pruebas que se han realizado para cada acción disponible.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
22. El usuario pulsa "Mostrar rutas" después de visualizar los filtros	Independiente	Las rutas se visualizan en el mapa, una de cada color por defecto, y se muestra la leyenda de las rutas.	**Prueba 1:** Las rutas no se visualizan. La leyenda no se visualiza.	Prueba incorrecta: El JSON de los datos de la respuesta no se estaba leyendo correctamente.
			Prueba 2: Las rutas no se visualizan. La leyenda no se visualiza.	Prueba incorrecta: Los datos al objeto *PolyLine* de Google Maps no se estaban pasando correctamente.
			Prueba 3: Las rutas se visualizan pero no aparece la leyenda del mapa.	Prueba incorrecta: Las rutas no se estaban analizando bien en la leyenda, y por ello esta no se mostraba.
			Prueba 4: La leyenda y las rutas se muestran.	Prueba correcta.

Tabla 81 - Pruebas de visualización de rutas en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
23. El usuario pulsa el botón de la leyenda para ver la valoración de la ruta	23A. Usuario no autorizado	El botón para visualizar la valoración no se debe mostrar en pantalla.	El botón no se visualiza en pantalla.	Prueba correcta.
	23B. Usuario autorizado	La valoración de la ruta se muestra de forma correcta en la ventana modal.	La ventana modal se muestra correctamente con la valoración	Prueba correcta.

Tabla 82 - Pruebas de visualización de la valoración de una ruta en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
24. El usuario pulsa "Eliminar valoración" en la ventana modal donde se visualiza la valoración	Usuario autorizado	Mensaje indicando la eliminación correcta de la ruta.	**Prueba 1**: Mensaje de error.	Prueba incorrecta: Los parámetros no se estaban mandando de forma correcta al back-end.
			Prueba 2: Mensaje indicando la eliminación de la valoración de la ruta.	Prueba correcta.

Tabla 83 - Pruebas de eliminar la valoración de una ruta en front-end

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
25. El usuario pulsa sobre el botón de eliminar una ruta en la leyenda del mapa	25A.Usuario no autorizado	El usuario no debe visualizar ese botón en el sistema.	El botón no se visualiza de forma correcta.	Prueba correcta.
	25B.Usuario autorizado	Mensaje indicando la eliminación de la ruta.	Mensaje indicando la eliminación correcta de la ruta.	Prueba correcta.

Tabla 84 - Pruebas de eliminar una ruta en front-end

7.2.8. Inicio y cierre de sesión

Para acceder al sistema desde el front-end de la aplicación web es necesario iniciar sesión. Por ello, a continuación, se detallan las pruebas de back-end realizadas para comprobar el buen funcionamiento de esta funcionalidad.

Prueba realizada	Rol del usuario	Resultado esperado	Resultado obtenido	¿Satisfactoria?
26. El usuario rellena correctamente su usuario y contraseña y pulsa "Iniciar Sesión"	Usuario no identificado	El usuario entra en el sistema y se le muestra la pantalla principal.	**Prueba 1:** No se inicia sesión.	Prueba incorrecta: El inicio de sesión es correcto pero el redireccionamiento de pantalla está mal realizado
			Prueba 2: El usuario inicia sesión y se le muestra la pantalla de "Mapa"	Prueba correcta.
27. El usuario rellena incorrectamente su usuario y contraseña y pulsa "Iniciar Sesión"	Usuario no identificado	Mensaje indicando que los datos no son correctos y no se puede iniciar sesión.	Mensaje indicando el error.	Prueba correcta.
28. Realizar alguna acción pasado el tiempo de validez del *Token*	Cualquier usuario que haya iniciado sesión en el sistema	Mensaje indicando que la sesión ha expirado y redireccionamiento al inicio de sesión.	Mensaje y expulsión del sistema.	Prueba correcta.
29. Cerrar sesión en el sistema	Cualquier usuario que haya iniciado sesión.	Se le muestra al usuario la pantalla de inicio de sesión.	El usuario es expulsado del sistema.	Prueba correcta.

Tabla 85 - Pruebas de inicio y cierre de sesión en front-end

7.3. Usabilidad de Walkability Analyzer

Por otra parte, uno de los objetivos de este proyecto, que se pueden visualizar en el apartado 2.2 Objetivos, es realizar una aplicación web intuitiva, en la que no se requieran amplios conocimientos informáticos para poder utilizar el sistema de una manera fácil y óptima. Una vez construido el sistema se ha evaluado sí realmente el sistema cumple con este objetivo mediante la realización de varias encuestas a diferentes usuarios.

En concreto, para evaluar la usabilidad de la aplicación web se ha utilizado el System Usability Scale (Sistema de Escalas de Usabilidad) (System Usability Scale, s.f.), un sistema sencillo y rápido para medir la usabilidad de un sistema. La encuesta consta de diez preguntas con un sistema de puntuación de cinco puntos, desde "Completamente de acuerdo" hasta "Completamente en desacuerdo" y un algoritmo rápido de puntuación para calcular el resultado.

El algoritmo del sistema de evaluación utilizado se basa en:

1. Al valor asignado por el usuario en las preguntas o afirmaciones impares, por ejemplo, la afirmación 1 o la 3, se le restará 1.

2. El valor de las afirmaciones pares se calculará como 5 menos el valor asignado por el usuario.

3. El resultado de sumar los valores obtenidos en los dos pasos anteriores se multiplicará por 2,5 para calcular el resultado final.

La muestra de personas a las que se le ha realizado el estudio es relativamente pequeña, siendo de únicamente seis usuarios: cuatro de ellos sin amplios conocimientos informáticos y dos de ellos con conocimientos propios de un usuario estándar. La muestra es así ya que, a pesar de que el objetivo es conocer sí el sistema es intuitivo para la gente sin amplia experiencia informática, también interesa la opinión de ese tipo de usuarios.

Las afirmaciones que se le realizan al usuario en el estudio son las siguientes:

- Creo que me gustaría utilizar este sistema frecuentemente.
- El sistema me resultó innecesariamente complejo.
- Creo que el sistema es bastante fácil de utilizar.
- Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.
- Creo que las diferentes funciones del sistema se encuentran muy bien integradas.
- Opino que hubo demasiada inconsistencia en el sistema.
- Imagino que la mayoría de las personas aprendería a utilizar el sistema rápidamente.
- Me sentí algo incómodo al utilizar este sistema.
- Me sentí muy seguro al utilizar este sistema.
- Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.

Una vez realizada la encuesta, y aplicando el algoritmo propio del método utilizado para calcular los resultados, la puntuación media obtenida se puede visualizar en la *Tabla 86*.

	Usuarios sin conocimientos informáticos	**Usuarios con conocimientos informáticos**
Puntuación media	80	72.5
Puntuación total	77.5	

Tabla 86 - Tabla de resultados de usabilidad

Analizando los resultados obtenidos, una puntuación total de 77.5 sobre 100 indica que, la usabilidad del sistema, tanto para usuarios con conocimientos informáticos como para usuarios sin conocimientos, resulta excelente. Esto es así ya que, según la escala de medición del System Usability Scale, las puntuaciones superiores a 68 ya se consideran superiores al promedio (System Usability Scale, s.f.). Por ello, tanto las pruebas realizadas al funcionamiento interno de la lógica del front-end para realizar las acciones, como a su interfaz gráfica, se dan por correctas.

8. Conclusiones y trabajo futuro

Después de realizar una aplicación web de esta magnitud es necesario recapitular y examinar los pasos que se han dado desde el comienzo de este proyecto hasta este punto. De esta manera se pueden analizar el trabajo realizado y los resultados obtenidos, ver qué objetivos se han cumplido realmente y si el proyecto se ha ajustado a lo establecido en el inicio, es decir, si la planificación del mismo se ha cumplido.

8.1. Evaluación de los objetivos

Al comienzo de este proyecto se marcaron unos objetivos que se deseaba cumplir, con lo que ahora, al finalizar el proyecto, es el momento de revisar sí realmente estos se han cumplido satisfactoriamente. Para ello, se analizará y se mostrarán las razones por las que un objetivo particular se da o no por cumplido.

Para Walkability Analyzer se marcaron los siguientes objetivos principales:

- *La creación de una aplicación web que permita a los investigadores el análisis de los datos recogidos permitiendo la visualización de las rutas en función de distintos criterios.*

Este objetivo, relacionado con la gestión de rutas, consistía en desarrollar una funcionalidad en la que los miembros del grupo CRIM-AP pudieran visualizar y filtrar las rutas, de manera que se pudiera centrar el estudio en determinadas características. En este aspecto, el objetivo está cumplido satisfactoriamente, ya que Walkability Analyzer permite no sólo filtrar los datos almacenados en el sistema, sino que también permite aplicar distintos criterios para visualizar dichas rutas, eliminar las rutas que puedan ser incorrectas o, dejar de visualizarlas en el mapa.

- *La aplicación debe permitir la descarga de los datos almacenados en el sistema en un formato que permita su posterior análisis con herramientas estadísticas más específicas.*

Este objetivo está cumplido ya que los usuarios que lo deseen, pueden descargar tanto las coordenadas de las rutas para analizarlas en herramientas QGIS, como la información de las mismas en formato *csv*. A pesar de ello, la ejecución de este objetivo no se ha realizado de la manera más eficiente posible, ya que el proceso de crear los objetos para descargar la información es ineficiente. Se podría haber realizado de una mejor manera, por ejemplo, creando una vista en la base de datos con la información del fichero ya en el formato adecuado, es decir, con todas las columnas necesarias, y de esta manera únicamente habría que solicitar la información necesaria a dicha vista.

- *Crear una aplicación web intuitiva, en la que no se requieran amplios conocimientos informáticos para poder utilizar el sistema de una manera fácil y óptima.*

Al definir este objetivo se indicó que, como forma de evaluación del mismo, se realizarían encuestas a diversos usuarios. Tras las encuestas realizadas, y detalladas en el capítulo O, el objetivo queda cumplido satisfactoriamente. Los resultados indican que el sistema desarrollado es intuitivo tanto para usuarios con conocimientos informáticos como para usuarios que no poseen experiencia en el ámbito informático, por lo que la usabilidad del sistema es alta. De esta manera, Walkability Analyzer puede ser utilizado de una manera sencilla por todos los usuarios que así lo requieran.

- *Aprendizaje personal en la realización de páginas web mediante la utilización de frameworks de desarrollo utilizados actualmente.*

Este objetivo personal, marcado para aumentar los conocimientos a nivel informático y de desarrollo, se ha cumplido con creces. En un principio no se conocía ningún framework de desarrollo de páginas web, por lo que el aprendizaje ha partido desde cero. En este aspecto, los conocimientos obtenidos son de gran valor, pudiendo haber desarrollado una aplicación web óptima desde el inicio. Por otro lado, el hecho de haber realizado un proyecto de esta magnitud de manera individual ha servido para aprender a gestionar otros aspectos no relacionados con la programación en sí, como puede ser la parte de estructurar un proyecto o gestionar los tiempos del mismo.

En función del grado de consecución de los objetivos principales se marcaron algunos objetivos secundarios que se analizarán a continuación:

- *Profundizar en el tipo de análisis que permita hacer la aplicación web a los investigadores.*

En este aspecto, el sistema ha sido desarrollado bajo las necesidades del grupo CRIM-AP y estas han sido cubiertas por completo, desarrollando un sistema que permite analizar las rutas en función de todos los parámetros que los usuarios finales han creído necesarios. Además, el sistema permite controlar todos los aspectos variables y configurables de la aplicación móvil, así como la corrección de municipios, por lo que este objetivo se ha dado por cumplido.

- *Desarrollo de un buscador para los usuarios en el que, en base a los datos recogidos y almacenados en el sistema, proporciona a los transeúntes la mejor ruta entre dos puntos atendiendo a los criterios que el usuario decida.*

Este objetivo no ha sido cumplido debido a la complejidad que se ha aplicado en otras partes de la aplicación web. No obstante, si se volviera a realizar el proyecto, se pondría más énfasis en este aspecto y se planificaría el proyecto de manera que pudiera ser desarrollado, ya que se considera una funcionalidad de utilidad para la ciudadanía. Los transeúntes, mediante esta funcionalidad, podrían visualizar qué partes de la ciudad están mejor acondicionadas para personas con problemas de movilidad, o podrían analizar qué rutas son más seguras e inseguras, aumentando así la calidad de vida de los ciudadanos.

- *Integración de la visualización de rutas con la visualización de espacios restauradores del proyecto Restorative City.*

Debido a la falta de tiempo en ambos proyectos, este objetivo no ha podido realizarse. A pesar de ello, tanto Walkability Analyzer como Restorative City están preparados para su integración, habiendo sido realizados bajo una estructura similar y con los mismos frameworks de desarrollo.

Por otro lado, una de las razones principales para seleccionar este Trabajo Fin de Grado y no otro era la importancia social del mismo, y la posibilidad de contribuir, aunque fuera de forma leve, en la mejora de las ciudades. En este aspecto, el grupo CRIM-AP ya ha realizado mediante Walkability Analyzer un primer estudio piloto para poder determinar las percepciones de inseguridad que posee la ciudadanía en zonas concretas de la ciudad.

Durante el primer estudio piloto, la aplicación móvil de Walkability Capturer ha permitido recopilar las rutas a pie de una muestra de estudiantes y trabajadores del campus de Ibaeta, y una vez al día recogía sus respuestas –incluyendo la percepción de seguridad- sobre una de las rutas realizadas. Sin embargo, toda la información recopilada sería de poco

valor si no existieran herramientas para la visualización y la gestión de dichos datos. Por ello, Walkability Analyzer permite al equipo investigador visualizar las rutas en el mapa de la ciudad estudiada y las respuestas al cuestionario asociado.

En el primer estudio piloto se ha empleado Walkability Analyzer para estudiar los patrones de respuesta de participantes, mujeres y hombres, para llevar a cabo un análisis de la movilidad y la percepción de inseguridad desde una perspectiva de género. Este estudio, además, ha constituido el Trabajo Fin de Grado de una alumna de la Facultad de Psicología, Elvira Blanco.

Continuar analizando datos sobre movilidad a pie y percepción de seguridad permitirá al equipo investigador localizar los lugares y momentos por los que los ciudadanos caminan sintiéndose inseguros e inseguras, estudiar. Además, permitirá estudiar otros aspectos de estos lugares, como el diseño, usos, paisaje social, y detectar así buenas o malas prácticas con las que contribuir al diseño de ciudades sostenibles y saludables, en las que toda la ciudadanía pueda moverse con una adecuada percepción de seguridad.

8.2. Planificación y su cumplimiento

Los objetivos principales del proyecto se han cumplido satisfactoriamente, ahora bien, en cuanto a la planificación temporal realizada al comienzo de este Trabajo Fin de Grado, hay que resaltar que esta ha sufrido un retraso considerable. En concreto, este retraso es debido a un parón en el proyecto de dos meses de duración, por lo que la mayoría de las tareas en sí no han sufrido grandes variaciones en cuanto al tiempo de ejecución, únicamente se han desplazado en el calendario. En la *Ilustración 34* se puede visualizar tanto el parón en el desarrollo del proyecto como el desplazamiento en el calendario de las tareas resultantes.

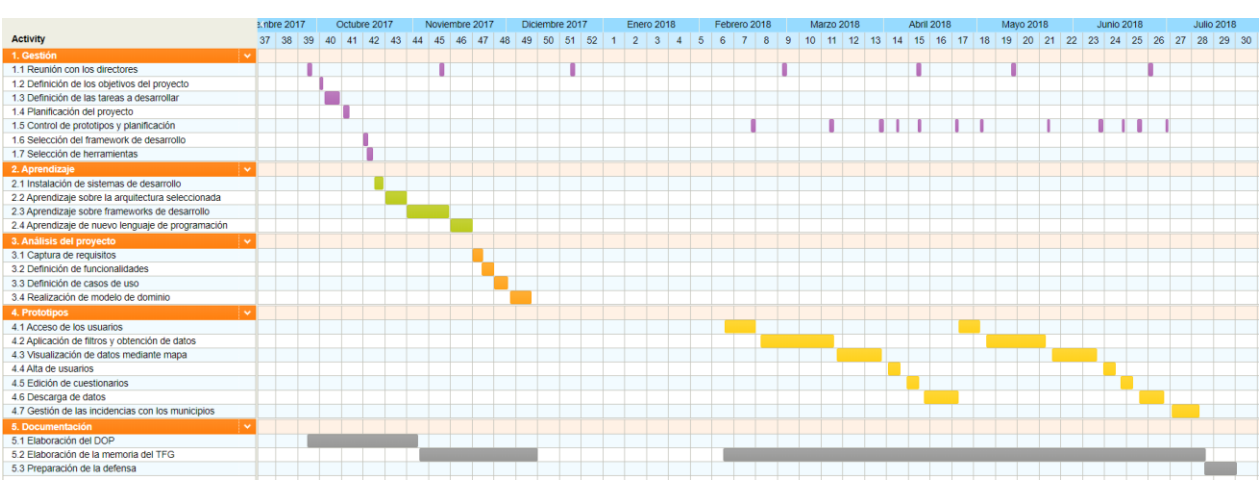


Ilustración 34 - Diagrama Gantt de la duración real del proyecto

Respecto a los paquetes de trabajo en sí que se detallaron en el apartado 2.3.2, el paquete de *Gestión*, cuyas tareas están representadas de morado, no se ha visto afectado en cuanto a tiempo total de ejecución. Esto es así ya que, a pesar de que se han reestructurado algunas de sus tareas en el calendario, por ejemplo, el *Control de prototipos y planificación*, el tiempo destinado a cada una ha sido prácticamente similar.

Por otro lado, los paquetes de trabajo tanto de *Aprendizaje*, representado en color verde, como de *Análisis del proyecto*, cuyas tareas son de color naranja, no se han visto afectados por los meses de retraso, ya que el total de sus tareas se realizó antes de la parada

en el proyecto. A pesar de ello, los tiempos totales de ejecución de sus tareas no se han ajustado fielmente a lo establecido en un inicio, aunque en el conjunto de los paquetes de trabajo el desajuste en la planificación no es grande. Por ejemplo, la tarea de "Aprendizaje sobre frameworks de desarrollo" se demoró más de lo planificado, pero como hubo otras, en concreto la de "Aprendizaje de nuevo lenguaje de programación", que se realizó en menor tiempo de lo esperado, el total de horas destinadas a la realización de las tareas de esos paquetes de trabajo no ha sufrido grandes variaciones.

La gran variación del proyecto, o la parte donde más se puede ver el desajuste entre las horas planificadas y las horas reales es en el paquete de trabajo de *Prototipos*. En él, en concreto, las tareas de "Aplicación de filtros y obtención de datos", "Visualización de datos en el mapa" y "Descarga de datos" han conllevado más horas de las esperadas debido a la complejidad del desarrollo. Por otro lado, ha sido necesario añadir un nuevo prototipo a la planificación final correspondiente a la "Gestión de las incidencias con los municipios". Este último prototipo se ha desarrollado al final del proyecto, por lo que se ha realizado de forma continua tanto la parte back-end del mismo como la front-end. El tiempo de desarrollo de este prototipo ha sido de 28 horas.

Por último, el paquete de trabajo de *Documentación* se ha realizado en menor tiempo de lo previsto, por lo que la estimación de horas iniciales ha sido superior a las horas dedicadas realmente al desempeño de sus tareas.

A continuación, en la *Ilustración 35* se puede visualizar un gráfico con las diferencias de horas establecidas al comienzo del proyecto y a su finalización, y en la *Tabla 87*, la comparativa de las mismas para indicar la diferencia total.

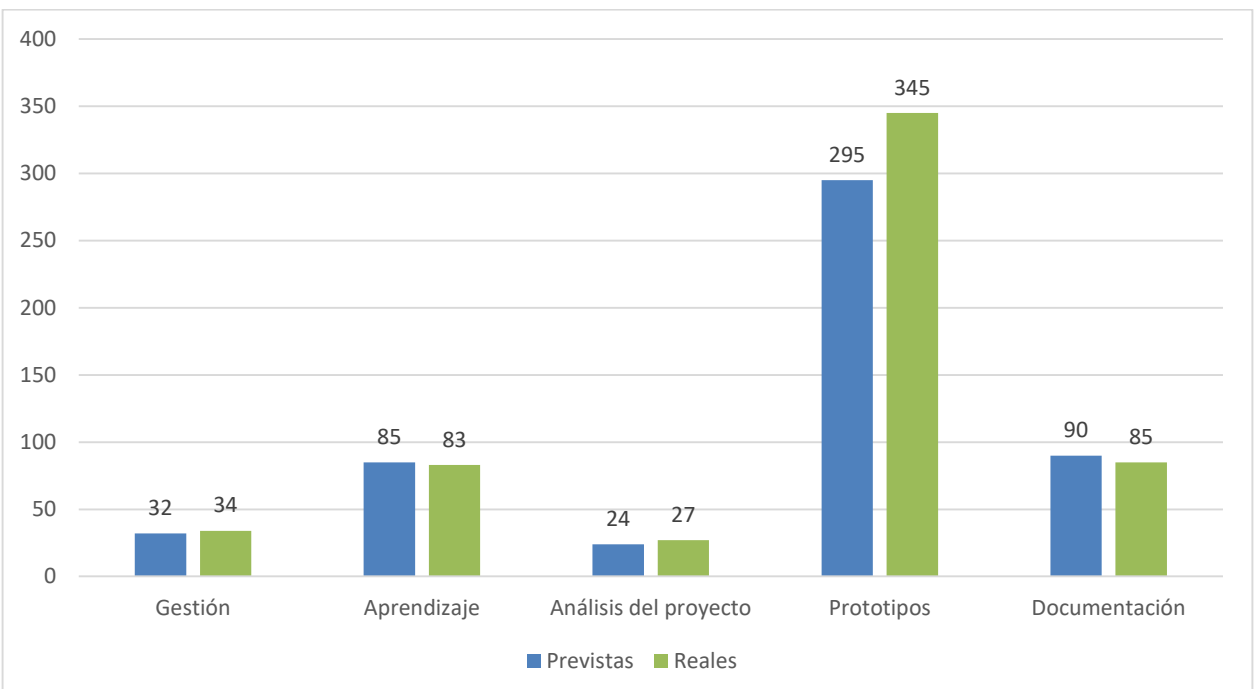


Ilustración 35 - Gráfica comparativa: horas estimadas y reales del proyecto

Paquete de trabajo	Horas estimadas	Horas reales
Gestión	32	34
Aprendizaje	85	83
Análisis del proyecto	24	27
Prototipos	295	345
Documentación	90	85
TOTAL	**526**	**574**

Tabla 87 - Tabla comparativa: horas estimadas y reales del proyecto

De la tabla anterior se deduce que el incremento a la finalización del proyecto asciende a 48 horas. Este incremento no afecta a la evaluación económica del proyecto realizada al inicio del mismo, y que se puede visualizar en el apartado 2.8. El gasto de personal para la realización del proyecto, para una duración de 9 meses, se estimó en 12688,2 €. Debido al parón de dos meses en la realización del proyecto este se ha visto retrasado, pero los meses de desarrollo se han desplazo en el calendario, siendo al final el mismo número de meses que los planificados. Por ello, el coste de personal, y a consecuencia el coste total de desarrollo de Walkability Analyzer, no se ve aumentado por el retraso total.

8.3. Riesgos y sus apariciones

Respecto a los riesgos que se analizaron en el apartado 2.7, varios de ellos han ocurrido durante la realización de este proyecto. En concreto, uno de los más destacables ha sido la *Planificación temporal incorrecta*, ya que, debido al parón en el proyecto, esta ha resultado incorrecta. Por ello, para ajustar los tiempos de dicha planificación, lo primero que se realizó fue un ajuste de la planificación en el diagrama Gantt del proyecto, para estimar de nuevo la fecha de finalización del proyecto y poder realizar los ajustes necesarios en los tiempos de los paquetes de trabajo. En este aspecto, el plan de contingencia se cumplió en su totalidad, y se pudo reajustar la planificación sin un mayor problema a posteriori.

Por otro lado, el otro riesgo que se ha producido durante el desarrollo de Walkability Analyzer son las *Variaciones en los requisitos o especificaciones del proyecto*. Cuando se encontraron los fallos geográficos de algunas de las rutas almacenadas en el sistema, así como el problema en la detección de los municipios de inicio y de fin en las rutas, se vio necesario realizar las funcionalidades oportunas para que los miembros del grupo CRIM-AP pudieran solventar dichos problemas en un futuro. Por ello, nuevos requisitos se añadieron al sistema, aumentando así la complejidad del mismo y, en consecuencia, desajustando también la planificación inicial del proyecto. A pesar de ello, al haber realizado un proyecto modular, la incorporación de dichas funcionalidades no complicó el proceso de desarrollo del resto del sistema, por lo que este riesgo, aparte de aumentar las horas de trabajo, no ha supuesto un mayor problema.

8.4. Líneas futuras

En cuanto a las futuras mejoras para el sistema de Walkability Analyzer, resaltar sobre todo la realización del buscador de rutas para la ciudadanía. Este aspecto se considera importante ya que, así, los propios ciudadanos pueden beneficiarse de primera mano de los datos almacenados en el sistema, y pueden encontrar rutas según las características que ellos deseen, por ejemplo, rutas consideradas seguras o aptas para todo tipo de movilidad.

Por otro lado, la integración de Walkability Analyzer con Restorative City, el proyecto que se ha realizado en paralelo, es un aspecto a implementar y que, debido a la comunicación existente entre los desarrolladores de ambos proyectos, es relativamente sencilla, ya que ambos sistemas están preparados para ello. Además, de esta manera se conseguiría una aplicación web común más potente y con muchas más funcionalidades.

Por último, y debido la ineficiencia en el proceso de la descarga de datos analizada en el apartado 8.1, se debería implementar esta funcionalidad de nuevo mediante la utilización de vistas en la base de datos. Al comienzo de este proyecto no se valoró dicha opción, pero estudiando la problemática existente, es lo más útil y practico ya que, aunque la utilización de vistas de gran tamaño puede aumentar un poco la rapidez del proceso, se gana en limpieza de código y se facilita la corrección de errores.

Si se analiza cuál de estas líneas futuras sería más ventajosa de aplicar al proyecto, a pesar del mayor coste que supondría su desarrollo, realizar el buscador de rutas para la ciudadanía aportaría mayor valor al proyecto, y no solo por el bien social que supondría, sino también por que daría a conocer a la sociedad un proyecto pensado por y para los ciudadanos.

8.5. Reflexión personal

A nivel personal, y comenzando por la parte técnica del proyecto, resaltar sobre todo los conocimientos adquiridos en el desarrollo de aplicaciones web mediante el uso de frameworks específicos, como Angujar o NodeJS. Después de todo el trabajo realizado y observando el resultado obtenido, así como el comportamiento final del sistema desarrollado, valoro en un futuro cercano seguir aprendiendo y profundizando más en el ámbito del desarrollo de aplicaciones. Además, he podido perfeccionar el lenguaje de JavaScript y conocer desde cero el lenguaje de TypeScript, por lo que la realización de este proyecto también me ha podido aportar conocimientos en lenguajes de programación que los navegadores son capaces de interpretar.

En cuanto a la parte personal, cabe destacar sobre todo la capacidad de superación en diferentes momentos durante el desarrollo de este proyecto. Ha sido complicado en ciertos momentos compaginar el desarrollo del proyecto, con la realización de prácticas y asistir a clase, por lo que en este aspecto mi capacidad de organización también ha mejorado. Además, el hecho de haber sido capaz de desarrollar un proyecto de esta magnitud individualmente, y sabiendo que este puede contribuir al desarrollo y mejora de las ciudades, y a mejorar la calidad de vida de los ciudadanos, supone un verdadero orgullo.

9. Bibliografía

BBVA. (23 de marzo de 2016). API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. Obtenido de https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

Blancarte, O. (6 de marzo de 2017). Software Architect.

BOLETÍN OFICIAL DEL ESTADO. (18 de enero de 2017). Obtenido de https://www.boe.es/boe/dias/2017/01/18/pdfs/BOE-A-2017-542.pdf

C. Kelly, M. T. (2010). A comparison of three methods for assessing the walkability of the pedestrian environment. Journal of Transport Geography.

Google Maps API. (s.f.). Google Maps Platform. Obtenido de https://cloud.google.com/maps-platform/?hl=es

JSON . (s.f.). Obtenido de https://www.json.org/json-es.html

Lenguajes de programación. (3 de febrero de 2015). Obtenido de https://lenguajesdeprogramacion.net/diccionario/que-es-un-framework/

Marqués, A. (11 de abril de 2013). Conceptos sobre APIs REST. Obtenido de http://asiermarques.com/2013/conceptos-sobre-apis-rest/

N. C. McDonald, E. D. (s.f.). Influence of the social environment on children's school travel. Preventive medicine.

Pain., R. (2000). Place, social relations and the fear of crime: a review. Progress in Human Geography.

Periodico ABC. (16 de febrero de 2015). ABC Consultorio. Obtenido de https://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html

QGIS. (s.f.). Obtenido de https://qgis.org/es/site/about/index.html

QuestionPro. (s.f.). Obtenido de https://www.questionpro.com/es/que-es-spss.html

Sacco., E. A. (1989). Crime and Victimisation of the Elderly. Springer- Verlag.

San Juan, C. V. (2012). Self-protective behaviours against crime in urban settings: an emprical approach to vulnerability and victimization models. European Journal of Criminology, 652-667.

SERPROGRAMADOR. (12 de abril de 2014). Obtenido de https://serprogramador.es/que-es-frontend-y-backend-en-la-programacion-web/

Solymosi, R. B. (2015). Mapping fear of crime as a context-dependent everyday experience that varies in space and time. Legal and Criminological Psychology.

System Usability Scale. (s.f.). Obtenido de https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

Vozmediano, L. S.-J.-A. (2017). “Watch out, Sweetie”: The Impact of Gender and Offence Type on Parents’ Altruistic Fear of Crime. Sex Roles, 1-11.

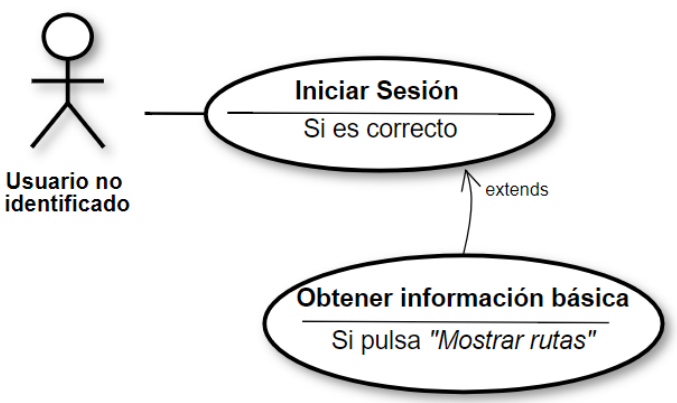
Anexo I: Casos de uso extendidos

En éste primer anexo se hará un análisis detallado de los casos de uso mencionados en la documentación. Para ello, se mostrará de forma explícita toda la información relativa al funcionamiento de cada caso de uso y a los roles que participan en él. También, se mostrará qué tipo de decisiones lógicas suceden y cómo se gestionan. Para estructurar este anexo se dividirán los casos de uso extendidos en función de su actor principal.

1. Usuario no identificado

El siguiente caso de uso extendido corresponde al usuario "Usuario no identificado".

1.1. Iniciar sesión



Nombre:	Iniciar sesión
Descripción: El usuario no identificado, mediante las credenciales correctas, puede iniciar sesión en el sistema. En el caso de que inicie sesión correctamente se le mostrará la pantalla de "Mapa".	
Actores: Usuario no identificado	
Precondiciones: Ninguna	
Requisitos no funcionales: Ninguno	
Flujo de Eventos: 1. El usuario no identificado introduce las credenciales en el formulario de inicio de sesión (*Ilustración 36*). [Si los datos son correctos] 2A. El usuario tiene acceso al sistema y se muestra la pantalla principal (*Ilustración 38*). [Si los datos no son correctos] 2B. El usuario no puede iniciar sesión en el sistema y se le muestra el error	

correspondiente (*Ilustración 37*).

Postcondiciones: En caso correcto el usuario ha iniciado sesión en el sistema

Interfaz Gráfica:

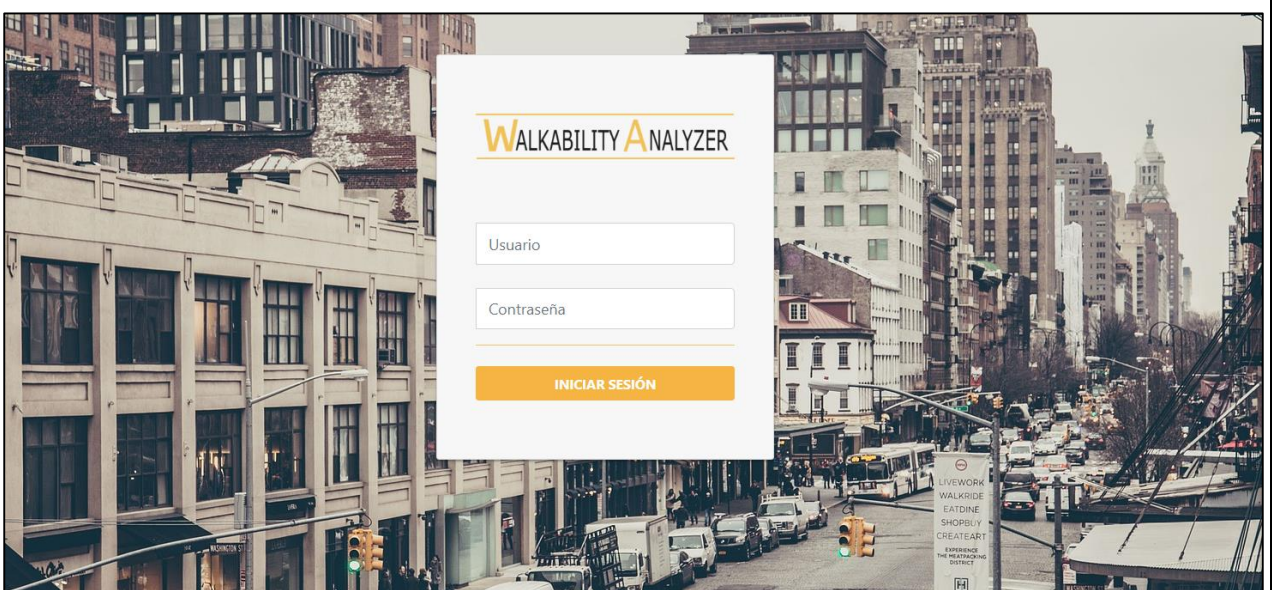


Ilustración 36 - Pantalla de iniciar sesión en el sistema

Ilustración 37 - Mensaje de error al iniciar sesión

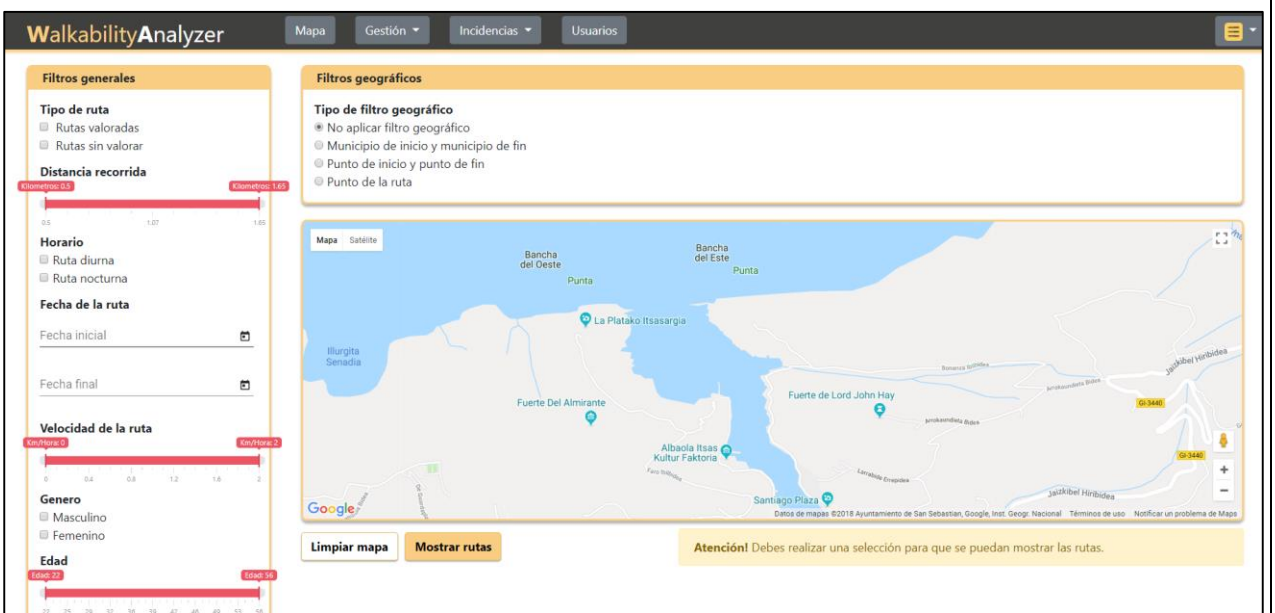
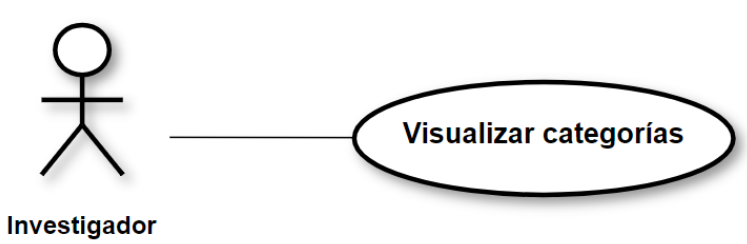


Ilustración 38 - Pantalla principal de la aplicación

2. Investigador

Los siguientes casos y sub-casos de uso son propios del usuario con rol "Investigador". Debido a la herencia existente en la jerarquía de actores de Walkability Analyzer, los casos de uso del usuario "Investigador" también pueden ser realizados por el rol de "Administrador".

2.1. Visualizar categorías



Nombre:	Visualizar categorías
Descripción: Permite al usuario obtener y visualizar las categorías de las preguntas del sistema.	

Actores: Investigador y Administrador

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa en el botón de "Gestión > Categorías" situado en el menú de navegación (*Ilustración 39*).

   [Si los datos se cargan correctamente]

   2A. Se le muestra una pantalla donde puede visualizar mediante una tabla todas las categorías existentes en el sistema (*Ilustración 40*).

   [Si los datos no se cargan correctamente]

   2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: La información se visualiza en pantalla.

Interfaz Gráfica:

Ilustración 39 - Botón de categorías en el menú de navegación

Ilustración 40 - Pantalla de visualizar categorías

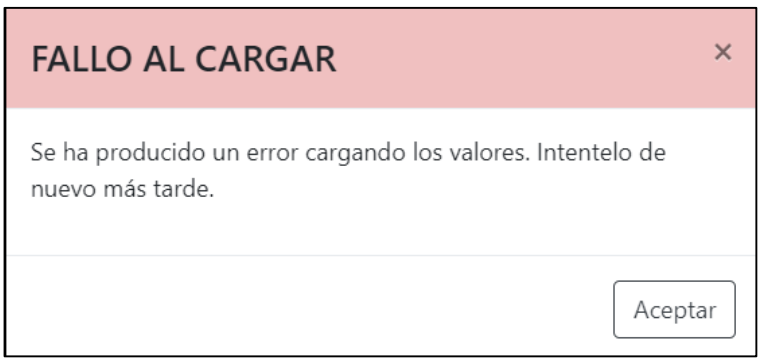
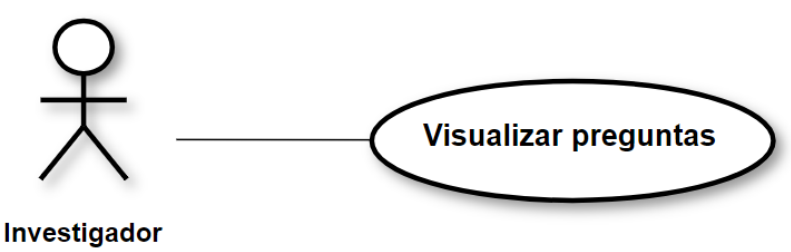


Ilustración 41 - Mensaje de error al cargar los datos

2.2. Visualizar preguntas



Nombre:	Visualizar preguntas
Descripción: Permite al usuario obtener y visualizar toda la información referente a las preguntas almacenadas en el sistema.	
Actores: Investigador y Administrador	
Precondiciones: Ninguna	
Requisitos no funcionales: Ninguno	

Flujo de Eventos:

2. El usuario pulsa en el botón de "Gestión > Preguntas" situado en el menú de navegación (*Ilustración 42*).

   [Si los datos se cargan correctamente]

   2A. Se le muestra una pantalla donde puede visualizar mediante una tabla todas las preguntas existentes en el sistema, así como información de las mismas (*Ilustración 43*).

[Si el usuario pulsa sobre una de las preguntas]

2A A. Se le muestra al usuario la información completa de la pregunta, así como sus posibles respuestas (*Ilustración 44*).

[Si los datos no se cargan correctamente]

2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: La información se visualiza en pantalla.

Interfaz Gráfica:

Ilustración 42 - Botón de preguntas en el menú de navegación

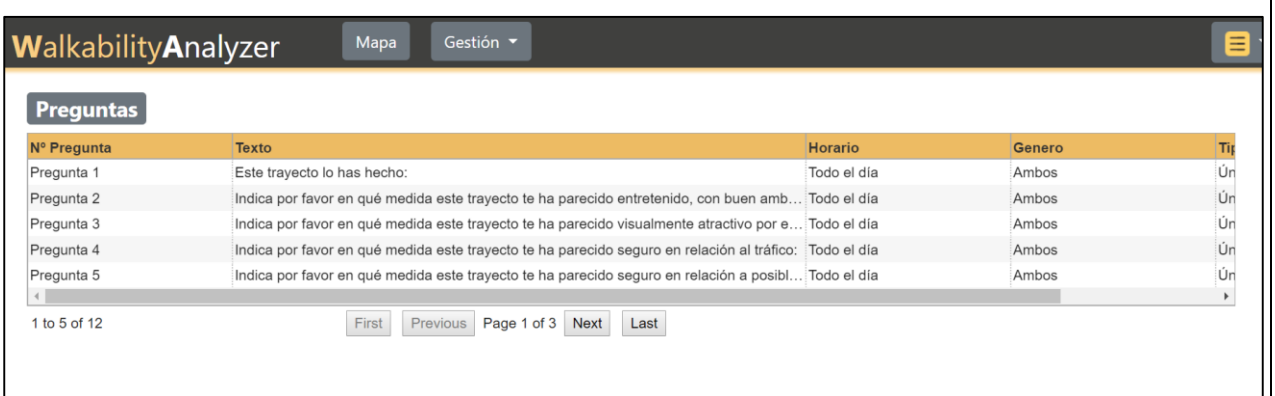


Ilustración 43 - Pantalla de visualizar preguntas

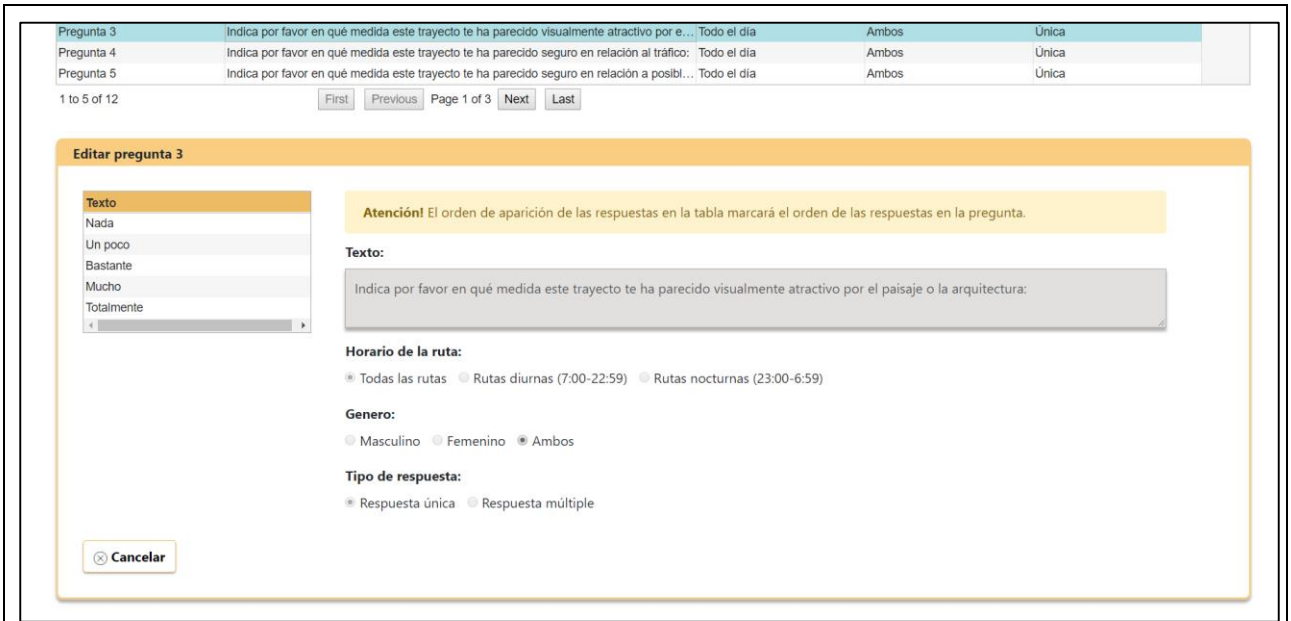
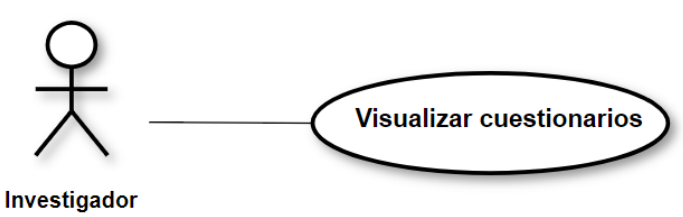


Ilustración 44 - Tabla de información completa sobre la pregunta

2.3. Visualizar cuestionarios



Nombre:	Visualiza cuestionarios
Descripción: Permite al usuario obtener y visualizar toda la información referente a los cuestionarios almacenados en el sistema.	
Actores: Investigador y Administrador	
Precondiciones: Ninguna	
Requisitos no funcionales: Ninguno	
Flujo de Eventos: 1. El usuario pulsa en el botón de "Gestión > Cuestionarios" situado en el menú de navegación (*Ilustración 45*). [Si los datos se cargan correctamente] 2A. Se le muestra una pantalla donde puede visualizar mediante una tabla todos	

los cuestionarios existentes en el sistema. (*Ilustración 46*).

[Si el usuario pulsa sobre uno de los cuestionarios]

2 AA. Se le muestra al usuario la información completa del cuestionario en particular, así como las preguntas que lo componen. (*Ilustración 47*).

[Si los datos no se cargan correctamente]

2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: La información se visualiza en pantalla.

Interfaz Gráfica:

Ilustración 45 - Botón de cuestionarios en el menú de navegación

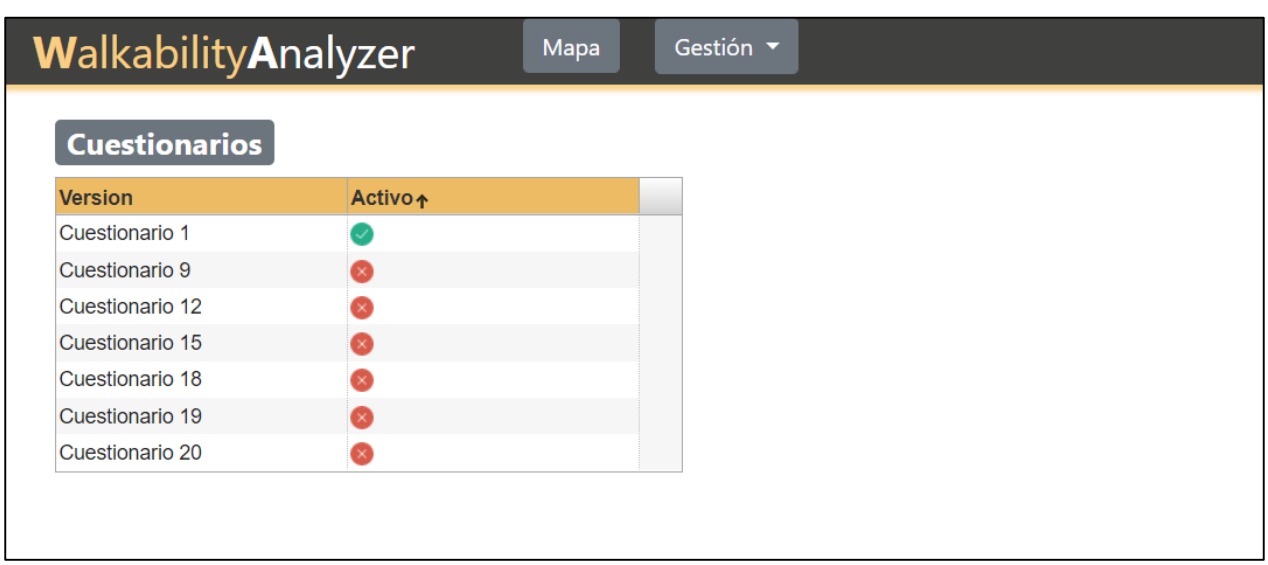


Ilustración 46 - Pantalla de visualizar cuestionarios

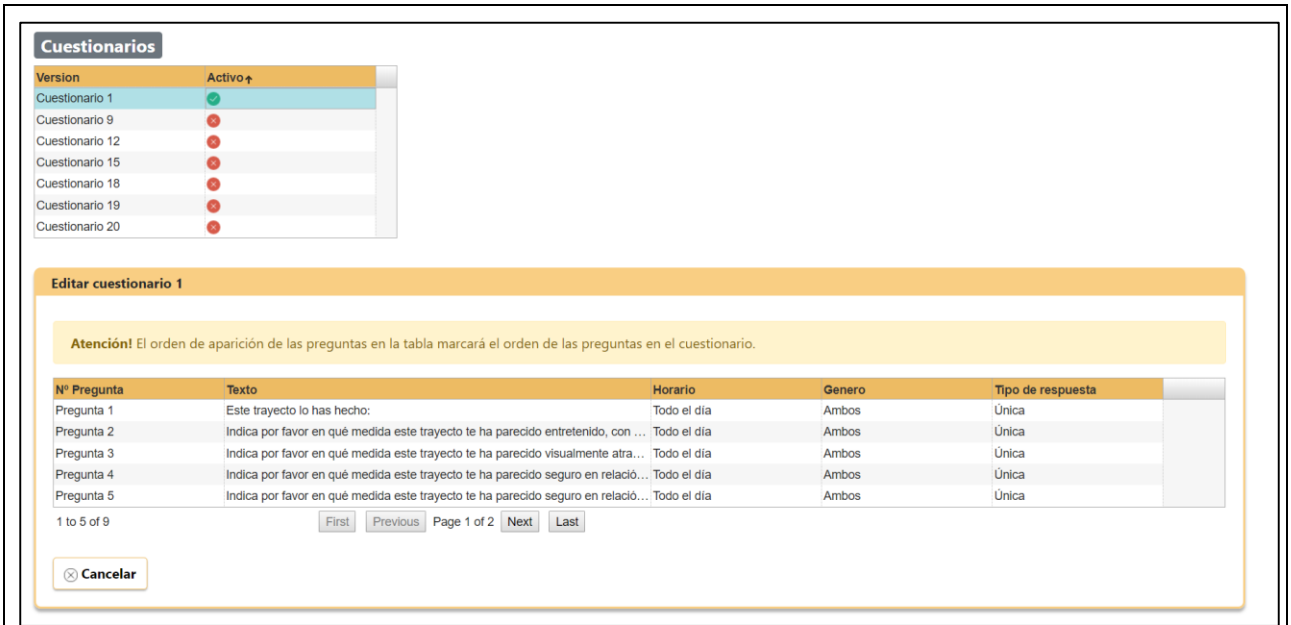
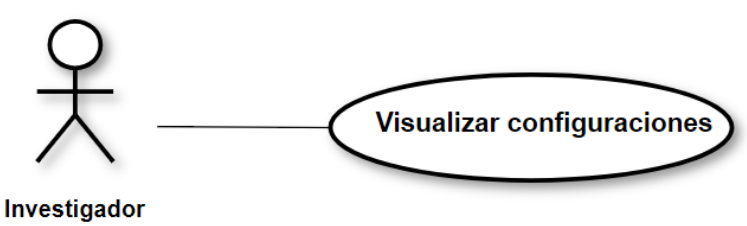


Ilustración 47 - Tabla de información completa sobre el cuestionario

2.4. Visualizar configuraciones



Nombre:	Visualizar configuraciones
Descripción: Permite al usuario obtener y visualizar toda la información referente a las configuraciones almacenadas en el sistema.	
Actores: Investigador y Administrador	
Precondiciones: Ninguna	
Requisitos no funcionales: Ninguno	
Flujo de Eventos: 1. El usuario pulsa en el botón de "Gestión > Configuraciones" situado en el menú de navegación (*Ilustración 48*). [Si los datos se cargan correctamente] 2A. Se le muestra una pantalla donde puede visualizar mediante una tabla	

todas las configuraciones existentes en el sistema, así como los valores que las componen. (*Ilustración 49*).

[Si los datos no se cargan correctamente]

2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

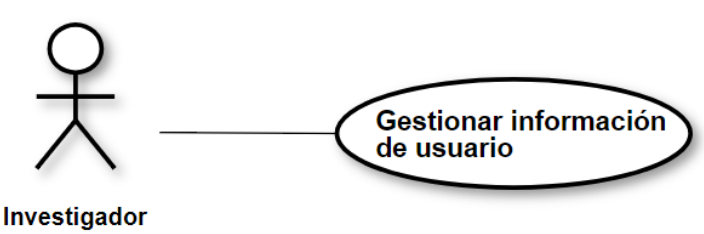
Postcondiciones: La información se visualiza en pantalla.

Interfaz Gráfica:

Ilustración 48 - Botón de configuraciones en el menú de navegación

Ilustración 49 - Pantalla de visualización de configuraciones

2.5. Gestionar información de usuario



Nombre:	Gestionar información de usuario

Descripción: Permite al usuario poder visualizar su información personal y modificarla si

así lo desea.

Actores: Investigador y Administrador

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa en el botón de "Mi perfil" situado en el menú de navegación en la zona derecha de la pantalla (*Ilustración 50*).

   [Si los datos se cargan correctamente]

   2A. Se le muestra una pantalla donde puede visualizar mediante una tabla todas las configuraciones existentes en el sistema, así como los valores que las componen. (*Ilustración 51*).

   [Si el usuario edita la información y pulsa "Guardar"]

   [Si los datos se almacenan correctamente]

   2 AA. Se le muestra al usuario un mensaje de confirmación. (*Ilustración 52*).

   [Si los datos no se almacenan correctamente]

   2 AB. Se le muestra al usuario un mensaje de error. (*Ilustración 53*).

   [Si los datos no se cargan correctamente]

   2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: La información del usuario se ha actualizado en el sistema.

Interfaz Gráfica:

Ilustración 50 - Botón de "Mi perfil" en el menú de navegación

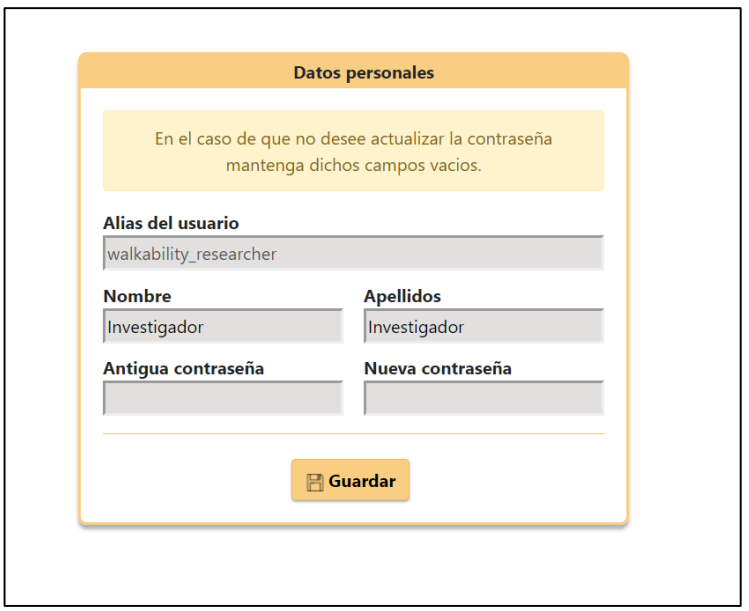


Ilustración 51 - Tabla de visualización y edición de datos personales

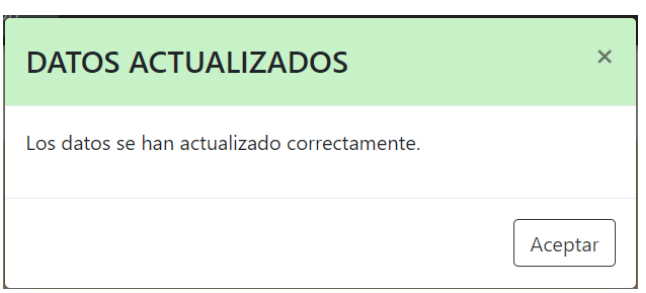


Ilustración 52 - Mensaje de confirmación en la actualización de datos

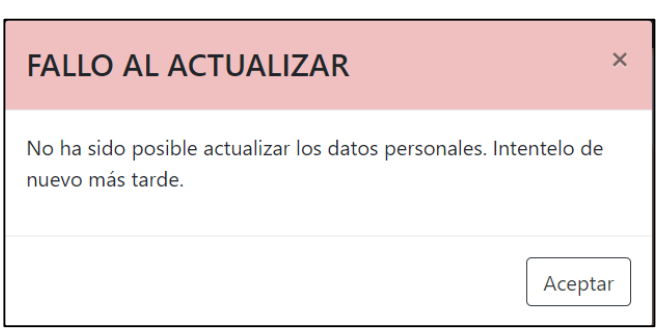
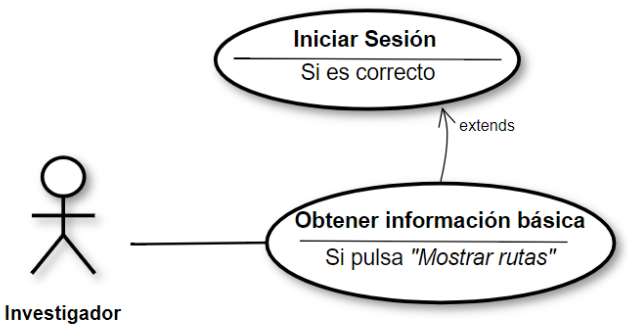


Ilustración 53 - Mensaje de error en la actualización de datos

2.6. Obtener información básica



Nombre:	Obtener información básica

Descripción: Tanto si el usuario inicia sesión en el sistema, como si pulsa el botón de "Mapa", se cargan los datos necesarios para mostrar la pantalla y poder operar con ella.

Actores: Investigador y Administrador

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario inicia sesión correctamente en el sistema o, cuando ya está utilizando la aplicación web, pulsa en el botón "Mapa" situado en la barra de navegación (*Ilustración 54*).

   [Si los datos se cargan correctamente]

   2A. Se muestra en la pantalla la información necesaria para poder utilizar los filtros del sistema: las categorías de las rutas y las versiones de cuestionarios existentes (*Ilustración 55*), la distancia mínima y máxima de las rutas realizadas alojadas en el sistema, el rango de edad de los usuarios de la aplicación y la velocidad de las rutas (*Ilustración 56*), y, por último, los municipios de inicio y de fin.

   [Si los datos no se cargan correctamente]

   2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: La información se visualiza en pantalla.

Interfaz Gráfica:

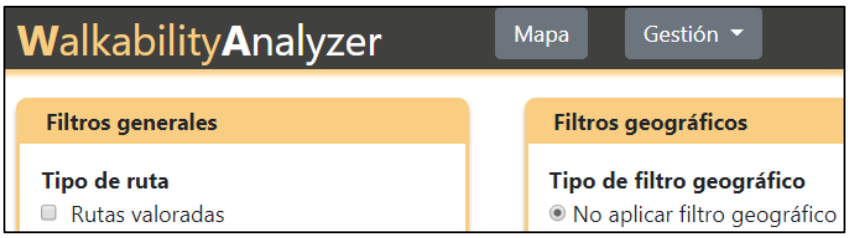


Ilustración 54 - Botón de mapa en el menú de navegación

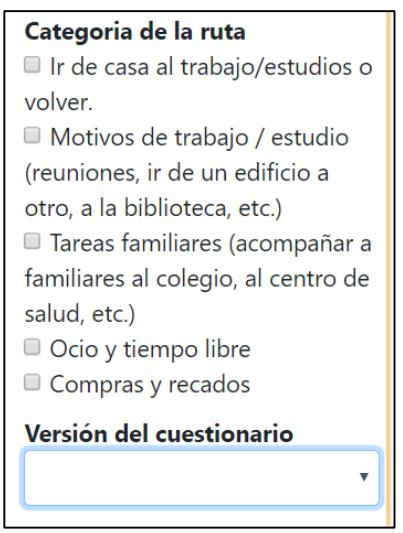


Ilustración 55 - Información básica a cargar (I)

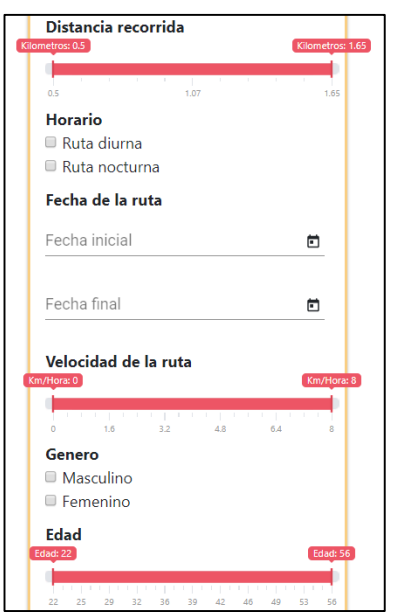
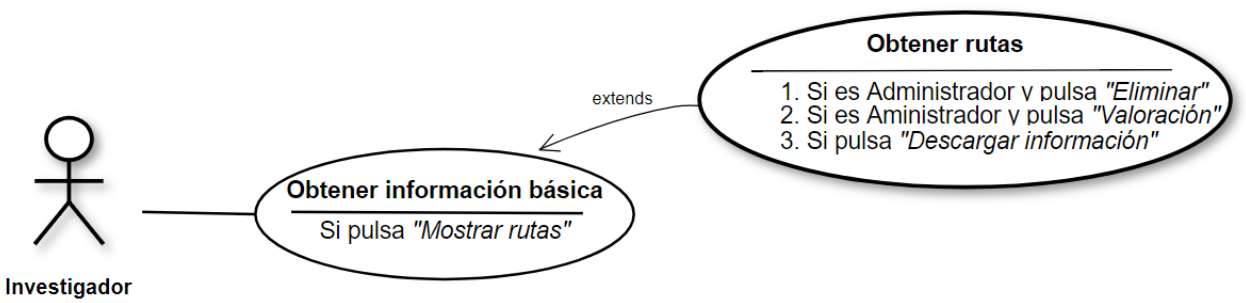


Ilustración 56 - Información básica a cargar (II)

En este punto se detallarán los sub-casos de uso del caso de uso "Obtener información básica". Los sub-casos de uso en los que se ha divido esta funcionalidad, de manera que se pueda analizar de una forma más detallada, son los siguientes:

Obtener rutas



Nombre:	Obtener rutas

Descripción: Una vez la información básica se ha cargado correctamente si el usuario lo desea, pulsando el botón "Mostrar rutas" puede obtener y visualizar las rutas que cumplan los filtros que haya aplicado.

Actores: Investigador y Administrador

Precondiciones: Es necesario tener la información básica cargada.

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario aplica los filtros deseados y pulsa el botón de "Mostrar rutas" para obtener las rutas y visualizarlas. (*Ilustración 57*)

   [Si las rutas se obtienen correctamente]

   2A. Se muestran en la pantalla las rutas que se ajustan a los filtros aplicados (*Ilustración 58*).

   [Si las rutas no se obtienen correctamente]

   2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: Las rutas se visualizan en el mapa y aparece en pantalla la leyenda del mapa junto con las opciones avanzadas de visualización.

Interfaz Gráfica:

Ilustración 57 - Botón de "Mostrar rutas" de la pantalla del mapa

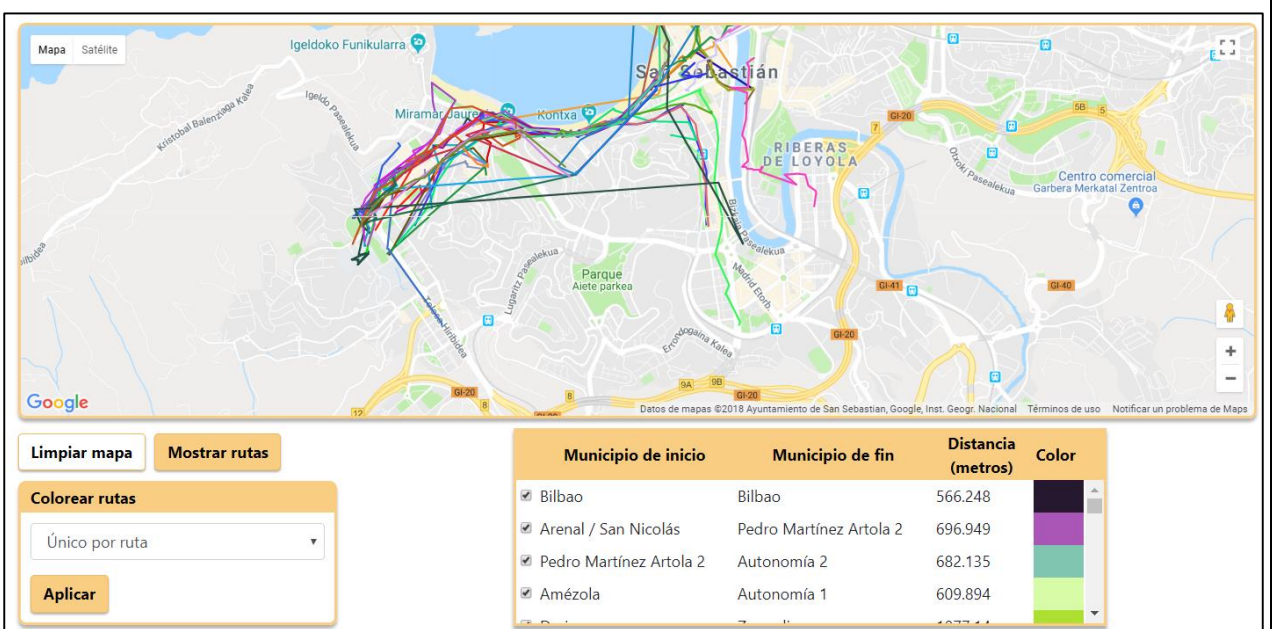
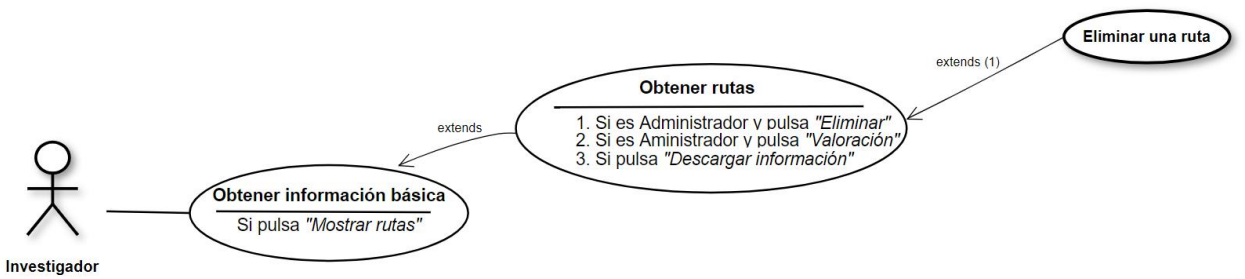


Ilustración 58 - Visualización de rutas en el mapa

Dentro del sub-caso de uso de “Obtener rutas” existen más sub-casos según la acción que realice el usuario, en concreto hay tres opciones:

- **Eliminar una ruta:** Este sub-caso de uso únicamente está disponible para los actores con rol de “Administrador”.
- **Gestionar valoración de una ruta**: Este sub-caso de uso únicamente está disponible para los actores con rol de “Administrador”.
- **Descargar información de las rutas:** Este sub-caso de uso está disponible tanto para los actores con rol de “Administrador” como para los actores con rol “Investigador”.

Eliminar una ruta



Nombre:	Eliminar una ruta

Descripción: Una vez la información básica se ha cargado correctamente y las rutas se están visualizando en el mapa, si el actor desea, puede eliminar las rutas.

Actores: Administrador

Precondiciones: Es necesario tener la información básica cargada y las rutas obtenidas en el mapa.

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa el botón de eliminar la ruta presente en la leyenda de las rutas que se muestran en el mapa. Este botón solo está disponible para el actor "Administrador". (*Ilustración 59*)

   [Si la ruta se ha eliminado correctamente]

   2A. Se muestra en la pantalla un mensaje de confirmación y la ruta se deja de visualizar en el mapa (*Ilustración 60*).

   [Si la ruta no se ha eliminado correctamente]

   2B. Se le muestra un mensaje de error al usuario. (*Ilustración 61*).

Postcondiciones: La ruta eliminada se deja de visualizar en el mapa.

Interfaz Gráfica:

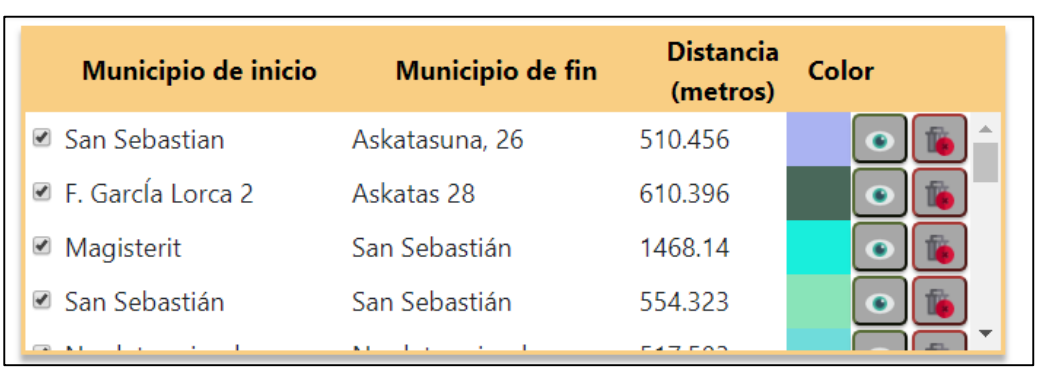


Ilustración 59 - Botón de eliminación de rutas en la leyenda del mapa

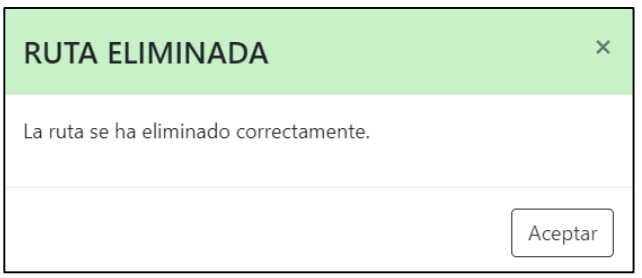


Ilustración 60 - Mensaje de confirmación al eliminar rutas

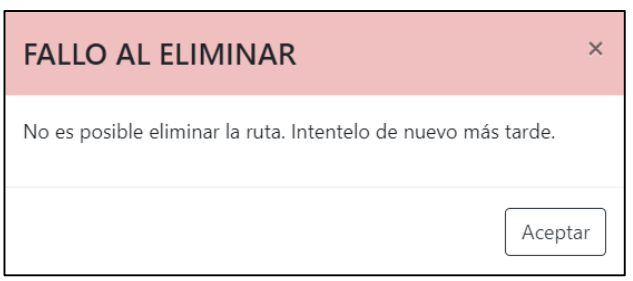
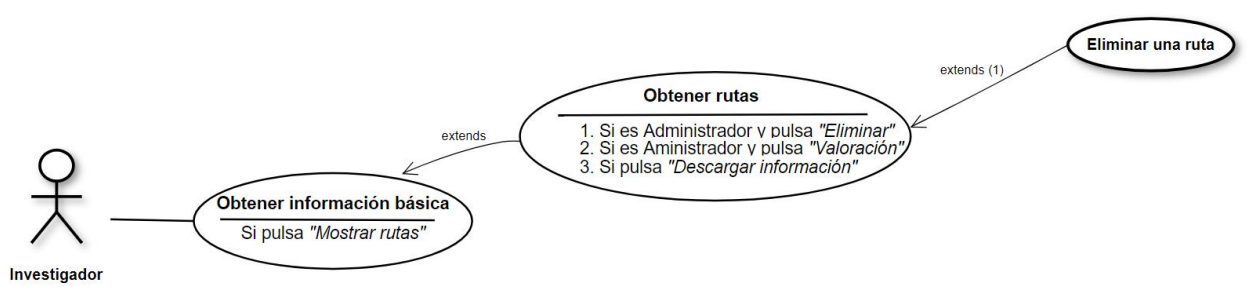


Ilustración 61 - Mensaje de error al eliminar rutas

Gestionar valoración de una ruta



Nombre:	Gestionar valoración de una ruta

Descripción: Una vez la información básica se ha cargado correctamente y las rutas se están visualizando en el mapa, si el actor desea, puede visualizar la valoración de las rutas que han sido valoradas y posteriormente, puede eliminar dichas valoraciones.

Actores: Administrador

Precondiciones: Es necesario tener la información básica cargada y las rutas obtenidas en el mapa.

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa el botón de visualizar la valoración de la ruta presente en la leyenda del mapa. Este botón solo está disponible para el actor “Administrador”.

(*Ilustración 62*)

[Si la información de la valoración de obtiene correctamente]

2A. Se muestra en la pantalla un pop-up con la valoración de la ruta seleccionada (*Ilustración 63*).

[Si el usuario pulsa "Eliminar valoración"]

[Si la valoración se ha eliminado correctamente]

3A. Se muestra en la pantalla un mensaje de confirmación. (*Ilustración 64*).

[Si la ruta no se ha eliminado correctamente]

3B. Se le muestra un mensaje de error al usuario. (*Ilustración 65*).

[Si la ruta no se ha eliminado correctamente]

2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: La valoración de la ruta se ha eliminado del sistema.

Interfaz Gráfica:

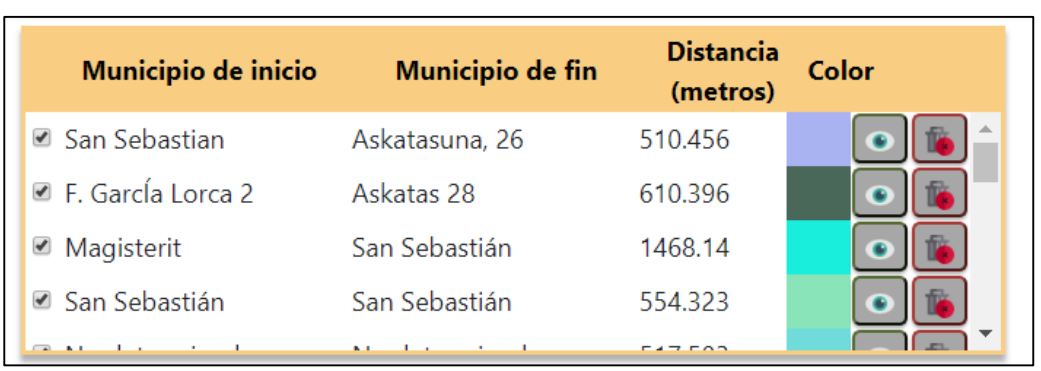


Ilustración 62 - Botón de visualización de la valoración de las rutas

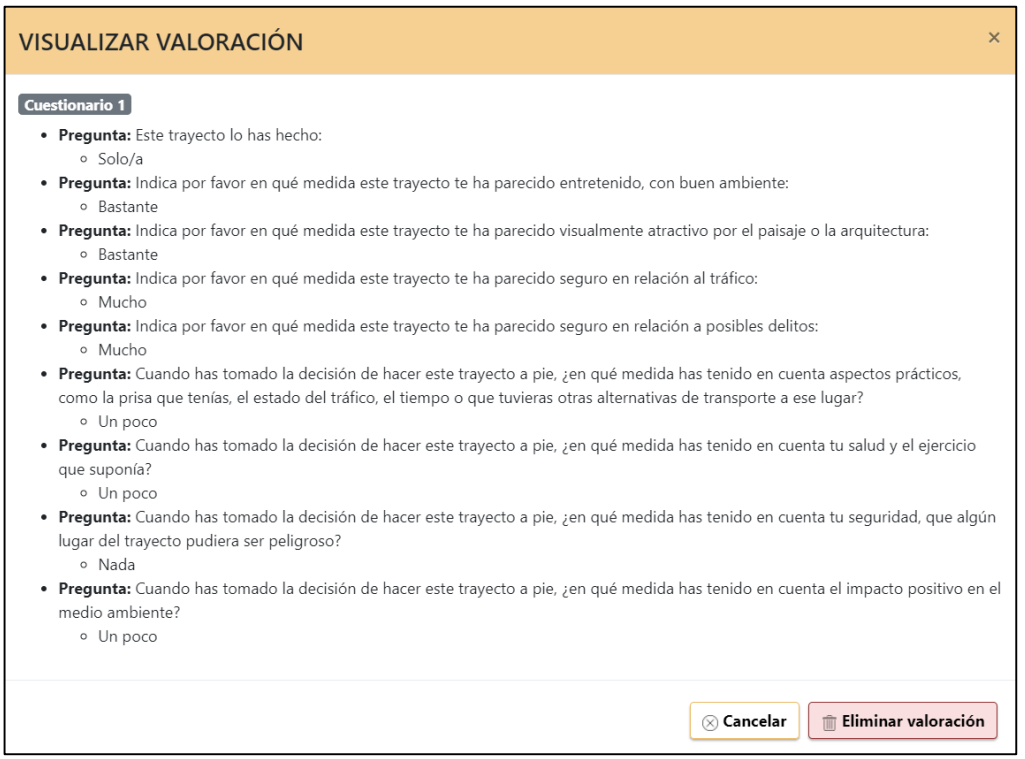


Ilustración 63 - Mensaje pop-up para la muestra de las valoraciones

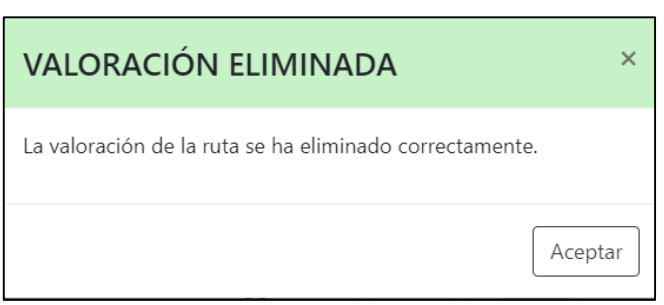


Ilustración 64 - Mensaje de confirmación de valoración eliminada

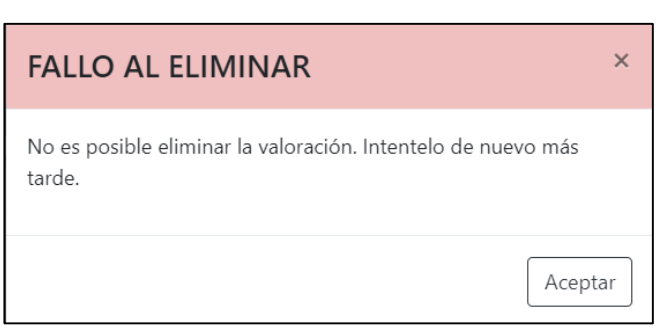
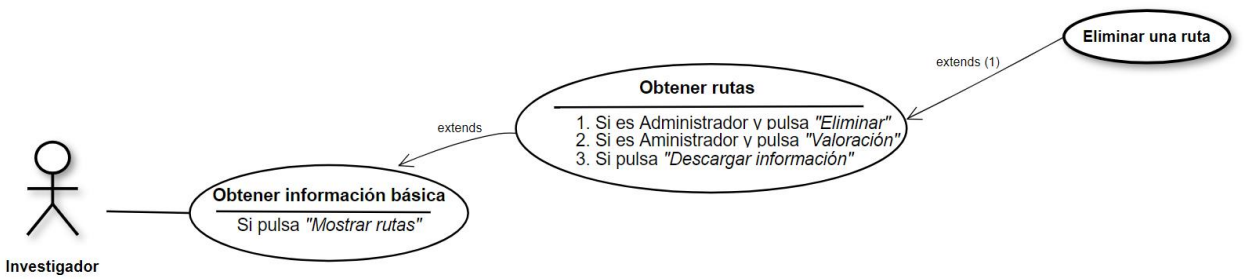


Ilustración 65 - Mensaje de error al eliminar una valoración

Descargar información de las rutas



Nombre:	Descargar información de las rutas

Descripción: Una vez la información básica se ha cargado correctamente y las rutas se están visualizando en el mapa, si el actor desea, puede descargar la información de las rutas en el formato que desee.

Actores: Investigador y Administrador

Precondiciones: Es necesario tener la información básica cargada y las rutas obtenidas en el mapa.

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario selecciona el formato deseado y pulsa el botón “Descargar”. (*Ilustración 66*)

   [Si la información de las rutas se ha descargado correctamente]

   2A. Se descarga el archivo en el propio navegador.

   [Si la información de las rutas no se ha descargado correctamente]

   2B. Se le muestra un mensaje de error al usuario. (*Ilustración 41*).

Postcondiciones: La información de las rutas se ha descargado en el formato deseado.

Interfaz Gráfica:

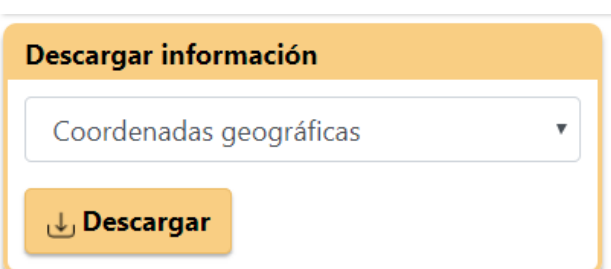
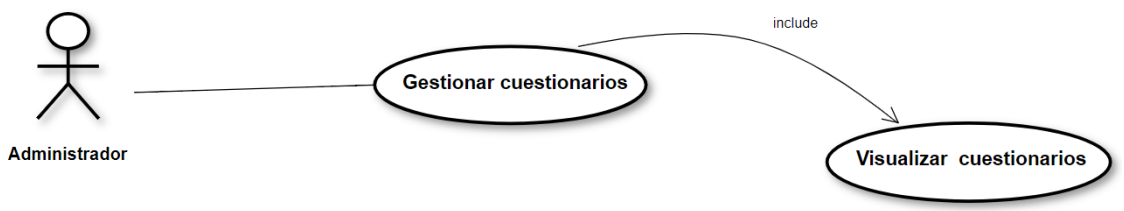


Ilustración 66 - Botón para descargar la información de las rutas

3. Administrador

Los siguientes casos de uso son propios del usuario con rol "Administrador". Debido a la herencia existente en la jerarquía de actores de Walkability Analyzer, los casos de uso del usuario "Investigador" también pueden ser realizados por el rol de "Administrador".

3.1. Gestionar cuestionarios



Nombre:	Gestionar cuestionarios

Descripción: El usuario "Administrador" puede visualizar los cuestionarios y posteriormente, si lo desea, puede realizar diversas gestiones sobre ellos: eliminarlos, activarlos, editarlos e incluso crear otros nuevos.

Actores: Administrador

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa en el botón de "Gestión > Cuestionarios" situado en el menú de navegación (*Ilustración 45*). ***INCLUDE → Gestionar cuestionarios***

   [Si el usuario pulsa "Añadir cuestionario" (*Ilustración 67*)]

   2A. Se muestra una pantalla donde el usuario debe crear el cuestionario y seleccionar las preguntas que lo forman (*Ilustración 68*).

   [Si el usuario pulsa "Guardar"]

   [Si el cuestionario se ha añadido correctamente]

   3AAA. Se le muestra al usuario un mensaje de confirmación (*Ilustración 69*).

   [Si el cuestionario no se ha añadido correctamente]

   3AAB. Se le muestra al usuario un mensaje de error (*Ilustración 70*).

   [Si el usuario pulsa "Cancelar"]

   3AB. La pantalla de la *Ilustración 68*, que permite añadir un cuestionario,

se deja de visualizar.

[Si el usuario pulsa para visualizar un cuestionario de la vista y pulsa “Eliminar cuestionario” (*Ilustración 71*)]

[Si es posible eliminar el cuestionario]

3BA. Se muestra un mensaje indicando que se ha eliminado correctamente. (*Ilustración 73*)

[Si no es posible eliminar el cuestionario]

3BB. Se muestra un mensaje indicando el error (*Ilustración 72*).

[Si el usuario pulsa para visualizar un cuestionario de la vista y pulsa “Guardar” (*Ilustración 71*)]

[Si es posible actualizar el cuestionario]

3CA. Se muestra un mensaje indicando que se ha actualizado correctamente. (*Ilustración 74*)

[Si no es posible actualizar el cuestionario]

3CB. Se muestra un mensaje indicando el error (*Ilustración 75*).

[Si el usuario pulsa para visualizar un cuestionario de la vista y pulsa “Activar” (*Ilustración 71*)]

[Si es posible activar el cuestionario]

3DA. Se muestra un mensaje indicando que se ha activado correctamente. (*Ilustración 76*)

[Si no es posible activar el cuestionario]

3DB. Se muestra un mensaje indicando el error (*Ilustración 77*).

Postcondiciones: Ninguna

Interfaz Gráfica:

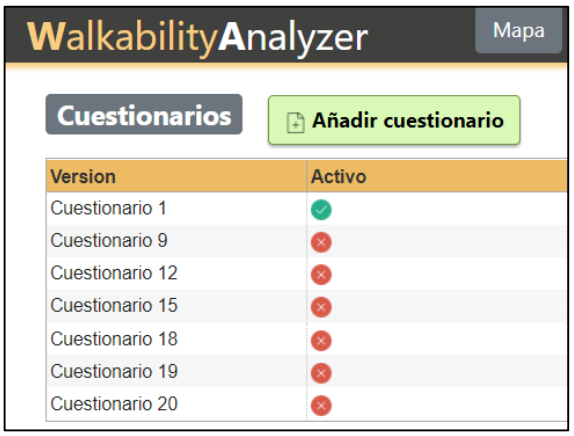


Ilustración 67 - Botón para añadir cuestionarios

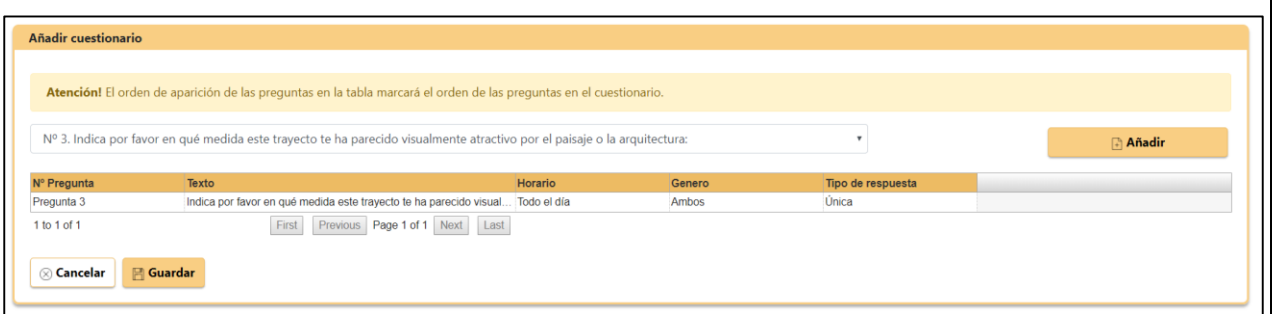


Ilustración 68 - Pantalla para añadir cuestionarios

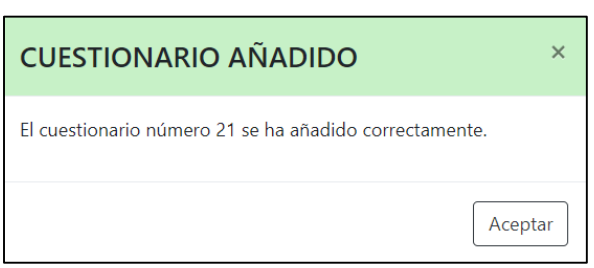


Ilustración 69 - Mensaje de confirmación para los cuestionarios añadidos

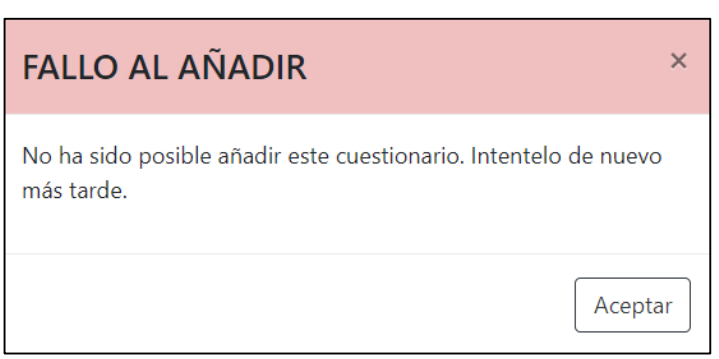


Ilustración 70 - Mensaje de error al añadir cuestionarios

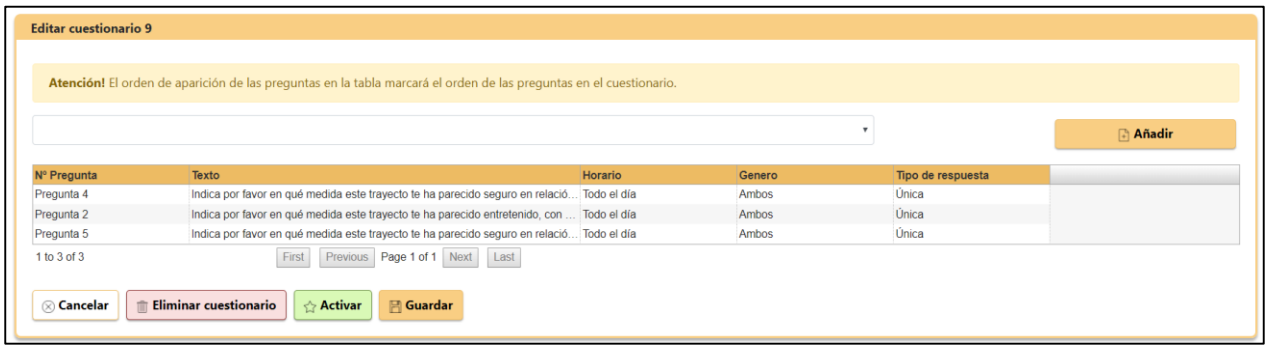


Ilustración 71 - Pantalla de visualización de cada cuestionario

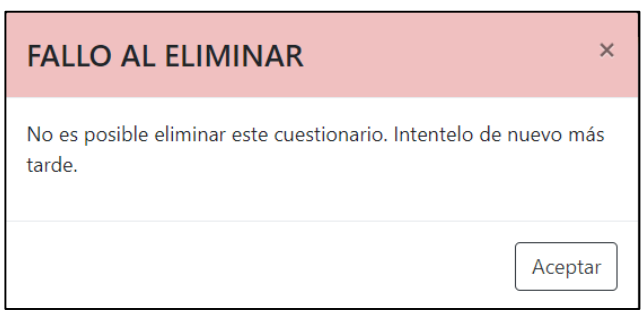


Ilustración 72 - Mensaje indicando el error al eliminar un cuestionario

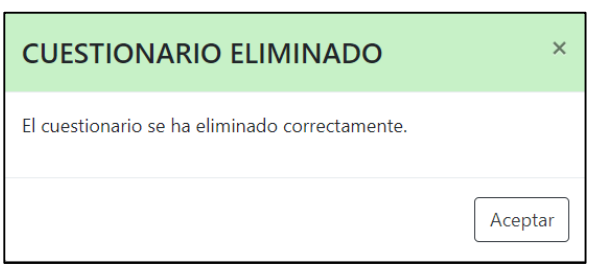


Ilustración 73 - Mensaje indicando que el cuestionario se ha eliminado correctamente

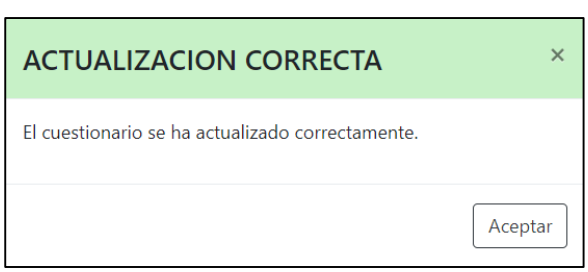


Ilustración 74 - Mensaje indicando la actualización correcta de cuestionarios

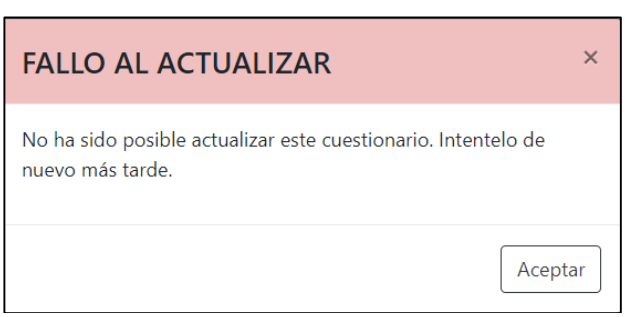


Ilustración 75 - Mensaje de error al actualizar un cuestionario

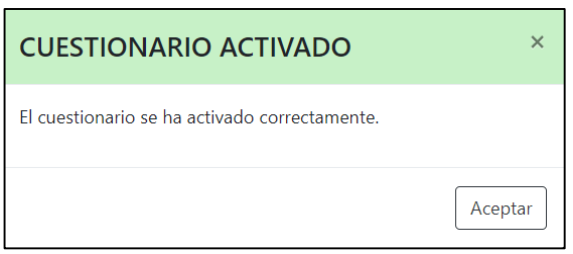


Ilustración 76 - Mensaje de confirmación al activar un cuestionario

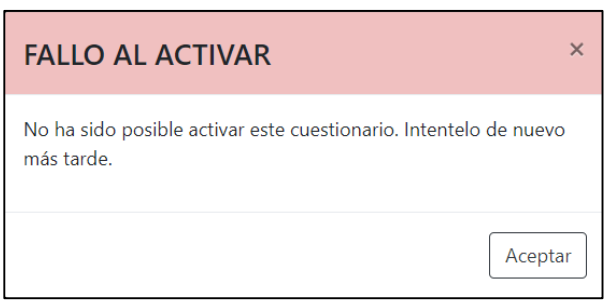


Ilustración 77 - Mensaje de error al activar un cuestionario

3.2. Gestionar configuraciones

Nombre:	Gestionar configuraciones
Descripción: El usuario "Administrador" puede visualizar las configuraciones y posteriormente, si lo desea, puede realizar diversas gestiones sobre ellas: eliminarlas, activarlas e incluso crear otros nuevos.	
Actores: Administrador	
Precondiciones: Ninguna	
Requisitos no funcionales: Ninguno	
Flujo de Eventos:	

1. El usuario pulsa en el botón de "Gestión > Configuraciones" situado en el menú de navegación (*Ilustración 45*). ***INCLUDE → Visualizar configuraciones***

   [Si el usuario pulsa "Añadir configuración" (*Ilustración 78*)]

   2A. Se muestra una pantalla donde el usuario debe crear la configuración e indicar los valores que la forman (*Ilustración 79*).

   [Si el usuario pulsa "Guardar"]

   [Si la configuración se ha añadido correctamente]

   3AAA. Se le muestra al usuario un mensaje de confirmación (*Ilustración 81*).

   [Si la configuración no se ha añadido correctamente]

   3AAB. Se le muestra al usuario un mensaje de error (*Ilustración 82*).

   [Si el usuario pulsa "Cancelar"]

   3AB. La pantalla de la *Ilustración 79*, que permite añadir una configuración, se deja de visualizar.

   [Si el usuario pulsa para visualizar una configuración de la vista y pulsa "Eliminar configuración" (*Ilustración 80*)]

   [Si es posible eliminar la configuración]

   3BA. Se muestra un mensaje indicando que se ha eliminado correctamente. (*Ilustración 83*)

   [Si no es posible eliminar la configuración]

   3BB. Se muestra un mensaje indicando el error (*Ilustración 84*).

   [Si el usuario pulsa para visualizar una configuración de la vista y pulsa "Activar" (*Ilustración 80*)]

   [Si es posible activar la configuración]

   3CA. Se muestra un mensaje indicando que se ha activado correctamente. (*Ilustración 85*)

   [Si no es posible activar la configuración]

   3CB. Se muestra un mensaje indicando el error (*Ilustración 86*).

Postcondiciones: Ninguna

Interfaz Gráfica:

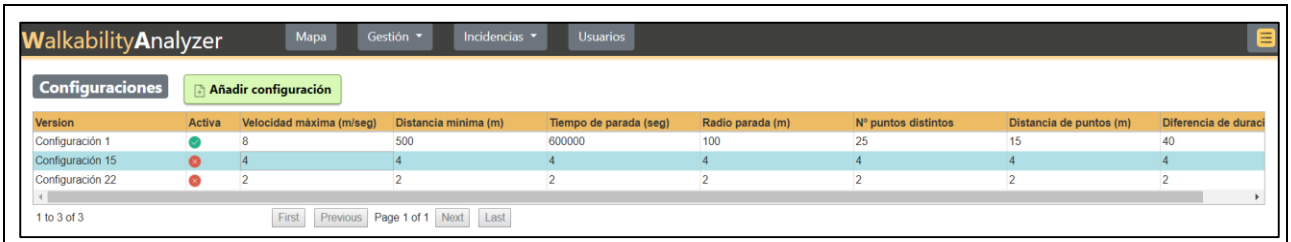


Ilustración 78 - Botón para añadir configuraciones

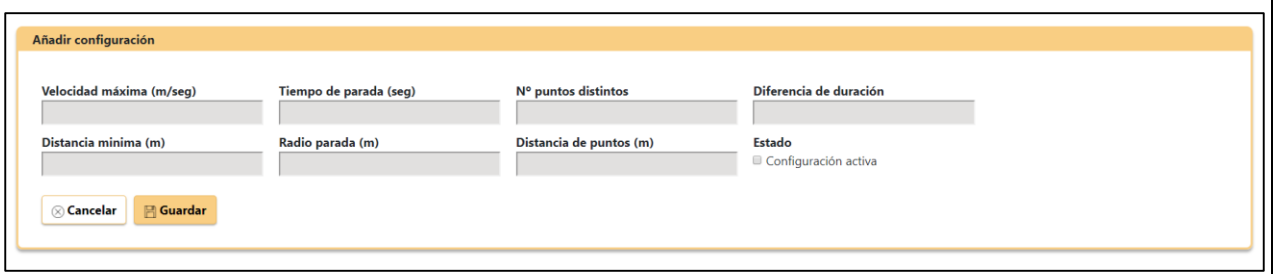


Ilustración 79 - Pantalla para añadir una configuración

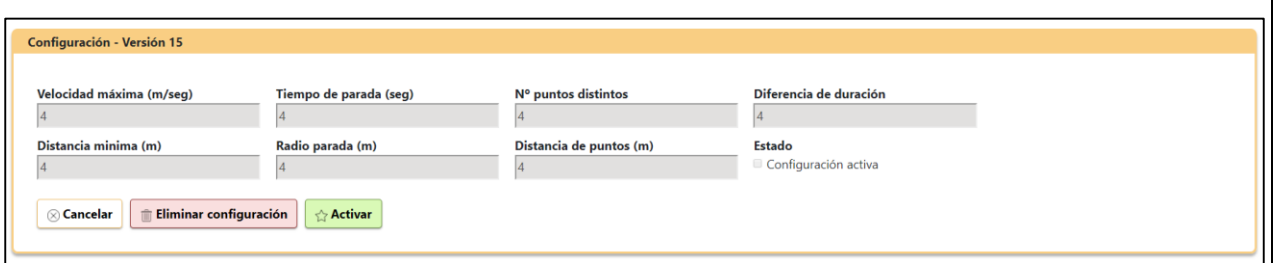


Ilustración 80 - Pantalla de visualización de los datos de cada configuración

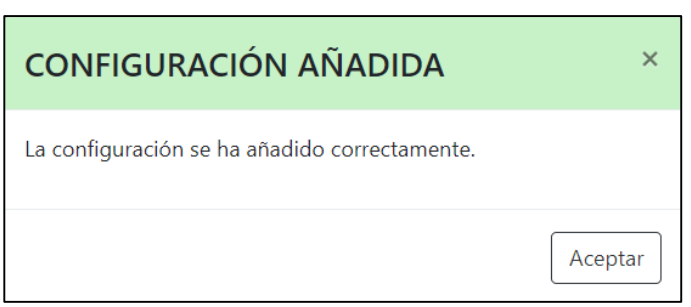


Ilustración 81 - Mensaje de confirmación de configuración añadida

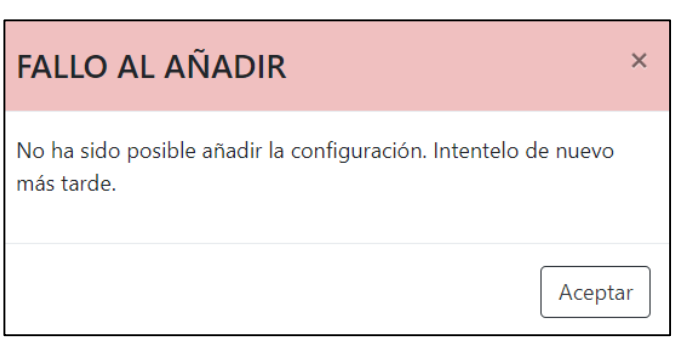


Ilustración 82 - Mensaje de error al añadir una configuración

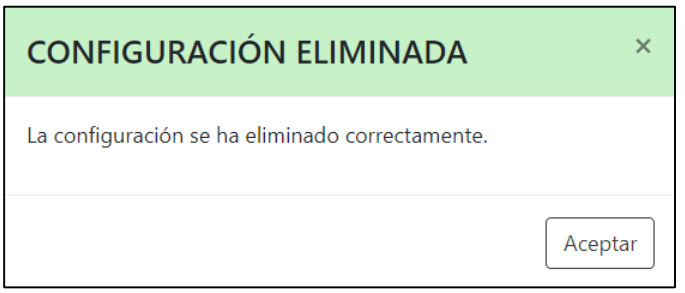


Ilustración 83 - Mensaje de confirmación de configuración eliminada

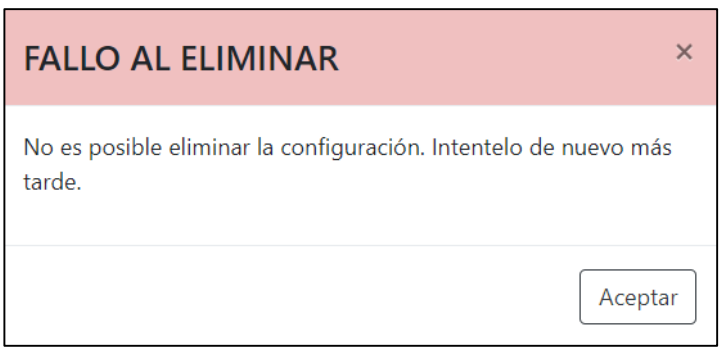


Ilustración 84 - Mensaje de error al eliminar una configuración

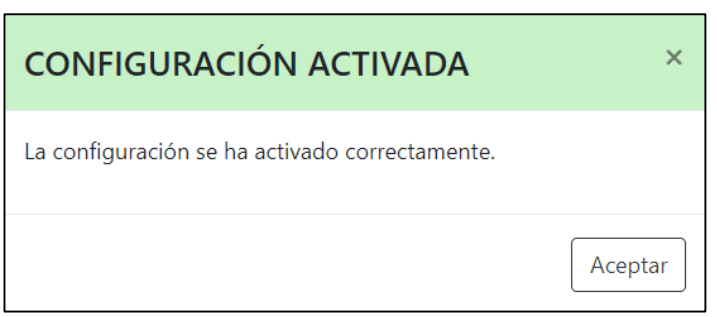


Ilustración 85 - Mensaje de confirmación de configuración activada

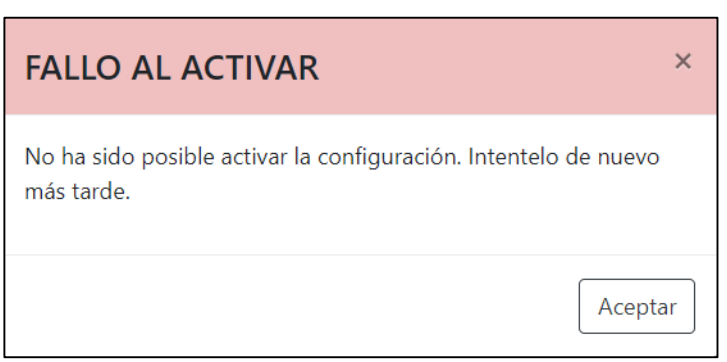
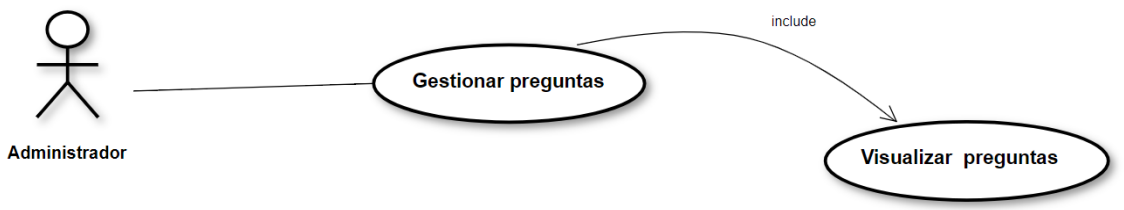


Ilustración 86 - Mensaje de error al activar una configuración

3.3. Gestionar preguntas



Nombre:	Gestionar preguntas

Descripción: El usuario "Administrador" puede visualizar las preguntas y posteriormente, si lo desea, puede realizar diversas gestiones sobre ellas: eliminarlas, editarlas e incluso crear otras nuevas.

Actores: Administrador

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa en el botón de "Gestión > Preguntas" situado en el menú de navegación (*Ilustración 45*). ***INCLUDE → Visualizar preguntas***

   [Si el usuario pulsa "Añadir pregunta" (*Ilustración 87*)]

   2A. Se muestra una pantalla donde el usuario debe crear la pregunta e indicar las respuestas que la forman (*Ilustración 88*).

   [Si el usuario pulsa "Guardar"]

   [Si la pregunta se ha añadido correctamente]

   3AAA. Se le muestra al usuario un mensaje de confirmación (*Ilustración 89*).

   [Si la pregunta no se ha añadido correctamente]

   3AAB. Se le muestra al usuario un mensaje de error (*Ilustración 90*).

   [Si el usuario pulsa "Cancelar"]

   3AB. La pantalla de la *Ilustración 88*, que permite añadir una pregunta, se deja de visualizar.

   [Si el usuario pulsa para visualizar una pregunta de la vista y pulsa "Eliminar pregunta" (*Ilustración 95*)]

   [Si es posible eliminar la pregunta]

   3BA. Se muestra un mensaje indicando que se ha eliminado correctamente.

(*Ilustración 93*)

[Si no es posible eliminar la pregunta]

3BB. Se muestra un mensaje indicando el error (*Ilustración 94*).

[Si el usuario pulsa para visualizar una pregunta de la vista y pulsa "Guardar" (*Ilustración 95*)]

[Si es posible actualizar la pregunta]

3CA. Se muestra un mensaje indicando que se ha actualizado correctamente. (*Ilustración 91*)

[Si no es posible actualizar la pregunta]

3CB. Se muestra un mensaje indicando el error (*Ilustración 92*).

Postcondiciones: Ninguna

Interfaz Gráfica:

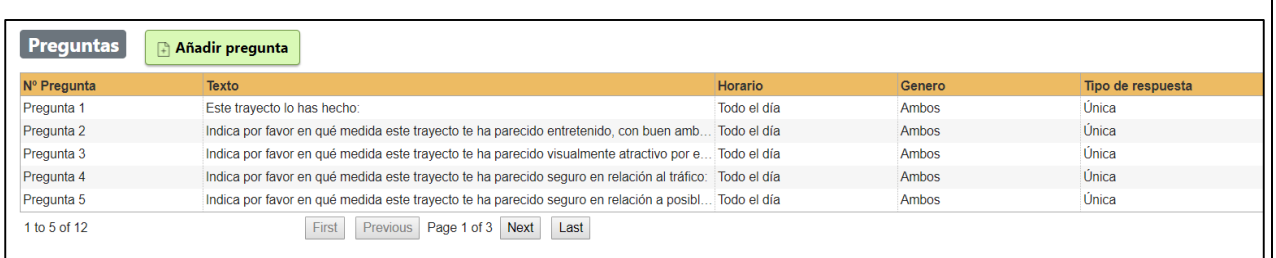


Ilustración 87 - Botón para añadir preguntas

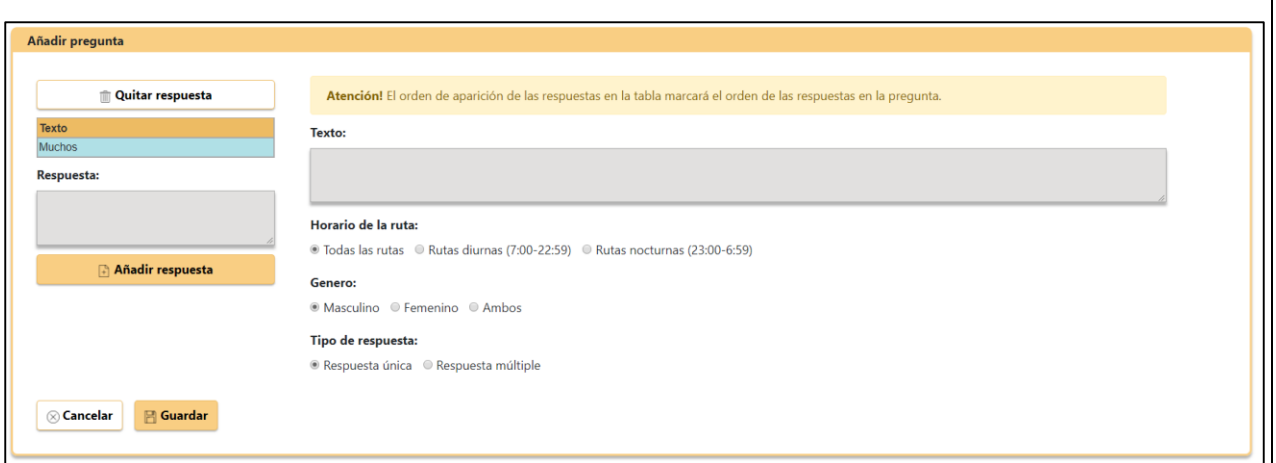


Ilustración 88 - Pantalla para añadir preguntas

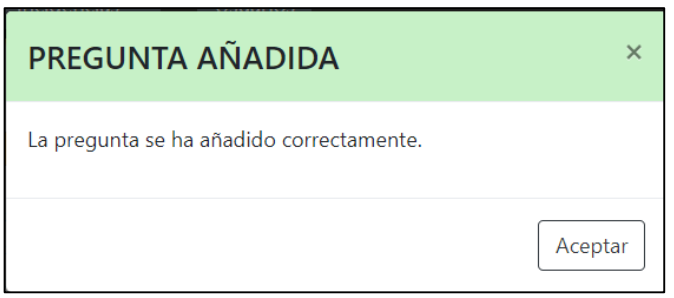


Ilustración 89 - Mensaje de confirmación de pregunta añadida

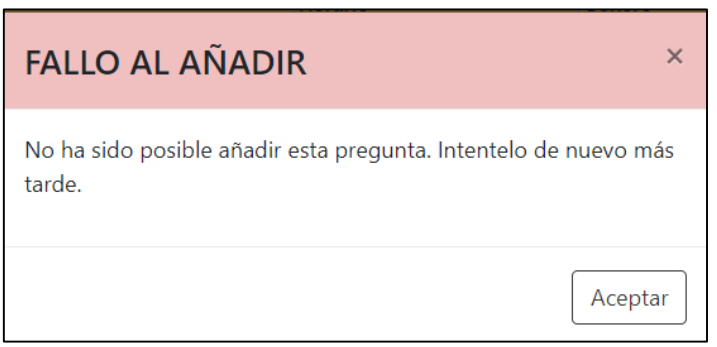


Ilustración 90 - Mensaje de error al añadir preguntas

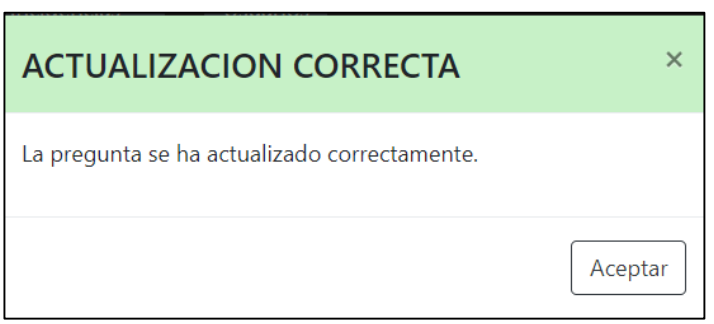


Ilustración 91 - Mensaje de confirmación de pregunta actualizada

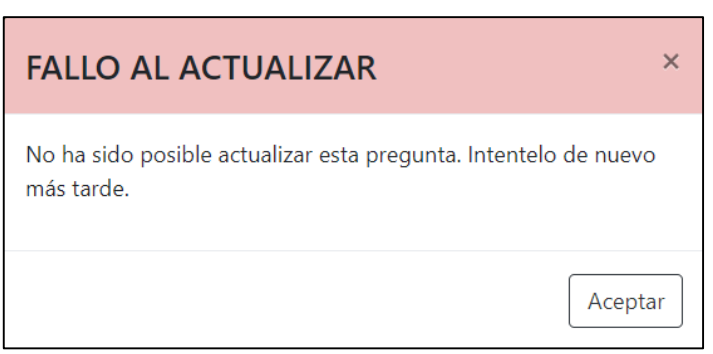


Ilustración 92 - Mensaje de error al actualizar preguntas

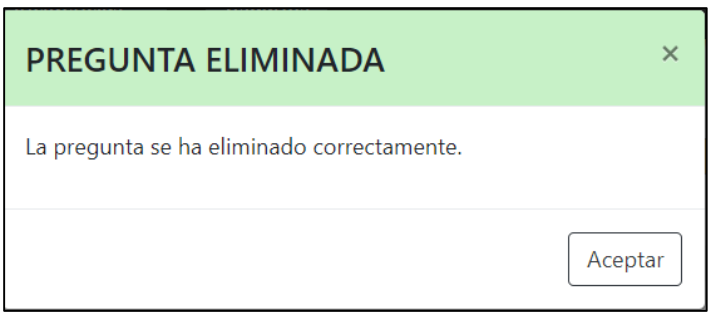


Ilustración 93 - Mensaje de confirmación de pregunta eliminada

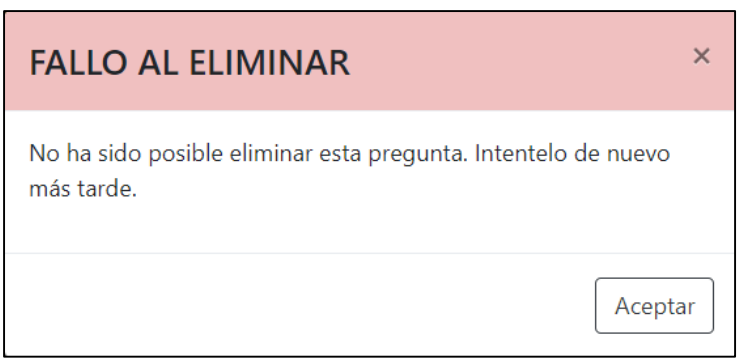


Ilustración 94 - Mensaje de error al eliminar una pregunta

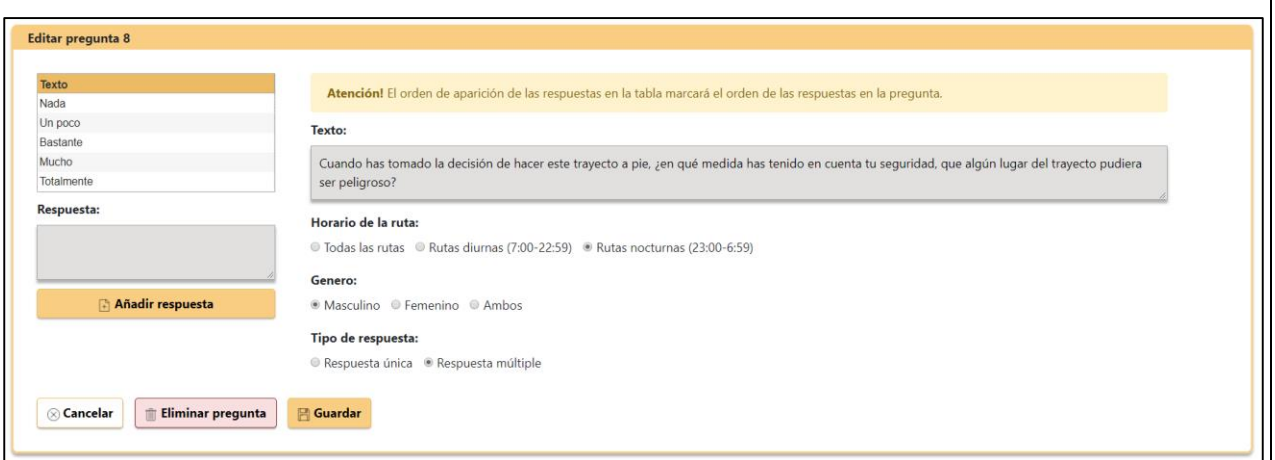


Ilustración 95 - Pantalla de visualización de la información de una pregunta

3.4. Gestionar categorías

Nombre:	Gestionar categorías

Descripción: El usuario "Administrador" puede visualizar las categorías y posteriormente, si lo desea, puede actualizarlas.

Actores: Administrador

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa en el botón de "Gestión > Categorías" situado en el menú de navegación (*Ilustración 45*). ***INCLUDE → Visualizar categorías***

   [Si el usuario pulsa para visualizar una categoría de la vista y pulsa "Guardar" (*Ilustración 96*)]

   [Si es posible actualizar la categoría]

   2A. Se muestra un mensaje indicando que se ha actualizado correctamente. (*Ilustración 97*)

   [Si no es posible actualizar la categoría]

   2B. Se muestra un mensaje indicando el error (*Ilustración 98*).

Postcondiciones: Ninguna

Interfaz Gráfica:

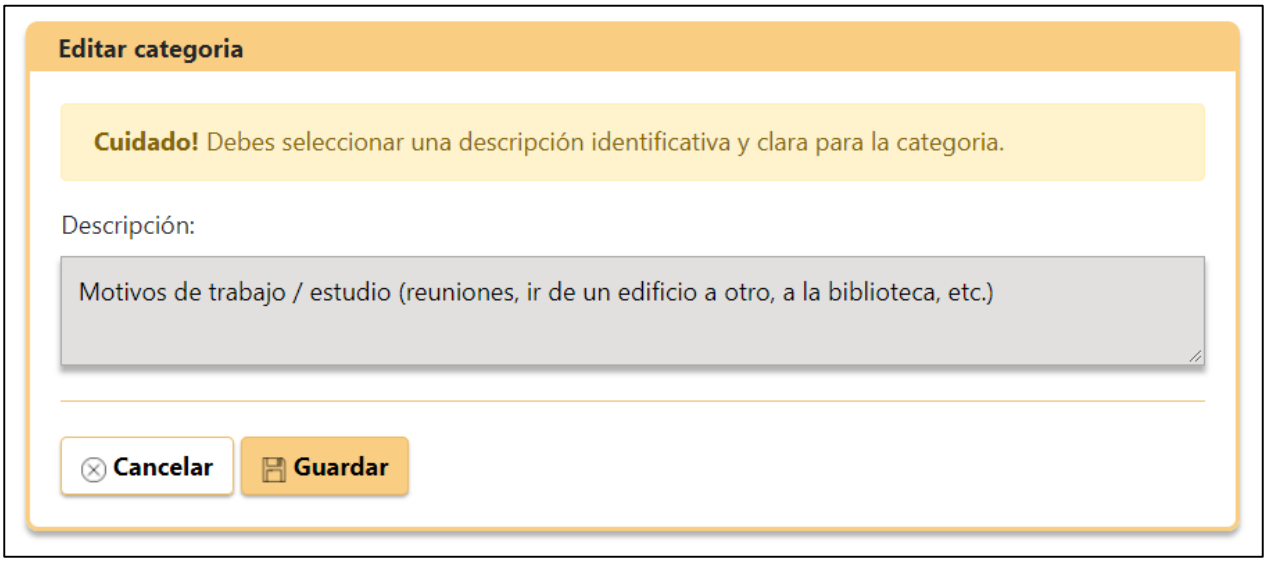


Ilustración 96 - Botón de guardar para actualizar categorías

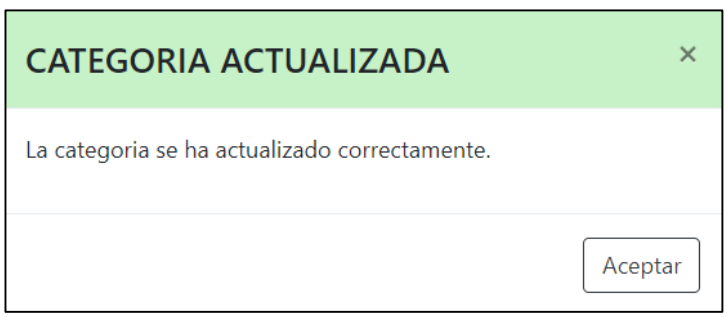


Ilustración 97 - Mensaje de confirmación de categoría actualizada

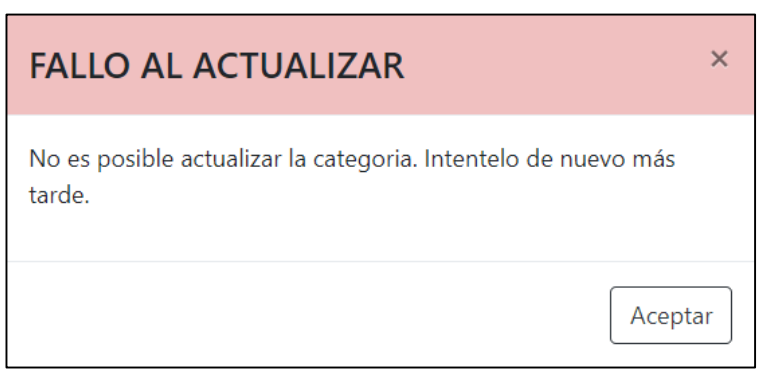
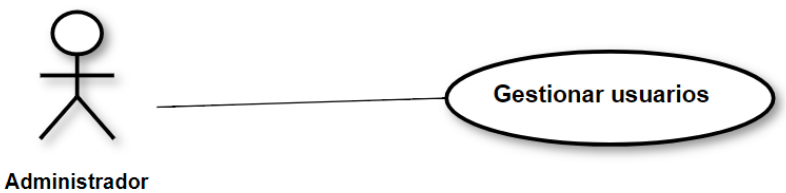


Ilustración 98 - Mensaje de error al actualizar categorías

3.5. Gestionar usuarios



Nombre:	Gestionar usuarios

Descripción: El usuario "Administrador" puede visualizar los usuarios del sistema, eliminarlos, actualizar los permisos de los usuarios existentes o crear nuevos.

Actores: Administrador

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de Eventos:

1. El usuario pulsa en el botón de "Usuarios" situado en el menú de navegación (*Ilustración 99*).

   [Si el usuario pulsa "Añadir" (*Ilustración 100*) se le visualiza la pantalla para poder introducir los valores (*Ilustración 101*)]

   [Si es posible añadir el usuario]

   2AA. Se muestra un mensaje indicando que se ha añadido correctamente. (*Ilustración 103*)

   [Si no es posible añadir el usuario]

   2AB. Se muestra un mensaje indicando el error (*Ilustración 104*).

   [Si el usuario pulsa un usuario y selecciona "Eliminar usuario" (*Ilustración 102*)]

   [Si es posible eliminar el usuario]

   2BA. Se muestra un mensaje indicando que se ha eliminado correctamente. (*Ilustración 107*)

   [Si no es posible eliminar el usuario]

   2AB. Se muestra un mensaje indicando el error (*Ilustración 108*).

   [Si el usuario pulsa un usuario y selecciona "Hacer investigador" (*Ilustración 102*) o "Hacer administrador", para actualizar los permisos]

   [Si es posible actualizar los permisos del usuario]

2CA. Se muestra un mensaje indicando que se ha actualizado correctamente. (*Ilustración 105*)

[Si no es posible actualizar los permisos del usuario]

2CB. Se muestra un mensaje indicando el error (*Ilustración 106*).

Postcondiciones: Ninguna

Interfaz Gráfica:

Ilustración 99 - Botón de usuario del panel de navegación

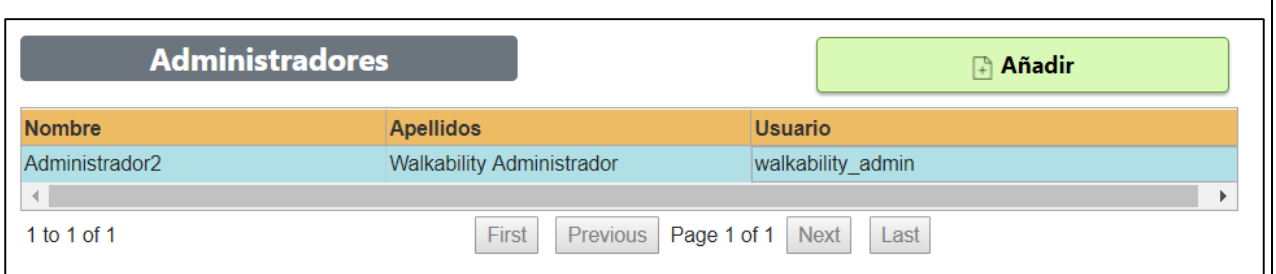


Ilustración 100 - Botón de añadir usuario

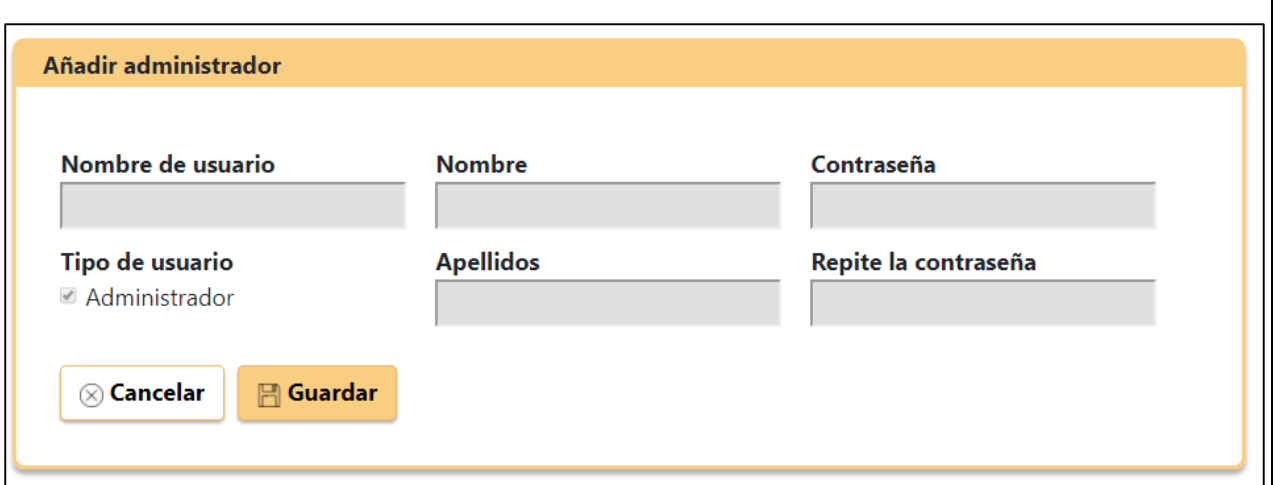


Ilustración 101 - Pantalla para añadir un usuario

Ilustración 102 - Botones de acciones para gestionar usuarios

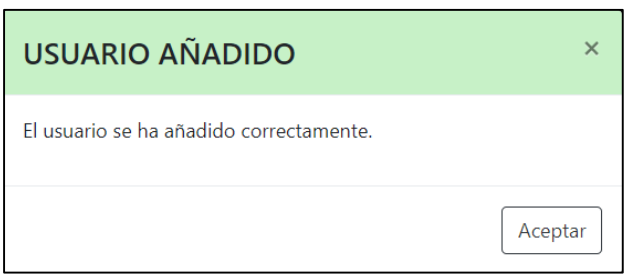


Ilustración 103 - Mensaje de usuario añadido correctamente

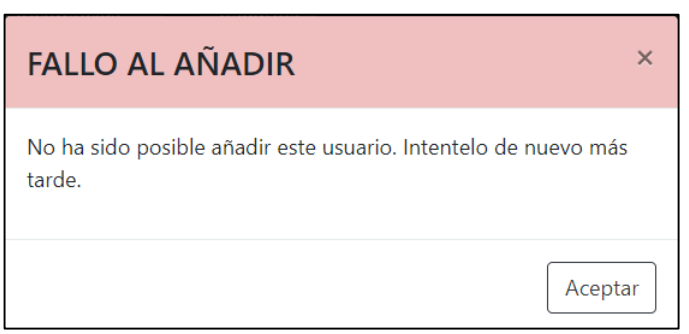


Ilustración 104 - Mensaje de error al añadir usuarios

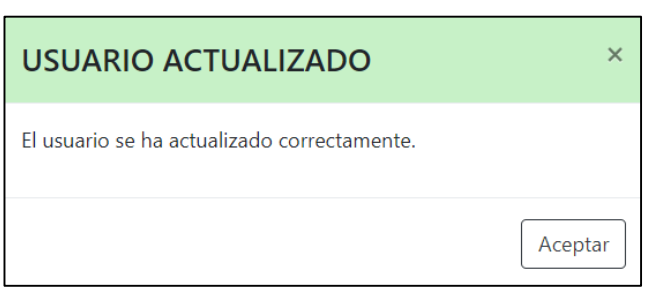


Ilustración 105 - Mensaje de usuario actualizado correctamente

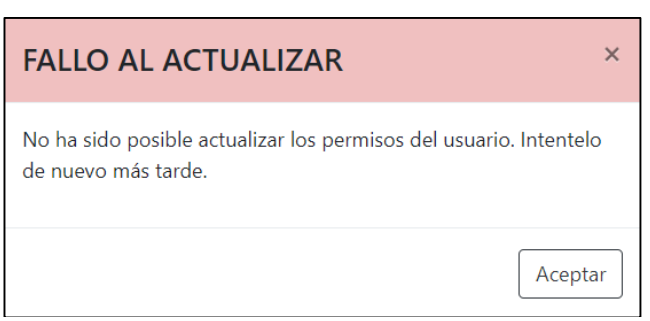


Ilustración 106 - Mensaje de error al actualizar un usuario

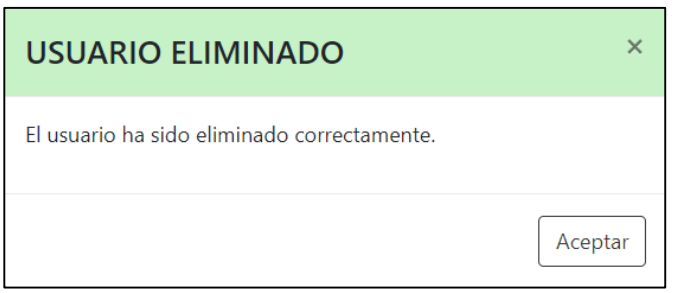


Ilustración 107 - Mensaje de usuario eliminado correctamente

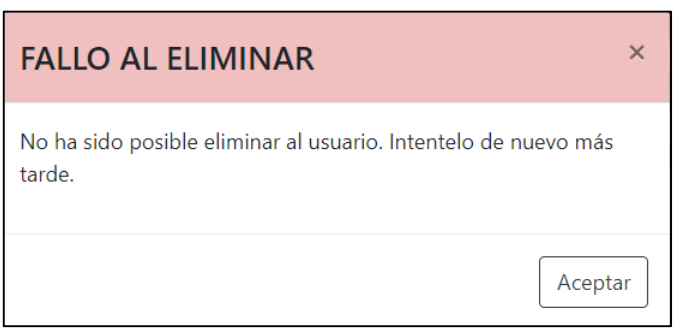
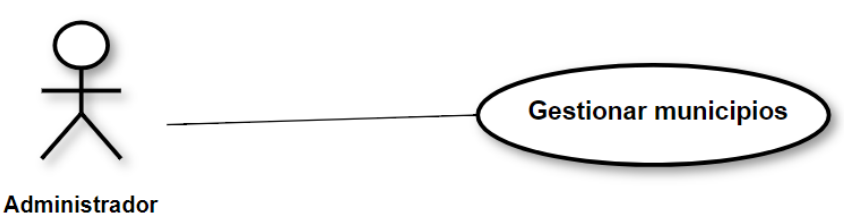


Ilustración 108 - Mensaje de error al eliminar un usuario

3.6. Gestionar municipios



Nombre:	Gestionar municipios
Descripción: El usuario "Administrador" puede visualizar los municipios de inicio y del fin de las rutas en el sistema y si lo desea, al encontrar municipios incorrectos, actualizarlos.	
Actores: Administrador	
Precondiciones: Ninguna	
Requisitos no funcionales: Ninguno	
Flujo de Eventos: 1. El usuario pulsa en el botón de "Incidencias > Gestión de municipios por nombre" o	

"Incidencias > Gestión de municipios por zona", y en este último caso se selecciona una zona. (*Ilustración 109*)

[Si los municipios de las rutas se cargan correctamente (*Ilustración 110*)]

[Si el usuario visualiza la ruta en el mapa (*Ilustración 111*) y desea actualizar un municipio pulsando "Guardar" (*Ilustración 112*)]

[Si los municipios se actualizan correctamente]

2AAA. Se le muestra un mensaje de confirmación al usuario. (*Ilustración 113*)

[Si los municipios no se actualizan correctamente]

2AAA. Se le muestra un mensaje de error al usuario. (*Ilustración 114*)

[Si los municipios de las rutas no se cargan correctamente]

2B. Se muestra un mensaje de error indicando que los datos no se han podido cargar (*Ilustración 41*).

Postcondiciones: Ninguna

Interfaz Gráfica:

Ilustración 109 - Botones de corrección de municipios

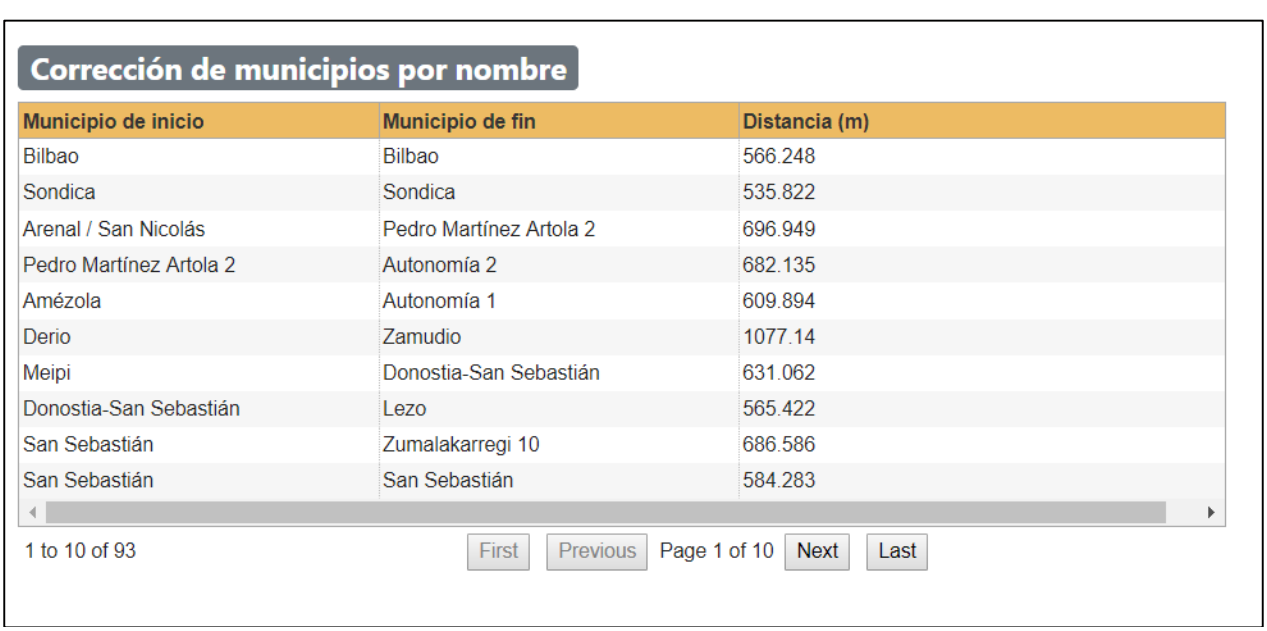


Ilustración 110 - Pantalla de visualización de municipios

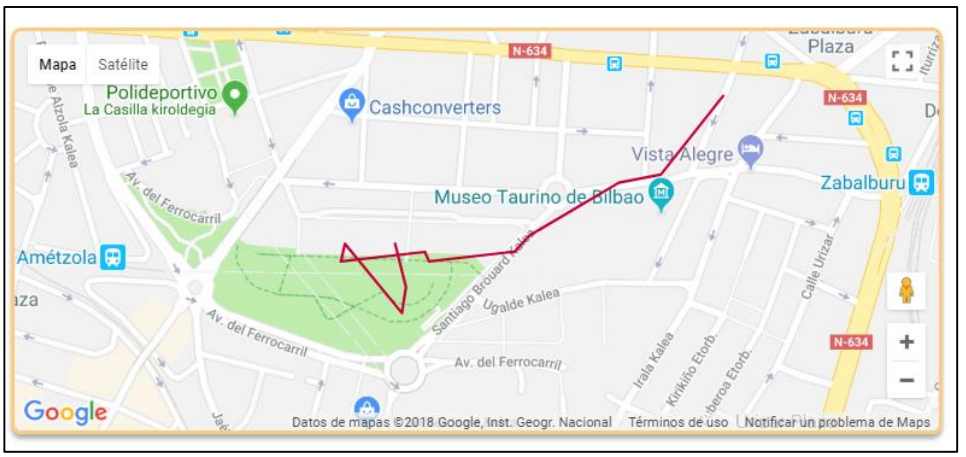


Ilustración 111 - Visualización de municipios en el mapa

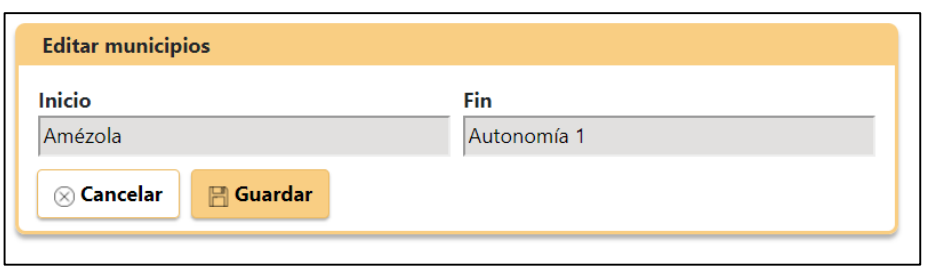


Ilustración 112 - Actualizar municipios

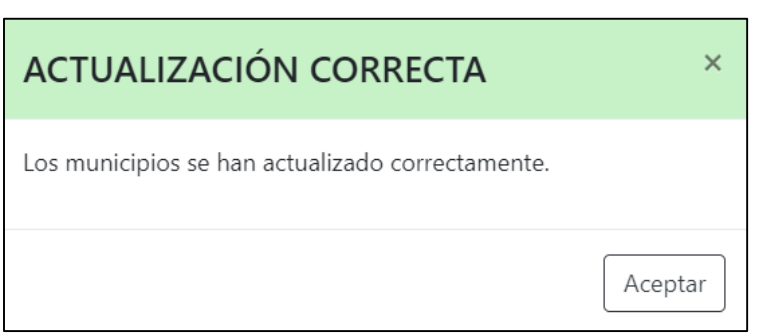


Ilustración 113 - Mensaje de actualización correcta de municipios

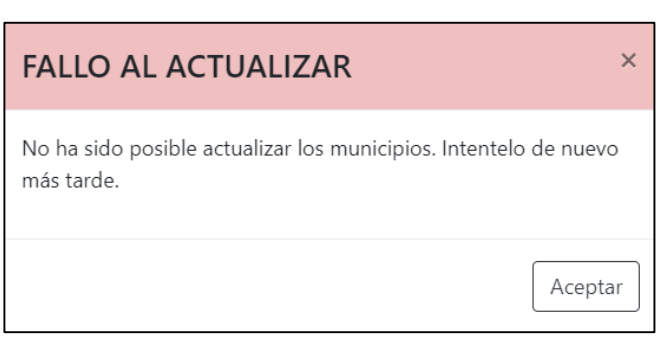


Ilustración 114 - Mensaje de error al actualizar municipios

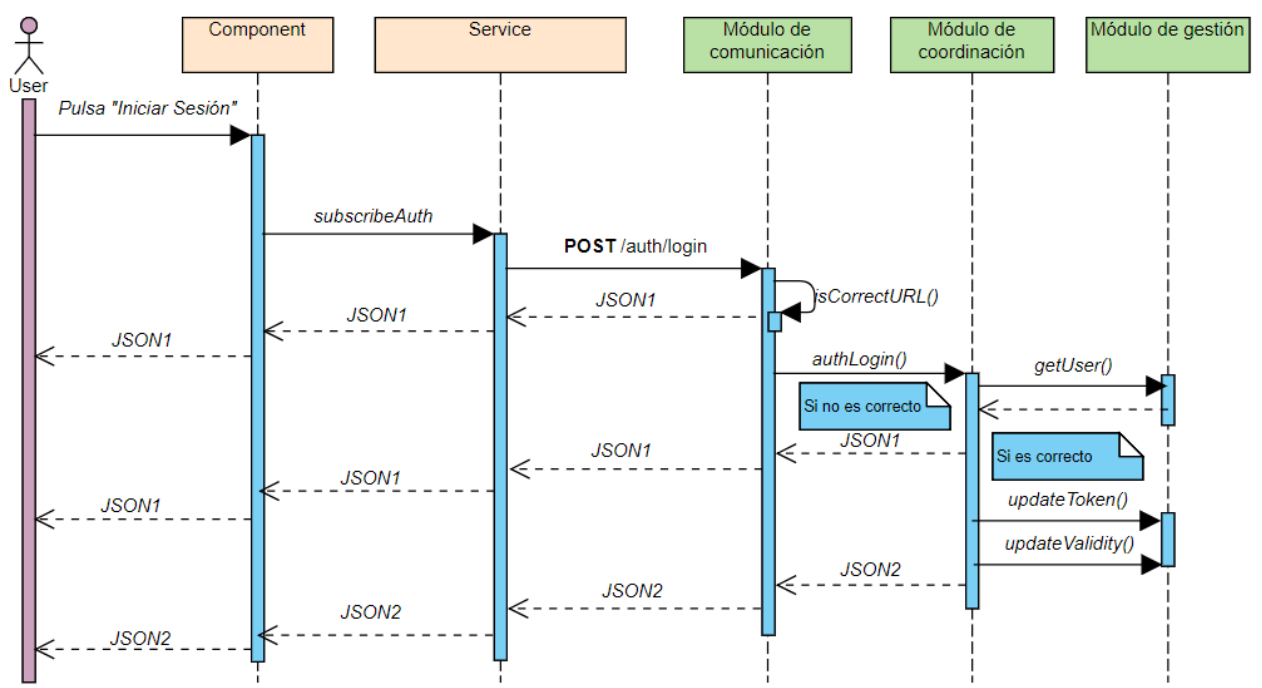
Anexo II: Diagramas de secuencia

En éste segundo anexo se hará un análisis detallado de los diagramas de secuencia de Walkability Analyzer. Para ello, se analizará cada diagrama de secuencia en función de su funcionalidad, y se detallará la estructura del objeto que se devuelve al usuario. Hay que destacar, que debido a que la aplicación web posee tanto back-end como front-end, los diagramas de secuencia están fragmentados en dos partes. Dentro de cada diagrama se podrá visualizar, en amarillo, la parte de front-end, y, en verde, el proceso de comunicación que sigue la parte back-end. Además, para estructurar este anexo se realizará un diagrama de secuencia por cada caso de uso extendido analizado en el anexo anterior, siguiendo la misma estructura.

1. Usuario no identificado

El siguiente diagrama de secuencia corresponde a la acción de Iniciar Sesión, propia del actor con rol "Usuario no identificado".

1.1. Iniciar sesión



El "Módulo de gestión", en primer lugar, realiza una llamada a la base de datos para obtener la información del usuario y así poder comprobar si sus credenciales son correctas. Posteriormente, actualiza en la base de datos tanto el *Token* del usuario como su validez. Las llamadas realizadas son las siguientes:

```
SELECT * FROM researcher WHERE nickResearcher = parameter
UPDATE researcher SET parameters WHERE tokenResearcher = parameter
UPDATE researcher SET parameters WHERE validityResearcher = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos, o si las credenciales del usuario no coinciden, se devuelve un mensaje de error, en el diagrama es denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*.


```
{
    "status": 500,
    "code": 5001,
    "msg": "Error: Internal Server Error",
    "body": null
}
```


Ilustración 115 - Estructura de un mensaje de error

En cambio, si el proceso de iniciar sesión transcurre correctamente se devuelve un objeto, JSON2, que presenta la estructura de la *Ilustración 116*.

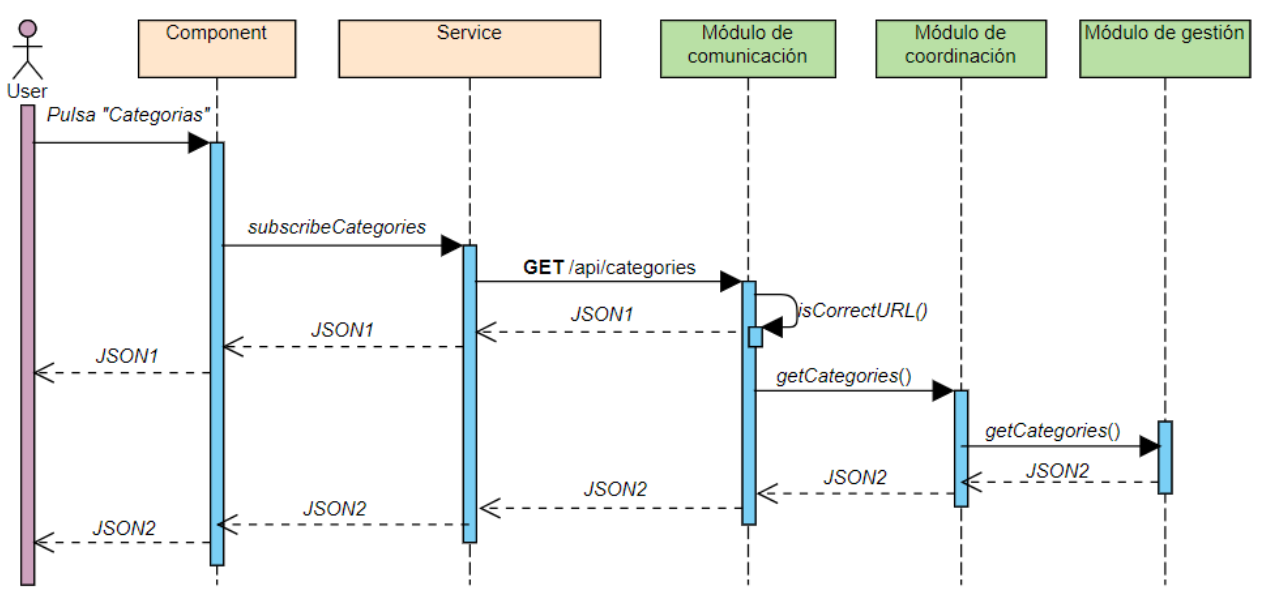

```
{
    "status": 200,
    "code": 2001,
    "msg": "Correcto",
    "body": {
        "researcherData": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
            .eyJjb2RlIjoid2Fsa2FiaWxpdHlfYW5hbHl6ZXIiLCJpZCI6NCwicmVzZWFyY2hlciI6IndhbGthYmlsaXR5X
            OjE1MzIxMDQ3MTcsImlhdCI6MTUzMjEwNDcxN30.FmwUENYrzcyLom4lZiyk-z-f8IpJ8pxtm_mELdU4LoU",
        "rolResearcher": 1
    }
}
```


Ilustración 116 - Estructura del objeto al iniciar sesión

2. Investigador

Los siguientes diagramas de secuencia corresponden a los casos de uso extendidos del usuario "Investigador", estructurados de la misma manera:

2.1. Visualizar categorías



El "Módulo de gestión" realiza una única llamada a la base de datos para obtener la información que necesita. En este caso la llamada es la siguiente:

```
SELECT * FROM categoria
```

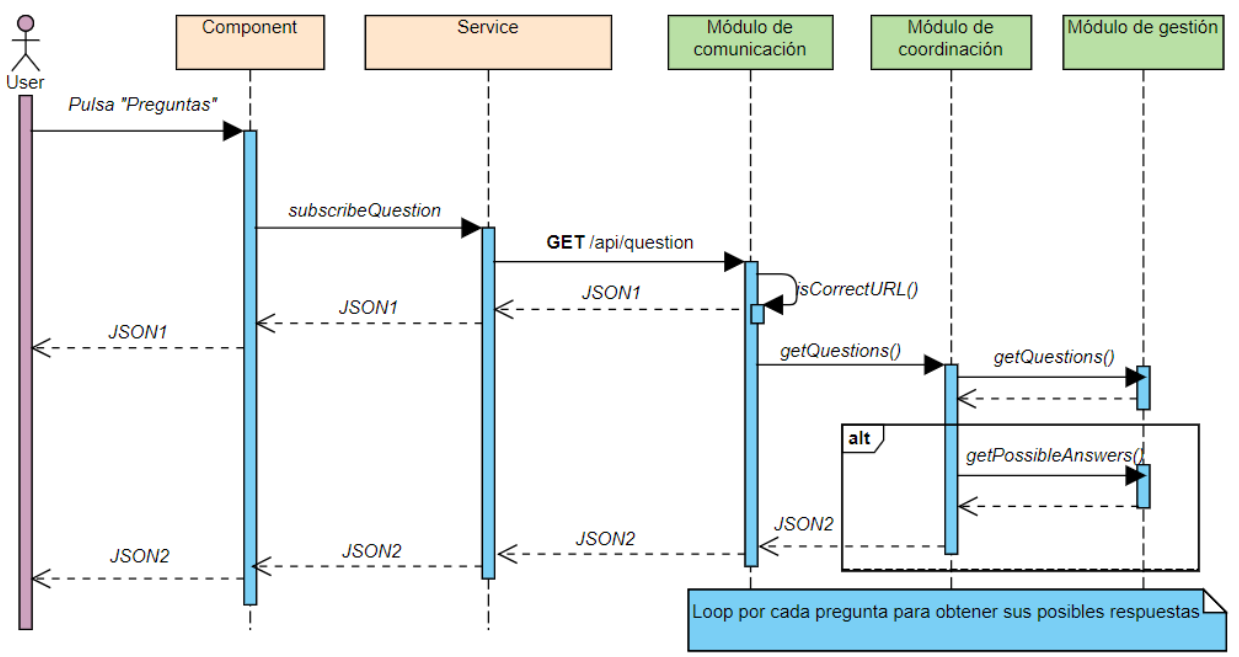
En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*.

En cambio, si las categorías se obtienen correctamente se devuelve un objeto JSON que posee, por cada categoría, la estructura que se puede visualizar en la *Ilustración 117*.

```
{
    "idcategoria": 5,
    "texto": "Compras y recados"
}
```

Ilustración 117 - Estructura para enviar una categoría

2.2. Visualizar preguntas



El "Módulo de gestión" realiza dos llamadas a la base de datos para obtener la información que necesita. En primer lugar recoge todas las preguntas, y, posteriormente, por cada pregunta recoge sus respuestas. El código de dichas llamadas es el siguiente:

```
SELECT * FROM pregunta

SELECT * FROM respuesta WHERE pregunta_idpregunta = parameter ORDER BY orden
ASC
```

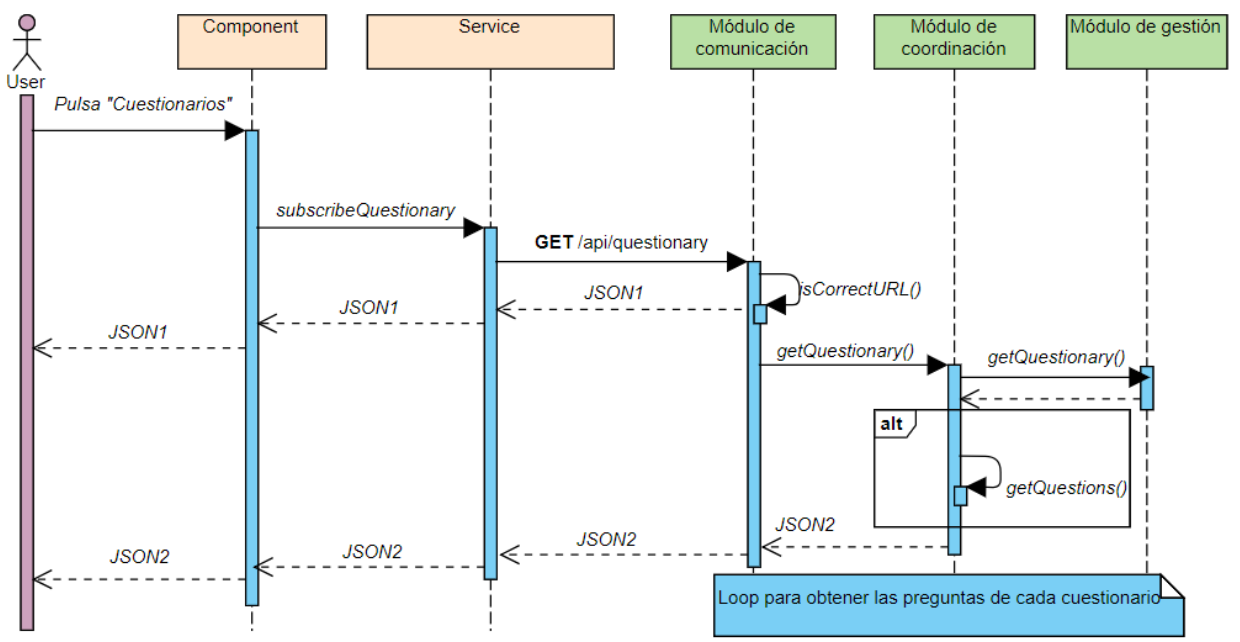
En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*.

En cambio, si las preguntas se obtienen correctamente, se devuelve un objeto JSON que posee, por cada pregunta, la estructura que se puede visualizar en la *Ilustración 118*. Dentro de esta estructura, por cada pregunta se devuelven todas sus posibles respuestas.

```
"idpregunta": 1,
"texto": "Este trayecto lo has hecho:",
"tipo_respuesta": 0,
"genero": 2,
"horario": 0,
"hasAnswers": 1,
"belongQuestionary": 1,
"answers": [
    {
        "idrespuesta": 1,
        "pregunta_idpregunta": 1,
        "texto": "Solo/a",
        "orden": 0
    },
```


Ilustración 118 - Estructura de cada pregunta

2.3. Visualizar cuestionarios



El "Módulo de gestión" realiza una única llamada a la base de datos para obtener los cuestionarios, y posteriormente, por cada cuestionario se recogen las preguntas que lo componen. Una vez que se poseen las preguntas de un cuestionario, las posibles respuestas a las preguntas se obtienen como se ha explicado en el apartado anterior.

```
SELECT * FROM cuestionario ORDER BY activo DESC

SELECT * FROM pregunta_cuestionario INNER JOIN pregunta on pregunta.idpregunta
= pregunta_cuestionario.pregunta_idpregunta WHERE cuestionario_version =
parameter ORDER BY orden ASC
```

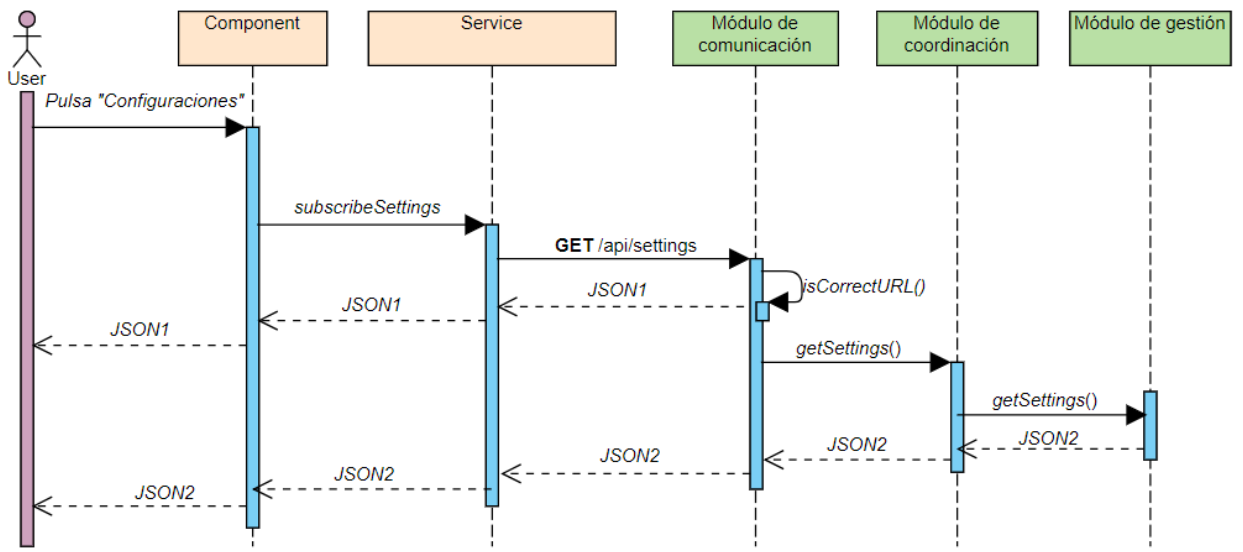
En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*.

En cambio, si los cuestionarios se obtienen correctamente se devuelve un objeto JSON que posee, por cada cuestionario, la estructura que se puede visualizar en la *Ilustración 119*. Por cada cuestionario se devuelve las preguntas que lo componen, con lo que dentro del objeto de cada cuestionario se poseerá la estructura de cada pregunta que se ha explicado anteriormente.

```
"version": 15,
"activo": 1,
"hasAnswers": 0,
"questions": [
    {
```


Ilustración 119 - Estructura de cada cuestionario

2.4. Visualizar configuraciones



El "Módulo de gestión" realiza una única llamada a la base de datos para obtener las configuraciones almacenadas en el sistema. En este caso la llamada es la siguiente:

```
SELECT * FROM configuracion INNER JOIN fecha_activacion on
configuracion.version = fecha_activacion.configuracion_version ORDER BY
fecha_activacion.activa DESC
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*.

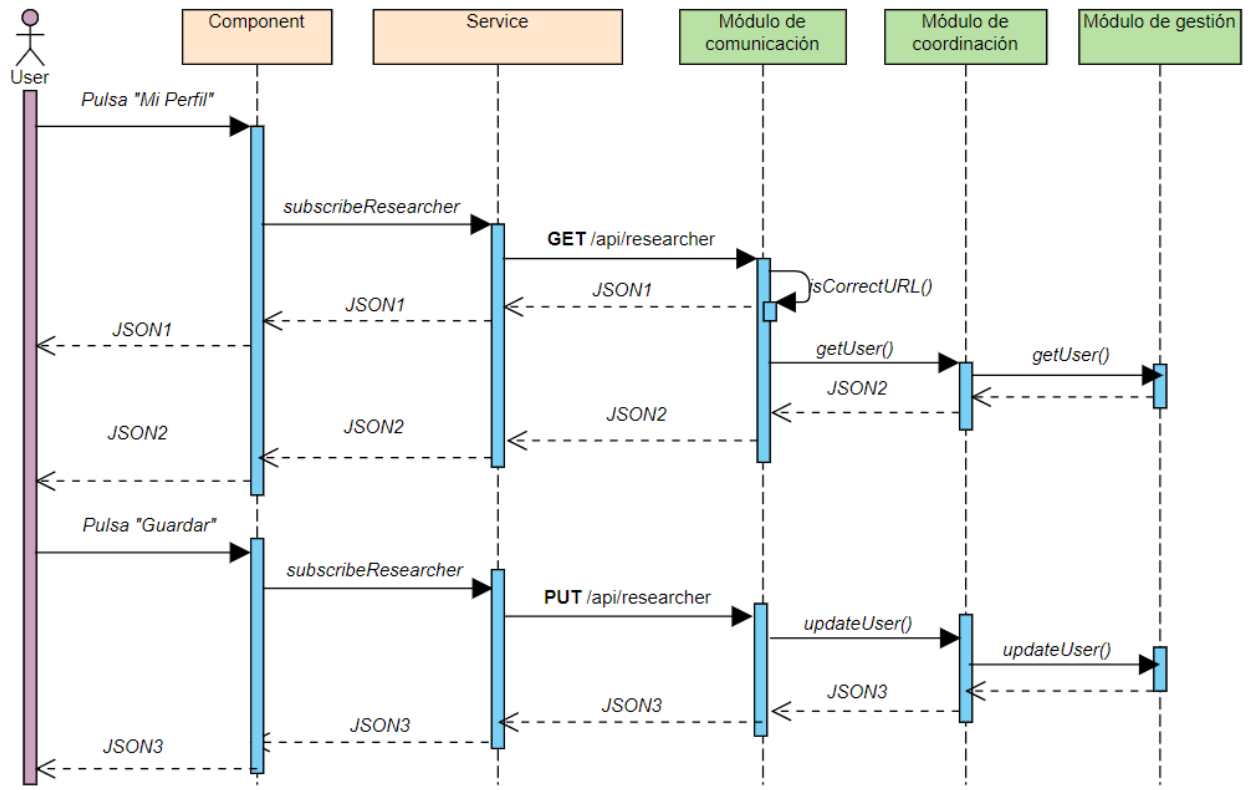
En cambio, si las configuraciones se obtienen correctamente, se devuelve un objeto que posee, por cada configuración, la estructura que se puede visualizar en la *Ilustración 120*.


```
"version": 22,
"vel_max": 2,
"dist_min_ruta": 2,
"tiempo_parada": 2,
"radio_parada": 2,
"puntos_distintos": 2,
"dist_puntos": 2,
"dif_duracion": 2,
"idactivacion": 22,
"dia": "2018-07-18T22:00:00.000Z",
"activa": 1,
"configuracion_version": 22
```


Ilustración 120 - Estructura de una configuración

2.5. Gestionar información de usuario

En el siguiente diagrama se puede visualizar el proceso que sigue el sistema para obtener la información de un usuario, así como, si el usuario lo desea, el proceso seguido para actualizar sus datos personales.



El "Módulo de gestión" realiza una única llamada a la base de datos para obtener la información de un usuario. En este caso la llamada es la siguiente:

```
SELECT nickResearcher, nameResearcher, surnameResearcher FROM researcher WHERE
nickResearcher = parameter
```

Posteriormente, si el usuario decide actualizar su información, se realiza una nueva llamada:

```
UPDATE researcher SET parameters WHERE nickResearcher = parameter
```

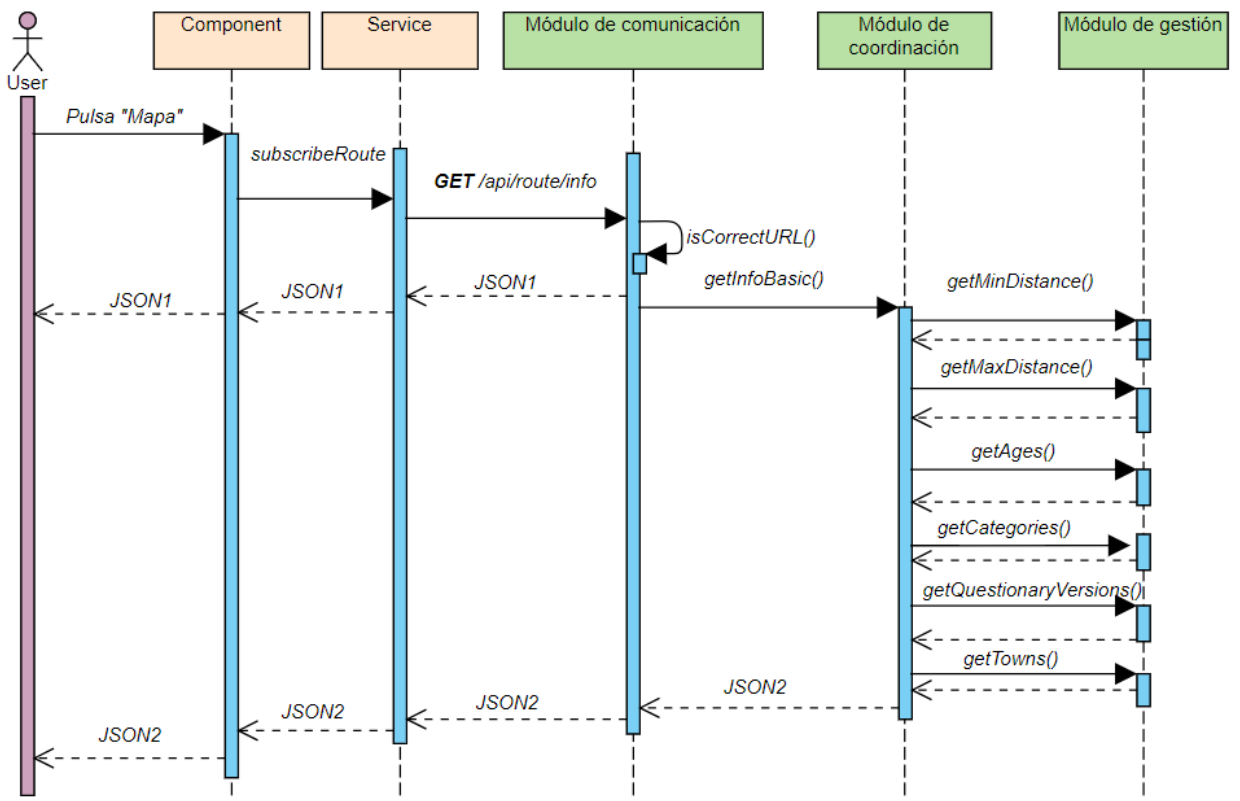
En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. El JSON3, posee el mismo formato que JSON1, pero puede ser tanto un mensaje con un código correcto como incorrecto, depende de si el proceso de actualizar la información ha concluido satisfactoriamente.

Por otro lado, la estructura del JSON2, que devuelve la información de un usuario, se puede visualizar en la *Ilustración 121*:

```
{
    "nickResearcher": "walkability_admin",
    "nameResearcher": "Administrador2",
    "surnameResearcher": "Walkability Administrador"
}
```


Ilustración 121 - Estructura de la información de un usuario

2.6. Obtener información básica

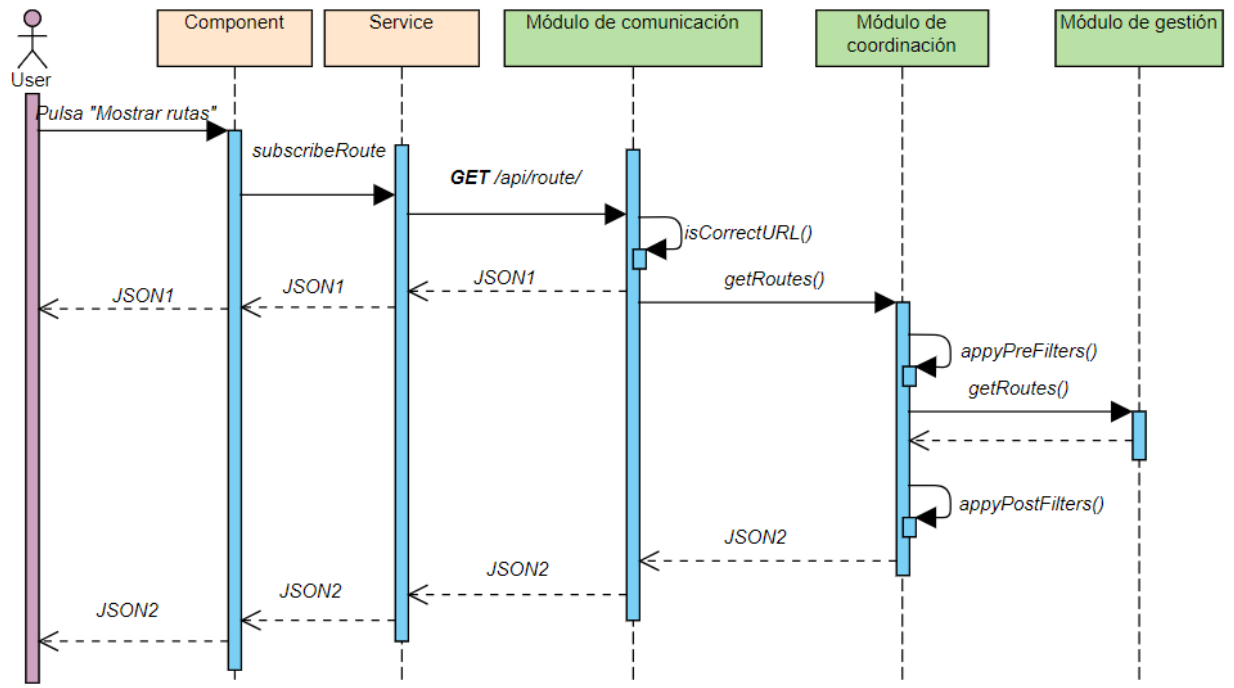


El "Módulo de gestión" realiza varias llamadas a la base de datos para solicitar toda la información necesaria para cargar la pantalla principal de la aplicación web. En este caso las llamadas son las siguientes:

```
SELECT distancia_recorrida FROM ruta ORDER BY distancia_recorrida ASC LIMIT 1
SELECT distancia_recorrida FROM ruta ORDER BY distancia_recorrida DESC LIMIT 1
SELECT * FROM usuario ORDER BY anyo_nacimiento ASC LIMIT 1
SELECT * FROM usuario ORDER BY anyo_nacimiento DESC LIMIT 1
SELECT * FROM categoría
SELECT DISTINCT municipio_inicio, municipio_fin FROM ruta
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. En cambio, si todas las llamadas se realizan correctamente se devuelve una estructura con todos los datos solicitados.

Obtener rutas



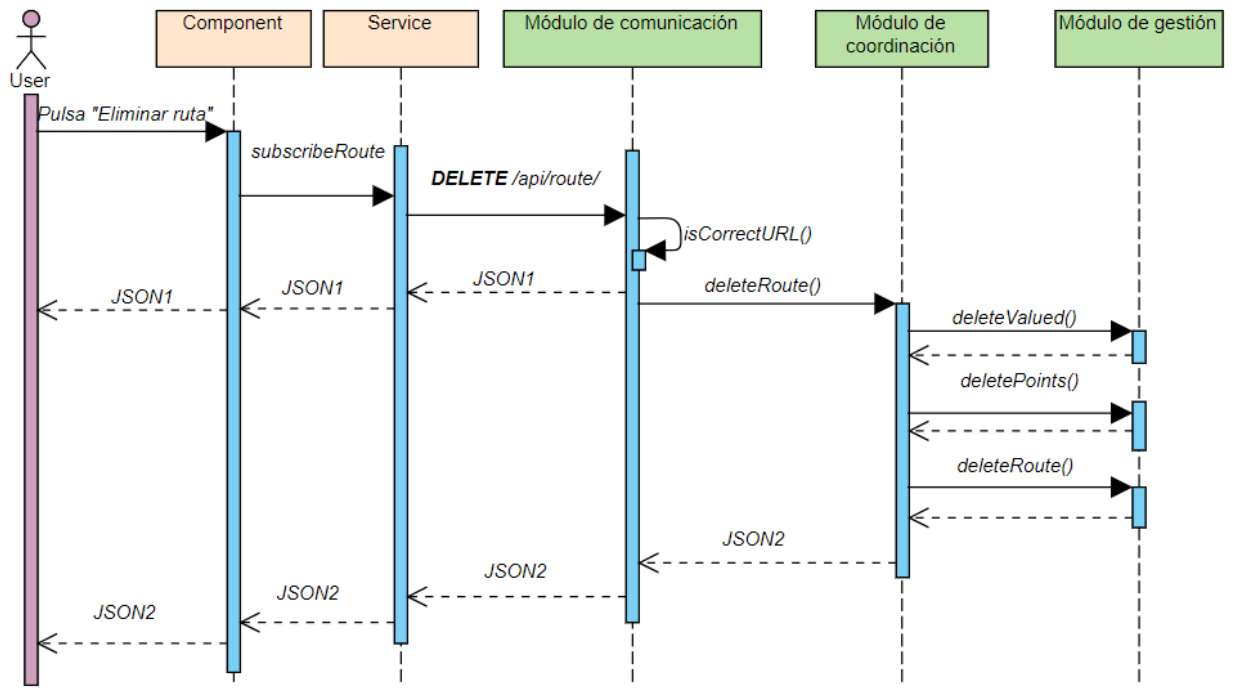
Para obtener las rutas, en primer lugar, en el "Módulo de coordinación" se gestionan todos los filtros aplicados, y una vez que se construye la sentencia, con las tablas de las que hay que extraer la información y la condición que hay que aplicar a la sentencia, está se ejecuta contra la base de datos.

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. En cambio, si todas las llamadas se realizan correctamente se devuelve una estructura, JSON2, con todos los datos solicitados. Esta estructura se puede visualizar en la *Ilustración 122*.


```
anyo_nacimiento: 1984
categoria_idcategoria: null
configuracion_version: 1
copia_de: null
dia: 20180411
distancia_recorrida: 566.248
duracion: 31
fecha_ruta_idfecha_ruta: "1747c755756f99331523460516682"
genero: 1
hora: 1728
idfecha_ruta: "1747c755756f99331523460516682"
idruta: "1747c755756f99331523460516620"
idusuario: "1747c755756f99331523454239196"
movilidad_reducida: 0
municipio_fin: "Bilbao"
municipio_inicio: "Bilbao"
municipio_procedencia: "Bilbao"
punto_fin: "1747c755756f99331523459907284"
punto_inicio: "1747c755756f99331523458078234"
▼route: Array(15)
  ▶0: {latitud: 43.2637, longitud: -2.95093, precision: 13.695}
  ▶1: {latitud: 43.2637, longitud: -2.95096, precision: 13.595}
  ▶2: {latitud: 43.263, longitud: -2.94772, precision: 30.618}
```


Ilustración 122 - Estructura para devolver las rutas

Eliminar una ruta

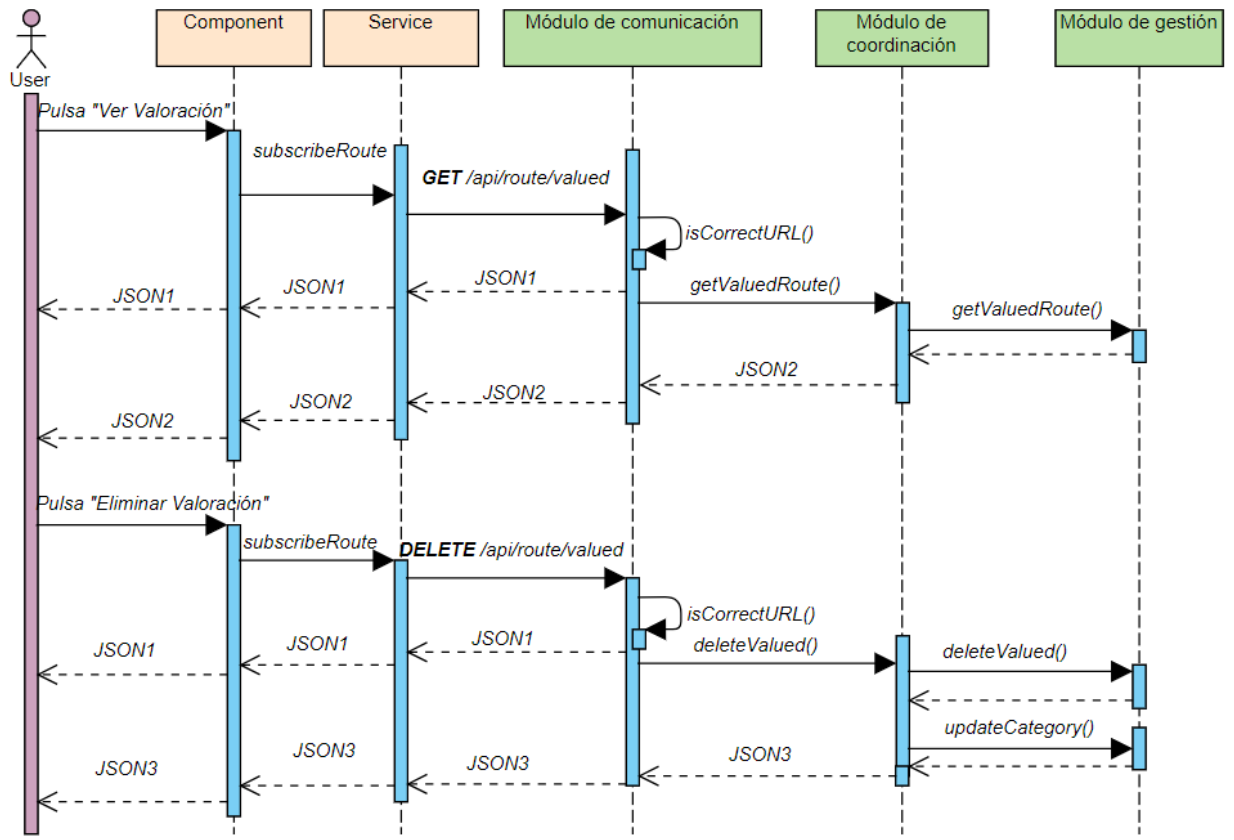


Para eliminar una ruta únicamente es necesario realizar tres llamadas desde el "Módulo de gestión" a la base de datos. De esta manera, y por este orden, se elimina la posible valoración de la ruta, se eliminan todos sus puntos geográficos y por último, se elimina la propia ruta en sí.

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. El JSON2, posee el mismo formato que JSON1, pero puede ser tanto un mensaje con un código correcto como incorrecto, depende de si el proceso de eliminar la ruta ha concluido satisfactoriamente.

Gestionar valoración de una ruta

En el siguiente diagrama de secuencia de puede visualizar el proceso mediante el cual se obtiene toda la información referente a la valoración de una ruta, y posteriormente, si el usuario lo desea, se puede eliminar dicha valoración.



Para obtener la valoración de una ruta es necesario realizar una llamada, desde el "Módulo de gestión", a la base de datos para solicitar dicha información. Posteriormente, en caso de que el usuario decida eliminar la valoración, se realiza una llamada a la base de datos, igual que la que se realiza al eliminar la ruta, para eliminar la valoración. Cuando la valoración de una ruta se ha eliminado, es necesario indicar que esa ruta ya no está valorada, por lo que ya no posee ninguna categoría de valoración. Para ello se realiza la siguiente llamada a la base de datos:

```
UPDATE ruta_realizada SET categoria_idcategoria = NULL WHERE ruta_idruta =
parameter
```

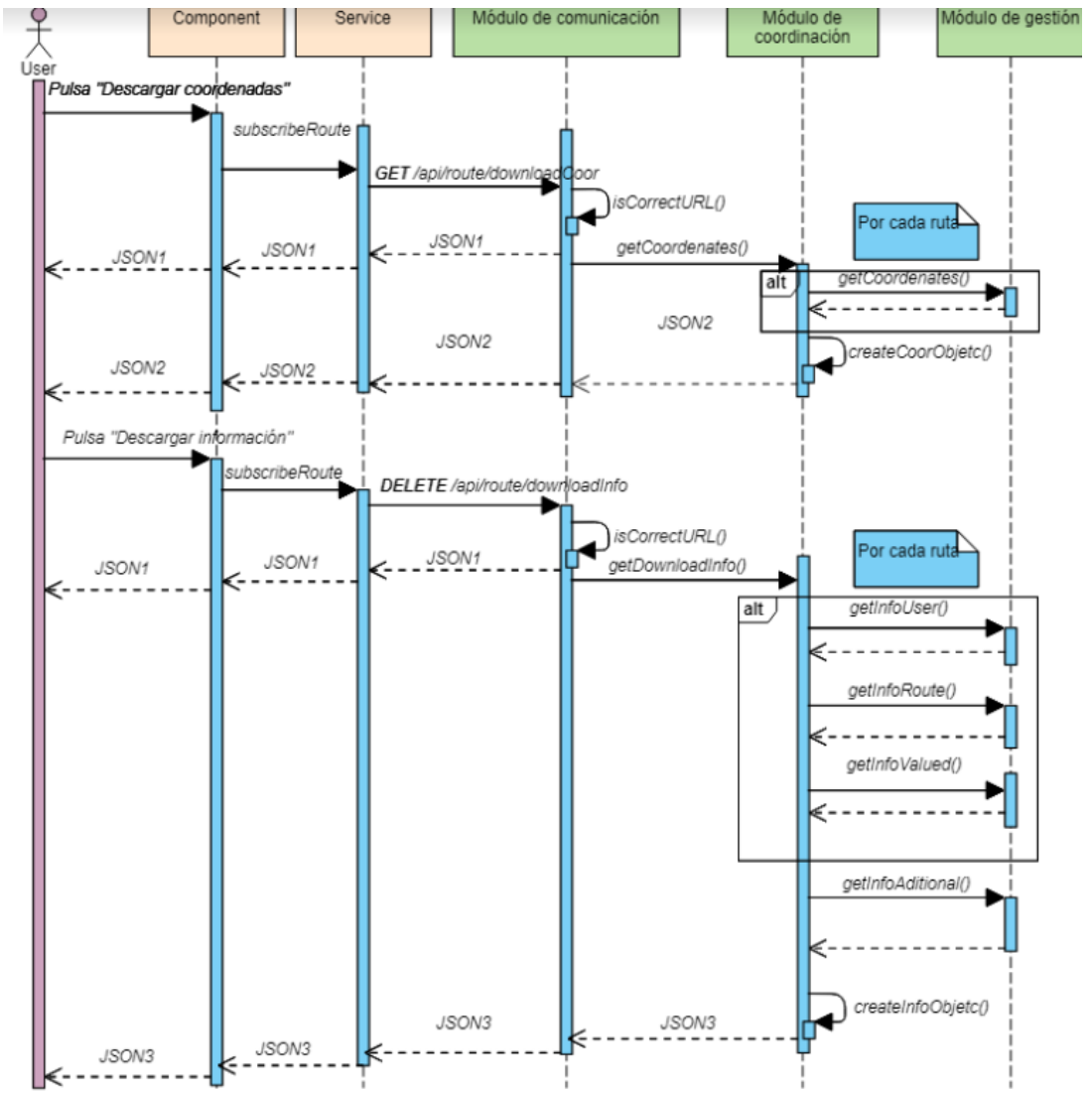
En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. El JSON3, posee el mismo formato que JSON1, pero puede ser tanto un mensaje con un código correcto como incorrecto, depende de si el proceso de eliminar la valoración de la ruta ha concluido satisfactoriamente. Por último, el JSON2, que devuelve la información de la valoración de la ruta tiene la estructura que se puede visualizar en la *Ilustración 123*.

```
{pregunta_idpregunta: 1, texto: "Este trayecto lo has hecho:", respuestas: Array(1)}
{pregunta_idpregunta: 2, texto: "Indica por favor en qué medida este trayecto te ha parecido entretenido, con buen ambiente:", respuestas: Array(1)}
{pregunta_idpregunta: 3, texto: "Indica por favor en qué medida este trayecto te ha…mente atractivo por el paisaje o la arquitectura:", respuestas: Array(1)}
{pregunta_idpregunta: 4, texto: "Indica por favor en qué medida este trayecto te ha parecido seguro en relación al tráfico:", respuestas: Array(1)}
{pregunta_idpregunta: 5, texto: "Indica por favor en qué medida este trayecto te ha parecido seguro en relación a posibles delitos:", respuestas: Array(1)}
{pregunta_idpregunta: 6, texto: "Cuando has tomado la decisión de hacer este trayec…ras otras alternativas de transporte a ese lugar?", respuestas: Array(1)}
{pregunta_idpregunta: 7, texto: "Cuando has tomado la decisión de hacer este trayec…do en cuenta tu salud y el ejercicio que suponía?", respuestas: Array(1)}
{pregunta_idpregunta: 8, texto: "Cuando has tomado la decisión de hacer este trayec…e algún lugar del trayecto pudiera ser peligroso?", respuestas: Array(1)}
{pregunta_idpregunta: 9, texto: "Cuando has tomado la decisión de hacer este trayec… cuenta el impacto positivo en el medio ambiente?", respuestas: Array(1)}
```

Ilustración 123 - Estructura de la valoración de una ruta

Descargar información de rutas

El usuario tiene dos opciones a la hora de descargar la información de las rutas que se visualizan en el mapa: descargar coordenadas o descargar la información de la ruta.



En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama denominado JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. La información del JSON2 y JSON3 viene preparada ya para que en el front-end, únicamente haya que poner dicha estructura en columnas, para poder así descargar el fichero. Por un lado, el JSON2, que devuelve las coordenadas de las rutas, posee la estructura que se puede visualizar en la *Ilustración 124*.

```
{route: "1747c755756f99331523460516620", latitud: 43.2637, longitud: -2.95093}
{route: "1747c755756f99331523460516620", latitud: 43.2637, longitud: -2.95096}
{route: "1747c755756f99331523460516620", latitud: 43.263, longitud: -2.94772}
{route: "1747c755756f99331523460516620", latitud: 43.2626, longitud: -2.94742}
```

Ilustración 124 - Estructura de la descarga de coordenadas

En cambio, la estructura del JSON3 para poder descargar la información de las rutas posee varias cosas: la cabecera de códigos que deben poseer las columnas del fichero destinada a la valoración de las rutas, y, por otro lado, toda la información del fichero. Esta

estructura se puede visualizar en la *Ilustración 125*.


```
▶ header: (9) ["C1P1", "C1P2", "C1P3", "C1P4", "C1P5", "C1P6", "C1P7", "C1P8", "C1P9"]
▼ info: Array(57)
  ▼ 0:
      C1P1: ""
      C1P2: ""
      C1P3: ""
      C1P4: ""
      C1P5: ""
      C1P6: ""
      C1P7: ""
      C1P8: ""
      C1P9: ""
      anyo_nacimiento: 1984
      categoria_idcategoria: ""
      configuracion_version: 1
      copia_de: ""
      cuantas_rutas: 0
      cuestionario: ""
      dia: 20180411
      distancia_recorrida: 566.248
      duracion: 31
      genero: 1
      hora: 1728
      idusuario: "1747c755756f99331523454239196"
      movilidad_reducida: 0
      municipio_fin: "Bilbao"
      municipio_inicio: "Bilbao"
      municipio_procedencia: "Bilbao"
      route: "1747c755756f99331523460516620"
```


Ilustración 125 - Estructura de la descarga de información

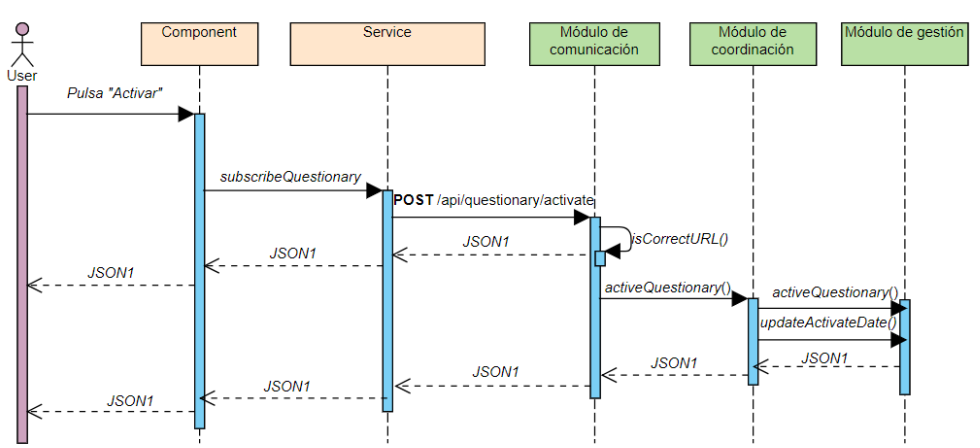
3. Administrador

Los siguientes diagramas de secuencia corresponden a los casos de uso extendidos del usuario "Administrador", estructurados de la misma manera:

3.1. Gestionar cuestionarios

La funcionalidad de "Gestionar cuestionarios" se divide en los siguientes diagramas de secuencia:

Activar un cuestionario

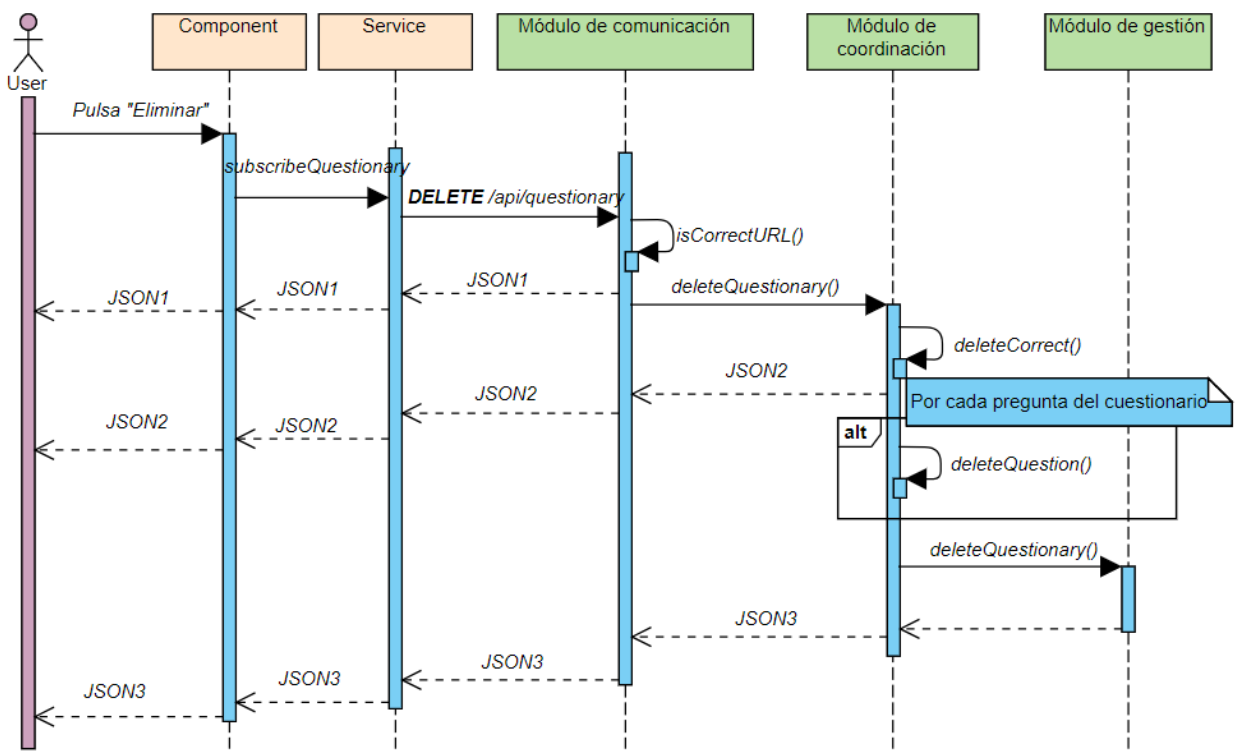


El "Módulo de gestión" realiza varias llamadas a la base de datos en el proceso de activar un cuestionario. En primer lugar activa el cuestionario, a la vez que desactiva el que está activo en ese momento, y posteriormente, almacena la fecha activación en la tabla correspondiente. En el siguiente fragmento de código se puede visualizar el proceso de activar y desactivar el cuestionario activo.

```
UPDATE cuestionario SET activo = 0 WHERE activo = 1
UPDATE cuestionario SET activo = 1 WHERE version = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos o el proceso no se resuelve satisfactoriamente se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*.

Eliminar un cuestionario



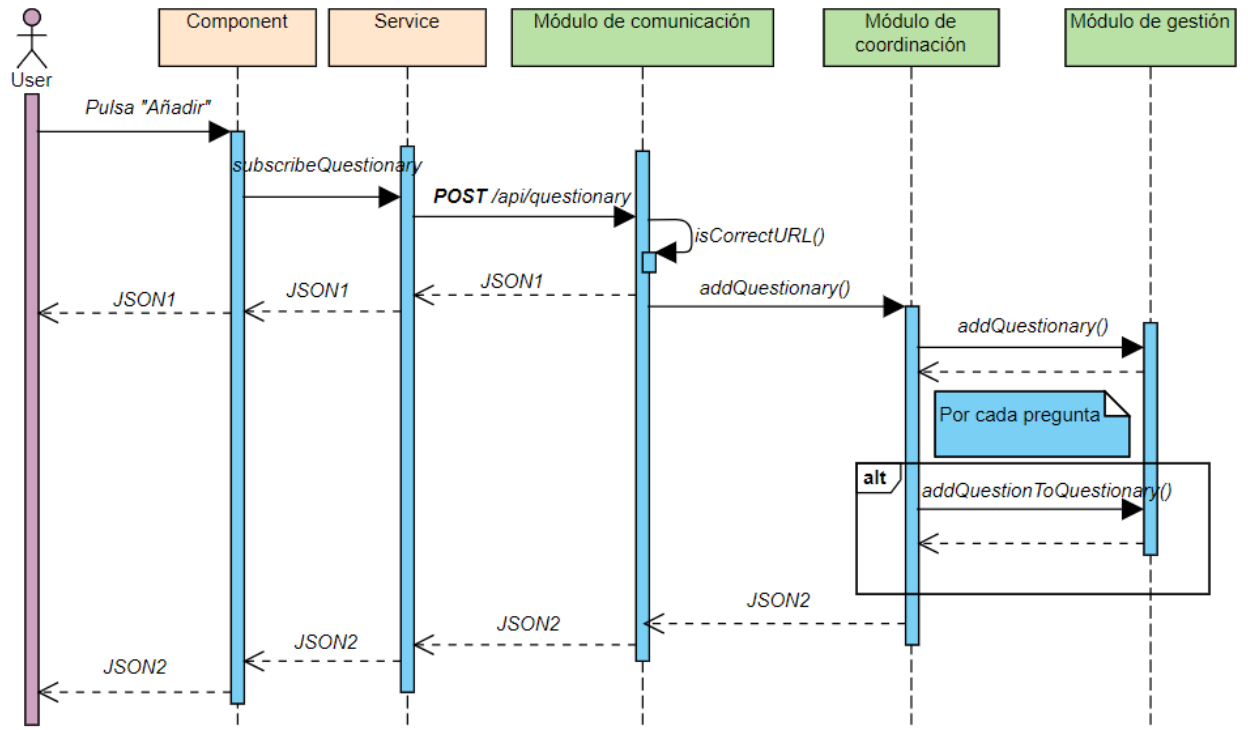
Para eliminar un cuestionario, en primer lugar, se elimina la relación entre el cuestionario y todas las preguntas que lo componen, y posteriormente, se elimina el cuestionario. El siguiente fragmento de código corresponde a esas acciones realizadas contra la base de datos.

```
DELETE FROM pregunta_cuestionario WHERE cuestionario_version = parameter
DELETE FROM cuestionario WHERE version = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Por otro lado, si no es posible eliminar el cuestionario, por ejemplo, si está activo, se devuelve un mensaje de error en el JSON2. Y, finalmente, el JSON3, que posee la misma estructura que el JSON1, devuelve un código de

error si el proceso no se ha completado o un mensaje correcto si el cuestionario se ha podido eliminar.

Añadir un cuestionario

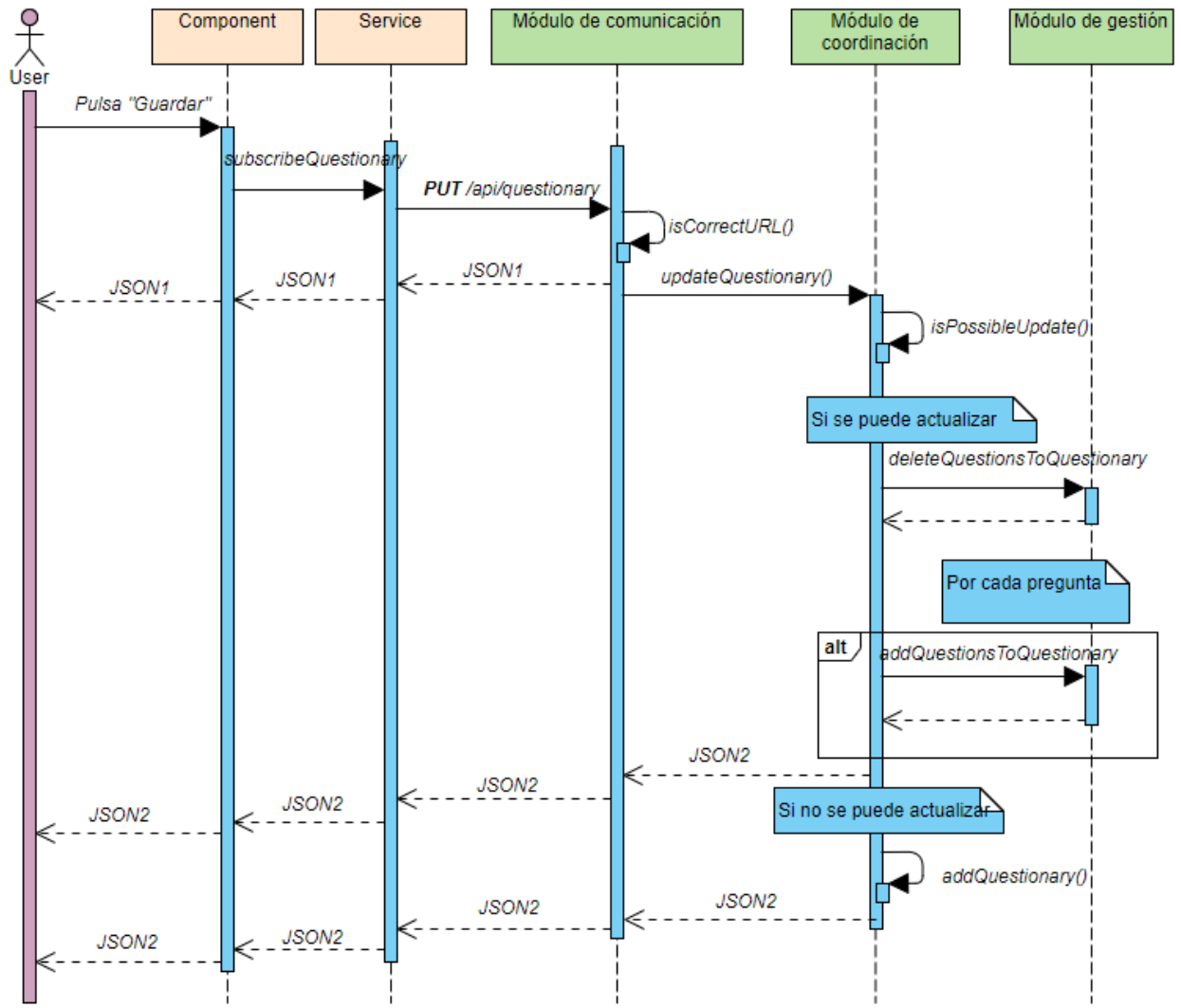


Para añadir un cuestionario, se crea dicho cuestionario y posteriormente, se relaciona el cuestionario creado con las preguntas que ha seleccionado el usuario para que formen parte de él. En el siguiente fragmento de código se pueden visualizar esas acciones.

```
INSERT INTO cuestionario SET parameters
INSERT INTO pregunta_cuestionario SET parameters
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. El JSON2 posee la misma estructura que el JSON1, y devuelve el resultado del proceso.

Editar un cuestionario



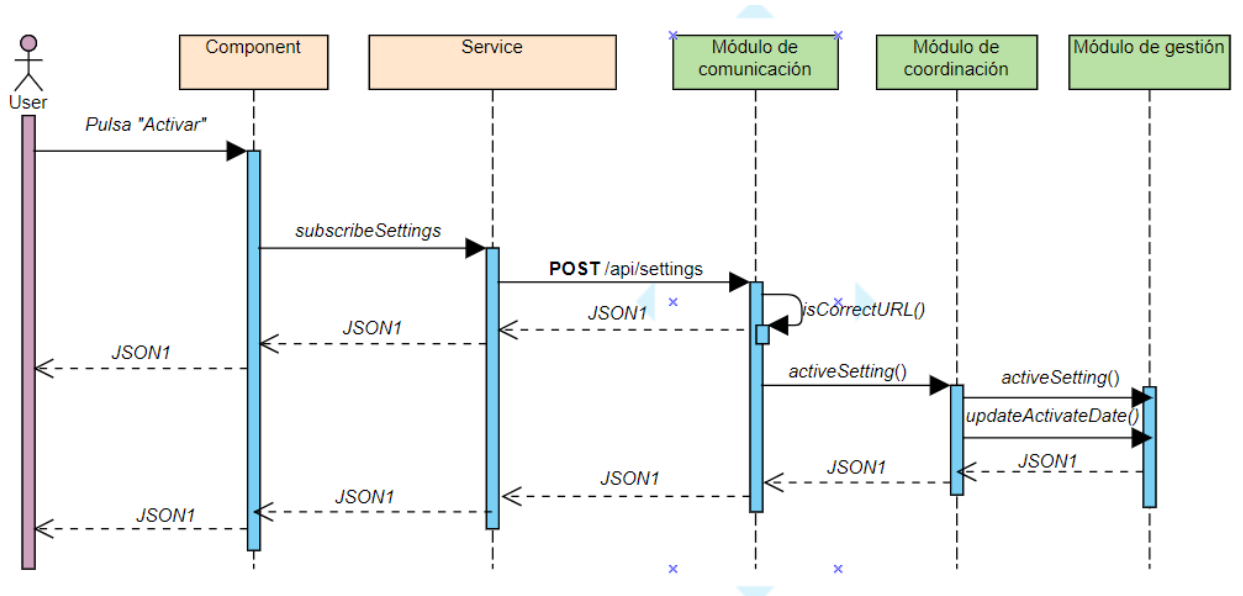
Para editar un cuestionario, se elimina la relación existente con las preguntas que ya poseía, y posteriormente, se relaciona el cuestionario creado con las nuevas preguntas que ha seleccionado el usuario. Estas dos acciones contra la base de datos ya se han detallado en las funcionalidades de añadir y eliminar una configuración.

En el diagrama anterior, si la petición no posee los parámetros correctos o el proceso no se resuelve satisfactoriamente se devuelve un mensaje de error, en el diagrama se denomina JSON1 o JSON2. El formato de este mensaje se puede visualizar en la *Ilustración 115*.

3.2. Gestionar configuraciones

La funcionalidad de "Gestionar configuraciones" se divide en los siguientes diagramas de secuencia:

Activar una configuración

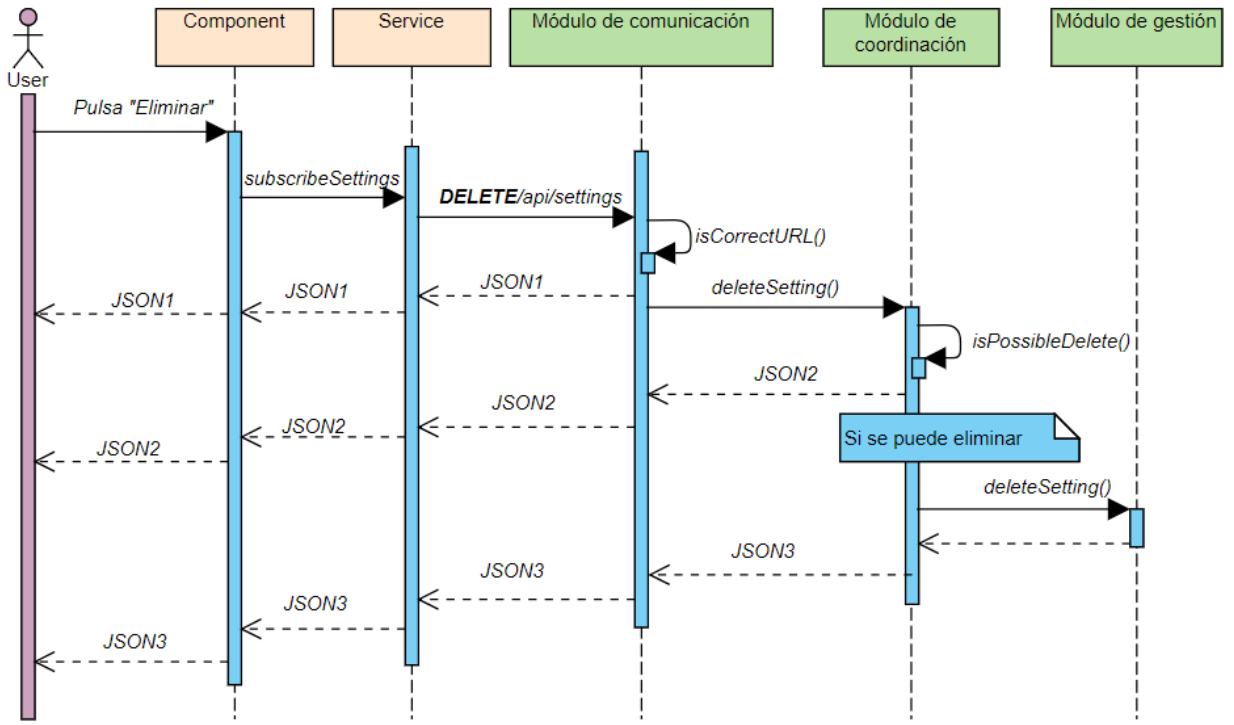


El "Módulo de gestión" realiza varias llamadas a la base de datos en el proceso de activar una configuración. En primer lugar activa la configuración, a la vez que desactiva la que está activa en ese momento, y posteriormente, almacena la fecha activación en la tabla correspondiente. En el siguiente fragmento de código se puede visualizar el proceso de activar y desactivar la configuración activa.

```
UPDATE configuracion SET activo = 0 WHERE activo = 1
UPDATE configuracion SET activo = 1 WHERE version = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos o el proceso no se resuelve satisfactoriamente se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*.

Eliminar una configuración

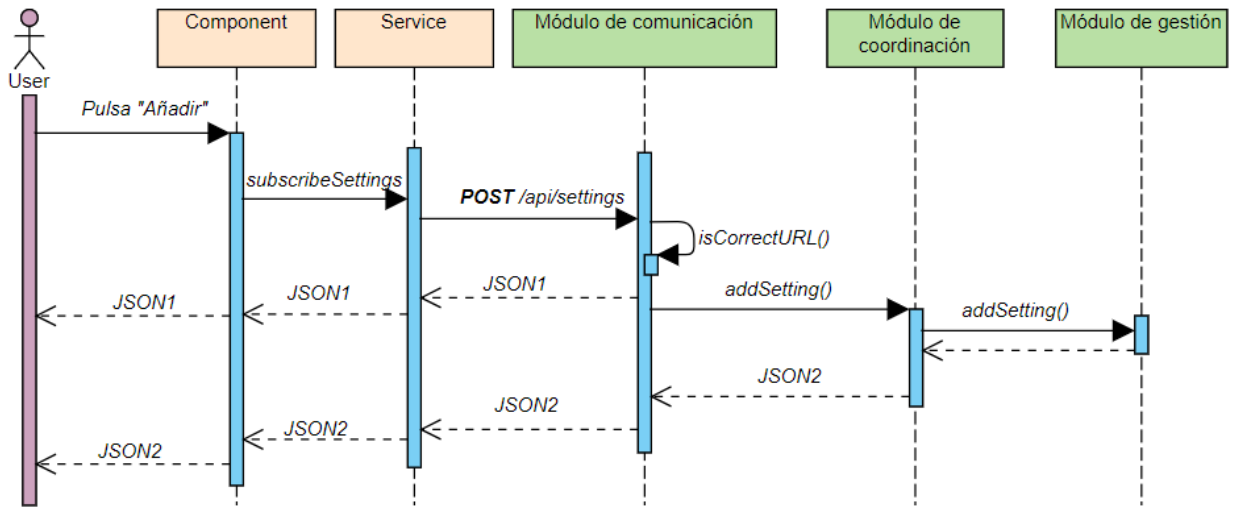


El "Módulo de gestión" realiza una única llamada a la base de datos, para poder eliminar una configuración, en caso de que esta se pueda eliminar. En el siguiente fragmento de código se muestra la sentencia que se ejecuta contra la base de datos.

```
DELETE FROM configuracion WHERE version = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos o el proceso no se puede realizar se devuelve un mensaje de error, en el diagrama se denomina JSON1 o JSON2. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, si la configuración se elimina correctamente se devuelve el JSON3, que posee un código para indicar que el proceso ha sido correcto.

Añadir una configuración



Para añadir una configuración se realiza una única llamada a la base de datos, ya que la información de cada configuración solo se almacena en una tabla. En el siguiente fragmento se puede visualizar la sentencia.

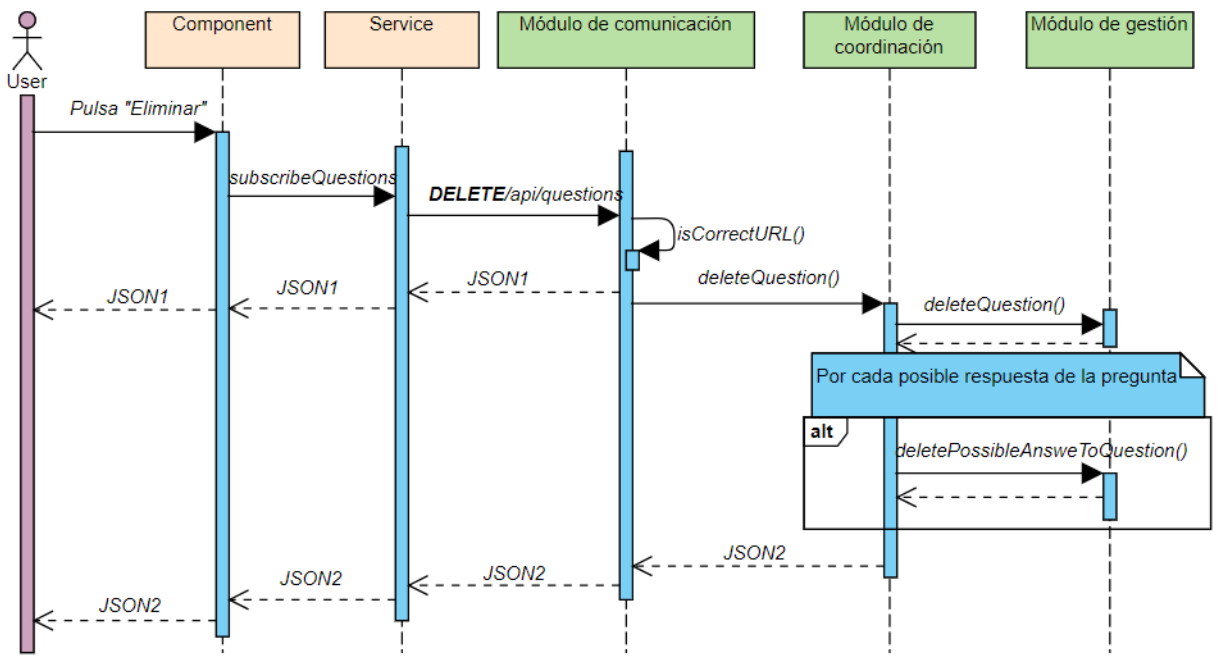
```
INSERT INTO configuracion SET parameters
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar al anterior.

3.3. Gestionar preguntas

La funcionalidad de “Gestionar preguntas” se divide en los siguientes diagramas de secuencia:

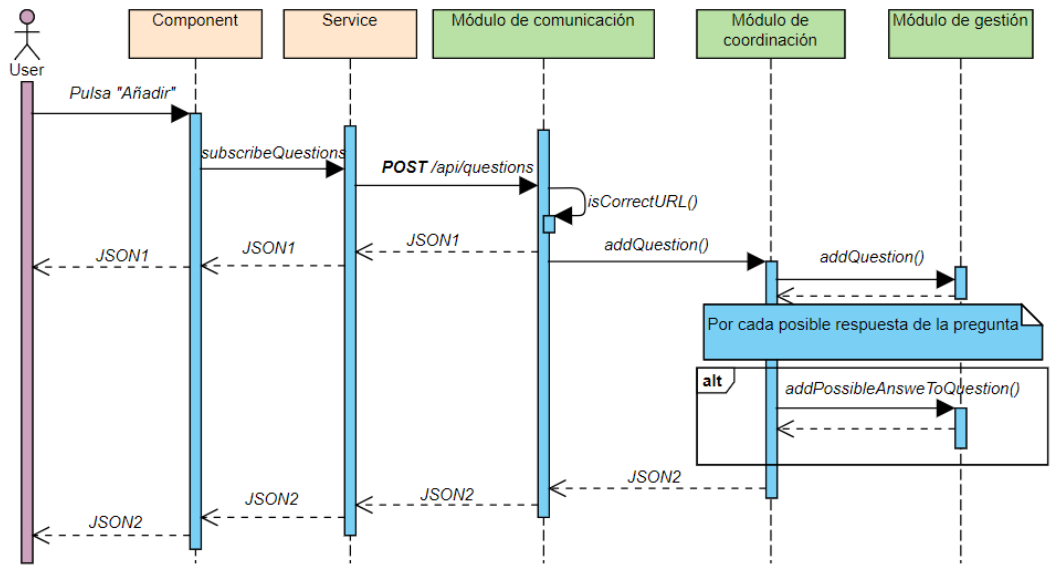
Eliminar una pregunta



El proceso de eliminar una pregunta realiza, en primer lugar, una llamada a la base de datos para eliminar la información de esa pregunta, y posteriormente, elimina todas las posibles respuestas que posee la pregunta.

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

Añadir una pregunta

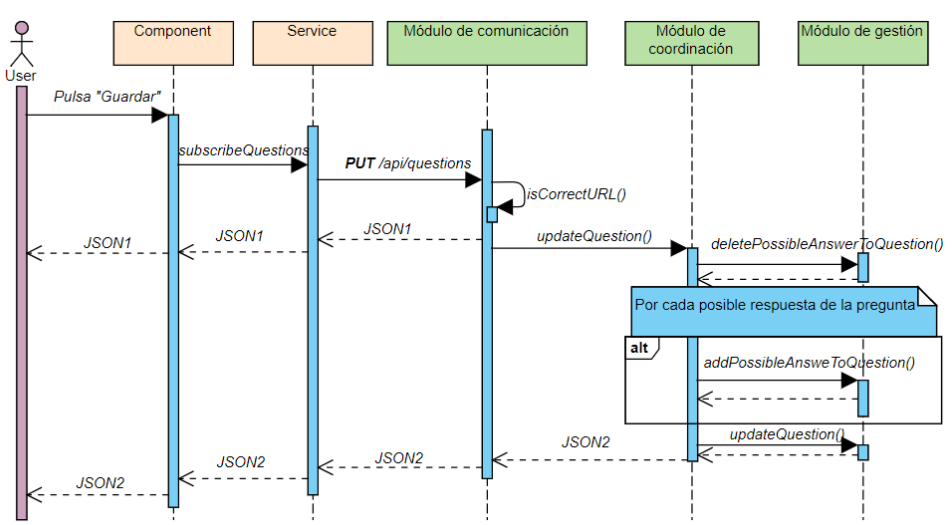


Para añadir una pregunta, en primer lugar, se añade su información a la base de datos, y posteriormente se relaciona la pregunta ya creada con las posibles respuestas que va a tener la pregunta. En el siguiente código se pueden visualizar estas inserciones en la base de datos.

```
INSERT INTO pregunta SET parameters
INSERT INTO respuesta SET parameters
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

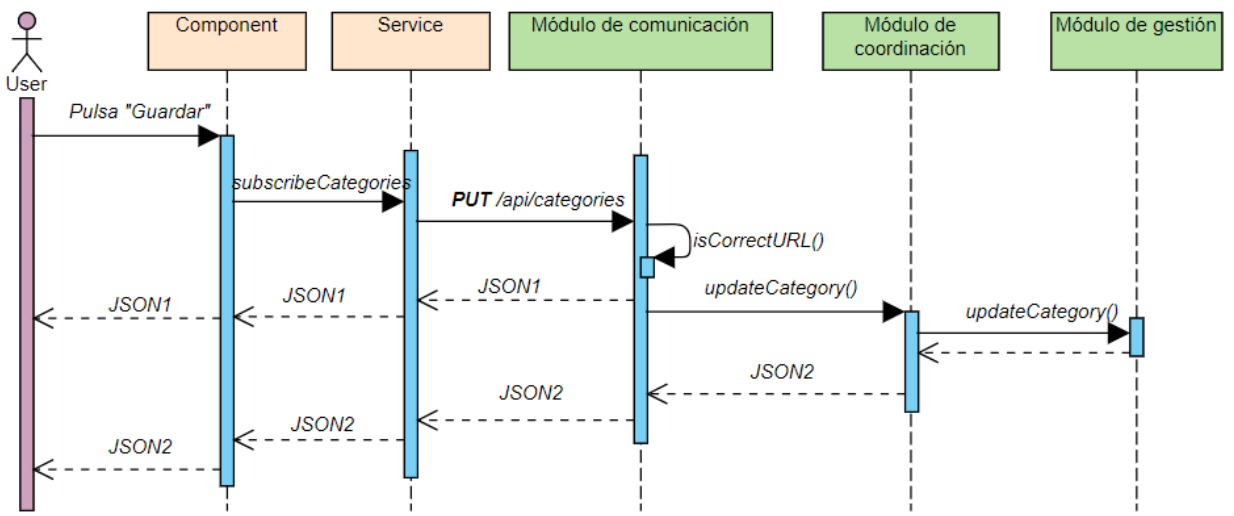
Editar una pregunta



El proceso de editar una pregunta es similar al de edición de cuestionarios. En primer lugar se eliminan las posibles respuestas que tenía dicha pregunta, para posteriormente añadir las que se desea que posea la pregunta a partir de ese momento. Finalmente se actualizan las características de la pregunta.

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

3.4. Gestionar categorías



El "Módulo de gestión" realiza una única llamada a la base de datos para actualizar el texto de una categoría. Esta llamada se puede visualizar en el siguiente fragmento de código.

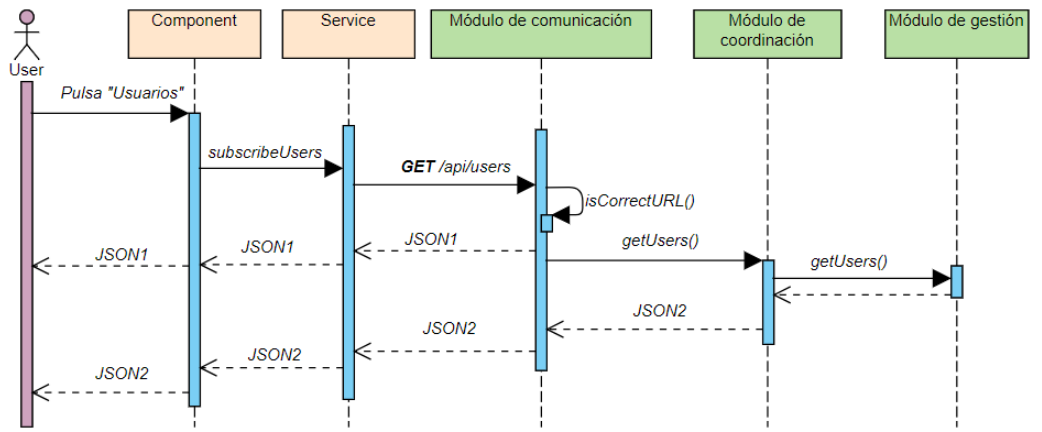
```
UPDATE categoria SET parameter WHERE idcategoria = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

3.5. Gestionar usuarios

La funcionalidad de “Gestionar usuarios” se divide en los siguientes diagramas de secuencia:

Obtener los usuarios

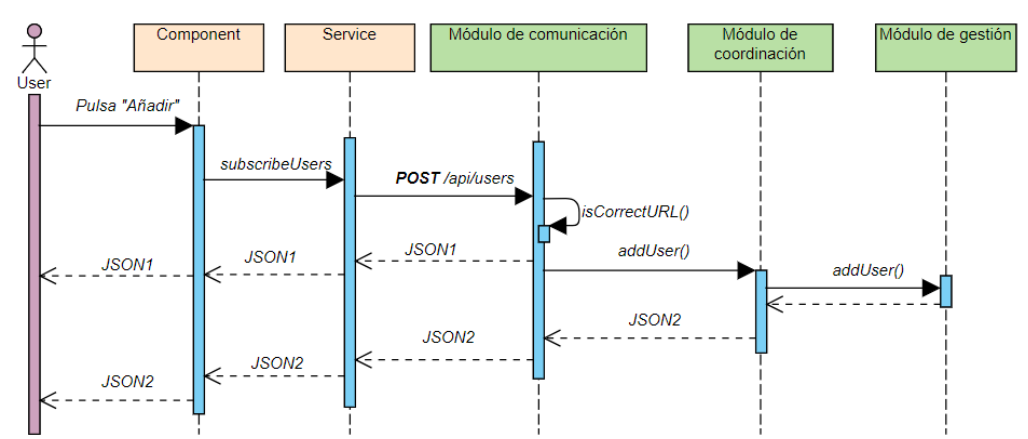


Para obtener todos los usuarios de la aplicación web, tanto investigadores como administradores, se realiza una llamada a la base de datos con la siguiente sentencia.

```
SELECT idResearcher, nickResearcher, rolResearcher, nameResearcher,
surnameResearcher FROM researcher
```

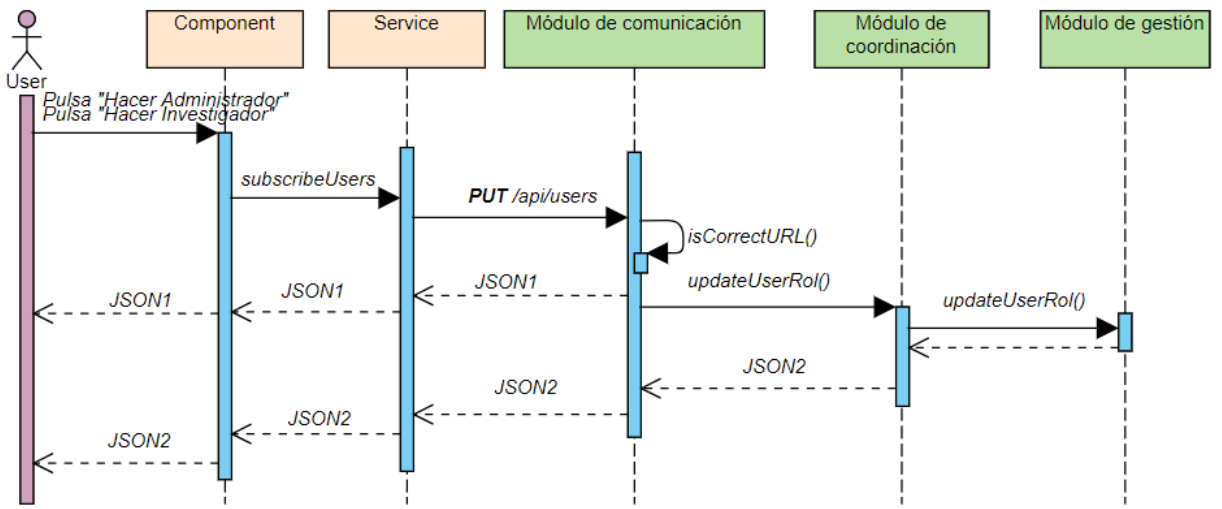
En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

Añadir un usuario



Para añadir un usuario, ya sea investigador o administrador, únicamente se realiza la llamada a la base de datos correspondiente a inserción. Por otro lado, en el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

Actualizar permisos de un usuario



Actualizar los permisos de un usuario es similar a actualizar su información, ya que únicamente se realiza la sentencia de actualización contra la base de datos. Dicha sentencia se puede visualizar en el siguiente fragmento de código.

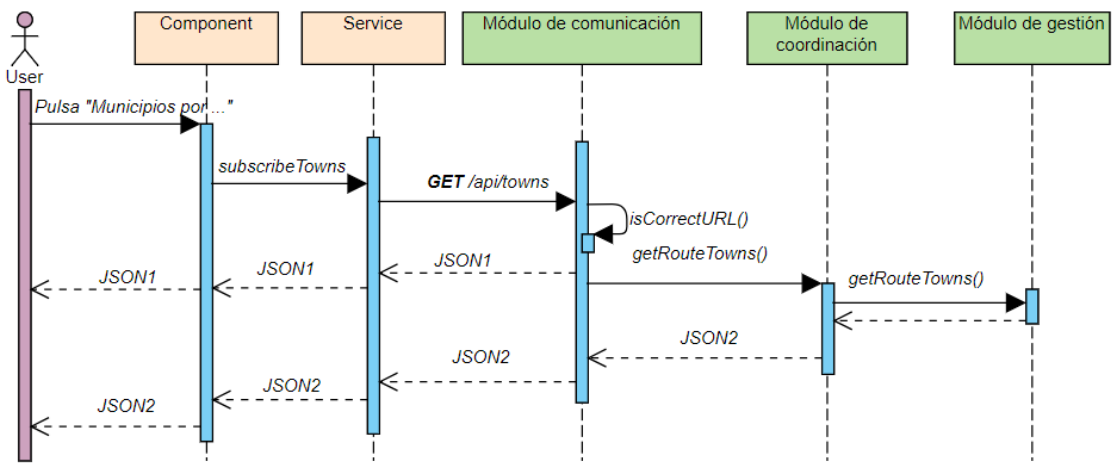
```
UPDATE researcher SET parameters WHERE nickResearcher = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

3.6. Gestionar municipios

La funcionalidad de “Gestionar municipios” se divide en los siguientes diagramas de secuencia:

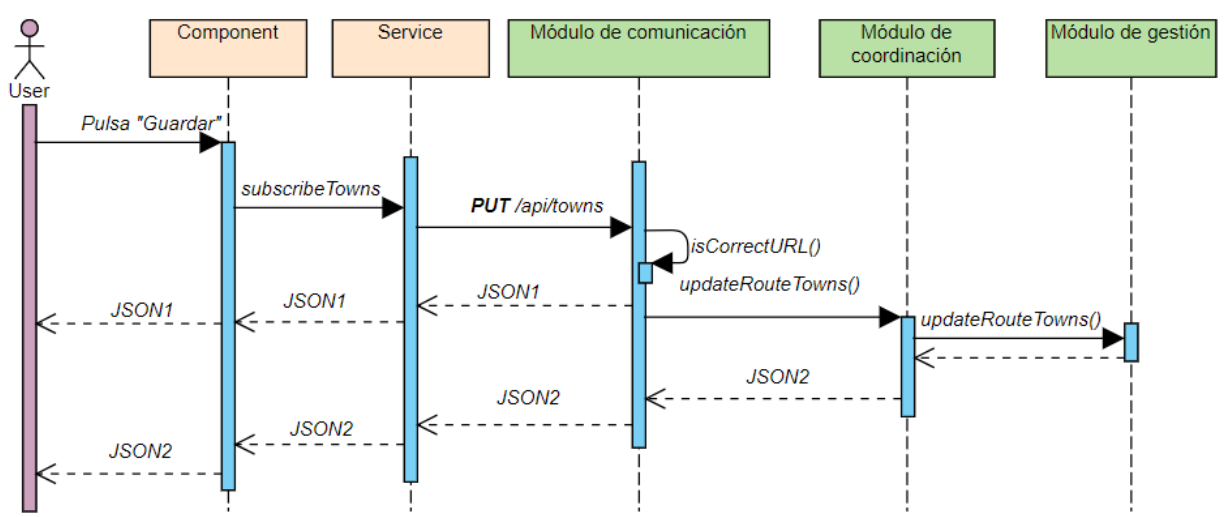
Obtener municipios



Para obtener los municipios de las rutas, de manera que así se puedan visualizar los errores, se realiza una sentencia contra la base de datos que recoge los municipios de inicio y de fin.

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.

Actualizar un municipio



Si el usuario ha encontrado un error en los municipios, únicamente se realiza una consulta a la base de datos para actualizar los municipios de la ruta concreta donde se ha

producido el error. Esta sentencia se puede visualizar en el siguiente fragmento de código.

```
UPDATE ruta SET parameters WHERE idruta = parameter
```

En el diagrama anterior, si la petición no posee los parámetros correctos se devuelve un mensaje de error, en el diagrama se denomina JSON1. El formato de este mensaje se puede visualizar en la *Ilustración 115*. Además, tanto si el proceso se ha resuelto correctamente como si se ha producido algún error, se devuelve el JSON2 como respuesta, con una estructura similar a la anterior.