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

RESTORATIVE CITY

Alumno/Alumna: Hernández, Rivas, Oscar

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

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

Curso: 2017-2018

Fecha: Bilbao, 18, julio, 2018

RESUMEN

Las demandas de la vida cotidiana, especialmente en entornos urbanos, requieren del empleo continuado de recursos psicológicos para cuestiones como procesar información, gestionar tareas o manejar la motivación y las emociones. Como consecuencia, a menudo experimentamos fatiga mental, aburrimiento, estrés e irritabilidad, que conducen a la disminución de la eficacia y el bienestar.

En Psicología Ambiental se descubrió que el contacto con ciertos espacios ayuda a la recuperación de dichas facultades menoscabadas por la actividad diaria, restaurándolas. La evidencia empírica muestra que el contacto con ciertos espacios naturales disminuye significativamente los niveles de estrés, emociones negativas y recupera la capacidad atencional.

Sin embargo, cabe la posibilidad de que ciertos espacios puramente urbanos, debido a que poseen una estética concreta, puedan contribuir también a la restauración y con ello también al bienestar de las personas.

Por ello, se pretende crear *Restorative City*: un entorno que, combinando una aplicación móvil y una aplicación web, permita medir el nivel de capacidad restauradora de un espacio urbano.

LABURPENA

Eguneroko bizitzaren eskakizunek baliabide psikologikoak etengabe erabiltzea eskatzen dute, esate baterako, informazioa prozesatzea, zereginak kudeatzea edo motibazioa eta emozioak kudeatzea, batez ere hiriguneetan. Ondorioz, adimen-nekea, asperdura, estresa eta suminkortasuna sortzen zaizkigu, eraginkortasuna eta ongizatea murriztuz.

Ingurumen Psikologian espazio jakin batzuekin kontaktuan izateak egunean eguneko jarduerek kaltetutako faktore horiek berreskuratzen laguntzen dutela aurkitu zen. Ebidentzia enpirikoak ingurune natural batzuekin kontaktuan egoteak estres maila eta emozio negatiboak nabarmenki murrizten dituela eta arreta gaitasuna berreskuratzen duela erakusten du.

Alabaina, posible da hiri-espazio jakin batzuk, estetika espezifikoa dutelako, berreskuratzeko balio izatea eta, era berean, pertsonen ongizaterako.

Horregatik, Restorative City sortu nahi da: mugikor aplikazio bat eta web aplikazio bat konbinatuz hiri ingurune baten berreskuratze ahalmena ahalbidetzeko aukera.

ABSTRACT

The demands of everyday life, especially in urban environments, require the continued use of psychological resources for issues such as information, management or motivational tasks and emotions. As a consequence, we often experience mental fatigue, boredom, stress and irritability, which lead to decreased effectiveness and well-being.

In Environmental Psychology it was discovered that contact with some spaces helps the recovery of faculties impaired by daily activity, restoring them. Empirical evidence shows that contact with certain natural spaces significantly decreases stress levels, negative emotions and regains attention capacity.

However, there is a chance that some urban spaces, due to their specific aesthetic, can also contribute to the restoration and well-being of people.

Therefore, we intend to create the Restorative City: an environment that, combining a mobile application and a web application, allows measuring the restorative capacity of an urban space.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

Las demandas de la vida cotidiana -familiares, laborales, académicas, etc.-, especialmente en entornos urbanos, requieren del empleo continuado de recursos psicológicos para cuestiones como procesar información, gestionar tareas o manejar la motivación y las emociones. Como consecuencia, a menudo experimentamos fatiga mental, aburrimiento, estrés e irritabilidad, que conducen a la disminución de la eficacia y el bienestar (Kaplan & Kaplan, 1989).

Por restauración psicológica se conoce el proceso de recuperación de esos recursos físicos, psicológicos y sociales que han disminuido ante las diferentes demandas cotidianas (Harting, 2004) (Ulrich, 1993).

En Psicología Ambiental se descubrió que el contacto con ciertos espacios ayuda a la recuperación de dichas facultades menoscabadas por la actividad diaria, restaurándolas (Kaplan S. , 1995). La evidencia empírica muestra que el contacto con ciertos espacios naturales disminuye significativamente los niveles de estrés, emociones negativas y recupera la capacidad atencional (Tyrväinen, y otros, 2014). Por ejemplo, pasear por el campo permite disminuir el estrés acumulado.

Sin embargo, cabe la posibilidad de que ciertos espacios puramente urbanos, debido a que poseen una estética concreta (una calle especialmente bonita), o los servicios de los que disponen (una zona de ocio), puedan contribuir también a la restauración y con ello también al bienestar de las personas (San Juan, Subiza-Pérez, & Vozmediano, 2017).

Por ello, este Trabajo Fin de Grado, en colaboración con el grupo CRIM-AP[1] de la UPV/EHU[2], pretende facilitar la recogida de datos de las experiencias de restauración de los ciudadanos en entornos urbanos, dándoles al mismo tiempo la posibilidad de comprobar qué lugares les ayudan a reducir el estrés y mejorar la capacidad de atención dirigida.

Para ello, se pretende crear *Restorative City*: un entorno que, combinando una aplicación móvil y una aplicación web, permita medir el nivel de capacidad restauradora de un espacio urbano. La aplicación móvil podrá ser usada por cualquier persona, mientras que la aplicación web está pensada para su uso por parte de los investigadores del grupo CRIM-AP.

Algunas de las razones por las que se ha elegido este TFG son las que se exponen a continuación:

[1] https://www.ehu.eus/es/web/dms/equipo
[2] https://www.ehu.eus

- Aprendizaje y perfeccionamiento de nuevas tecnologías, muy utilizadas y demandadas hoy en día, para el desarrollo de aplicaciones web y móviles.
- Creación de una herramienta que será dedicada para un uso real, con datos reales, y que además podrá ayudar tanto a los miembros del grupo CRIM-AP, como a los ciudadanos que usen la aplicación.
- Aprendizaje de nuevos modelos y estructuras de trabajo muy utilizados actualmente en el desarrollo de aplicaciones web y móviles.
- Tener como meta que de que la aplicación global resultante (web y móvil) pueda expandirse y pueda ser usada en un ámbito mayor, por ejemplo, por administraciones públicas, tales como ayuntamientos para mejorar la calidad de sus ciudades, pueblos, municipios, etc. y por tanto el bienestar de sus habitantes.
- Pensamiento de una posible ampliación de la herramienta, integrándose con otros proyectos de temáticas similares (como por ejemplo *Walkability Capturer* y *Walkability Analyzer*), y como resultado de ello surja una aplicación mucho más completa, útil y beneficiosa para los usuarios.
- Capacidad de aplicar los conocimientos adquiridos durante el grado, tanto para el análisis, diseño e implementación, como para la resolución de problemas que surjan durante el desarrollo del proyecto.

Por último, cabe destacar que 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.

2. PLANTEAMIENTO INICIAL

En este capítulo se detalla todo lo que se llevará a cabo y tendrá que tenerse en cuenta para la gestión y organización del proyecto, esto es, se exponen cuáles son los objetivos y el alcance del proyecto, se especifica qué arquitectura se va a usar para el desarrollo de *Restorative City*, se describe qué herramientas o entornos se usarán para el desarrollo, se realiza una estimación económica y temporal, y se plantea una gestión de riesgos.

2.1. Objetivos

Los principales objetivos del proyecto, tanto funcionales como de aprendizaje personal, son los siguientes:

1. Realización de una aplicación web que permita a los integrantes del grupo CRIM-AP definir qué puntos urbanos desean examinar para comprobar si son restauradores o no, y analizar los resultados obtenidos.

   Es importante puntualizar que, para un mejor entendimiento, y facilitar la lectura, las referencias a los puntos urbanos que definen los miembros del grupo CRIM-AP pueden ser las siguientes: puntos, puntos posiblemente restauradores (PPR) y puntos en estudio.

2. Realización de una aplicación móvil que facilite al grupo CRIM-AP la recogida de datos sobre las personas en los puntos urbanos que hayan definido. Además, dichos datos deben permitir a las personas medir su nivel de estrés y la capacidad de atención dirigida y comprobar qué lugares les ayudan a reducir el estrés y mejorar su atención. Para ello, cada vez que un usuario entre en un PPR, deberá realizar una prueba de atención y carga cognitiva basada en una de las alternativas al SDMT (Symbol Digit Modalities Test) propuestas por Hinton et al. (Hinton-Bayre, Anton, Geffen, & Gina, 2005).

   La prueba consiste en presentar al usuario durante 90 segundos una secuencia de símbolos (Ilustración 1) y pedirle que indique el dígito que se corresponde con el símbolo presentado. Para ello se proporciona también una leyenda con la correspondencia entre los símbolos y los dígitos numéricos. En base al número de respuestas, errores y aciertos se mide la capacidad de atención y carga cognitiva del usuario. Tras esta primera prueba se le pide al usuario que valore tanto su nivel de estrés como de felicidad en una escala de 0 a 100.

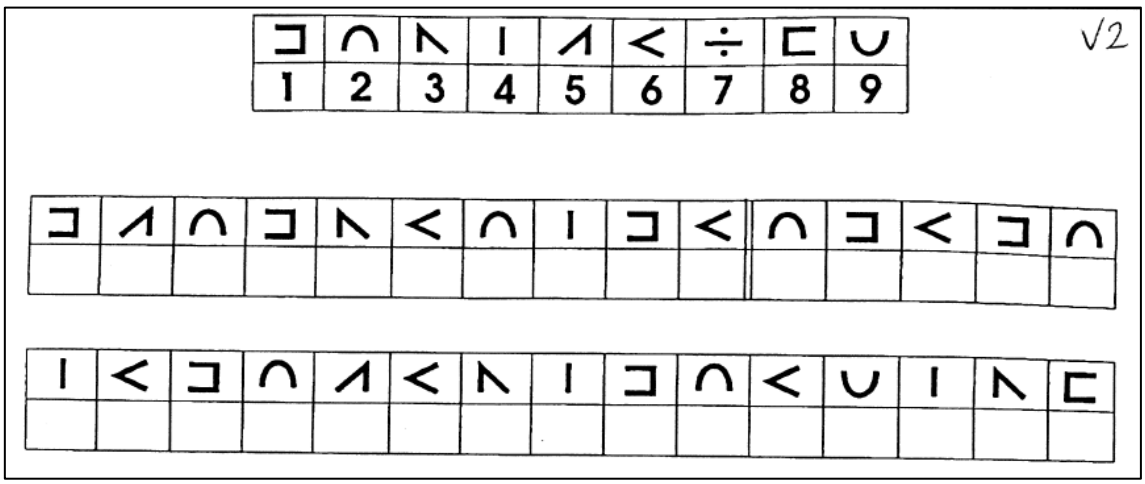


Ilustración 1 - Ejemplo de secuencia de símbolos

Al abandonar el PPR se le vuelve a someter al usuario a las mismas pruebas y además se le pide que rellene un cuestionario aportando información extra sobre sus sensaciones.

3. Aprendizaje de nuevas tecnologías para cada desarrollo, tanto para las partes *front-end* como *back-end*.
4. Estudio y análisis de las arquitecturas y modelos de trabajo que se utilizan actualmente para cada desarrollo y aplicarlos.

2.2. Definiciones, acrónimos y abreviaturas

A continuación, se listarán y explicarán cada uno de los términos relacionados con este proyecto:

- ***JSON:*** Acrónimo de *JavaScript Object Notation*. Es un formato de texto ligero para el intercambio de datos.
- ***REST:*** Acrónimo de *Representational State Transfer*, es un modelo de arquitectura. Esta arquitectura se caracteriza por ser *stateless* (sin estado), es decir, que en cada petición que se haga a cualquier servicio, el usuario debe identificarse ya que el sistema no lo recordará. Generalmente, dicha identificación se realiza mediante un *token* el cual es proporcionado en la cabecera de las peticiones HTTP o HTTPS. REST nos permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP.

- ***SOAP:*** Acrónimo de *Simple Object Access Protocol, e*s un protocolo que permite la comunicación entre aplicaciones a través de mensajes por medio de Internet. Los mensajes SOAP son un fichero XML.
- ***WSDL:*** Acrónimo de *Web Services Description Language*, es una notación XML para describir un servicio web. Una definición WSDL indica a un cliente cómo componer una solicitud de servicio web.
- ***API:*** Acrónimo de *Application Programming Interfaces*, es una especificación formal sobre cómo un módulo de software se comunica o interactúa con otro. Generalmente, el intercambio de mensajes o datos suele ser en formato JSON o XML.
- ***API REST:*** Una API que sigue el modelo de arquitectura REST.
- ***Back-end:*** Desarrollo que se corresponde con la lógica de la aplicación y que generalmente suele estar alojado en un servidor. Este se encarga de dar respuesta a las peticiones que se le realizan. Es el encargado de la manipulación de datos e interacción con las bases de datos. El *back-end* no dispone de interfaz gráfica con lo cual para el usuario es invisible.
- ***Front-end:*** Dicho de una manera simple, es lo que el usuario ve al utilizar la aplicación. Es la parte visible de cualquier aplicación. Este desarrollo se encarga de interactuar con el usuario y facilitar el uso del sistema completo a través de su interacción con un *back-end*.
- ***SPA:*** Acrónimo de *Single Page Application*, es un tipo de aplicación web donde todas las vistas son mostradas en la misma página, sin recargar el navegador. En este tipo de aplicaciones web solo existe un único punto de entrada, esto es, no es posible a través de ninguna dirección URL acceder de manera separada a algún contenido o parte de la aplicación. La forma de proceder de estas aplicaciones cuando el usuario realiza una interacción es mostrar u ocultar vistas. Por ello, se podría definir finalmente que una aplicación SPA dispone de múltiples vistas, no páginas. Comúnmente, este tipo de aplicaciones web se comunican con una API REST a modo de servidor.
- ***Framework:*** *"Es una estructura en capas que indica qué tipo de programas pueden o deben ser construidos y cómo se interrelacionan. […] Algunos también incluyen programas reales, especificaciones de interfaces de programación u ofrecen herramientas de programación"* (Rouse, 2016)

- ***IDE:*** Un Entorno Integrado de Desarrollo *(Integrated Development Environment)* es una aplicación informática que proporciona servicios para facilitarle al desarrollador o programador el desarrollo de software.

2.3. Arquitectura

Actualmente existen diversas arquitecturas que pueden ser válidas para resolver las necesidades que se plantean.

Para este proyecto en concreto, se pretende hacer uso de servicios web, esto es, servicios a los que se les realiza una petición esperando una respuesta a cambio con la información correspondiente. Partiendo de esa necesidad, se analizarán dos de las arquitecturas que mejor podrían aplicarse para el desarrollo: SOAP y REST.

Una arquitectura SOAP es comúnmente utilizada para comunicaciones entre sistemas o aplicaciones. El intercambio de datos se realiza mediante el lenguaje XML. Las peticiones a los servicios SOAP se realizan, por lo general, a través del protocolo HTTP, que suele ser el más común para invocar servicios web. Sin embargo, SOAP no está limitado únicamente a dicho protocolo, sino que puede usar además otros como FTP, POP3, TCP y colas de mensajería (JMS, MQ, etc.).

Uno de los puntos fuertes de una arquitectura SOAP es la seguridad. Se definen "contratos", usando el formato WSDL, que el cliente debe cumplir para poder hacer uso de los servicios web que la API expone. En dichos contratos, se define la manera y las condiciones en las que la información debe ser enviada. Estos suelen ser utilizados para la integración de varios sistemas.

El protocolo SOAP, por lo tanto, es robusto, pero a su vez pesado, tanto en tamaño como en procesamiento, ya que hace uso de los XML. Para aclarar este tipo de arquitectura, se presenta a continuación la siguiente ilustración (Ilustración 2) donde se muestra como una aplicación cliente (App A) realiza una petición a otra aplicación (App B):

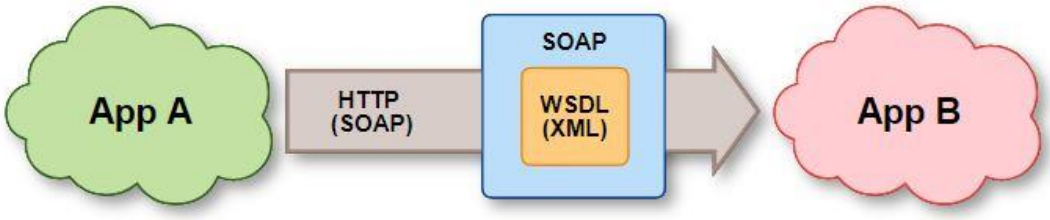


Ilustración 2 - Arquitectura SOAP

Por otro lado, se encuentra la arquitectura REST. Esta es una tecnología mucho más flexible que transporta datos por medio del protocolo HTTP, pero que permite utilizar los diversos métodos que proporciona HTTP para comunicarse (GET, POST, PUT, DELETE y PATCH), y a la vez, utiliza los códigos de respuesta nativos de HTTP (404, 200, 204, 409, etc.).

REST es tan flexible que permite transmitir prácticamente cualquier tipo de datos, ya que dicho tipo es definido en las cabeceras de la propia petición, lo que permite enviar, XML, JSON, Binarios (imágenes, documentos, etc.), Text, etc., y contrasta con SOAP, que solo permite la transmisión de datos en formato XML. Sin embargo, pese a poder soportar varios formatos de datos, el más usado es el formato JSON, puesto que este es interpretado de forma natural en JavaScript y, por lo tanto, muestra alta compatibilidad y facilidad a la hora de tratar con este formato. Además, el formato JSON es muy ligero en tamaño y rápido en procesamiento debido a su sencillez.

La arquitectura REST es la más recomendada para aplicaciones que necesiten realizar peticiones simples y, por lo general, con una alta velocidad de respuesta.

Finalmente, y una vez analizadas las ventajas de cada una de las arquitecturas y sus aplicaciones recomendadas, *Restorative City* hará uso de una API siguiendo la arquitectura REST, como se muestra en la siguiente ilustración (Ilustración 3) donde se muestra como una aplicación cliente (App A) realiza una petición a otra aplicación servidor (App B):

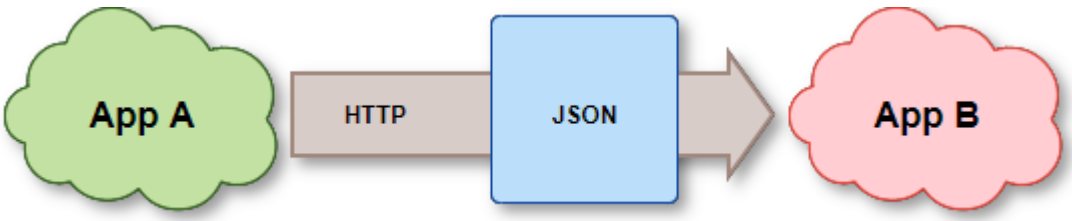


Ilustración 3 - Arquitectura REST

La estructura a seguir para *Restorative City* será la de Cliente-Servidor, a modo de *front-end* y *back-end* (respectivamente). Siguiendo el esquema de la ilustración anterior, los clientes (tanto web como móvil) estarían representados por “App A” y el servidor, la API REST, estaría representada por “App B”.

2.4. Herramientas

A continuación, se detallarán las herramientas que se utilizarán en este Trabajo Fin de Grado, y se explicará para qué sirven y para qué será usada cada una de ellas en particular:

- ***NodeJS*** [3]***:*** Entorno de programación basado en el lenguaje de programación JavaScript. Este será utilizado en la parte del servidor (*back-end*).

- ***TypeScript:*** Lenguaje de programación de código abierto, desarrollado por Microsoft, y que cuenta con herramientas de programación orientada a objetos. Este lenguaje será utilizado para el desarrollo *front-end* en la aplicación web ya que combina las características de JavaScript (en cuanto a flexibilidad) y la programación orientada a objetos. Además, es totalmente compatible con los navegadores ya que el resultado final que se genera es en JavaScript.

- ***Angular:*** Framework desarrollado en TypeScript que sirve para el desarrollo de la parte *front-end* de aplicaciones web y que permite flexibilidad a la hora de visualizar dichas aplicaciones en resoluciones más pequeñas (móviles, tabletas, etc.). Este framework será utilizado para el desarrollo web del cliente (*front-end*).

- ***Android Studio*** [4]***:*** Es el entorno de desarrollo recomendado por el sistema operativo Android para el desarrollo de aplicaciones nativas en este sistema operativo. Ofrece múltiples herramientas para la ayuda al desarrollo, entre las más destacadas está su máquina virtual, que permite emular las aplicaciones para cualquier versión de Android y en cualquier dispositivo. Este entorno de desarrollo se utilizará para el desarrollo de la aplicación móvil.

- ***WebStorm*** [5]***:*** Es un IDE para JavaScript desarrollado por JetBrains. Permite el desarrollo de capa cliente en JavaScript, pero también de la capa servidor con NodeJS, de hecho, ofrece soporte para ambas tecnologías. Esta herramienta será usada para el desarrollo del *back-end* (utilizando JavaScript y NodeJS).

- ***MySQL:*** Es un sistema de gestión de bases de datos (SGBD) relacional desarrollado por Oracle Corporation. Está considerada como la base de datos de código abierto más popular del mundo[6] [7]. Esta herramienta será el SGBD de la API de *Restorative City*.

[3] https://nodejs.org/es/
[4] https://developer.android.com/studio/
[5] https://www.jetbrains.com/webstorm/
[6] https://www.oracle.com/mysql/index.html
[7] https://db-engines.com/en/ranking

- ***Postman*** [8]***:*** Es una plataforma para la creación de solicitudes API. Esta herramienta será usada para realizar las pruebas necesarias del *back-end*.
- ***Visual Studio Code*** [9]***:*** Es un editor de código fuente ligero pero potente desarrollado por Microsoft y está disponible para Windows, MacOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y tiene un rico ecosistema de extensiones para otros lenguajes (como C ++, C #, Java, Python, PHP, Go) y runtimes (como .NET y Unity). Esta herramienta será usada para el desarrollo del *front-end* de la aplicación web.
- ***Office 365 Hogar:*** Es un paquete ofimático desarrollado por Microsoft (incluye Word, Excel, PowerPoint, OneNote, entre otros servicios). Este paquete será usado para realizar la documentación y presentación necesaria.
- ***Visual Paradigm for UML standard edition*** [10]***:*** Se trata de una herramienta que permite realizar diagramas UML. Esta herramienta será usada para realizar los diagramas necesarios para poder realizar la implementación del trabajo.
- ***Dropbox:*** Es un servicio de alojamiento de archivos en la nube. Esta herramienta será usada para realizar copias de seguridad de ficheros y documentos. Este servicio se utilizará para alojar documentos relacionados con el proyecto y además también alojará las copias de seguridad que se vayan realizando.
- ***Google Chrome:*** Es un navegador web desarrollado por Google. Este navegador será usado para el desarrollo de la aplicación web y comprobar que visualmente todo funciona como se espera.
- ***Mozilla Firefox:*** Es un navegador web libre, de código abierto, desarrollado por Fundación Mozilla. Este navegador será usado para el desarrollo de la aplicación web y comprobar que visualmente todo funciona como se espera.
- ***Microsoft Internet Explorer:*** Es un navegador web desarrollado por Microsoft. Este navegador será usado para el desarrollo de la aplicación web y comprobar que visualmente todo funciona como se espera.

[8] https://www.getpostman.com/
[9] https://code.visualstudio.com/
[10] https://www.visual-paradigm.com/

2.5. Alcance

Una de las primeras tareas de todo proyecto, es la definición de su alcance, que en lo que a *Restorative City* respecta, se detalla en los siguientes apartados.

2.5.1. Organización del proyecto

Este proyecto está enfocado a seguir una metodología basada en prototipos durante su desarrollo. La metodología mencionada consiste en la iteración de varios ciclos de vida en cascada. Así, con cada iteración se obtiene un prototipo de la aplicación con más funcionalidades que el anterior. De esta manera, lo que se consigue es cada vez una versión más completa.

El motivo por el que se ha decidido seguir esta metodología es que presenta algunas ventajas frente a otras metodologías. Una de estas ventajas es que, si se produjera un error, este solo afectaría a la última iteración, es decir, al último prototipo y por lo tanto su recuperación sería fácil y rápida. Otra ventaja es que cada prototipo se puede probar de manera individual. Finalmente, otro motivo a destacar es que esta metodología se adecua más a la forma de trabajar que se pretende seguir.

Para este proyecto, se han definido un total de 17 prototipos de desarrollo, que son los siguientes:

- ***En cuanto al desarrollo web (9 prototipos):***
    1. Login y Logout.
    2. Crear un punto a estudiar.
    3. Mostrar mapa con puntos a estudiar.
    4. Gestionar preguntas y cuestionarios.
    5. Modificar un punto a estudiar.
    6. Eliminar un punto a estudiar.
    7. Descargar datos.
    8. Gestionar sugerencias de puntos a estudiar.
    9. Gestionar usuarios.

- ***En cuanto al desarrollo móvil (8 prototipos):***
    1. Pedir datos al usuario y guardarlos.
    2. Mostrar mapa con puntos en estudio activos.
    3. Desarrollar pruebas.
    4. Completar un punto en estudio.
    5. Notificar punto en estudio cercano.
    6. Gestionar puntos en estudio.
    7. Sugerir un punto a estudiar.
    8. Modificar datos del usuario.

2.5.2. Estructura de Descomposición del Trabajo (EDT)

La Estructura de Descomposición del Trabajo (EDT) es un diagrama en el cual son representadas cada una de las tareas que forman parte del proyecto, y posteriormente se detallarán cada una de estas.

A continuación, se presenta el diagrama EDT (Ilustración 4):

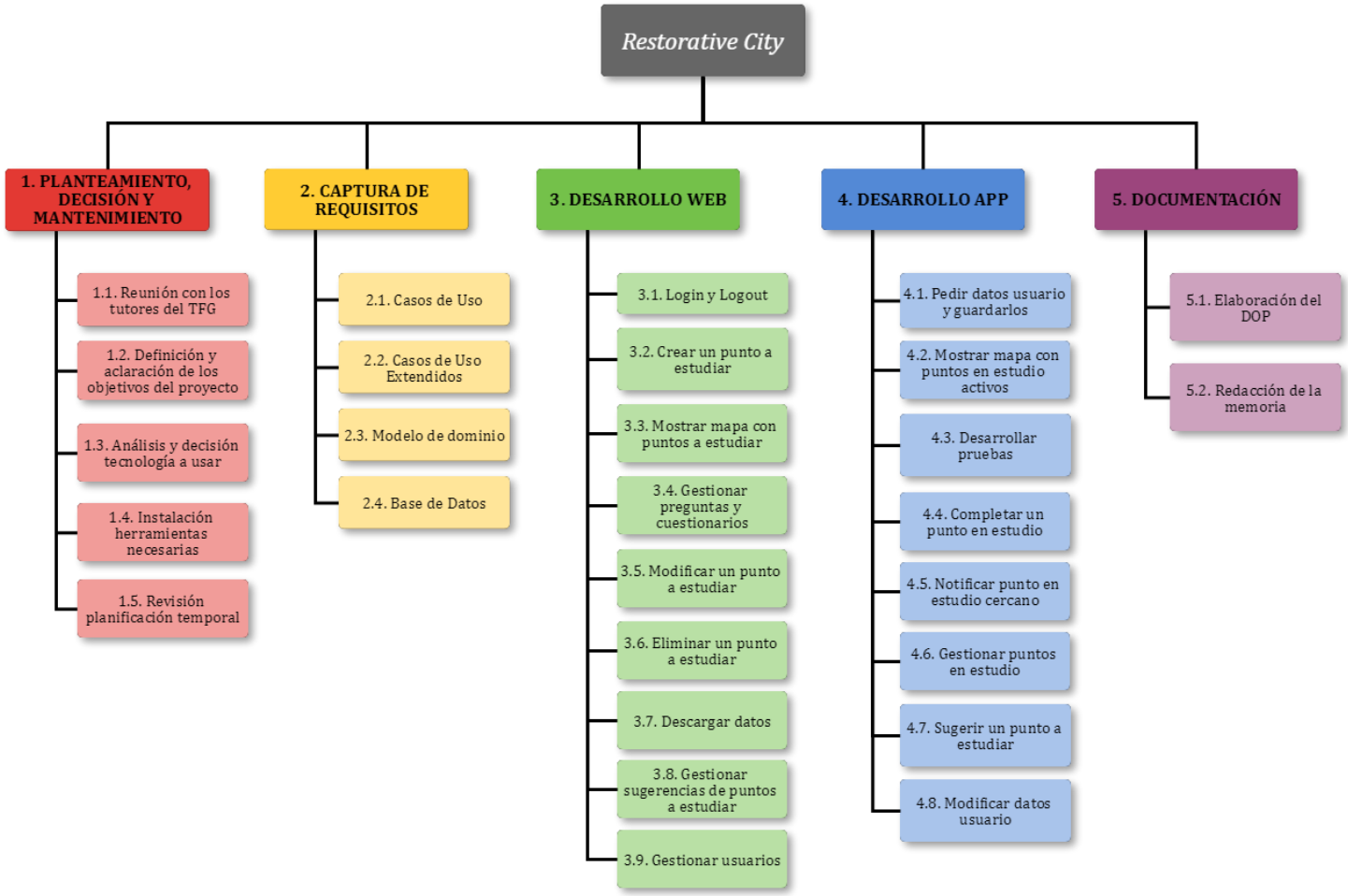


Ilustración 4 - Diagrama EDT

La explicación de cada uno de los paquetes de trabajo principales representados en la ilustración anterior, es la que sigue:

1. **Planteamiento, decisión y mantenimiento:** Este será un paquete de trabajo muy activo durante todo el proyecto, ya que se compone de subtareas que se realizarán de manera rutinarias. Este paquete consiste en tres partes fundamentales. La primera de ellas es la de definir las especificaciones y requerimientos del proyecto. La segunda es el estudio de las herramientas posibles a emplear para la realización del proyecto y la toma de las decisiones por cada una de ellas. La tercera y última, es la supervisión constante de la planificación temporal para detectar pronto los posibles problemas que puedan surgir.
2. **Captura de requisitos:** Este paquete de trabajo engloba las subtareas necesarias para realizar el análisis y diseño previos al proyecto.
3. **Desarrollo web:** Este paquete consiste en la realización de la aplicación web para los miembros del grupo CRIM-AP.
4. **Desarrollo app:** Este paquete consiste en la realización de la aplicación móvil.
5. **Documentación:** Este paquete, al igual que el paquete número 1 anteriormente mencionado, será un paquete muy activo, puesto que contiene tareas que se ejecutarán en prácticamente durante todo el proyecto. Estas tareas consisten en realizar la documentación del proyecto.

Una vez detallado cada paquete de trabajo y explicado la función que desempeñarán en el proyecto, a continuación se desglosan, detallando las subtareas que los componen.

Planteamiento, decisión y mantenimiento

A continuación, se presentan las subtareas del paquete 1:

Tabla 1 - Tarea: Reunión con los tutores del TFG

1.1. Reunión con los tutores del TFG
Paquete de trabajo: PLANTEAMIENTO, DECISIÓN Y MANTENIMIENTO. **Duración:** 6 horas.
Descripción: Reunirse con los tutores del TFG para informar de los avances que haya habido, así como también dudas, problemas surgidos, y demás cuestiones acerca del proyecto.
Entradas: Ninguna. **Salidas:** Avances mostrados. Dudas y problemas revisados y, si procede, solucionados.

Tabla 2 - Tarea: Definición y aclaración de los objetivos del proyecto

1.2. Definición y aclaración de los objetivos del proyecto
Paquete de trabajo: PLANTEAMIENTO, DECISIÓN Y MANTENIMIENTO **Duración:** 1 hora.
Descripción: Reunirse con los tutores del TFG para detallar cada uno de los objetivos del proyecto y resolver las dudas que surjan.
Entradas: Ninguna. **Salidas:** Descripción y objetivos del proyecto definidos.

Tabla 3 - Tarea: Análisis y decisión tecnología a usar

1.3. Análisis y decisión tecnología a usar
Paquete de trabajo: PLANTEAMIENTO, DECISIÓN Y MANTENIMIENTO **Duración:** 30 horas.
Descripción: Analizar las tecnologías actuales que puedan ser usadas para llevar a cabo el proyecto, y tras haberlas probado un mínimo o buscado información al respecto, decidir finalmente cuáles utilizar.
Entradas: Descripción y objetivos del proyecto. **Salidas:** Tecnología a utilizar decidida.

Tabla 4 - Tarea: Instalación herramientas necesarias

1.4. Instalación herramientas necesarias
Paquete de trabajo: PLANTEAMIENTO, DECISIÓN Y MANTENIMIENTO **Duración:** 3 horas.
Descripción: Instalación de las herramientas necesarias para llevar a cabo el proyecto.
Entradas: Tecnología a utilizar decidida. **Salidas:** Herramientas necesarias instaladas.

Tabla 5 - Tarea: Revisión planificación temporal

1.5. Revisión planificación temporal
Paquete de trabajo: PLANTEAMIENTO, DECISIÓN Y MANTENIMIENTO **Duración:** 2 horas.
Descripción: Revisión de la situación actual del proyecto con respecto a la planificación temporal establecida, y en caso de haber un desfase, replantear la planificación para así disminuir el impacto global sobre el proyecto.
Entradas: Ninguna. **Salidas:** Planificación temporal revisada y modificada en caso de que fuera necesario.

Captura de requisitos

A continuación, se presentan las subtareas del paquete 2:

Tabla 6 - Tarea: Casos de Uso

2.1. Casos de Uso
Paquete de trabajo: CAPTURA DE REQUISITOS **Duración:** 3 horas.
Descripción: Realización del diagrama de casos de uso de la tarea correspondiente.
Entradas: Especificación de la tarea a realizar. **Salidas:** Diagrama de casos de uso de la tarea correspondiente.

Tabla 7 - Tarea: Casos de Uso Extendidos

2.2. Casos de Uso Extendidos
Paquete de trabajo: CAPTURA DE REQUISITOS **Duración:** 30 horas.
Descripción: Realización del diagrama de casos de uso extendidos de la tarea correspondiente.
Entradas: Especificación y diagrama de casos de uso de la tarea a realizar. **Salidas:** Diagrama de casos de uso extendidos de la tarea correspondiente.

Tabla 8 - Tarea: Modelo de dominio

2.3. Modelo de dominio
Paquete de trabajo: CAPTURA DE REQUISITOS **Duración:** 10 horas.
Descripción: Realización del modelo de dominio de cada desarrollo (aplicación web y móvil).
Entradas: Especificaciones del proyecto. **Salidas:** Modelo de dominio.

Tabla 9 - Tarea: Base de Datos

2.4. Base de Datos
Paquete de trabajo: CAPTURA DE REQUISITOS **Duración:** 5 horas.
Descripción: Conversión del modelo de dominio de cada desarrollo a una base de datos relacional.
Entradas: Modelo de dominio. **Salidas:** Base de datos creada.

Desarrollo Web

A continuación, se presentan las subtareas del paquete 3. Cabe destacar que todas las subtareas de este paquete incluyen el trabajo de análisis, diseño, implementación y pruebas, tanto en el *back-end* como en el *front-end* de las funcionalidades en ellas contenidas. Además, en la estimación de la duración de cada una de las subtareas de este paquete, están consideradas tanto la parte *back-end* como la *front-end*.

Tabla 10 - Tarea: Login y Logout

3.1. Login y Logout
Paquete de trabajo: DESARROLLO WEB **Duración:** 10 horas.
Descripción: Desarrollar la funcionalidad con la que el usuario (perteneciente al grupo CRIM-AP) podrá iniciar sesión en el sistema y salir del mismo.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de inicio y cierre de sesión. Funcionalidad probada.

Tabla 11 - Tarea: Crear un punto a estudiar

3.2. Crear un punto a estudiar
Paquete de trabajo: DESARROLLO WEB **Duración:** 25 horas.
Descripción: Desarrollar la funcionalidad con la que el usuario (perteneciente al grupo CRIM-AP) podrá crear un nuevo punto a estudiar, esto es, indicar en el mapa un lugar, ajustar los detalles que vaya a tener dicho punto (nombre, radio, descripción, etc.) y asociarle las preguntas que se quieran a dicho punto. Finalmente, este quedará guardado.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de creación de un punto a estudiar. Funcionalidad probada.

Tabla 12 - Tarea: Mostrar mapa con puntos a estudiar

3.3. Mostrar mapa con puntos a estudiar
Paquete de trabajo: DESARROLLO WEB **Duración:** 20 horas.
Descripción: Desarrollar la funcionalidad que mostrará al usuario (perteneciente al grupo CRIM-AP) un mapa con todos los puntos a estudiar que ya estuvieran definidos.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad que muestra el mapa con todos los puntos a estudiar que hubiera. Funcionalidad probada.

Tabla 13 - Tarea: Gestionar preguntas y cuestionarios

3.4. Gestionar preguntas y cuestionarios
Paquete de trabajo: DESARROLLO WEB **Duración:** 30 horas.
Descripción: Desarrollar la funcionalidad que permitirá al usuario (perteneciente al grupo CRIM-AP) añadir, modificar y eliminar cuestionarios y preguntas.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de gestión de preguntas y cuestionarios. Funcionalidad probada.

Tabla 14 - Tarea: Modificar un punto a estudiar

3.5. Modificar un punto a estudiar
Paquete de trabajo: DESARROLLO WEB **Duración:** 5 horas.
Descripción: Desarrollar la funcionalidad que permitirá al usuario (perteneciente al grupo CRIM-AP) seleccionar un punto a estudiar del mapa, ver los resultados obtenidos hasta ese momento, y modificarlo si quisiera.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de modificación de un punto de estudio. Funcionalidad probada.

Tabla 15 - Tarea: Eliminar un punto a estudiar

3.6. Eliminar un punto a estudiar
Paquete de trabajo: DESARROLLO WEB **Duración:** 5 horas.
Descripción: Desarrollar la funcionalidad que permitirá al usuario (perteneciente al grupo CRIM-AP) seleccionar un punto a estudiar del mapa y eliminarlo.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de eliminación de un punto a estudiar. Funcionalidad probada.

Tabla 16 - Tarea: Descargar datos

3.7. Descargar datos
Paquete de trabajo: DESARROLLO WEB **Duración:** 30 horas.
Descripción: Desarrollar la funcionalidad que permitirá al usuario (perteneciente al grupo CRIM-AP) descargar los resultados de uno, varios o todos los puntos a estudiar.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de descarga de datos de los puntos a estudiar. Funcionalidad probada.

Tabla 17 - Tarea: Gestionar usuarios y permisos

3.8. Gestionar sugerencias de puntos a estudiar
Paquete de trabajo: DESARROLLO WEB **Duración:** 20 horas.
Descripción: Desarrollar la funcionalidad que permitirá al usuario (perteneciente al grupo CRIM-AP) revisar las sugerencias de puntos a estudiar enviadas por los usuarios de la aplicación móvil y podrá aceptar la sugerencia o rechazarla. En caso de aceptarla, dicho punto sugerido quedará almacenado.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de sugerencias de puntos a estudiar. Funcionalidad probada.

Tabla 18 - Tarea: Gestionar usuarios

3.9. Gestionar usuarios
Paquete de trabajo: DESARROLLO WEB **Duración:** 5 horas.
Descripción: Desarrollar la funcionalidad que permitirá al usuario (perteneciente al grupo CRIM-AP) dar de alta a nuevos usuarios y eliminarlos.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de gestión de usuarios. Funcionalidad probada.

Desarrollo App

A continuación, se presentan las subtareas del paquete 4. Cabe destacar que todas las subtareas de este paquete incluyen el trabajo de análisis, diseño, implementación y pruebas, tanto en el *back-end* como en el *front-end* de las funcionalidades en ellas contenidas. Además, en la estimación de la duración de cada una de las subtareas de este paquete están consideradas tanto la parte *back-end* como la *front-end*.

Tabla 19 - Tarea: Pedir datos al usuario y guardarlos

4.1. Pedir datos al usuario y guardarlos
Paquete de trabajo: DESARROLLO APP **Duración:** 5 horas.
Descripción: Desarrollar la funcionalidad que pedirá al usuario los datos que se deseen almacenar de este.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad de petición y almacenamiento de datos del usuario. Funcionalidad probada.

Tabla 20 - Tarea: Mostrar mapa con puntos en estudio activos

4.2. Mostrar mapa con puntos en estudio activos
Paquete de trabajo: DESARROLLO APP **Duración:** 15 horas.
Descripción: Desarrollar la funcionalidad que mostrará al usuario un mapa con todos los puntos en estudio activos y además le indicará cuál es su ubicación actual.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad que muestra la ubicación del usuario en el mapa junto con todos los puntos en estudio activos. Funcionalidad probada.

Tabla 21 - Tarea: Desarrollar pruebas

4.3. Desarrollar pruebas
Paquete de trabajo: DESARROLLO APP **Duración:** 30 horas.
Descripción: Desarrollar las pruebas que el usuario deberá realizar al entrar y salir de un punto en estudio.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Pruebas desarrolladas.

Tabla 22 - Tarea: Completar un punto en estudio

4.4. Completar un punto en estudio
Paquete de trabajo: DESARROLLO APP **Duración:** 120 horas.
Descripción: Desarrollar la funcionalidad que permita a un usuario completar un punto en estudio, esto es, que tras haber hecho las pruebas de entrada al punto y las de salida, los resultados queden almacenados en el sistema. Esta funcionalidad también controlará si los resultados deben ser almacenados o no (ya que en caso de que el usuario abandone el experimento, estos no serán almacenados).
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad que permite al usuario completar un punto en estudio. Funcionalidad probada.

Tabla 23 - Tarea: Notificar punto en estudio cercano

4.5. Notificar punto en estudio cercano
Paquete de trabajo: DESARROLLO APP **Duración:** 40 horas.
Descripción: Desarrollar la funcionalidad que se encargará de notificar al usuario cuando este se encuentre cercano a un punto en estudio.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad que notifica al usuario cuando se encuentre cerca de un punto en estudio. Funcionalidad probada.

Tabla 24 - Tarea: Gestionar puntos en estudio

4.6. Gestionar puntos en estudio
Paquete de trabajo: DESARROLLO APP **Duración:** 10 horas.
Descripción: Desarrollar la funcionalidad que permita al usuario gestionar los puntos en estudio, esto es, el usuario podrá silenciar aquellos puntos en estudio que no le interesen realizar o de los cuales no quiera ser avisado. De la misma manera, podrá revertir estas acciones cuando quiera.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad que permite al usuario gestionar los puntos en estudio. Funcionalidad probada.

Tabla 25 - Tarea: Sugerir un punto a estudiar

4.7. Sugerir un punto a estudiar
Paquete de trabajo: DESARROLLO APP **Duración:** 15 horas.
Descripción: Desarrollar la funcionalidad que permita al usuario sugerir nuevos puntos a estudiar. Para ello, deberá rellenar un formulario desde la aplicación móvil.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad que permite al usuario sugerir nuevos puntos a estudiar. Funcionalidad probada.

Tabla 26 - Tarea: Modificar datos del usuario

4.8. Modificar datos del usuario
Paquete de trabajo: DESARROLLO APP **Duración:** 2 horas.
Descripción: Desarrollar la funcionalidad que permita al usuario modificar sus datos.
Entradas: Caso de uso de la tarea y modelo de dominio. **Salidas:** Funcionalidad que permite al usuario modificar sus datos. Funcionalidad probada.

Documentación

A continuación, se presentan las subtareas del paquete 5:

Tabla 27 - Tarea: Elaboración del DOP

5.1. Elaboración del DOP
Paquete de trabajo: DOCUMENTACIÓN **Duración:** 60 horas.
Descripción: Elaboración del DOP.
Entradas: Descripción y objetivos del proyecto definidos y tecnología a utilizar decidida. **Salidas:** DOP redactado.

Tabla 28 - Tarea: Redacción de la memoria

5.2. Redacción de la memoria
Paquete de trabajo: DOCUMENTACIÓN **Duración:** 120 horas.
Descripción: Elaboración de la memoria del proyecto.
Entradas: Ninguna. **Salidas:** Memoria redactada.

Una vez detallado cada paquete de trabajo y sus correspondientes subtareas, a continuación se presenta una tabla (Tabla 29) a modo de resumen de la estimación en horas:

Tabla 29 - Resumen dedicación estimada en horas

Paquete de trabajo	***Tarea***	***Dedicación estimada (en horas)***	***Dedicación estimada total (en horas)***
PLANTEAMIENTO, DECISIÓN Y MANTENIMIENTO	Reunión con los tutores del TFG	6	**42**
	Definición y aclaración de los objetivos del proyecto	1	
	Análisis y decisión de la tecnología a usar	30	
	Instalación de las herramientas necesarias	3	
	Revisión de la planificación temporal	2	

Paquete de trabajo	***Tarea***	***Dedicación estimada (en horas)***	***Dedicación estimada total (en horas)***
CAPTURA DE REQUISITOS	Casos de uso	3	**48**
	Casos de uso extendidos	30	
	Modelo de dominio	10	
	Base de datos	5	
DESARROLLO WEB	Login y Logout	10	**150**
	Crear un punto a estudiar	25	
	Mostrar mapa con puntos a estudiar	20	
	Gestionar preguntas y cuestionarios	30	
	Modificar un punto a estudiar	5	
	Eliminar un punto a estudiar	5	
	Descargar datos	30	
	Gestionar sugerencias de puntos a estudiar	20	
	Gestionar usuarios	5	
DESARROLLO APP	Pedir datos al usuario y guardarlos	5	**237**
	Mostrar mapa con puntos en estudio activos	15	
	Desarrollo pruebas	30	
	Completar un punto en estudio	120	
	Notificar punto en estudio cercano	40	
	Gestionar puntos en estudio	10	
	Sugerir un punto a estudiar	15	
	Modificar datos del usuario	2	
DOCUMENTACIÓN	Elaboración del DOP	60	**180**
	Redacción de la memoria	120	
		Total	***657***

2.6. Planificación temporal

Una vez definidos y detallados cada paquete de trabajo junto con las subtareas que los componen, se debe especificar cómo se distribuirán durante el tiempo. Para ello, se empleará un diagrama de Gantt como el que se muestra a continuación (Ilustración 5).

Respecto a la carga de trabajo, se han estimado 30 horas semanales (5 horas diarias de lunes a sábado).

Por lo tanto, sabiendo que el trabajo comienza el 29 de noviembre de 2017, éste terminaría, si se sigue el plan previsto, el 4 de junio de 2018.

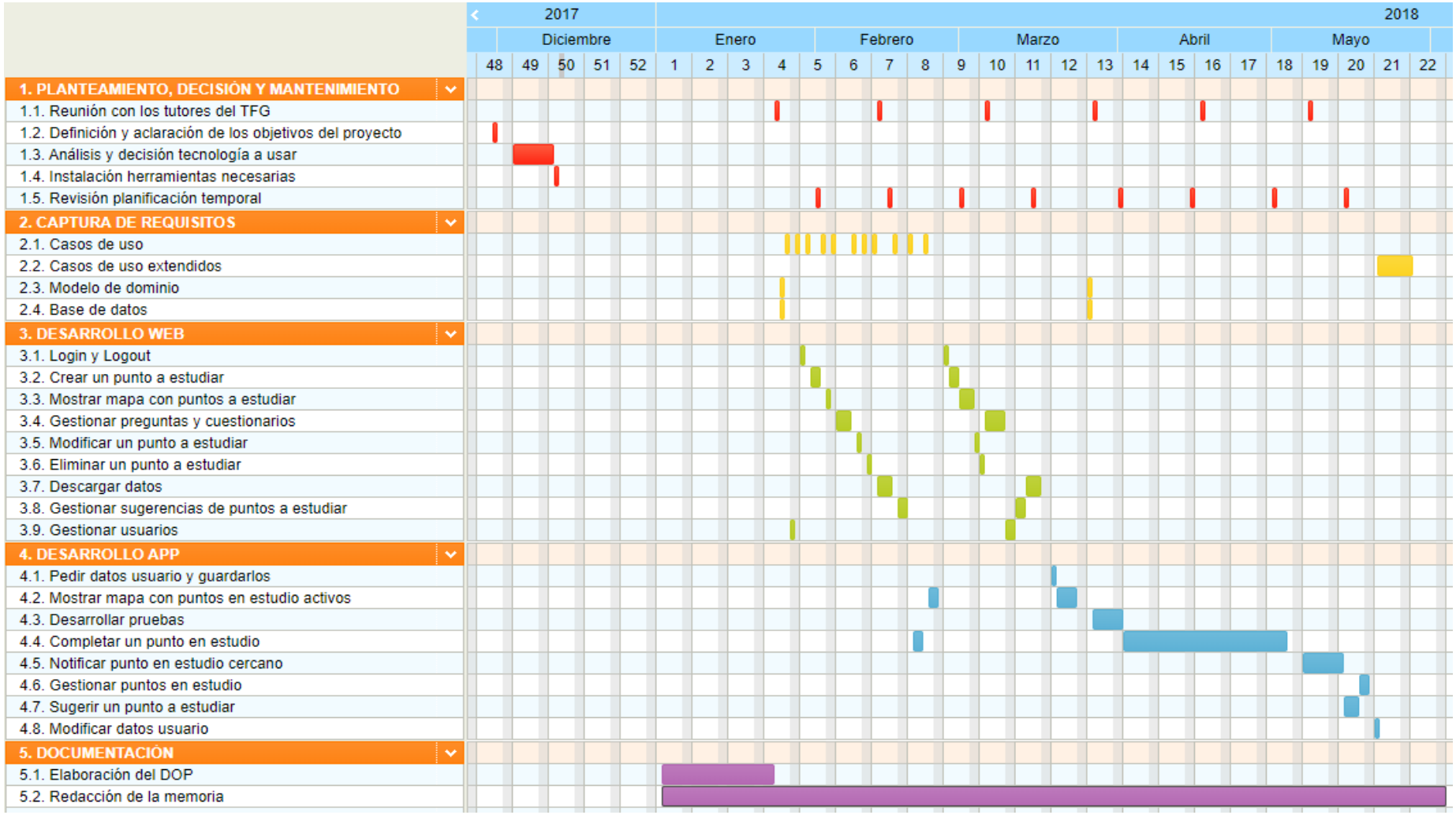


Ilustración 5 - Diagrama Gantt

2.7. Gestión de riesgos

En este apartado, se detectarán y valorarán los riesgos a los que puede estar sometido este proyecto, y, además, se detallará un plan de contingencia para cada uno de ellos. La necesidad y utilidad de generar un plan de gestión de riesgos es poder mitigar al máximo los efectos que estos puedan tener sobre el proyecto, así como prevenir las causas que puedan conducir a ellos, y en el caso de que lleguen a producirse, saber cuáles serán las medidas a tomar.

Antes de proceder a examinar los riesgos del proyecto, cabe destacar que se realizarán copias de seguridad de tanto la parte de documentación (archivos, diagramas, etc.) como de partes directas del desarrollo. Toda la parte de documentación se guardará automáticamente en Dropbox cuando sufra una modificación, y en cuanto a la parte de desarrollo, se harán copias de seguridad cada 4 días, tanto en Dropbox como en un disco duro externo.

A continuación, y para comenzar con la gestión de riesgos, se examinarán cuáles son los riesgos de este proyecto. Todos los riesgos detectados se muestran a continuación en la siguiente tabla (Tabla 30):

Tabla 30 - Riesgos

Riesgos
1. Planificación temporal mal estimada. 2. Actualización de alguna de las herramientas de desarrollo. 3. Actualización de alguna API usada para el desarrollo. 4. Modificación de funcionalidades propias del lenguaje de programación utilizado. 5. Riesgos del sistema. 5.1. Infección malware. 5.2. Rotura/pérdida del equipo. 5.3. Rotura/pérdida del dispositivo móvil. 6. Indisposición del desarrollador. 6.1. Enfermedad. 6.2. Cambio de país de residencia. 6.3. Empleo/prácticas.

Ahora que los riesgos ya han sido identificados, se definirá una nueva tabla (Tabla 31) que recogerá las probabilidades posibles, que posteriormente serán asociadas a cada riesgo.

Tabla 31 - Probabilidades

Probabilidad	***Porcentaje***
Improbable	0 - 20 %
Poco probable	20 - 40 %
Probable	40 - 70 %
Muy Probable	70 - 100 %

Además de la probabilidad de que ocurra un determinado riesgo, otro factor a tener en cuenta es el impacto que causaría en caso de que este se produjera. Habrá riesgos que no supongan demasiado impacto, otros en cambio, podrían retrasar y afectar en el proyecto de forma mucho más considerable. Por ello, y al igual que se ha hecho anteriormente con las probabilidades, se definirá otra tabla (Tabla 32) que indicará de forma más precisa el retraso que supondría los diferentes tipos de impactos.

Tabla 32 - Impactos

Impacto	***Retraso (días)***
Muy leve	< 1
Leve	1
Medio	2 - 3
Grave	4 - 5
Muy grave	> 5

Una vez identificados los posibles riesgos (Tabla 30), definidas las probabilidades (Tabla 31), y los impactos (Tabla 32), es hora de analizarlos individualmente. A continuación, se detallará cada riesgo, con su correspondiente plan de prevención, probabilidad, impacto y plan de contingencia asociado.

En la siguiente tabla (Tabla 33), se detallan los riesgos expuestos anteriormente:

Tabla 33 - Riesgos detallados

Riesgos detallados	
1. Planificación temporal mal estimada	
Descripción	Haber realizado una estimación temporal no realista.
Prevención	Realizar una planificación temporal lo más precisa y realista posible e intentar ajustarse a ella lo máximo posible. Además, revisar frecuentemente la planificación para detectar retrasos lo antes posible.
Plan de contingencia	Reorganizar la planificación temporal para así poder ajustarla a la situación real e intentar así recuperar el desfase de tiempo.
Probabilidad	Muy probable.
Impacto	Medio.
2. Actualización de alguna de las herramientas de desarrollo.	
Descripción	Actualización de alguna de las herramientas de desarrollo que impida seguir con este.
Prevención	Comprobar los detalles de la actualización y asegurarse de que es posible volver a la versión anterior antes de llevarla a cabo para decidir si hacerlo o no.
Plan de contingencia	Volver a la versión anterior si la nueva versión diera algún problema.
Probabilidad	Muy probable.
Impacto	Medio.
3. Actualización de alguna API usada para el desarrollo.	
Descripción	Actualización de alguna de las APIs utilizadas para el desarrollo que impida seguir con este.
Prevención	Comprobar los detalles de la actualización antes de llevarla a cabo para decidir si hacerlo o no.
Plan de contingencia	Volver a la versión anterior de dicha API si fuera posible.
Probabilidad	Poco probable.
Impacto	Alto.

4. Modificación de funcionalidades propias del lenguaje de programación utilizado.	
Descripción	Actualización del lenguaje de programación utilizado para el desarrollo.
Prevención	Estar al corriente de las nuevas funcionalidades del lenguaje utilizado para así emplearlas y evitar la obsolescencia.
Plan de contingencia	Aprendizaje de las nuevas funcionalidades del lenguaje.
Probabilidad	Poco probable.
Impacto	Medio.
5. Riesgos del sistema.	
5.1. Infección malware**.**	
Descripción	Infección de cualquier tipo de malware en los equipos (PC o móvil) que requiera su limpieza o reparación para continuar con el desarrollo.
Prevención	Instalación, mantenimiento y actualización del antivirus gratuito Avast. Además, se realizarán copias de seguridad en un disco duro externo y en la plataforma Dropbox cada 4 días.
Plan de contingencia	Eliminar el malware del equipo (si fuera posible) y recuperar la última copia de seguridad. En el caso de que no fuera posible eliminar el malware, se utilizaría otro equipo, y se recuperarían en él todos los datos de la última copia de seguridad, así como también las herramientas necesarias.
Probabilidad	Improbable.
Impacto	Grave.
5.2. Rotura/pérdida del equipo.	
Descripción	Rotura o pérdida del equipo PC utilizado para el desarrollo.
Prevención	Mantener el equipo protegido de agentes externos dañinos, y siempre guardado y protegido en un sitio seguro. Hacer uso responsable del mismo. Además, se realizarán copias de seguridad en un disco duro externo y en la plataforma Dropbox cada 4 días.
Plan de contingencia	Utilizar otro equipo, y recuperar en ese todos los datos de la última copia de seguridad, así como también las herramientas necesarias.
Probabilidad	Improbable.
Impacto	Grave.

5.3. Rotura/pérdida del dispositivo móvil.	
Descripción	Rotura o pérdida del dispositivo móvil utilizado para el desarrollo.
Prevención	Mantener el equipo protegido de agentes externos dañinos, y siempre guardado y protegido en un sitio seguro. Hacer uso responsable del mismo.
Plan de contingencia	Utilizar otro dispositivo móvil de características similares.
Probabilidad	Improbable.
Impacto	Leve.
6. Indisposición del desarrollador.	
6.1. Enfermedad.	
Descripción	Cualquier enfermedad que pudiera surgirle al desarrollador.
Prevención	Extremar las precauciones para evitar exponerse a enfermedades.
Plan de contingencia	Tomar las medidas necesarias para recuperarse lo antes posible, y reorganizar la planificación temporal para recuperar el tiempo perdido.
Probabilidad	Probable.
Impacto	Leve.
6.2. Cambio de país de residencia.	
Descripción	Cambio de país de residencia del desarrollador.
Prevención	Seguir usando el equipo propio en el extranjero.
Plan de contingencia	Realizar las reuniones y cualquier comunicación con los tutores del proyecto de manera no presencial (online).
Probabilidad	Improbable.
Impacto	Muy grave.
6.3. Empleo/prácticas.	
Descripción	Comienzo de prácticas en la empresa o en cualquier empleo.
Prevención	Trabajar a media jornada de manera que permita tener suficientes horas disponibles para trabajar en el proyecto y así que la planificación no se vea afectada.
Plan de contingencia	Reorganizar el resto de actividades cotidianas.
Probabilidad	Muy probable.
Impacto	Muy leve.

Tras detallar todos los riesgos y asignarle a cada uno de ellos el plan de prevención, el plan de contingencia, la probabilidad y el impacto correspondiente, llega el momento de hacer una valoración individual para determinar la periodicidad con la que deben ser revisados cada uno de los impactos previamente expuestos.

Para ello, se generará una nueva tabla (Tabla 34) en la cual se indica cada uno de los riesgos, con su probabilidad e impacto, y dependiendo de esos dos factores, se le asignará una valoración:

Tabla 34 - Valoración de riesgos

Riesgos	***Probabilidad***	***Impacto***	***Valoración***
1. Desfase en la planificación temporal.	Muy probable	Medio	Media
2. Actualización de alguna de las herramientas de desarrollo.	Muy probable	Medio	Media
3. Actualización de alguna API usada para el desarrollo.	Poco probable	Alto	Leve
4. Modificación de funcionalidades propias del lenguaje de programación utilizado.	Poco probable	Medio	Leve
5. Riesgos del sistema.			
5.1. Infección malware.	Improbable	Grave	Medio
5.2. Rotura/pérdida del equipo.	Improbable	Grave	Leve
5.3. Rotura/pérdida del dispositivo móvil.	Improbable	Leve	Leve
6. Indisposición del desarrollador.			
6.1. Enfermedad.	Probable	Leve	Leve
6.2. Cambio de país de residencia.	Improbable	Muy grave	Leve
6.3. Empleo/prácticas.	Muy probable	Muy leve	Medio

En la siguiente tabla (Tabla 35) se detallan los 3 tipos de valoración que pueden tener cada uno de los impactos y cada cuánto se revisa:

Tabla 35 - Valoraciones

Valoración	***Revisión***
Leve	Cada mes
Media	Cada 15 días
Importante	Cada semana

2.8. Evaluación económica

En esta sección, se detallará y calculará el coste global que conllevaría este proyecto desarrollado en un entorno productivo real. Para llevar a cabo este cálculo, se deberán tener en cuenta lo siguientes datos:

- Según el diagrama Gantt resultante (Ilustración 5), el proyecto tardaría 6 meses en completarse.

- El sueldo al mes es de 1.265 € [11].
- Coste total de las licencias de herramientas de software.
- Coste total del hardware empleado.
- Gastos indirectos.
- Beneficios que generará el proyecto en el futuro.

Según la planificación temporal, se estima que el proyecto durará 6 meses aproximadamente. Teniendo en cuenta el sueldo definido anteriormente, se obtiene un gasto total de **7.590 €**.

Respecto a las licencias de herramientas de software, el gasto es de 231,18 €, teniendo en cuenta que se obtendrán las licencias mensuales de cada herramienta. En la siguiente tabla (Tabla 36) se muestran los detalles de los costes:

Tabla 36 - Coste de licencias de herramientas de software

Coste de horas			
Descripción	***Coste / Mes***	***Meses***	***Coste***
Office 365 Hogar	10 €	6	60 €
WebStorm	12,90 €		77,40 €
Visual Paradigm UML Standard	15,63 €		93,78 €
		Coste total:	***231,18 €***

En cuanto al hardware, lo que se empleará para el proyecto es un ordenador portátil de la marca ASUS valorado en 600 € (aprox.) y un smartphone Android de la marca Google valorado en 959 € [12]. Debemos tener en cuenta de que la vida útil de un ordenador portátil es de 5 años (aprox.) y la de un smartphone es de 4 años (aprox.). En la siguiente tabla (Tabla 37) se muestran los cálculos del gasto:

Tabla 37 - Coste de hardware

Coste de hardware			
Descripción	***Coste***	***Amortización***	***Coste en 6 meses***
Ordenador portátil ASUS X555LD	600 €	5 años	60 €
Smartphone Google Pixel 2 XL	959 €	4 años	120 €
		Coste total:	***180 €***

Para finalizar con los gastos, se debe tener en cuenta también los llamados gastos indirectos, esto es, los gastos de mantenimiento que se producen durante el proceso de elaboración del proyecto (luz, agua, electricidad, etc.). Para este

[11] https://www.boe.es/boe/dias/2017/01/18/pdfs/BOE-A-2017-542.pdf
[12] https://store.google.com/es/product/pixel_2_xl

proyecto, se considerará que dichos gastos suponen el 5 % del coste total del proyecto.

Así, una vez calculados todos los gastos, se obtiene el gasto total del proyecto, que se muestra en la siguiente tabla (Tabla 38):

Tabla 38 - Gastos del proyecto

Gastos del proyecto	
Descripción	***Coste***
Coste de sueldo	7.590 €
Coste de licencias	231,18 €
Coste de hardware	180 €
Gastos indirectos	400 €
Coste total:	***8.401,18 €***

Dado que este proyecto está desarrollado para un bien social, y por tanto no lucrativo, y para dar soporte al grupo de investigación CRIM-AP, no aplica hablar sobre beneficios económicos.

3. ANTECEDENTES

En este punto, lo que se expone es una visión general de la situación actual, esto es, aquellas aplicaciones ya existentes que tengan una idea o funcionalidad similar a la aplicación que se va a realizar; así como también de las posibles alternativas que se podrían aplicar para la realización de este proyecto.

3.1. Descripción de la situación actual

En este caso concreto, como la aplicación que se va a realizar está hecha por y para unos requisitos y unas funciones muy concretas y detalladas, no existe aplicación similar que permita realizar completamente dichas funciones.

Actualmente, las labores que desempeñará *Restorative City*, están siendo realizadas de las siguientes 3 formas:

1. **Pruebas en el laboratorio:** Las personas que formarán parte del experimento, son llevadas al laboratorio. Una vez allí, los participantes inicialmente son fatigados mentalmente realizando una prueba de atención sostenida. Posteriormente, se les muestra una serie de imágenes tanto de espacios restauradores como de no restauradores. Finalmente, los participantes son nuevamente sometidos a la prueba de atención sostenida. De esta manera, logran saber si las imágenes que han visto, mejoran su rendimiento en la prueba final o no, y, por consiguiente, si dichos espacios que aparecen en las imágenes son restauradores o no.
2. **Pruebas en espacios posiblemente restauradores:** Las personas que formarán parte del experimento, son llevadas al punto en estudio. Una vez allí, son sometidos a las pruebas explicadas en el punto 2 del apartado de objetivos (ver apartado 2.1) y todos los datos son recogidos en papel para su futura transcripción a herramientas que faciliten su análisis y la extracción de conclusiones sobre el poder restaurador del punto visitado.
3. **Encuestas a la ciudadanía:**
    a. Bien puede ser que el equipo de investigación acuda a un lugar específico (el cual quieran saber si es restaurador o no) y realicen la encuesta allí, o también puede ser que este pregunte a la ciudadanía sobre lugares concretos.
    b. El equipo de investigación lanza encuestas a la ciudadanía vía online o por correo electrónico.

3.2. Estudio de las posibles alternativas

Hoy en día, el abanico de posibilidades del que se dispone para realizar un desarrollo es muy amplio y variado. Es por ello que en este apartado se pretende presentar un pequeño estudio de las posibles alternativas existentes actualmente tanto para el desarrollo web como para el móvil, y razonar finalmente el motivo de cada elección.

3.2.1. Servicio de mapas y localización

Restorative City hará uso de mapas, tanto en la parte web como en la móvil. A continuación, se presentan algunas opciones que podrían cumplir con los requerimientos:

- **MapBox:** Es un proveedor de mapas online que hace uso de software Open Source. Cada vez es más frecuente su uso, sobre todo en aplicaciones móviles. Entre sus características principales se podrían destacar que es Open Source, que tiene una amplia libertad para la personalización de los mapas y además dispone de varios SDKs para diferentes plataformas, lo cual permite que sea utilizado en distintos entornos.
- **OpenStreetMap**: Es un mapa mundial gratuito, una iniciativa abierta impulsada por voluntarios para crear en colaboración un mapa del mundo, y exponer los datos del mapa bajo una licencia libre y abierta. Dispone de una API a la cual se le pueden realizar peticiones para obtener la información.
- **Google Maps:** Es uno de los proveedores de mapas más conocidos y utilizado tanto por los desarrolladores como por los usuarios. El proveedor de mapas de Google dispone de diferentes APIs con las que da soporte a las diferentes plataformas. Además, dispone de una amplia compatibilidad con los diferentes lenguajes de programación y plataformas.

Tras analizar las diferentes opciones a usar para implementar el servicio de mapas y localización, finalmente, *Restorative City* hará uso de las APIs de Google Maps, tanto para el desarrollo móvil como web. Se ha elegido este proveedor ya que es uno de los más conocidos, y posee una gran cantidad de ejemplos disponibles, lo cual es una ventaja añadida a la hora de realizar el desarrollo. Además, el diseño que este proveedor ofrece se ha considerado el más adecuado para este proyecto, ya que al ser uno de los más populares y extendidos, ofrece al usuario final un aspecto de familiaridad y comodidad.

3.2.2. Aplicación web

Para el desarrollo de una aplicación web, se disponen de las tecnologías comunes para el lado del cliente (*front-end*). Dichas tecnologías son aquellas que permiten crear interfaces de usuario y establecer comunicación con el servidor

basadas en HTML, CSS y JavaScript, en este caso, donde el navegador actúa como intérprete.

Existen dos arquitecturas a la hora de desarrollar un cliente web: cliente estático y cliente dinámico. Sin embargo, para este proyecto en concreto, la utilización de un cliente estático no sería viable, puesto que estos, generalmente, son utilizados para mostrar información que no vaya a sufrir grandes variaciones a lo largo del tiempo.

Por lo tanto, el tipo de cliente que se implementará para *Restorative City* será un cliente dinámico, puesto que se adapta mejor a las necesidades que este proyecto requiere y debe cumplir. Dicha aplicación web será una aplicación SPA, para aprovechar todas las ventajas que esta ofrece como son la fluidez, rapidez y mejor experiencia para el usuario final. Además, hará uso de una API REST como servidor como este tipo de aplicaciones suelen demandar.

Tras haber decidido realizar una aplicación web SPA, se analizarán las distintas tecnologías que facilitan la creación de dicho tipo de aplicaciones web.

- **React:** Es una librería en JavaScript de código abierto desarrollada y mantenida principalmente por Facebook. La única función de React es encargarse de la interfaz de usuario dinámica de la página. Para ello, proporciona una API sencilla y estable orientada a la creación de pequeños componentes. Un punto importante de React es que puede ser completamente independiente del resto de tecnologías que se utilicen en la aplicación web, ya que es una librería y no un framework.

- **Vue:** Es un framework en JavaScript de código abierto. Está basado en otro framework desarrollado por Google: AngularJS. Este framework principalmente está mantenido por la comunidad de desarrolladores. Al igual que React, Vue se centra en la interfaz de usuario dinámica, aunque también dispone de las herramientas necesarias para la construcción de aplicaciones web SPA y para la resolución de problemas como el enrutado, renderizado, etc.

- **Angular:** Es un framework de código abierto desarrollado en TypeScript, creado y mantenido por Google. Es una evolución de su antecesor: AngularJS; aun así, es un framework completamente distinto y no es retrocompatible. El objetivo principal de Angular es el desarrollo de SPAs, por lo que tiene un ecosistema completo de librerías para su construcción que incluye, entre muchas utilidades más, el sistema de enrutado y herramientas para la terminal que facilitan las tareas típicas del desarrollo. Este framework está orientado a componentes.

- **Ember:** Es un framework de código abierto en JavaScript. Este framework posee unas características muy similares a las de Angular. Sin embargo, para la creación de las plantillas y

componentes, dispone de un motor propio que utiliza el lenguaje HTMLBars.

Después de analizar las distintas herramientas que se podrían utilizar para desarrollar la aplicación web SPA, este proyecto se desarrollará utilizando el framework Angular, ya que es uno de los más conocidos y, en este caso en particular, se conocen ligeros aspectos previos de su antecesor, AngularJS, que pese a ser completamente distintos, comparten algunas ideas técnicas.

3.2.3. Aplicación móvil

A la hora de desarrollar una aplicación móvil, se debe tener en cuenta a quién va dirigida. Además, también se debe escoger bien en qué sistema operativo se podrá sacar mayor provecho a la aplicación. Por ello, a continuación, se expondrán las alternativas valoradas a la hora de realizar el desarrollo móvil de este proyecto.

- **Android:** Este sistema operativo es, hasta la fecha, el sistema operativo más extendido por todo el mundo y que se encuentra alojado en una gran cantidad de dispositivos de distintas marcas, tipos y tamaños. Es un sistema operativo personalizable, lo cual le permite abarcar una gran cuota de mercado, y gracias a esta, existe más posibilidad de que Android siga creciendo aún más.
- **Apache Cordova:** Es un framework para el desarrollo de aplicaciones móviles, utilizando herramientas web genéricas como JavaScript, HTML y CSS. Como resultado, se obtienen aplicaciones híbridas, esto es, aplicaciones que comparten una misma interfaz sin importar el sistema operativo en el que se esté ejecutando, pero que internamente, poseen código nativo de cada sistema operativo.
- **Xamarin:** Es un framework similar al mencionado anteriormente, Apache Cordova, que permite el desarrollo de aplicaciones móviles multiplataforma. En este caso, los lenguajes utilizados son .NET y C++.

Para este proyecto, la aplicación móvil se realizará para el sistema operativo Android, puesto que es uno de los más extendidos hasta la fecha. Además, por lo general, una aplicación nativa resulta más ligera y rápida que una híbrida. También es cierto que, si bien las aplicaciones híbridas permiten compartir una única interfaz, lo cual ahorra tiempo en ese aspecto, pueden surgir limitaciones posteriores que afecten o limiten las funcionalidades del proyecto, ya que el código no es totalmente compatible ni compartido.

3.2.4. API REST

Para el desarrollo de una API REST existen también varias opciones que serán expuestas a continuación:

- **Spring:** Es un framework para el lenguaje de programación Java. Este framework permite desarrollar aplicaciones de manera más eficaz y rápida, ya que automatiza y ahorra al desarrollador algunas tareas básicas de la programación. Spring dispone de muchas dependencias que le aportan multitud de utilidades. Dichas dependencias son fácilmente instaladas a través de Maven. Además, este framework sigue creciendo en la actualidad.
- **Express:** Es un framework para JavaScript y que se aloja dentro del entorno de ejecución NodeJS. La creación de una aplicación servidor con este framework resulta sencilla y no posee extensas dependencias.

En el desarrollo de la API REST a modo de servidor, en este proyecto, se hará uso del framework Express. Su sencillez en cuanto a la creación y puesta en marcha de un servidor favorece al desarrollador. Además, NodeJS es más rápido y eficaz ante situaciones de lectura y escritura de datos, que, al fin y al cabo, es lo que el servidor estará realizando en muchas ocasiones.

"When it comes to real world scenarios seen in typical web applications, Node truly is faster and more scalable than Java. For web application development Node's performance is hard to beat. Because web applications spend most of their time doing IO, and require high concurrency, Node is the clear winner." (Jenson, 2017).

Además, gracias a que Express se aloja en un entorno NodeJS, permite la instalación de otras herramientas o paquetes de trabajo a través del gestor de paquetes NPM *(Node Package Manager)* que NodeJS posee.

Para finalizar, y tras haber realizado un pequeño estudio de las posibles alternativas planteadas a la hora de realizar este proyecto, se presenta a continuación una ilustración (Ilustración 6) a modo de resumen de las decisiones tomadas.

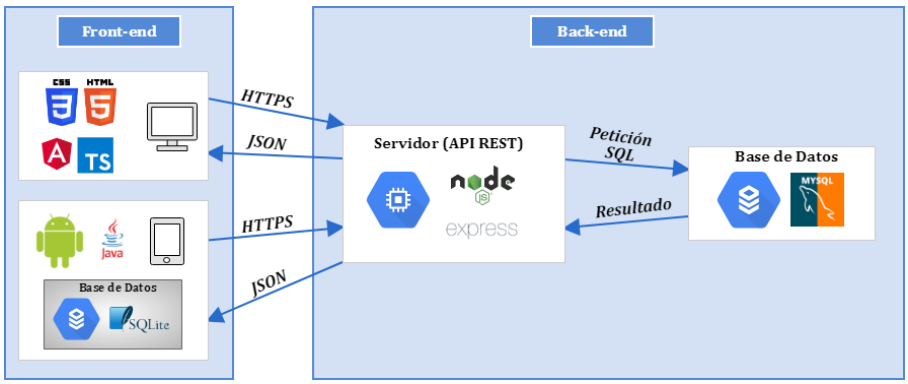


Ilustración 6 - Representación tecnológica del proyecto

4. CAPTURA DE REQUISITOS

En este capítulo se presentará la captura de requisitos de *Restorative City*. La captura de requisitos es el primer paso y uno de los más importantes para poder llevar a cabo de una manera coherente cualquier proyecto de desarrollo.

En el presente capítulo, se pretende dejar especificado cuales son los requisitos que debe cumplir este proyecto, así como detallar las posibles funcionalidades que podrían desarrollarse de manera opcional. Además, se detallará qué usuarios serán los que harán uso de las aplicaciones resultantes, en definitiva, hacia los que va dirigido este proyecto.

4.1. Requisitos funcionales

En este apartado, se expondrán los requisitos funcionales de ambas aplicaciones. Estos requerimientos definen las funciones que los sistemas deben cumplir al finalizar el desarrollo.

En cuanto a la aplicación web, cabe destacar que solo harán uso de esta las personas investigadoras del grupo CRIM-AP. Los requerimientos son los siguientes:

- Permitir gestionar espacios potencialmente restauradores. Los investigadores podrán añadir, eliminar, modificar y activar o desactivar los puntos potencialmente restauradores que consideren.
- Permitir gestionar diferentes cuestionarios que se les hará a los usuarios de la aplicación móvil para obtener unos resultados concretos. Los investigadores podrán añadir, eliminar, modificar y activar o desactivar dichos cuestionarios. Cabe destacar que estas acciones estarán sujetas a unas restricciones concretas para evitar la pérdida de información.
- Permitir gestionar las preguntas que formarán parte del cuestionario que se encuentre activo en cada momento. Los investigadores podrán añadir, eliminar y modificar cada pregunta individualmente. Cabe destacar que estas acciones estarán sujetas a unas restricciones concretas para evitar la pérdida de información.
- Permitir consultar la información recibida sobre un punto potencialmente restaurador en concreto. Visualizar cada respuesta obtenida. Además, de manera general, visualizar la información obtenida de dicho punto en unos gráficos y en un informe general.
- Permitir descargar en formato *.CSV* la información recibida sobre un punto potencialmente restaurador en concreto.

- Permitir gestionar investigadores. Podrán dar de alta nuevos investigadores y eliminarlos.
- Permitir la modificación del perfil personal. Cada investigador podrá modificar sus propios datos personales como el nombre, apellidos o contraseña de acceso al sistema.

En lo referente a la aplicación móvil, hay que mencionar que será la ciudadanía la que la utilice. Los requerimientos que debe cumplir son los siguientes:

- Permitir el registro en el sistema. La aplicación pedirá unos datos mínimos al usuario para completar el registro. Dichos datos serán totalmente anónimos de manera que no sea posible identificar personalmente al usuario a través de estos.
- Mostrar sobre un mapa los puntos potencialmente restauradores activos en ese momento.
- La aplicación, a través de tareas programadas y servicios en segundo plano del sistema operativo, consultará periódicamente la posición geográfica del usuario y comprobará si este se encuentra cercano a un punto posiblemente restaurador activo. De ser el caso, la aplicación lanzará una notificación al usuario preguntándole si desea participar en el estudio de dicho punto.
- En el caso de detectar que el usuario está abandonando el punto en estudio, la aplicación lanzará una notificación indicándole que está abandonando el punto y si desea completar las pruebas finales.
- La aplicación deberá ser capaz de discriminar aquellas pruebas o evaluaciones que no hayan quedado totalmente completadas, por ejemplo, en el caso de que un usuario abandone el punto sin completar las pruebas finales.

4.2. Requisitos funcionales secundarios

En lo referente a la página web, a continuación se detallan los requisitos funcionales no obligatorios que podrían completarse:

- Disponer de un sistema de roles de manera que no todos los investigadores puedan realizar las mismas acciones ni tengan acceso a la misma información.
- Permitir la gestión de puntos a estudiar sugeridos por los usuarios de la aplicación móvil. Los investigadores que posean permisos para ello, podrán consultar las sugerencias de los usuarios y tomar las decisiones que estimen oportunas. Por cada sugerencia, podrán consultar dónde se encuentra dicho punto, la descripción que el

usuario haya proporcionado y la fecha en la que se hizo dicha sugerencia.

- Proporcionar una serie de filtros en la sección donde los investigadores pueden ver la información recibida hasta el momento acerca de un punto en estudio en concreto y descargarla si desean. Haciendo uso de los filtros, los investigadores podrán limitar las respuestas que visualizan y los datos que descargan.

En cuanto a la aplicación móvil, los requisitos funcionales no obligatorios correspondientes son los siguientes:

- Permitir al usuario sugerir nuevos puntos posiblemente restauradores. A través de la aplicación deberá seleccionar en un mapa el punto que desea sugerir e indicar un motivo por el que lo sugiere.
- Permitir gestionar el aviso de cada punto. El usuario podrá silenciar y volver a activar el aviso sobre los puntos que desee, de manera que aquellos puntos que el usuario haya silenciado no le serán notificados como propuesta a participar.

4.3. Requisitos no funcionales

Los requisitos no funcionales definen requisitos que se deben cumplir siendo independientes de las funcionalidades propias del desarrollo que se va a realizar. Los requisitos no funcionales que este proyecto presenta son los siguientes:

- Ambos sistemas deben ser estables y accesibles.
- Ambos desarrollos deben mostrar alta compatibilidad, esto es, la aplicación debe funcionar en varios navegadores, y la aplicación móvil por su parte, debe funcionar tanto en diferentes versiones del sistema operativo Android como en distintos tamaños y resoluciones de dispositivos Android.

4.4. Jerarquía de actores

A continuación, se expondrán los actores que participan en cada de uno de los desarrollos y la jerarquía que forman entre sí.

En el desarrollo web, los actores implicados son los que se pueden ver en la siguiente ilustración (Ilustración 7):

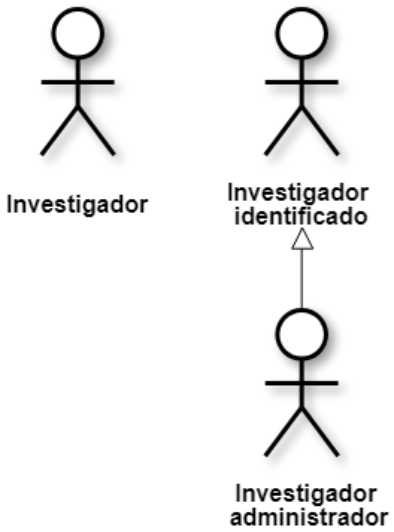


Ilustración 7 - Jerarquía de actores del desarrollo web

- **Investigador:** Representa a todo investigador del grupo CRIM-AP que llega a la aplicación y no se identifica.
- **Investigador identificado:** Representa a todo investigador que se haya identificado satisfactoriamente en el sistema web.
- **Investigador administrador:** Representa a todo investigador identificado que posea además permisos de administrador.

Respecto al desarrollo móvil, los actores implicados se muestran representados en la siguiente ilustración (Ilustración 8):

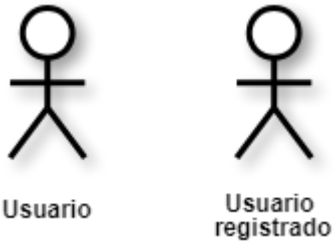


Ilustración 8 - Jerarquía de actores del desarrollo móvil

- **Usuario:** Representa a toda persona que haya instalado la aplicación móvil.
- **Usuario registrado:** Representa a toda persona que haya completado satisfactoriamente el registro en la aplicación móvil.

4.5. Casos de uso

Los casos de uso se emplean como una representación gráfica de las funcionalidades del sistema. A continuación, se presentarán los diferentes diagramas de casos de uso correspondientes a cada desarrollo. Además, cabe destacar, que los casos de uso extendidos se encuentran en el Anexo I (Anexo I: Casos de uso extendidos). Además, en esta memoria también se incluyen los diagramas de secuencia correspondientes en el Anexo II (Anexo II: Diagramas de secuencia).

4.5.1. Aplicación web

En la siguiente ilustración (Ilustración 9), se pueden observar los casos de uso correspondientes al desarrollo web:

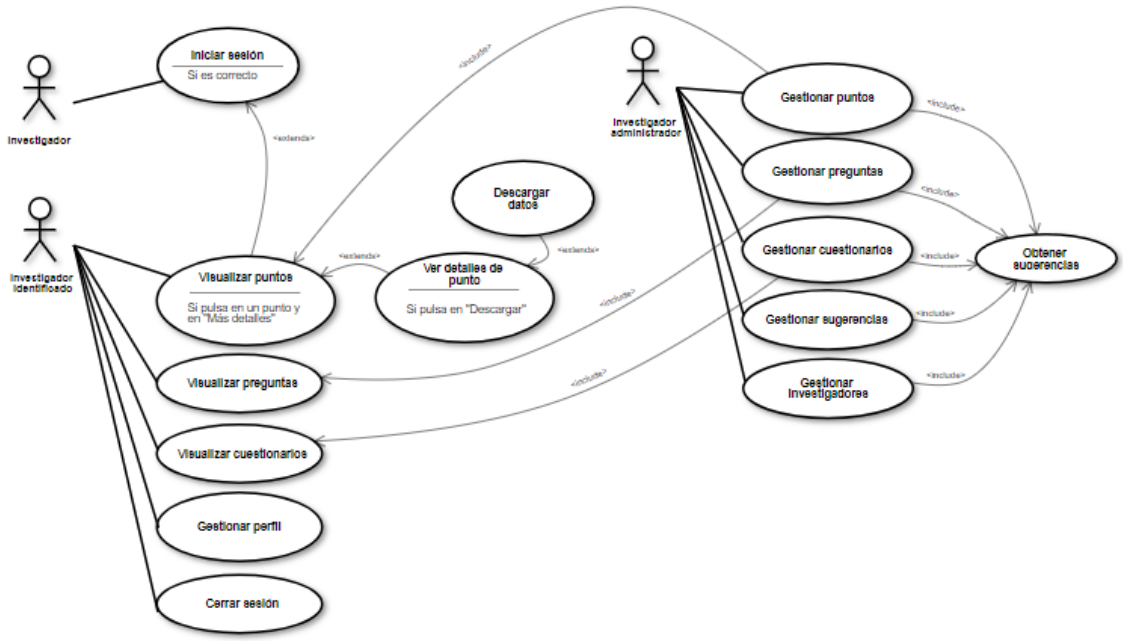


Ilustración 9 - Casos de uso del desarrollo web

A continuación, se describe cada uno de los casos de uso representados en las ilustraciones anteriores. En cuanto al actor *"Investigador"*, el caso de uso que presenta es el siguiente:

- **Iniciar sesión:** Permite al usuario identificarse en el sistema.

Sobre el actor *"Investigador identificado"*, los casos de uso de los que dispone son los siguientes:

- **Visualizar puntos:** Obtiene todos los puntos posiblemente restauradores (PPR) almacenados en la base de datos y los muestra en el mapa. Además, geolocaliza la posición del investigador identificado y lo indica también en el mapa. Permite, además, ver

más detalles de cada punto si el investigador identificado lo desease, en cuyo caso, este tendría disponible un subcaso de uso que se detalla a continuación.

- **Ver detalles de punto:** Muestra los resultados obtenidos hasta la fecha del PPR seleccionado. Además, estos resultados son acompañados con unas representaciones gráficas y un resumen estadístico. El investigador identificado podrá, si lo desea, filtrar los resultados. Finalmente, este también tendrá disponible el subcaso de uso "Descargar datos" que se detalla a continuación.
  - **Descargar datos:** Descarga los resultados obtenidos del punto posiblemente restaurador en formato *.CSV*.

- **Visualizar preguntas:** Obtiene todas las preguntas almacenadas en la base de datos y las muestra en una tabla.
- **Visualizar cuestionarios:** Obtiene todos los cuestionarios almacenados en la base de datos y su información asociada para finalmente mostrarlos en una tabla.
- **Gestionar perfil:** Obtiene la información asociada al investigador identificado. Además, este permite la modificación de los datos del perfil del investigador identificado, si desea.
- **Cerrar sesión:** Finaliza la sesión del investigador actualizado y elimina el *token* de este de la base de datos.

Para finalizar, los casos de uso disponibles para el actor *"Investigador administrador"* son los siguientes:

- **Gestionar puntos:** Se visualizan los puntos en estudio y, además, si el investigador administrador lo desea, podrá añadir, editar o eliminar puntos.
- **Gestionar preguntas:** Se visualizan todas las preguntas almacenadas en el sistema y, además, si el investigador administrador lo desea, podrá añadir, editar o eliminar preguntas.
- **Gestionar cuestionarios:** Se visualizan todos los cuestionarios almacenados en el sistema junto con la información perteneciente a cada uno de estos y, además, si el investigador administrador lo desea, podrá añadir, editar, eliminar, activar, desactivar cuestionarios.
- **Gestionar sugerencias:** Se obtienen todas las sugerencias realizadas por los usuarios de la aplicación móvil y, además, si el investigador administrador lo desea, podrá aceptar, rechazar o posponer cada sugerencia. Al aceptar una sugerencia, esta se convertirá en un PPR.

- **Gestionar investigadores:** Se visualizan todos los investigadores almacenados en el sistema y, además, si el investigador administrador lo desea, podrá añadir, cambiar de rol o eliminar investigadores.

Estos cinco casos de uso, harán uso del subcaso de uso *"Obtener sugerencias"*, que se detalla a continuación, porque cada vez que se ejecuta un caso de uso, se comprueba si hay sugerencias o no, y en caso de haberlas, se le indica al usuario a través de un aviso gráfico.

- **Obtener sugerencias:** Obtiene todos los PPR sugeridos por los usuarios de la aplicación.

4.5.2. Aplicación móvil

En la siguiente ilustración (Ilustración 10), se pueden observar los casos de uso correspondientes a la aplicación móvil:

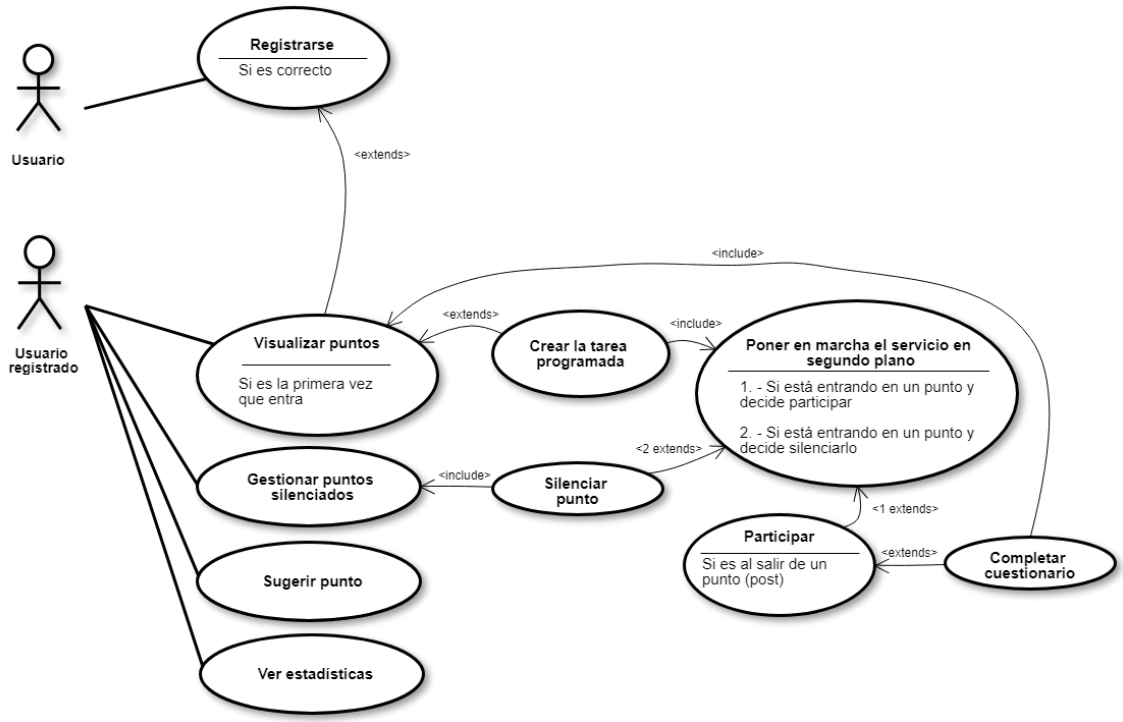


Ilustración 10 - Casos de uso del desarrollo móvil

A continuación, se describe cada uno de los casos de uso representados en la ilustración anterior. Empezando por el actor *"Usuario"*, el caso de uso que presenta es el siguiente:

- **Registrarse:** Permite al usuario registrarse en la aplicación móvil.

Por otro lado, los casos de uso de los que dispone el actor *"Usuario registrado"*, son los siguientes:

- **Visualizar puntos:** Obtiene los PPR que se encuentren activos y los muestra sobre el mapa de la aplicación. Además, este caso de uso dispondrá del subcaso de uso *"Crear la tarea programada"* en caso de que sea la primera vez que el usuario registrado accede a la aplicación. Dicho subcaso de uso se detalla a continuación.
    - **Crear la tarea programada:** Crea y pone en marcha una tarea programada que se ejecutará cada cierto tiempo. Este subcaso de uso dispondrá del caso de uso *"Poner en marcha el servicio en segundo plano"* que se detalla en el siguiente punto.
- **Poner en marcha el servicio en segundo plano:** Crea y pone en marcha el servicio en segundo plano. Este servicio tendrá como labor comprobar periódicamente la ubicación del usuario registrado, y obtener los puntos cercanos a dicha localización. El servicio se mantendrá activo mientras que existan puntos posiblemente restauradores activos cercanos a la posición del usuario registrado. Además, el servicio detectará cuando el usuario registrado esté entrando o saliendo de un punto en estudio, y en ese momento, le avisará a través de una notificación de dicho evento. En ese caso, el usuario registrado dispondrá de dos subcasos de uso (*"Participar"* y *"Silenciar punto"*) que se detallarán a continuación.
    - **Participar:** Mostrará y guiará al usuario registrado a través de las pruebas necesarias que debe completar para la recogida de datos y medidas de niveles necesarias. En el caso de que la participación se realice al salir de un punto en estudio, este caso de uso dispondrá del subcaso de uso *"Completar cuestionario"* que se detalla a continuación.
        - **Completar cuestionario:** Mostrará al usuario registrado el cuestionario que debe completar para finalizar la participación y que esta sea almacenada en el sistema. Una vez finalizado este subcaso de uso, se volverá al caso de uso *"Visualizar puntos"* explicado anteriormente.
    - **Silenciar punto:** Silencia para ese usuario registrado en concreto, dicho punto en estudio. Este subcaso de uso, llevará al caso de uso *"Gestionar puntos silenciados"* que se detalla a continuación.
- **Gestionar puntos silenciados:** Muestra los puntos silenciados del usuario registrado. Además, permite a este elegir qué puntos en estudio desea mantener como silenciados y cuáles no.

- **Sugerir punto:** Permite al usuario registrado sugerir un punto geográfico para que sea considerado en estudio por los miembros del grupo CRIM-AP.
- **Ver estadísticas:** Muestra al usuario registrado el historial de puntos posiblemente restauradores en los que haya participado junto con los resultados obtenidos en cada punto.

4.6. Modelo de dominio

El objetivo de un modelo de dominio es comprender y describir las clases más importantes dentro del contexto de un sistema. En este apartado, se presentarán y detallarán los diagramas de modelo de domino que se han generado por cada desarrollo, siguiendo y atendiendo los requerimientos funcionales previamente expuestos.

Para este proyecto, se ha decidido utilizar el inglés como idioma de desarrollo ya que suele ser lo más habitual en los desarrollos profesionales y, además, el inglés es el idioma más estandarizado y utilizado en la programación.

4.6.1. Aplicación web

Tras entender y estudiar los requerimientos que este proyecto demandaba, se ha generado el modelo de dominio que se muestra en la siguiente ilustración (Ilustración 11):

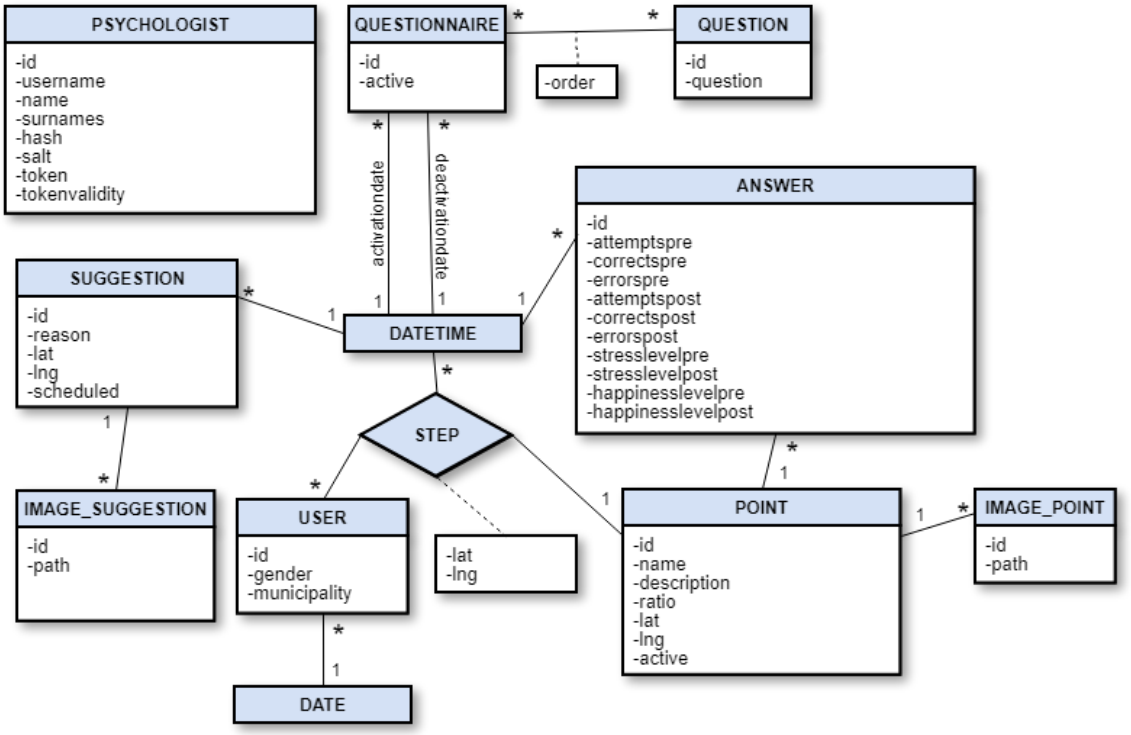


Ilustración 11 - Modelo de dominio del desarrollo web

Como puede observarse en la ilustración anterior, existen varias entidades y relaciones que se detallan a continuación:

- **Psychologist:** Esta entidad representa todos los investigadores (miembros del grupo CRIM-AP) que hayan sido dados de alta en el sistema.
- **Suggestion:** Esta entidad representa todas las sugerencias que los usuarios hayan realizado desde la aplicación móvil.
- **Image_Suggestion:** Esta entidad representa las imágenes que pertenecen a cada sugerencia. Estas imágenes son añadidas por los usuarios de la aplicación móvil a la hora de sugerir un punto.
- **Questionnaire:** Esta entidad representa a los cuestionarios almacenados en el sistema.
- **Question:** Esta entidad representa las preguntas almacenadas en el sistema.
- **Datetime:** Representa la fecha y la referencia temporal (hora, minutos y segundos).
- **Answer:** Esta entidad representa una participación completa de un usuario en un PPR, esto es, las respuestas y los resultados obtenidos en cada prueba tanto en la entrada del punto (pre) como en la salida del mismo (post).

- **Step:** Representa cada paso que da cada usuario mientras se encuentra participando dentro del área de un punto en estudio.
- **User:** Esta entidad representa a los usuarios registrados a través de la aplicación móvil.
- **Date:** Representa la fecha.
- **Point:** Esta entidad representa los puntos posiblemente restauradores que los investigadores definen desde la web.
- **Image_Point:** Esta entidad representa las imágenes que pertenecen a cada PPR. Estas imágenes son añadidas por los investigadores desde la web.

4.6.2. Aplicación móvil

Atendiendo al desarrollo móvil, el modelo de dominio resultante es el que puede observarse en la siguiente ilustración (Ilustración 12):

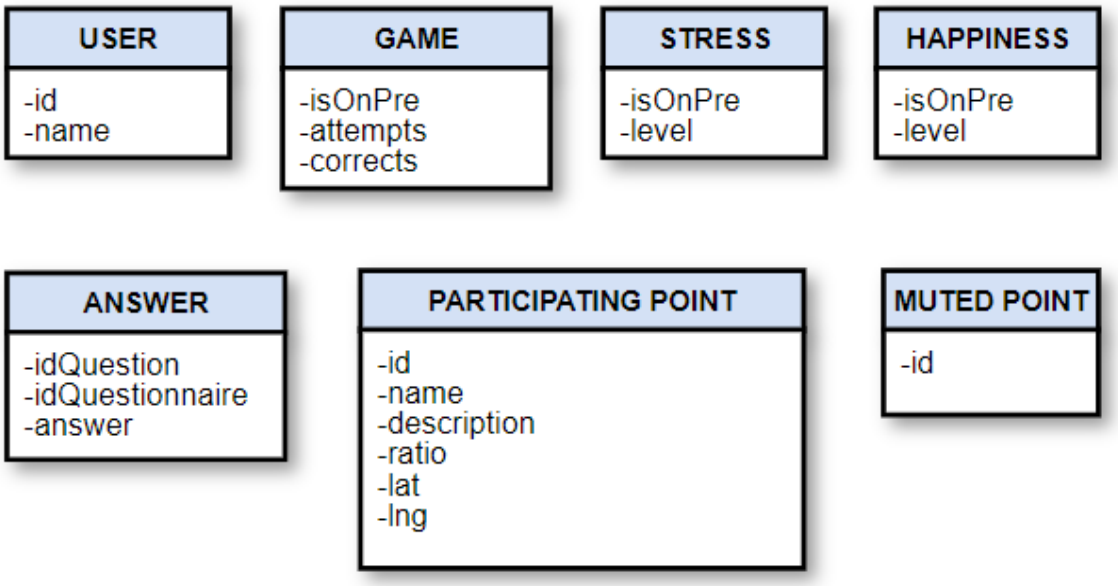


Ilustración 12 - Modelo de dominio del desarrollo móvil

Como puede verse en la ilustración anterior, no hay ninguna relación entre las entidades. Esto se debe a que la base de datos resultante, será una base de datos temporal en gran parte. Concretamente, excepto las entidades **User** y **Muted Point**, el resto de entidades únicamente almacenarán los datos relacionados con un PPR, que al finalizarlo o al abandonar la participación, serán vaciadas. Por lo tanto, siguiendo esta forma de trabajo, no es necesario que estas estén relacionadas. Esto da como resultado una estructura más simple y, además, da ligereza a la aplicación final ya que la base de datos no irá subiendo de tamaño a lo largo del tiempo.

En este modelo de dominio, intervienen varias entidades que se explican a continuación:

- **User:** Representa al usuario registrado en la aplicación móvil.
- **Game:** Representa a los resultados obtenidos en la prueba del juego que se realiza en cada participación, tanto en la entrada (pre) como en la salida (post).
- **Stress:** Esta entidad representa el nivel de estrés indicado por el usuario en el termómetro de estrés en cada participación.
- **Happiness:** Esta entidad representa el nivel de felicidad indicado por el usuario en el termómetro de felicidad en cada participación.
- **Answer:** Representa la respuesta dada por el usuario en cada pregunta del cuestionario.
- **Paricipating Point:** Esta entidad representa el punto posiblemente restaurador en el que el usuario se encuentra participando.
- **Muted Point:** Representa los PPR que el usuario ha silenciado.

5. ANÁLISIS Y DISEÑO

En este capítulo, se pretende detallar las pautas de funcionamiento de los sistemas desarrollados para este proyecto, así como también la estructura seguida para cada uno de estos. Además, se presentarán también los diseños creados para las bases de datos utilizadas y los diagramas de clases.

5.1. Diseño de la API REST

Para el desarrollo de la API REST (*back-end*) se seguirá una estructura multicapa, donde cada una de las capas cumple con unas funciones muy concretas, favoreciendo así a la modularidad y escalabilidad del software. Por lo tanto, las capas mencionadas serán las siguientes: capa de seguridad, capa del núcleo y capa de la base de datos.

Para favorecer el entendimiento de cada una de las capas y partes que tendrá esta API REST, se presenta a continuación (Ilustración 13) un diagrama a modo de resumen:

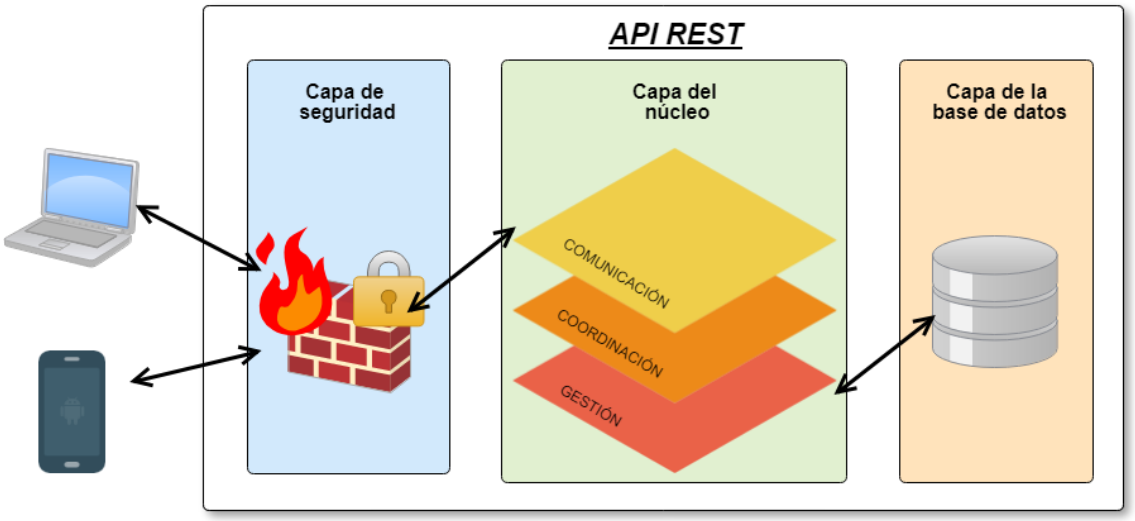


Ilustración 13 - Diseño de la API REST

A continuación, se detallarán cada una de las capas explicando cuál es su función y presentando los elementos que las componen.

5.1.1. Capa de seguridad

Esta capa será la primera por la que las peticiones entrantes deberán pasar. Esta será la encargada de comprobar las credenciales de cada petición y actuar en consecuencia: rechazándolas o permitiendo el acceso, en cada caso.

En esta capa se alojará el *middleware*. El *middleware* será el encargado de actuar como un filtro, esto es, solo permitirá el acceso a aquellas peticiones cuyo *token* sea válido.

Estos *tokens* no son más que, a primera vista, una secuencia de valores alfanuméricos a través del cual el cliente se identifica ante la API REST. Cada usuario dispondrá de un único e irrepetible *token*. La creación del *token* se realiza a través de la mezcla de información relevante del usuario que ha iniciado la sesión, una palabra secreta que el sistema proporciona y solo es conocida por el desarrollador del mismo, y un tiempo de expiración o caducidad. Posteriormente, el *token* generado se cifra para aportarle mayor seguridad. Finalmente, el *token* es entregado al cliente que realizó el inicio de sesión y este deberá usarlo a modo de identificación para realizar cada una de las peticiones futuras a la API REST.

Así, cada vez que reciba una petición, se examinará el *token* recibido (comprobaciones de validez, caducidad, etc.) para determinar si proceder al procesamiento o no de dicha petición. En caso de no recibir *token* alguno, o de no ser válido, el *middleware* rechazará la petición devolviendo al cliente el código de error estándar *401 UNAUTHORIZED* de HTTP. Por el contrario, si el *token* supera las pruebas de verificación, entonces la petición pasará a la siguiente capa: capa del núcleo.

A continuación, se muestra una ilustración (Ilustración 14) a modo de aclaración del comportamiento de esta capa:

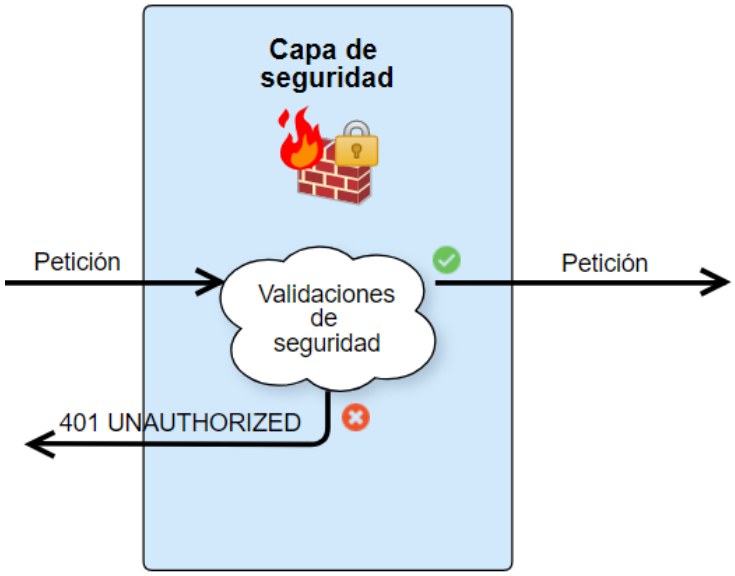


Ilustración 14 - Capa de seguridad de la API REST

5.1.2. Capa del núcleo

La capa núcleo tendrá la labor de alojar la gran parte de la lógica de negocio referente al tratamiento de peticiones y preparación de las respuestas oportunas para estas.

En esta capa, se podrán diferenciar tres módulos independientes entre sí, que trabajan de manera conjunta, comunicándose entre ellos para dar respuesta a las peticiones que se realizan a la API REST. Por tanto, los módulos propuestos serían:

1. **Módulo de comunicación:** Este módulo será el punto de entrada de las peticiones. En este se alojarán las clases que se denominarán "router". Estas clases, serán las encargadas de validar los datos que llegan en las peticiones y que en función de si superan o no la validación correspondiente, la clase actuará en consecuencia, continuando con el tratamiento de la petición o rechazándola, respectivamente.
2. **Módulo de coordinación:** Este módulo será el encargado del tratamiento de datos y recopilación de la información necesaria para proporcionar una respuesta a la petición.
3. **Módulo de gestión:** Este módulo tendrá la labor de realizar todas las comunicaciones con la base de datos y devolver los datos. Será el único módulo que tenga permiso para realizar consultas a la base de datos.

Una vez detallado cada módulo, se presenta a continuación (Ilustración 15) una representación de esta capa, donde se puede observar cómo la capa de seguridad accede al fichero "router" correspondiente del módulo de comunicación, para el tratamiento de la petición. El fichero "router", por su parte, una vez realizada la validación de los datos, si estos son correctos, accede al fichero "functions" correspondiente del módulo de coordinación. Finalmente, el fichero "functions", accede al fichero correspondiente del módulo de gestión, para que este, realice las consultas necesarias a la capa de base de datos.

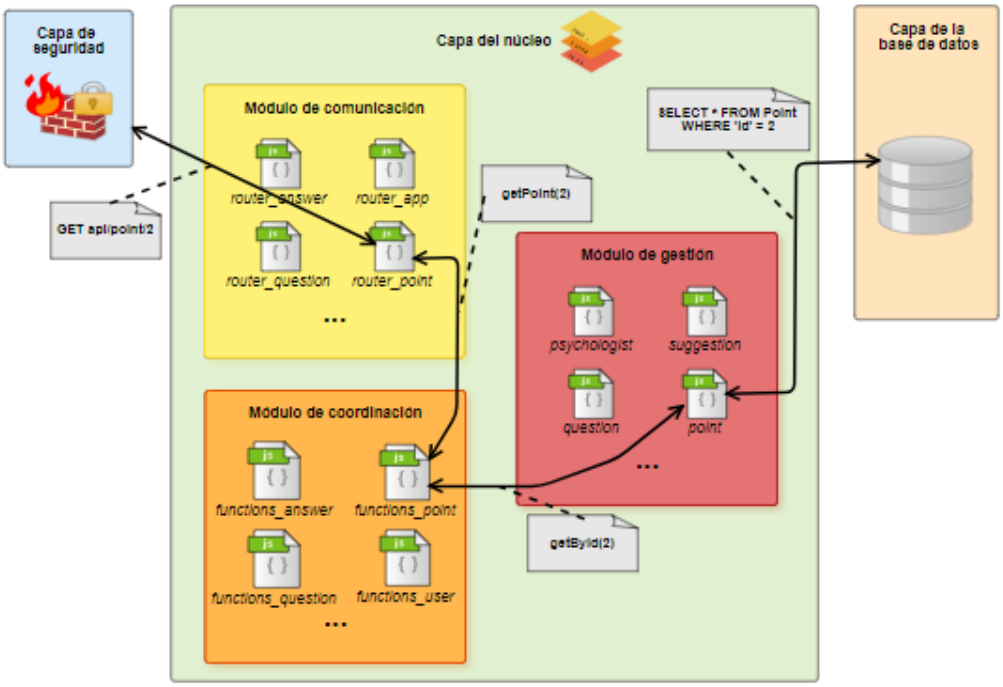


Ilustración 15 - Capa del núcleo de la API REST

5.1.3. Capa de la base de datos

Por último, en esta capa se encuentra la base de datos de la API REST de este proyecto. Para entender mejor el esquema de la base de datos, se muestra en la siguiente ilustración (Ilustración 16) el diagrama relacional correspondiente.

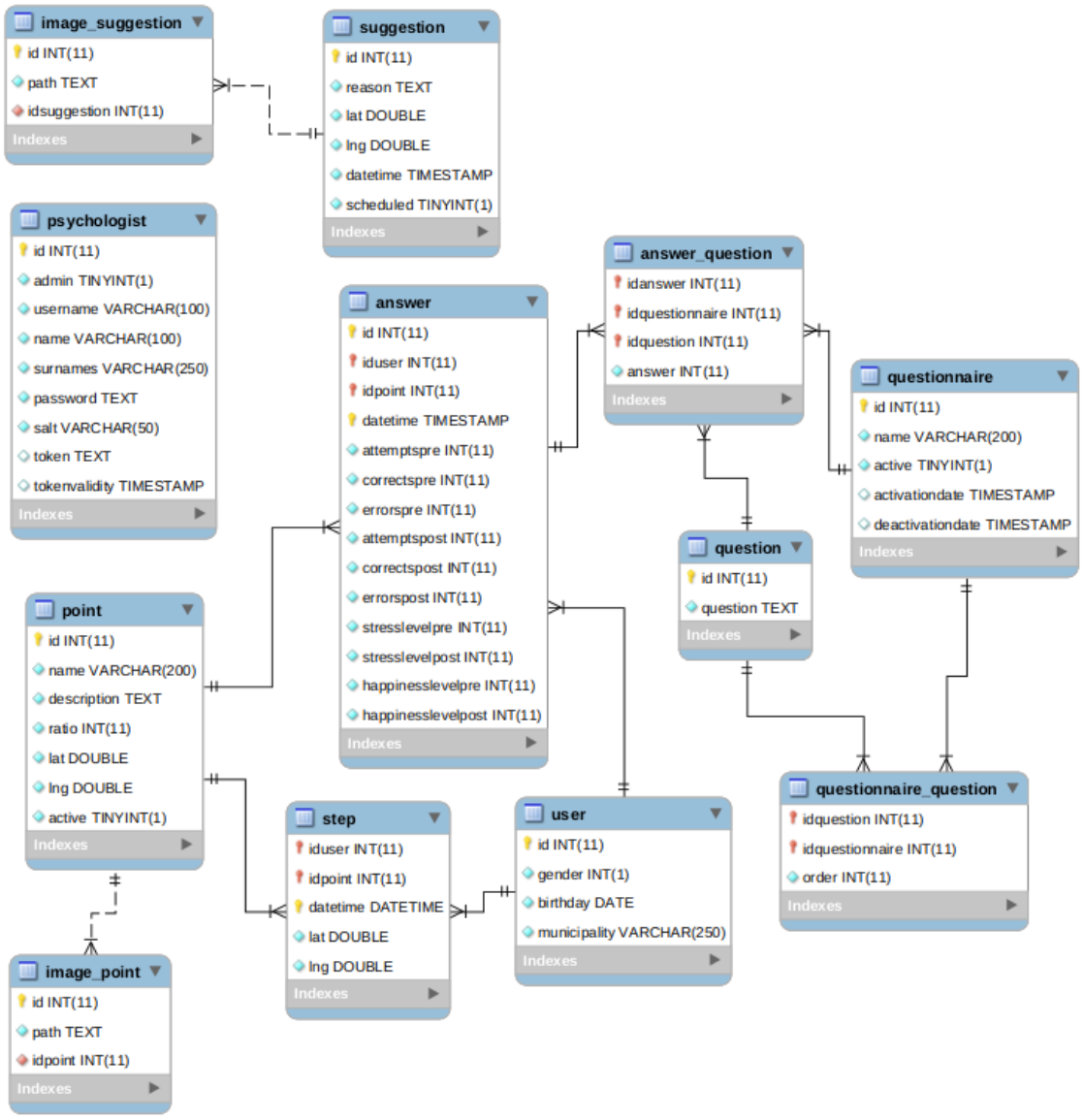


Ilustración 16 - Diagrama relacional de la base de datos de la API REST

A continuación, se detallarán cada una de las tablas resultantes del diagrama anterior.

Para comenzar, la tabla ***suggestion*** almacena los puntos sugeridos por los usuarios de la aplicación móvil. Esta tabla, está relacionada con la tabla ***image_suggestion***, que almacena las imágenes correspondientes a cada sugerencia.

Una relación similar a la anterior descrita se encuentra entre las tablas ***point***, que almacena la información de un PPR, e ***image_point***, que almacena las imágenes correspondientes a cada PPR.

En la tabla ***psychologist*** se almacenan los perfiles de los miembros del grupo CRIM-AP que tengan acceso a la aplicación web. Una tabla similar a esta, es

la tabla ***user***, que almacena la información de los usuarios que se hayan registrado correctamente en la aplicación móvil.

En este proyecto, interesa almacenar la ruta que el usuario realiza mientras se encuentra participando en un punto en estudio. Dicha información se almacena en la tabla ***step***, que se relaciona con las tablas ***point*** y ***user***, anteriormente descritas.

Para el almacenaje de las preguntas que los investigadores administradores definen, se utiliza la tabla ***question***. De la misma manera, en la tabla ***questionnaire*** se almacenan los cuestionarios que los investigadores administradores definen en la web. Ambas tablas están relacionadas. Por un lado, se relacionan dando como resultado la tabla ***questionnaire_question***, que almacena qué preguntas tiene cada cuestionario y el orden de las mismas. Por otro lado, se relacionan junto con la tabla ***answer***, que almacena los resultados obtenidos en una participación completa (es decir, *pre* y *post*), dando como resultado la tabla ***answer_question***, que almacena cada respuesta que se ha dado a cada pregunta en cada participación.

5.2. Diseño de la aplicación web

El diseño de la web se verá influenciado por la forma de trabajar del framework Angular, que ha sido el elegido para llevar a cabo este desarrollo. Antes de proceder a detallar el diseño de la web, se explicará brevemente la forma de trabajar de Angular, para así facilitar el entendimiento del diseño posteriormente.

En Angular se desarrolla en base a componentes. Al ser un framework para desarrollar webs SPA, la aplicación solo tiene un punto de entrada, y por lo tanto, tiene que comenzar por un único componente (generalmente llamado “app.component”). De esta manera, la aplicación web se construirá, en base a un árbol de componentes de "n" niveles, desde un componente principal a sus hijos, nietos, etc.

Estos componentes son como etiquetas HTML nuevas, y pueden ser creados tantos componentes como se quiera para realizar funciones concretas. Los componentes pueden ser cualquier porción o elemento de la vista, desde una sección de navegación a un formulario, o un campo de formulario, hasta un simple botón personalizado.

A continuación, se presenta una representación gráfica (Ilustración 17) de esta composición, donde puede verse como varios componentes (“formulario.component”, “cabecera.component”, “mapa.component” y “botón.component”) son inyectados en el componente “app.component” para formar así una interfaz gráfica completa.

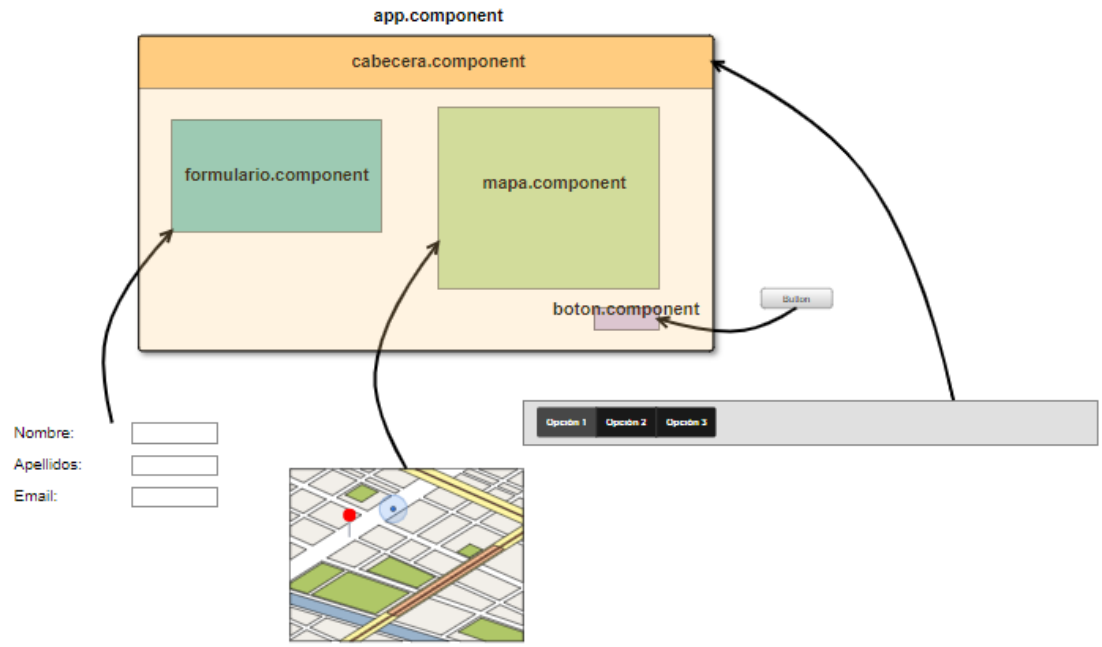


Ilustración 17 - Estructura de componentes de Angular

Cada componente se compone a su vez de tres ficheros fundamentales, a través de los cuales se define su estructura, diseño y funcionalidad. Esos ficheros son:

- **.html:** Se define la vista del componente.
- **.css:** Se define el estilo de la vista.
- **.ts:** Contiene la funcionalidad del componente.

Básicamente eso es un componente, una de las piezas fundamentales de las aplicaciones en Angular, que a su vez aporta diversos beneficios que mejoran sensiblemente la organización de la aplicación, su mantenimiento y reutilización.

Una vez entendido qué son y para qué sirven los componentes en Angular, es razonable pensar que esta aplicación web tendrá una gran cantidad de componentes. La forma que Angular proporciona una solución para ese caso, es mediante el uso de módulos.

Un módulo sirve para encapsular o empaquetar componentes, tantos como se quieran. En este proyecto se generarán varios módulos y cada uno de estos alojarán en su interior aquellos componentes que sean de una temática o lógica similar. De esta manera, se mejorará la escalabilidad, reusabilidad y modularidad de la aplicación.

Cada módulo además, contiene un fichero *.module*, donde se definen qué componentes acoge dicho módulo y donde se realiza además la inyección de

dependencias[13] para el módulo. De esa manera, todos los componentes de este módulo, podrán acceder a dichas dependencias.

A modo de resumen, se presenta a continuación la siguiente ilustración (Ilustración 18), donde puede verse que el módulo "module_A" está compuesto por dos componentes ("component_1" y "component_2"), otro módulo de la aplicación ("module_B"), y dos módulos de librerías externas, ajenas a la aplicación (iconos con símbolo de interrogación).

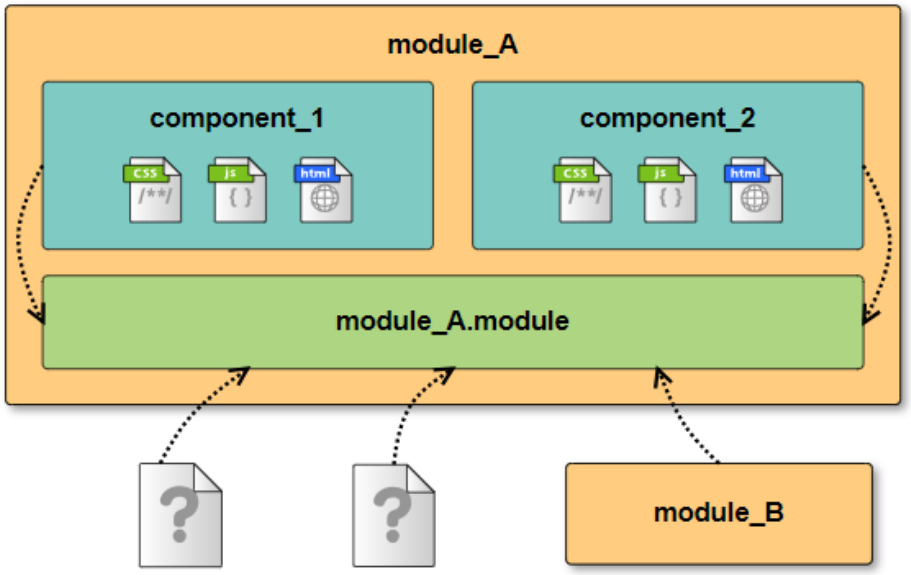


Ilustración 18 - Estructura de módulo de Angular

Por último, falta explicar otro elemento importante en una estructura Angular: los servicios.

Un *service* o servicio, se puede definir como una clase de la que se hace uso desde cualquier componente (siempre que dicho servicio sea previamente inyectado). El propósito de un servicio es contener lógica de negocio o funciones de acceso a datos. Por ejemplo, una clase de utilidades, es un servicio; y una clase que realiza peticiones a una API REST, también.

Una vez detallado los principales elementos de un proyecto Angular y entendido cómo se relacionan entre estos, se procederá a detallar el diseño que se seguirá para este proyecto en concreto.

Para comenzar, se dispondrá del componente principal, es decir, el componente padre donde posteriormente el resto de componentes se irán introduciendo en este. Este componente se denominará "app.component".

[13] La inyección de dependencias no es más que realizar la importación necesaria de librerías o módulos externos que se vayan a utilizar.

Además, se han creado dos carpetas: “modules” y “services”. La carpeta “modules” contiene los módulos creados para la aplicación web, que se detallan a continuación.

Por cada vista o página de la web se creará un módulo. Adicionalmente, se creará un módulo llamado “shared” que contendrá aquellos componentes que sean necesarios reutilizar en varias vistas, es decir, en varios módulos. En este último módulo se incluirán, por ejemplo, componentes como la cabecera del menú de la web, ventanas de avisos, mensajes de alertas, etc.

En la carpeta “services”, en cambio, se crearán los servicios. Como se ha mencionado anteriormente, los servicios pueden ser clases de utilidades simplemente o clases que realizan peticiones a la API REST. Por ese motivo, se dividirán los servicios en estos dos grupos. De esa manera, dentro de la carpeta “services” se crearán dos nuevas carpetas: “generalServices”, que contendrá las clases de utilidades y clases de lógica de negocio; y “webServices” que contendrá aquellas clases que realizarán peticiones a la API REST de *Restorative City*.

Por último, se creará también un *router*, que será el encargado de cambiar las vistas cada vez que se produzca un cambio en la URL.

A continuación, puede verse el esquema general que se ha descrito anteriormente (Ilustración 19):

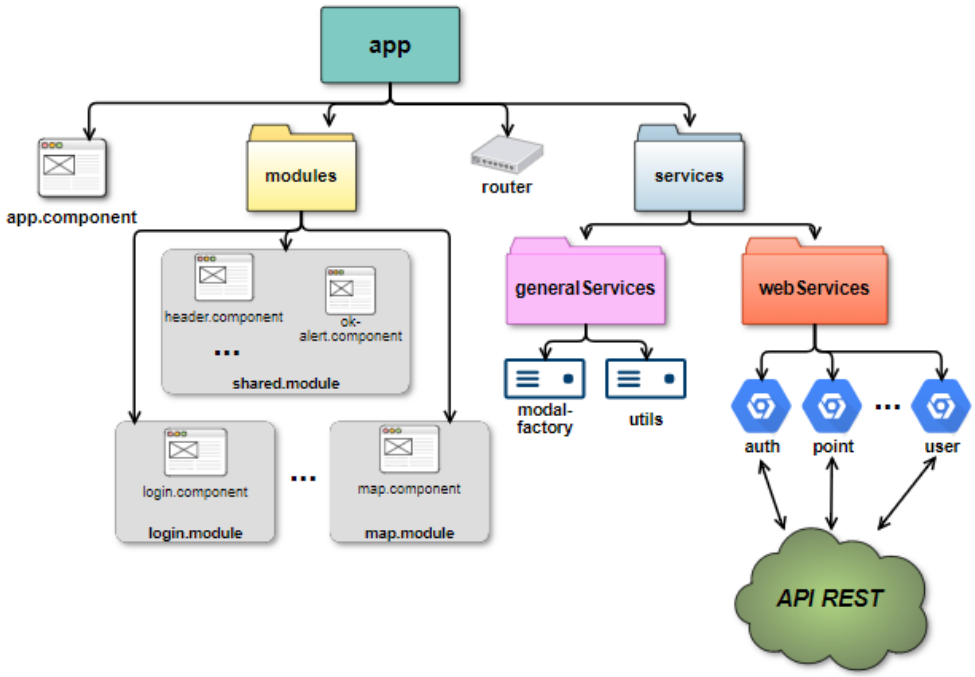


Ilustración 19 - Esquema aplicación web

5.3. Diseño de la aplicación móvil

Para realizar el diseño de la aplicación móvil, primero es importante explicar algunos de los elementos más característicos o fundamentales del sistema operativo Android.

Uno de los elementos principales de Android es la *activity* o actividad. Una *activity* es una clase java en la que se define un componente de la aplicación. En la mayoría de las ocasiones, las actividades presentan una interfaz gráfica con la que los usuarios pueden interactuar para realizar una acción, como por ejemplo, enviar un correo electrónico o ver un mapa. A cada actividad se le asigna una interfaz gráfica en la que se definen los componentes gráficos que la componen (botones, cuadros de texto, etc.). Una aplicación, generalmente, consiste en múltiples actividades vinculadas entre sí de manera que cuando el usuario va cambiando de interfaz gráfica, en realidad está navegando entre actividades.

Por lo tanto, una actividad se compone de dos elementos: vista y controlador. La lógica de negocio, esto es, el controlador, reside en la propia actividad. La vista, en cambio, se define en uno o varios ficheros XML. A las vistas, comúnmente se les llama “layout”.

Otro elemento importante a destacar antes de proceder a la especificación del diseño llevado a cabo para este proyecto, son los *services* o servicios. Un *service* es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario. Otro componente de la aplicación puede iniciar un servicio en segundo plano y este continuará ejecutándose, aunque el usuario cambie a otra aplicación.

Por último, también es importante tener en cuenta el *Manifest* o fichero de manifiesto. Todas las aplicaciones deben tener un archivo “AndroidManifest.xml” (con ese nombre exacto) en el directorio raíz. El archivo de manifiesto proporciona información esencial al sistema operativo Android como el nombre de la aplicación, su icono o la actividad de inicio de la misma.

Una vez expuestos los elementos básicos de Android, se detalla a continuación el diseño decidido para la aplicación móvil de este proyecto. La aplicación se dividirá en tres bloques: manifests, java y res; como es habitual en las aplicaciones Android.

En el primero de ellos, se aloja el fichero de manifiesto que se ha mencionado anteriormente.

El segundo, podría considerarse el núcleo de la aplicación, ya que contiene toda la lógica de esta. Esta será nuevamente dividida en cinco paquetes, separando así las actividades, los adaptadores (que son las clases que gestionan las listas que se muestran en la aplicación), los objetos (clases simples de Java), los diálogos y los servicios.

Por último, en el tercer bloque, se encontrarán todos los recursos de la aplicación, tales como imágenes, ficheros de traducción (para permitir que la aplicación sea multiidioma), las interfaces gráficas de las actividades, etc.

A continuación, puede verse el esquema general que se ha descrito anteriormente (Ilustración 20):

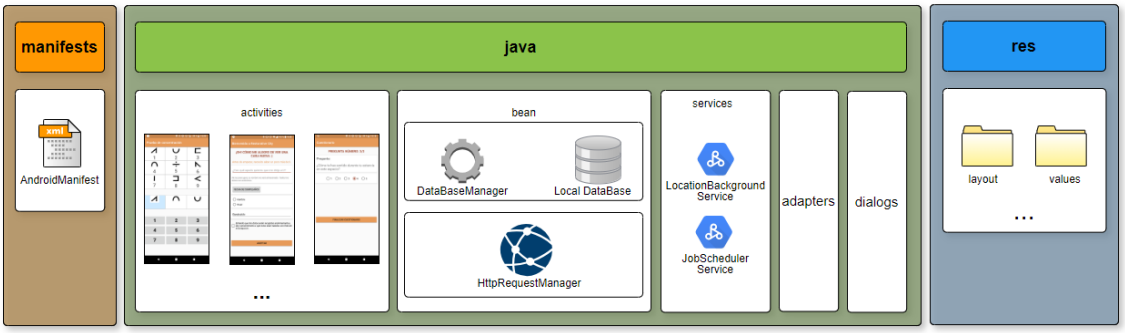


Ilustración 20 - Esquema aplicación móvil

5.3.1. Diagrama de clases

A continuación, se presenta el diagrama de clases de la aplicación móvil (Ilustración 21):

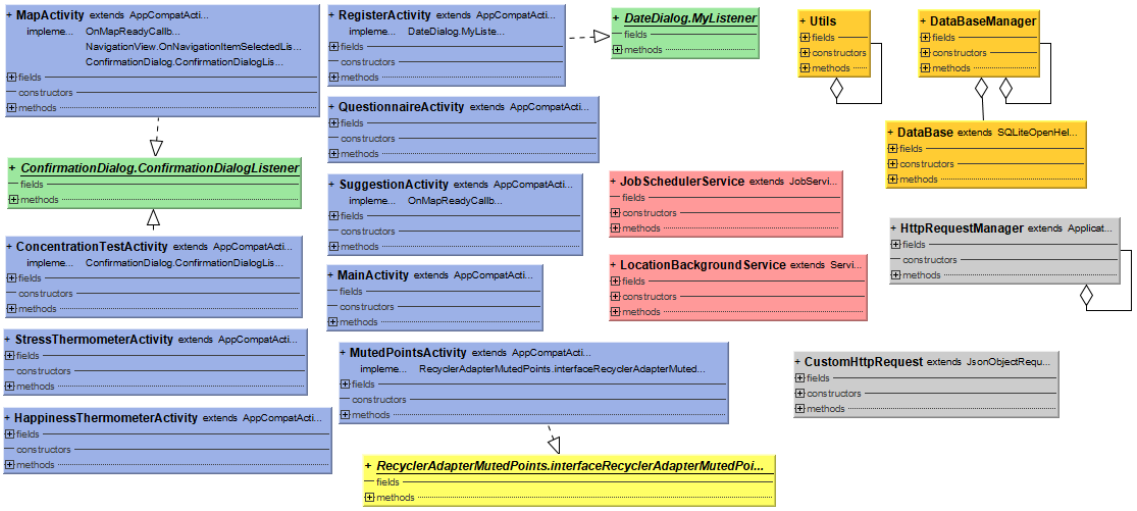


Ilustración 21 - Diagrama de clases de la aplicación móvil

5.3.2. Base de datos

Como se ha comentado anteriormente, la aplicación móvil contará con una base de datos local. Siguiendo el modelo de dominio correspondiente a este desarrollo presentado anteriormente (Ilustración 12), se ha generado el diagrama relacional correspondiente (Ilustración 22):

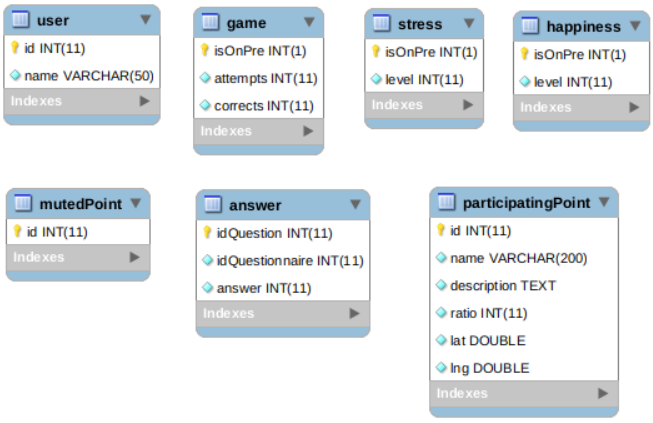


Ilustración 22 - Diagrama relacional base de datos aplicación móvil

6. DESARROLLO

En este capítulo se explicarán en detalle los diferentes desarrollos realizados en este proyecto: API REST, aplicación web y aplicación móvil. Además, por cada desarrollo, se explica su instalación y preparación previa, sus aspectos generales y sus aspectos destacados.

6.1. API REST

Antes de comenzar con el detalle de este desarrollo, se procederá explicar el proceso de instalación y preparación de las herramientas necesarias que se necesitan para llevarlo a cabo.

6.1.1. Instalación y preparación

Como se ha mencionado anteriormente, la API REST se realizará usando NodeJS y el framework Express y el entorno de desarrollo WebStorm.

En este caso, se han utilizado las siguientes versiones:

- **NodeJS:** 8.9.4
- **Express:** 4.16.0
- **WebStorm:** 2017.3.1

Una vez instaladas las herramientas anteriormente mencionadas, se procederá a la creación del esqueleto principal de la aplicación. Este será creado utilizando Express de una manera muy sencilla, tan solo se debe ejecutar el siguiente comando:

```
express restorativeCity
```

El comando anterior, creará una carpeta llamada “restorativeCity” que contendrá el esqueleto de la aplicación (Ilustración 23).

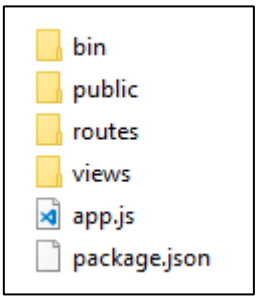


Ilustración 23 - Esqueleto creado por Express

En este caso, se podrá prescindir de la carpeta “routes” puesto que la organización de la aplicación será diferente como se ha explicado en el capítulo anterior (ver apartado 5.1). Además, se ha creado una carpeta nueva llamada

"modules", y dentro de esta, tres nuevas carpetas, una por cada nivel de la capa del núcleo de la API REST, como puede verse a continuación (Ilustración 24):

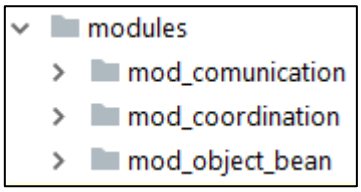


Ilustración 24 - Carpeta "modules"

Además, también se ha creado otra carpeta más llamada "utils". La utilidad de esta carpeta y de cada uno de los ficheros en ella contenidos, se explicará más adelante. El resultado de la creación de esta carpeta es la siguiente (Ilustración 25):

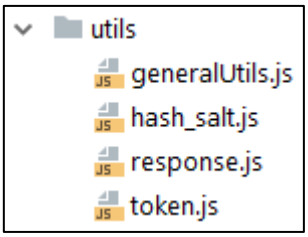


Ilustración 25 - Carpeta "utils"

Por último, se ha creado un fichero llamado "config.js" donde se podrá almacenar la información de configuración que se considere relevante para la API REST. En este caso, este fichero contiene la información acerca de las conexiones a la base de datos, el puerto en el que el servidor estará escuchando, entre otros datos.

6.1.2. Aspectos generales

Antes de comenzar con el desarrollo, se explicarán brevemente los elementos más importantes de los ficheros que se han creado previamente.

Por un lado, se ha creado un fichero llamado "package.json". Este fichero contiene información importante de la aplicación: el nombre, la versión y las dependencias que la aplicación necesita, entre otros datos.

Por otro lado, se encuentra el fichero "app.js". Este fichero es el equivalente a la clase principal de la aplicación. En este se llevan a cabo tareas tales como arranque del servidor y configuración del mismo, definición de las rutas o *endpoints* y asociación con sus clases "router" correspondientes, etc.

Por último, la carpeta "utils" contiene ficheros que sirven de utilidad para toda la aplicación. A continuación, se expone una descripción muy breve de cada uno de los ficheros que contiene esta carpeta:

- **generalUtils.js:** Contiene funciones generales que pueden requerirse en cualquier punto del desarrollo. La utilización de este fichero, favorece la modularidad, y, además, evita la repetición de código.
- **hash_salt.js:** Contiene funciones que aportan seguridad a la API REST.
- **response.js:** Es una factoría de respuestas. Se encarga de preparar y estructurar las respuestas que ofrece la API REST.
- **token.js**: Se encarga de la gestión de los *tokens* (creación, validación, etc.).

Al resto de carpetas no se les dará mayor importancia ya que no serán prácticamente usadas durante el desarrollo. Únicamente detallar que en la carpeta "views" es posible introducir un fichero *.html* en el caso de querer tener una pantalla a modo de portada de la API REST. Este fichero resulta útil, por ejemplo, para saber si la API REST está en marcha y disponible, o no.

Hasta ahora, la estructura de la que se dispone es la siguiente (Ilustración 26):

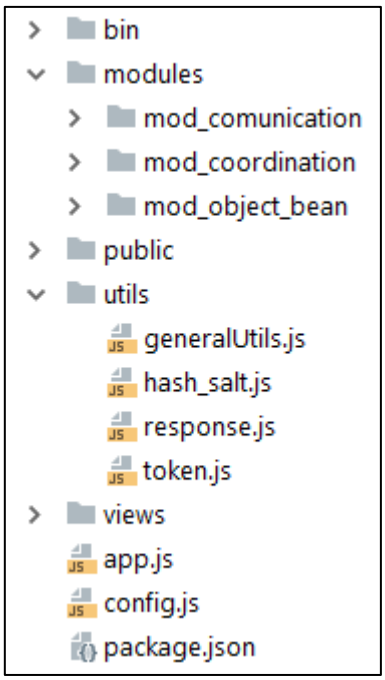


Ilustración 26 - Esqueleto modificado

Para finalizar con la preparación previa al desarrollo, se ejecutará el siguiente comando desde el directorio donde se haya creado el esqueleto de la aplicación:

```
npm install
```

El comando anterior sirve para que se instalen todas las dependencias del proyecto (las dependencias que estaban definidas en el fichero "package.json"). Tras finalizar la instalación, se habrá creado una nueva carpeta llamada "node_modules". En esta se encuentran todas las dependencias y librerías externas instaladas para esta aplicación.

Por último, es necesario instalar el módulo de MySQL ya que la base de datos que se ha usado es de este tipo. Este módulo permitirá realizar las conexiones y las consultas a la base de datos. Para instalarlo, simplemente se debe ejecutar el siguiente comando:

```
npm install mysql
```

En este punto, las configuraciones y pasos básicos quedan listas para comenzar con el desarrollo.

Se comenzará por crear y poner en marcha el servidor, que estará escuchando el puerto 3000. Para ello, en el fichero "app.js" se escribirá lo siguiente (Ilustración 27):

```
// Paquetes necesarios
let express = require('express');
let http = require('http');

let app = express();

//Se inicia el servidor (http)
let server = http.createServer(app);
server.listen(3000);
```

Ilustración 27 - Creación y puesta en marcha del servidor

En el caso de no tener instalado alguna librería (en este caso, la librería "http"), tan solo hará falta buscar cómo se instala dicha librería usando NPM. Para el caso de la librería "http", el comando a ejecutar es:

```
npm install http
```

Para poner en marcha el servidor, tan solo se deberá ejecutar en la terminal el siguiente comando:

```
node app.js
```

A continuación, se explicará cómo crear cada uno de los *endpoints* o rutas para cada fichero "router". Estos ficheros pertenecen al primer módulo dentro de la capa del núcleo (ver apartado 5.1.2).

Para crear un *endpoint*, tan solo se deberá definir en el fichero "app.js" la ruta de dicho *endpoint* y a qué fichero hace referencia. A continuación, se muestra un ejemplo (Ilustración 28):

```
// Módulos de comunicación, definición
let router_auth_login = require('./modules/mod_comunication/router_auth_login');

//Rutas
app.use('/auth/login', router_auth_login);
```

Ilustración 28 - Ejemplo de comienzo de endpoint

De esta manera, todas las peticiones que comiencen por */auth/login* serán redirigidas al fichero "router_auth_login.js" y este será el encargado de gestionarlas.

A continuación, se explicará cómo gestionan los ficheros "router" cada una de las peticiones y cómo se definen cada *endpoint* para que estos sean expuestos en la API REST.

En cada fichero "router" se definen los *endpoint* que este vaya a manejar. Por cada *endpoint*, se define el método de HTTP con el que ha de ser llamado y la URL. Además, se reciben los parámetros "req" y "res". El parámetro "req" permite acceder a los datos de la petición, tales como las cabeceras o los parámetros que vengan agregados en la propia URL. El segundo parámetro en cambio ("res"), sirve para devolver una respuesta al cliente. A continuación, se muestra el ejemplo en concreto de un *endpoint* que ha de ser llamado a través del método *POST* (Ilustración 29):

```
router_auth_login.post('/', function (req, res) {...});
```

Ilustración 29 - Ejemplo de final de endpoint

De igual manera, se han definido el resto de *endpoints*, utilizando los métodos *GET*, *POST*, *PUT* y *DELETE*.

Cada *endpoint* ejecuta una función. Estas funciones, son las encargadas de validar los parámetros introducidos en la petición. En caso de ser estos válidos, delegará el procesamiento de la petición al siguiente módulo, al módulo de coordinación, haciendo uso de los ficheros "functions"; y en caso contrario, rechazará la petición.

Por un lado, se comprueba que se hayan recibido todos los parámetros que se esperaban. Además, estos parámetros deben ser del tipo de datos que se esperaba recibir. Por otro lado, en las peticiones que lo requieran, esto es, en las peticiones securizadas, se accede a la cabecera "Authorization", de donde se extrae el *token* enviado en la petición. Posteriormente, el *token* es sometido a un

proceso de descifrado. Gracias al proceso anterior, del *token* se obtiene el nivel de permisos del que dispone el usuario (rol) que ha realizado la petición. Más adelante se explicará con detalle cómo se generan y tratan los *tokens*.

Suponiendo que la validación de parámetros ha sido satisfactoria, se continúa con el tratamiento de la petición. De igual manera que se ha visto en una ilustración anterior (Ilustración 28), en esta ocasión, se importa el fichero "functions" correspondiente a través de la palabra reservada *require*, para hacer uso de los métodos que ese fichero contiene. Los ficheros "functions" pertenecen al segundo módulo dentro de la capa del núcleo (ver apartado 5.1.2).

Las funciones contenidas en los ficheros "functions", se encargan de la mayor parte de lógica de negocio de la API REST. Estas pueden comunicarse tanto con otros ficheros "functions", como con los ficheros que se encuentran en el último módulo de la capa del núcleo.

En el último módulo de la capa del núcleo de la API REST, se encuentran los ficheros "object". Estos ficheros, son los que interactúan directamente con la base de datos. Las funciones contenidas en estos ficheros, se encargan de ejecutar consultas SQL, y de devolver los resultados o errores obtenidos, en cada caso.

6.1.3. Aspectos destacados

A continuación, se detallan los puntos más destacables de esta API REST, haciendo hincapié en aquellos aspectos más curiosos o en los problemas que hayan surgido durante su desarrollo.

Asincronía

Uno de los principales problemas que se han encontrado a la hora de trabajar con la base de datos, es la asincronía. Este problema reside básicamente, en que cuando se realiza una petición a la base de datos, el código no se queda esperando a la respuesta. Por este motivo, resulta un gran problema cuando se pretende trabajar con datos que dependan de la respuesta de una petición a la base de datos.

Para la resolución del problema de la asincronía, se ha hecho uso de las promesas o *promises* y las funciones *async* y *await*, que se explican a continuación. Es importante detallar que el uso de *async* y *await* solo está disponible a partir de la versión de JavaScript *ECMAScript 6*.

Las promesas son objetos que ejecutan código y pueden devolver algo o no, y en caso de devolverlo, puede ser al momento, más tarde o incluso nunca. Por eso, una promesa se rige por estados. Esos estados son los siguientes:

- **Pendiente *(pending)*:** Estado inicial, pendiente o esperando a ser resuelta.
- **Cumplida *(fulfilled)*:** Estado final, significa que la operación se completó satisfactoriamente.
- **Rechazada *(rejected)*:** Estado final, significa que la operación falló.

Como se puede ver, cuando una promesa es resuelta, solo puede haber dos casos: satisfactorio o erróneo. Una promesa utiliza el método *resolve()* para indicar que ha finalizado correctamente, esto es, que ha alcanzado el estado *fullfilled*; en cambio, para indicar que falló (estado *rejected*), hace uso del método *reject()*. Estos dos métodos, en definitiva, pueden ser interpretados como un *return*.

Para clarificar el funcionamiento de una promesa, se muestra a continuación un esquema simplificado (Ilustración 30):

```
new Promise(function (resolve, reject) {

        // Código que se ejecuta
        // ...

        // Si ha ido bien
        resolve();

        // Si ha ido mal
        reject();

    });
```

Ilustración 30 - Estructura de una promesa

La forma de saber cuándo ha terminado una promesa y de qué manera lo ha hecho, es suscribiéndose a esta. Para ello, se hace uso de las funciones *.then()*, para el caso de finalización satisfactoria; y *.catch()*, para el caso de finalización errónea. La siguiente ilustración representa estas situaciones (Ilustración 31):

```
new Promise(
    // Definicion de la promesa
    // Operaciones de la promesa
    // ...
    // resolve() ó reject()

).then(
    // La promesa ha finalizado correctamente
    // ...

).catch(
    // La promesa ha finalizado erróneamente
    // ...
);
```

Ilustración 31 - Interpretación del resultado de una promesa

Una vez entendido cómo funcionan las promesas, se explicarán las funciones *async* y *await* mencionadas anteriormente.

El uso de la función *async* se utiliza para la declaración de una función asíncrona. Una función asíncrona se caracteriza por no tener la necesidad de ejecutar su código secuencialmente, esto es, es posible mantener la ejecución en espera hasta obtener una respuesta. Para mantener la ejecución en espera en un punto determinado, se hace uso de la función *await*.

A continuación, se muestra un ejemplo (Ilustración 32) de la utilización de ambas funciones:

```
// Función asíncrona
async function getUserId(){

    // ...

    let id = await DataBase.getId();

    // La variable 'id' está informada
    // ...
}
```

Ilustración 32 - Ejemplo async y await

Conexión a la base de datos

Como se ha comentado anteriormente, la base de datos con la que se ha trabajado es MySQL, por ello, se ha hecho uso del módulo *mysql* instalado previamente.

Para favorecer la modularidad y el mantenimiento de la API REST, la configuración de la conexión a la base de datos se ha definido en el fichero "config.js". De esta manera, para futuros cambios en la configuración de conexión, bastaría con modificar este fichero.

Para comenzar, se abre la conexión con la base de datos. En el siguiente ejemplo (Ilustración 33) puede verse cómo, usando el módulo *mysql*, se realiza una petición de conexión a la base de datos, pasándole como parámetro la configuración especificada en el fichero "config.js".

```
let connection = mysql.createConnection(config.dbConfig.galanTEST);
connection.connect(function (error) {
    if (error) {
        console.error('Error al conectar con la base de datos: ' + error.stack);
        return null;
    } else {
        console.log('Conexión con la base de datos establecida (' + connection.threadId + ')');
        return connection;
    }
});
```

Ilustración 33 - Petición de conexión a BD

La responsabilidad de realizar las peticiones de conexión a la base de datos es de los ficheros del módulo de gestión. Esos ficheros, antes de realizar una petición, solicitan la conexión, realizan la consulta y finalmente destruyen la conexión. Ese último paso, es realmente importante, puesto que la base de datos tiene una limitación de conexiones abiertas y si estas no se cerrasen, la base de datos dejaría de dar acceso a nuevas conexiones.

Una vez obtenida la conexión a la base de datos, se realiza la consulta SQL que se requiera. En el siguiente ejemplo (Ilustración 34), se muestra cómo se define la SQL que se quiere ejecutar; un tiempo de respuesta máximo (en este caso 1 minuto) que, si pasado ese tiempo, no se ha obtenido respuesta alguna, se aborta la consulta para dejar libre la base de datos; y finalmente, los parámetros de la consulta SQL. Además, se ve cómo una vez se haya obtenido la respuesta, se destruye la conexión a la base de datos, y dependiendo de si la respuesta ha sido de error o no, se actúa en consecuencia.

```
connection.query({
    sql: 'INSERT INTO answer_question SET ?',
    timeout: GeneralUtils.getDataBaseDefaultTimeout(),
    values: [pAnswerQuestioData]

}, function (error, result) {

    connection.destroy();

    if (!error) {

        resolve(result.insertId);
    } else {
        reject(error);
    }
});
```

Ilustración 34 - Ejemplo consulta a BD

Seguridad

Uno de los aspectos más en boga hoy en día en el área de las aplicaciones, es el tema de la seguridad. Debido a que cada vez los ciberataques son más frecuentes y las técnicas con los que se realizan son más sofisticadas, es importante implementar medidas de seguridad para evitar, por ejemplo, accesos no deseados al sistema. En esta API REST, se han implementado varias medidas de seguridad que se expondrán a continuación.

Para comenzar, se ha desarrollado un sistema de gestión de *tokens*. Para ello, se ha utilizado la librería *JSON Web Tokens*[14]. Esta gestión, se realiza mediante el fichero "token.js".

En un objeto *token* se almacena:

- El identificador único del investigador (miembro del grupo CRIM-AP) que ha iniciado sesión.
- Si ese investigador posee o no, permisos de administrador.
- Una fecha de caducidad para ese *token* en concreto.

Una vez almacenada la información anteriormente detallada en un mismo objeto, este es cifrado a través de una palabra secreta que solo el desarrollador de la API REST conoce. El cifrado se realiza usando la función *sign()* de la librería *JWT* anteriormente mencionada. De manera inversa, cuando se recibe un *token* se hace uso de la función *decode()* de la librería *JWT* para obtener la información que este contiene.

Además, para garantizar la seguridad de los *tokens*, se ha añadido un extra para la validación de estos. Por ello, para que un *token* sea reconocido y pase a ser validado, es imprescindible que este empiece por una palabra secreta que solo el desarrollador conoce. Si este no empieza por dicha palabra, directamente se considerará como un *token* no válido.

Otra medida de seguridad que se ha implementado es el uso de un *middleware*. Como se ha comentado anteriormente, todas las peticiones que requieran estar securizadas, pasan por un filtro previo, antes de llegar al fichero "router" correspondiente. En este filtro (*middleware*), se intercepta la petición que se ha realizado, y se accede a las cabeceras de la misma, concretamente, a la cabecera "Authorization", que es donde se debe alojar el *token*. Es aquí donde se realizan las validaciones y comprobaciones anteriormente mencionadas, y en el caso de resultar correctas las validaciones, el *middleware* permite a la petición seguir su curso; en caso contrario, rechaza la petición. Esta tarea, se realiza en el fichero "middleware.js".

Es importante mencionar también que se han controlado la caducidad de las sesiones. Cuando un miembro del grupo CRIM-AP inicie sesión, se le generará un *token* con una fecha y hora de validez, como se ha comentado anteriormente. Ese tiempo de validez, en este proyecto, es de 30 minutos. Así, cada vez que el investigador realice cualquier operación contra la API REST, dicha validez se incrementará. El encargado del incremento del tiempo de validez del token es el *middleware*. De esa manera, mientras el miembro del grupo CRIM-AP esté realizando acciones en la web, su sesión se mantendrá activa. Por el contrario, si pasados 30 minutos, el investigador no ha realizado ninguna acción, entonces el *token* se caducará y, por lo tanto, la sesión también. En ese caso, se le avisa al

[14] https://jwt.io/

investigador a través de la interfaz gráfica y se le expulsa de la sesión, para que, si lo desea, vuelva a iniciar sesión.

Por último, cabe mencionar que las contraseñas de los usuarios de la web no se guardan directamente en la base de datos. Las contraseñas del inicio de sesión de cada usuario, antes de ser almacenadas en la base de datos, pasan por un proceso de cifrado, que se explica a continuación.

A cada investigador, cuando es dado de alta en el sistema, se le genera una secuencia alfanumérica aleatoriamente. A esta secuencia, se le llama *salt*. La *salt* se almacena en la base de datos junto con los datos de registro del investigador.

Cuando se le asigna una contraseña a un investigador, la *salt* de este y la contraseña son fusionadas para formar una única cadena de caracteres alfanuméricos. Esta cadena es encriptada mediante el algoritmo *AES-256* [15], dando como resultado una nueva cadena de caracteres, llamada *hash*. Finalmente, el *hash* es lo que se almacena en la base de datos. De esta manera, en caso de haber robo de información en la base de datos, no es posible saber cuál es la contraseña de cada miembro investigador puesto que estas no son legibles.

Para realizar el inicio de sesión se sigue la misma técnica. Se obtiene la *salt* del miembro investigador que quiere iniciar la sesión, se fusiona con la contraseña introducida y se encripta. Por último, el *hash* resultante se compara con el *hash* almacenado en la base de datos para ese miembro investigador. Si coinciden, entonces quiere decir que la contraseña introducida es correcta, si no, la contraseña es incorrecta.

Factoría de respuestas

Para la generación de respuestas que ofrece la API REST, se ha usado el fichero “response.js”. Este fichero es una factoría que, dado un código de respuesta y la información que se quiera añadir a la respuesta, genera el formato y el contenido de la respuesta correspondiente a dicho código.

El uso de este fichero, proporciona centralización del código, esto es, que en caso de tener que añadir nuevas respuestas o modificar alguna ya definida, facilitaría muchísimo la labor, puesto que tan solo se tendría que modificar este fichero.

El esquema de la respuesta que se genera es el que se muestra a continuación (Ilustración 35):

```
let templateGenerica = '{ "status" : _VAR0status_, "code" : _VAR1code_ , "msg" : _VAR2msg_ , "body" : _VAR3data_ }';
```

Ilustración 35 - Estructura de respuesta

[15] https://es.wikipedia.org/wiki/Advanced_Encryption_Standard

Un ejemplo concreto de la definición de una respuesta, siguiendo el esquema que se muestra en la ilustración anterior, es el siguiente (Ilustración 36):

```
{status: 200, code: 20010, msg: '"Inicio de sesión correcto."'},
```

Ilustración 36 - Ejemplo definición de respuesta

De esa manera, desde el fichero "router" correspondiente, se llamaría a esta factoría haciendo uso de la función *generateResponse(code,data)*, pasándole el código correspondiente y la información que quiere que la respuesta lleve en el cuerpo o *body*. En la siguiente ilustración (Ilustración 37), se muestra un ejemplo de respuesta, en este caso, de un inicio de sesión correcto.

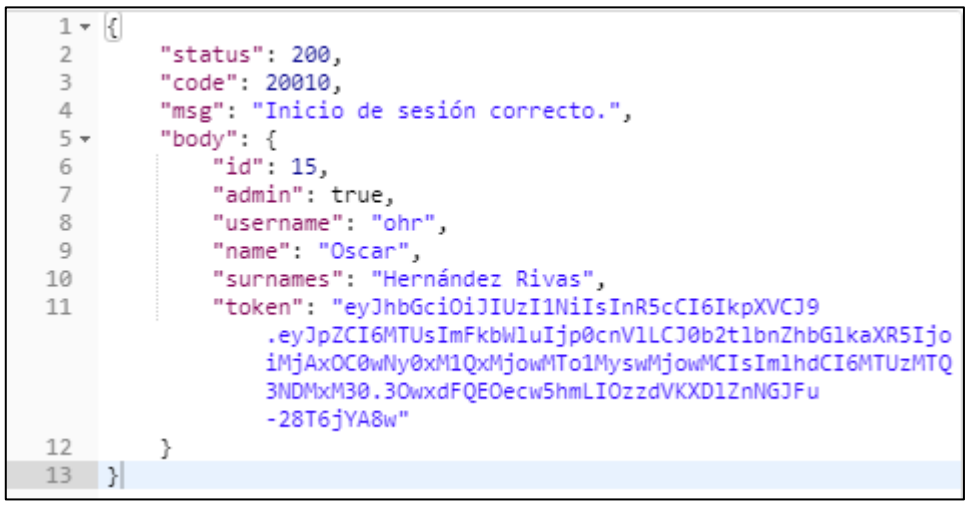


Ilustración 37 - Ejemplo de respuesta

6.2. Aplicación web

Antes de comenzar con el detalle de este desarrollo, se procederá explicar el proceso de instalación y preparación de las herramientas necesarias que se necesitan para llevarlo a cabo.

6.2.1. Instalación y preparación

Como se ha mencionado anteriormente, la aplicación web se realizará utilizando el framework Angular y el entorno de desarrollo Visual Studio Code.

En este caso, se han utilizado las siguientes versiones:

- **Angular:** 5.2.0
- **Angular CLI:** 1.7.0
- **Visual Studio Code:** Se ha ido actualizando durante todo el desarrollo (última versión).

En Angular, el encargado de generar el esqueleto de la aplicación es el *Angular CLI*. A través de este se generarán más adelante los elementos que hagan falta para el desarrollo de la aplicación (componentes, módulos, servicios, etc.).

Para instalar el cliente de Angular *(Angular CLI)*, se deberá ejecutar el siguiente comando:

```
npm install -g @angular/cli
```

Una vez finalizada la instalación, se podrá crear proyectos de Angular. La forma simple y normal de crear un proyecto de Angular, es usando el comando:

```
ng new restorativeCityWeb
```

Sin embargo, para este desarrollo, se hará uso del *router* de Angular. Este *router* será el encargado de redirigir la aplicación entre las diferentes vistas de las que dispone, es decir, contendrá las diferentes rutas de cada interfaz gráfica e irá mostrando unas u otras, dependiendo de la ruta en la que se encuentre. Por ello, la creación del proyecto se hará usando el siguiente comando:

```
ng new restorativeCityWeb --routing
```

Al finalizar, se habrá creado la estructura del proyecto. Como en este desarrollo no se han desarrollado pruebas automáticas utilizando las herramientas que Angular ofrece por defecto, se puede prescindir de todos los ficheros y carpetas que están destinados para ese fin, y que se han creado por defecto.

6.2.2. Aspectos generales

Una de las carpetas más importantes, es la carpeta “app”, que en realidad se trata de un módulo, que se encuentra dentro de la carpeta “src”. En ella, estarán contenidos todos los elementos de la aplicación, es decir, el componente principal (“app.component”), los módulos con sus respectivos componentes en el interior, y los servicios.

Otro de los ficheros más importantes de la aplicación, es el fichero “.angular-cli.json”. En este, se define información relevante del proyecto, como el nombre; la dirección de destino cuando se construya y empaquete la aplicación; cuál es el fichero *.html* principal (“index.html”); los ficheros de estilos *.css* que se desean importar; los ficheros JavaScript de librerías externas que se quieran importar; etc.

El fichero “package.json” también tiene una gran importancia, puesto que es en este donde se indican las dependencias del proyecto. Además, se pueden definir acciones personalizadas para poder utilizarlas posteriormente por consola.

El fichero encargado de arrancar la aplicación y por lo tanto elige qué módulo debe ser inicializado en el arranque, es el “main.ts”. En este caso, el módulo de arranque es el “AppModule”.

Por último, como se ha mencionado anteriormente, el fichero *.html* principal de la aplicación, se llama "index.html". En su interior, entre las etiquetas <*body*>, se hace referencia al componente que debe ser inyectado en el cuerpo del fichero, y por lo tanto, será el padre de todos los componentes sucesores, que serán inyectados en este. Como se ha comentado anteriormente, en una aplicación SPA, solo existe un punto de entrada, y en este fichero se ve reflejada esta estructura, ya que toda la aplicación comienza únicamente a partir del componente que sea inyectado en el "index.html", que en este caso, es el "app.component".

En cuanto a la carpeta "services", se han creado dos carpetas más en su interior: "generalServices" y "webServices". Ambas contendrán servicios, pero la diferencia entre ambas está en el tipo de servicios que contienen en su interior.

La carpeta "generalServices" contiene servicios que no realizan ninguna interacción con la API REST. En esta carpeta, se ha creado, por un lado, un servicio de utilidades, que contiene funciones útiles que pueden ser utilizadas por cualquier otra función; y por otro lado, una factoría de ventanas, que se detallará en el siguiente apartado.

En cambio, la carpeta "webServices" contiene aquellos servicios que se encargan de realizar peticiones a la API REST y devolver la respuesta obtenida.

6.2.3. Aspectos destacados

A continuación, se hará hincapié en los aspectos más importantes o curiosos que se han desarrollado para esta aplicación web.

Peticiones a la API REST

Como se ha comentado anteriormente, las peticiones a la API REST las realizan los servicios contenidos en la carpeta "webServices".

Para realizar estas peticiones, se ha utilizado la clase *HttpClient* que se encuentra en el módulo *@angular/common/http* de Angular.

La forma de trabajar de esta clase es similar a las promesas, esto es, cuando *HttpClient* realiza una petición, este devuelve una subscripción a quien lo llamó. Además, a través de esta clase se pueden realizar peticiones utilizando todos los métodos de HTTP, sin embargo, en este proyecto tan solo se han utilizado los métodos *GET*, *POST*, *PUT* y *DELETE*.

En el caso del inicio de sesión, por ejemplo, desde el componente "login.component.ts" se llama al servicio "auth.service.ts" para que este realice la petición de *login* a la API REST. El servicio devuelve la subscripción al componente y cuando esta sea resuelta, es decir, cuando se obtenga una respuesta (puede ser correcta o incorrecta), se ejecutará el código correspondiente en consecuencia. Para entender mejor esta interacción, se muestra a continuación un ejemplo (Ilustración 38):

```
this.authService.login(userData).subscribe(
  (data) => {
    // La petición ha sido resuelta y ha ido bien
  },
  (error) => {
    // La petición ha sido resuelta y ha ido mal
  }
);
```

Ilustración 38 - Ejemplo de gestión de petición

El servicio por su parte, realiza lo siguiente (Ilustración 39):

```
login(userData) {
  let url = "/auth/login";
  return this.httpClient.post(url, userData);
}
```

Ilustración 39 - Ejemplo de petición POST

De igual manera, se realizan el resto de peticiones en la aplicación. Cabe destacar que, en las ocasiones en las que ha requerido añadir el *token* en la cabecera de la petición, se ha utilizado además la clase *HttpHeaders* que pertenece al mismo módulo que *HttpClient*. La forma de añadir el *token* a la cabecera "Authorization" ha sido la siguiente (Ilustración 40):

```
let _headers = new HttpHeaders();
_headers = _headers.set('Authorization', token);
```

Ilustración 40 - Ejemplo de añadir cabeceras

Sesión

Cuando un miembro investigador realiza correctamente el inicio de la sesión, se debe recoger el *token* que se haya generado para ese usuario en esa sesión concreta y almacenarlo para, posteriormente, enviar ese *token* en cada petición.

Existen varias formas de resolver esta cuestión. En este desarrollo, se ha utilizado el objeto *sessionStorage* para almacenar la información del usuario devuelta por la API REST, donde, entre otros datos, se encuentra el *token*. Se ha utilizado este objeto por su sencillez, porque es donde se recomienda guardar este tipo de datos y porque es donde habitualmente se hace (Alligator, 2018). Este objeto es reconocido nativamente por *HTML5* con lo cual puede ser usado sin necesidad de instalar ni importar ninguna librería adicional.

La forma de almacenar el *JSON* recibido por el *back-end* en el objeto *sessionStorage* es haciendo uso de la función *setItem(key,value)*. De igual manera, para obtener posteriormente el objeto almacenado, se hace uso de la función *getItem(key)*.

Finalmente, cuando la sesión es cerrada, la información almacenada relacionada con dicha sesión se elimina del objeto *sessionStorage* haciendo uso de la función *removeItem(key)*.

Mapa

Cuando un miembro investigador realiza correctamente el inicio de la sesión, automáticamente es redirigido a la interfaz gráfica que le permite ver sobre un mapa los PPR definidos hasta la fecha. Además, se geolocaliza la posición del investigador y también se le indica sobre el mismo mapa. Para facilitar la búsqueda de lugares a los miembros investigadores, desde la interfaz podrán buscar una dirección y el mapa se centrará automáticamente en dicha ubicación.

Para poder llevar a cabo las funciones mencionadas, se ha necesitado hacer uso de dos APIs de Google: *Maps JavaScript API*, para poder mostrar un mapa y sus elementos; y *Geocoding API*, para la funcionalidad de búsqueda de ubicaciones.

Para el uso de ambas APIs, se ha necesitado crear un proyecto en la *Google Cloud Platform*, que es la plataforma a través de la cual Google otorga las claves de uso a cada cliente para el uso de sus APIs.

Una vez obtenidas las credenciales, estas se han colocado en el fichero "index.html" de la aplicación, a través de una etiqueta <*script*>. De esta manera, la aplicación se identifica con las APIs anteriormente mencionadas.

Adicionalmente, se ha instalado los tipos de datos que Google Maps utiliza *(@types/googlemaps)*. Este paso no es estrictamente necesario, pero sí facilita el desarrollo y garantiza el correcto formato de los datos que se utilizan en el mapa.

Para colocar un mapa sobre la interfaz gráfica, se ha definido un <*div*> con el identificador *#gmap* en el fichero *.html* del componente correspondiente. Posteriormente, desde el fichero *.ts* de ese mismo componente, se ha generado el mapa sobre el <*div*>, como se muestra a continuación (Ilustración 41):

```
@ViewChild('gmap') gmapElement: any;

ngOnInit() {

  let mapProp = {
    center: new google.maps.LatLng(0, 0),
    zoom: 1,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };

  this.map = new google.maps.Map(this.gmapElement.nativeElement, mapProp);
}
```

Ilustración 41 - Ejemplo de creación de mapa

El siguiente paso que se hace una vez creado el mapa, es geolocalizar al usuario. Para ello, se utiliza la geolocalización nativa que ofrece *HTML5* en los navegadores. A continuación, se muestra un ejemplo (Ilustración 42):

```
// Compruebo si el navegador permite la geolocalización
if (navigator.geolocation) {
  // El navegador permite la geolocalización

  navigator.geolocation.getCurrentPosition(
    (position) => {
      // Ha obtenido la posición del usuario
    },
    (error) => {
      // Error al obtener la posición
    }
  );

} else {
  // El navegador no permite la geolocalización
}
```

Ilustración 42 - Ejemplo de geolocalización de HTML5

Finalmente, en el caso de haber obtenido la geolocalización del miembro investigador, se realiza una petición a la API REST para obtener los PPR. Una vez estos son obtenidos, se muestran sobre el mapa en objetos *marker* diferenciados en dos colores: verde, si el punto está activo; y rojo, si no está activo. Además, si el miembro investigador posee permisos de administrador, podrá pinchar sobre el mapa y saldrá un nuevo *marker* de color azul.

A continuación, se muestra un ejemplo (Ilustración 43) del mapa creado para este desarrollo donde pueden verse los PPR activos de color verde, y los no activos de color rojo:

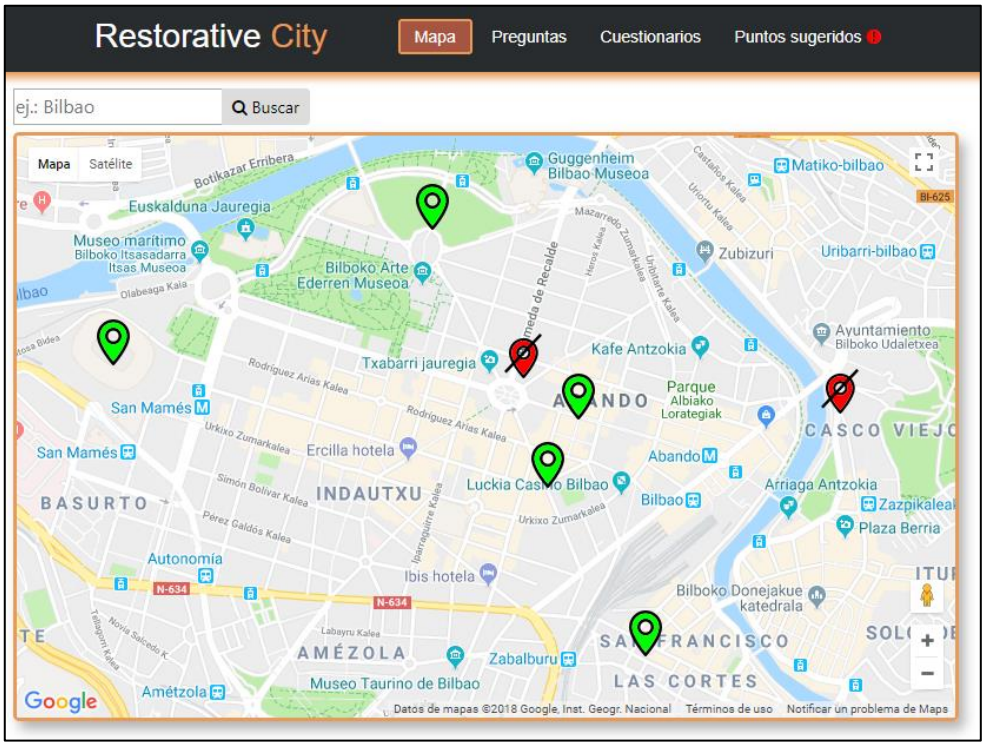


Ilustración 43 - Ejemplo mapa aplicación web

Además, a cada *marker* se le añade una ventana de información que aparece cuando se hace click sobre este. En esa ventana de información se muestra una pequeña estadística sobre la situación actual de dicho punto y también ofrece la posibilidad de acceder a la vista detallada del PPR donde se pueden ver los resultados obtenidos en este.

A continuación, se presenta un ejemplo de la ventana de información de un PPR (Ilustración 44):

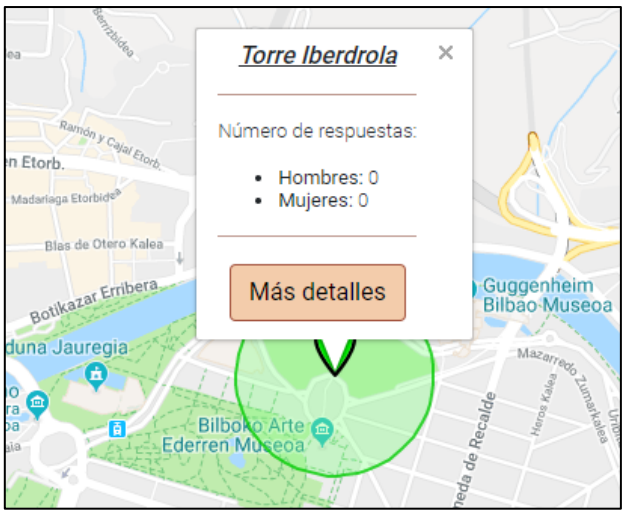


Ilustración 44 - Ejemplo de ventana de información sobre un PPR

Búsqueda de ubicaciones

Para facilitar y agilizar la labor de los miembros investigadores, se ha incorporado un buscador de ubicaciones.

Los miembros investigadores, podrán introducir en este buscador el nombre de la ciudad, región, provincia, etc. que deseen y automáticamente el mapa se centrará en la ubicación indicada.

Para implementar esta funcionalidad, se ha hecho uso de los tipos de datos de Google Maps, utilizados anteriormente para la creación del mapa.

Para comenzar, se ha generado el objeto *geocoder*, que será el encargado de realizar la búsqueda de la ubicación introducida (Ilustración 45):

```
this.geocoder = new google.maps.Geocoder();
```

Ilustración 45 - Creación del geocoder

Una vez creado el *geocoder*, simplemente se le debe introducir la ubicación que se desea buscar y este puede responder de dos formas posibles, con la ubicación encontrada o con un error, dependiendo de si ha sido capaz de encontrar la ubicación solicitada o no.

A continuación, se muestra un ejemplo (Ilustración 46) de la implementación de esta funcionalidad, donde puede verse como a través de la propiedad *address* se le indica al *geocoder* la ubicación que debe buscar.

```
geocodeAddress() {
  this.geocoder.geocode({ 'address': this.address }, (results, status) => {
    if (status === 'OK') {
      this.map.setCenter(results[0].geometry.location);
    } else {
      this.modalFactoryService.openErrorModal("ERROR", "No se ha podido encontrar el lugar especificado.");
    }
  });
};
```

Ilustración 46 - Ejemplo de uso geocoder

En la interfaz gráfica, esta funcionalidad se ha representado de la siguiente manera (Ilustración 47):

Ilustración 47 - Buscador de ubicaciones

Este buscador, agiliza enormemente la labor de los miembros investigadores, puesto que de manera automática pueden moverse por todo el mapa y encontrar una ubicación concreta en tan solo unos segundos.

Routing

Como se ha comentado anteriormente, esta aplicación hace uso del *router* de Angular. Este fichero llamado "app.router.ts", contiene en su interior las asociaciones de cada URL con cada componente que este debe mostrar en cada caso. Este será el encargado de cambiar los componentes visibles en la interfaz gráfica, dependiendo de la URL en la que el usuario se encuentre.

La utilización de este fichero, permite gestionar todas las URL posibles de la aplicación desde un mismo lugar.

Además, es posible configurar algunas direcciones para que estas automáticamente se redirijan a otra URL. Este es el caso de la pantalla de inicio de sesión, ya que el usuario accede a la aplicación desde una URL similar a *http://servidor/proyecto* y es labor del *router* redirigirle al usuario a *http://servidor/proyecto/login*. A continuación, se presenta un ejemplo de la situación planteada (Ilustración 48):

```
const routes: Routes = [
    { path: '', redirectTo: 'login', pathMatch: 'full' },
    { path: 'login', component: LoginComponent }
];
```

Ilustración 48 - Ejemplo de routing (I)

De igual manera, se han controlado aquellas URL que no correspondan con ningún componente de esta aplicación, para que, en el caso de no encontrar coincidencia alguna, se le muestre al miembro investigador una vista indicándole la situación. La manera de implementarlo ha sido la siguiente (Ilustración 49):

```
{ path: '**', component: PageNotFoundComponent }
```

Ilustración 49 - Ejemplo de routing (II)

Angular Grid (ag-Grid)

En esta aplicación se ha hecho uso de varias tablas, por ejemplo, cuando se muestran las preguntas, los cuestionarios o las respuestas de los usuarios de la aplicación en un PPR en concreto. Si bien es cierto que en *HTML* existe la posibilidad de crear tablas a mano, en esta ocasión, se ha optado por utilizar la librería de creación de tablas *ag-Grid*[16].

Esta librería no solo aporta una estética profesional a las tablas, sino que además facilita muchísimo la creación de las mismas y reduce el código necesario para gestionarlas, lo que mejora la fluidez de la interfaz gráfica.

[16] https://www.ag-grid.com/

La declaración de la tabla en el fichero *.html* se realiza a través de la etiqueta <*ag-grid-angular*> como se muestra a continuación (Ilustración 50):

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

Ilustración 50 - Ejemplo etiqueta tabla ag-Grid

Como se puede ver en la ilustración anterior, se han añadido dos propiedades a la etiqueta <*ag-grid-angular*>: *class* y *gridOptions*.

La primera de ellas, define el estilo de la tabla. *ag-Grid* posee varios estilos de tablas que pueden ser utilizados e incluso modificados a través de *CSS*.

La segunda propiedad, hace referencia al objeto que realmente define la tabla, esto es, los eventos de los que dispone, la estructura de la tabla, las cabeceras, los datos que contendrá, etc. A continuación, se muestra un ejemplo simple del objeto *gridOptions* (Ilustración 51):

```
this.gridOptions = <GridOptions>{
  columnDefs: this.columnDefs,
  rowData: this.rowData,
  enableSorting: true,
  enableColResize: true,
  rowSelection: 'single',
  rowStyle: { 'text-align': 'left' },
  domLayout: 'autoHeight',
  pagination: true,
  paginationAutoPageSize: true,
  paginationPageSize: 10,
  onRowClicked: (row) => {
    this.rowSelected = row.data;
  }
};
```

Ilustración 51 - Ejemplo definición tabla ag-Grid

Como puede observarse en la ilustración anterior, es posible determinar el tipo de selección que se desee permitir sobre las filas de la tabla *(rowSelection)*; es posible añadir una paginación automática y definir el número de filas por cada página *(pagination* y *paginationPageSize)*; se le asignan las columnas de la tabla *(columnDefs)*; la forma de añadir eventos a la tabla es muy sencilla y además posee una larga lista de ellos disponibles *(onRowClicked)*; entre otros tantas configuraciones que pueden hacerse.

Cabe destacar la sencillez con la que se definen las cabeceras de la tabla. Simplemente se define una lista de objetos, en el que cada objeto representa una columna. Cada objeto, principalmente tiene dos atributos: “headerName”, que será el nombre que se muestre en la cabecera de la tabla; y “field” que será el valor del atributo que debe mostrar, esto es, cuando se introduce una lista de objetos que la tabla debe mostrar, si esos objetos poseen un atributo que se llame igual que el campo “field” de la cabecera, el valor de dicho campo será colocado en dicha columna. A continuación, se muestra un ejemplo de una cabecera (Ilustración 53):

```
this.columnDefs = [
  { headerName: "Pregunta", field: 'question' }
];
```

Ilustración 52 - Ejemplo cabecera tabla ag-Grid

Además, *ag-Grid* cuenta con una API que permite, modificar la tabla una vez se haya construido. Esta API se ha utilizado, entre otras cosas, para asignar los datos obtenidos del *back-end* a la tabla y que estos sean visibles en la interfaz gráfica. Para ello, la forma de proceder es como se muestra a continuación (Ilustración 53):

```
this.gridOptions.api.setRowData(data['body']);
```

Ilustración 53 - Ejemplo asignar datos a tabla de ag-Grid

Finalmente, este ha sido el resultado obtenido utilizando las tablas de *ag-Grid* (Ilustración 54):

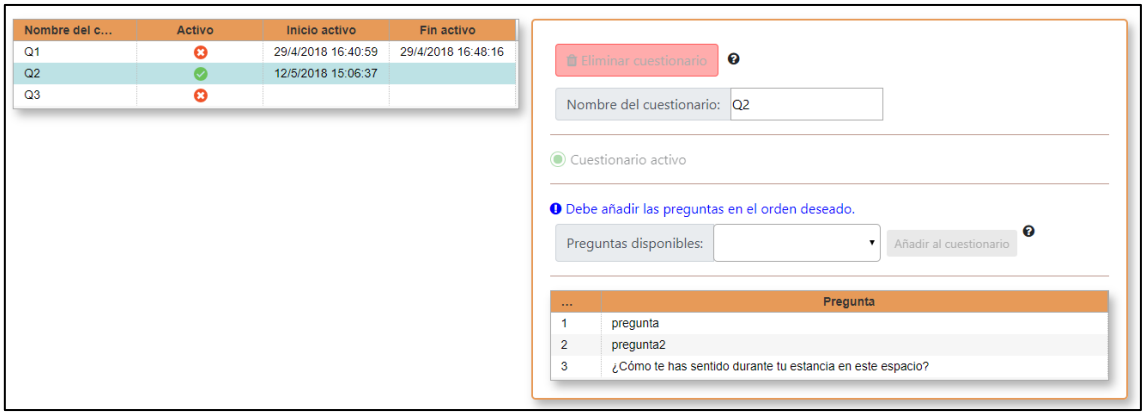


Ilustración 54 - Ejemplo de tablas ag-Grid

Descarga de datos

Para este proyecto, se requería que los miembros investigadores pudieran descargar de cada PPR, en formato *CSV*, los resultados que los usuarios de la aplicación móvil hubieran obtenido en cada participación.

Mientras un PPR se mantiene activo, puede que los administradores decidan cambiar de cuestionario. Por ello, existe la posibilidad de que un PPR contenga respuesta de varios cuestionarios. Para solucionar ese problema, se ha hecho una distinción de las preguntas a la hora de descargar los datos, indicando cuáles son de cada cuestionario.

Por otro lado, surgió otro problema con la codificación de los datos. Y es que las preguntas descargadas poseían tildes, espacios, signos de interrogación,

etc., que las herramientas de análisis que los investigadores usan para tratarlos, no admitían. Por ello, fue necesario no incluir el texto de las preguntas en el fichero *CSV* resultante.

En consecuencia, el problema se ha resuelto de la siguiente manera: los títulos de las preguntas son sustituidos por códigos identificativos a la hora de descargarlas al fichero *.csv*, y en un fichero de texto (*.txt*) aparte, se introduce cada código de pregunta con su título correspondiente. De esa manera, cuando los miembros investigadores descargan los datos, se descargan ambos ficheros, y así el fichero *.csv* queda libre de caracteres especiales.

Gracias al uso de la librería *ag-Grid*, anteriormente mencionada, se ha simplificado mucho la descarga de datos, puesto que su API posee una función que descarga todos los datos de la tabla a un fichero *.csv*. A continuación, se muestra un ejemplo del uso de esta función (Ilustración 55), donde se puede ver cómo se le asigna el nombre al fichero resultante a través de la propiedad *fileName*; además, se decide que el separador de cada valor del fichero *CSV* sea la coma; y por último, se suprimen las comillas que por defecto aparecen al comienzo y final de cada palabra.

```
exportData() {
  var params = {
    fileName: this.namePoint,
    columnSeparator: ',',
    suppressQuotes: true
  };

  this.gridOptionsDownload.api.exportDataAsCsv(params);
}
```

Ilustración 55 - Descargar datos CSV

Por otro lado, la descarga de datos para el fichero de texto, se ha tenido que hacer de manera manual, como se muestra a continuación (Ilustración 56):

```
exportDataAsTxt(filename, content) {
  var element = document.createElement('a');
  element.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(content));
  element.setAttribute('download', filename);

  element.style.display = 'none';
  document.body.appendChild(element);

  element.click();

  document.body.removeChild(element);
}
```

Ilustración 56 - Descargar datos TXT

Gráficos

Durante el desarrollo de la aplicación, surgió la idea de poder visualizar de manera gráfica algunas de las estadísticas de los resultados obtenidos en un PPR concreto. Para dar respuesta a esa idea, se ha utilizado la librería *Chart.js*[17].

La declaración de los gráficos en los ficheros *HTML*, se hace mediante etiquetas <*canvas*> a las que se les asigna un identificador. A continuación, se presenta un ejemplo (Ilustración 57):

```
<canvas id="myChart"></canvas>
```

Ilustración 57 - Ejemplo declaración gráfico

Posteriormente, desde el controlador (fichero *.ts* correspondiente), se obtiene el *canvas* anteriormente declarado, y sobre este, se define y crea el gráfico como puede observarse a continuación (Ilustración 58):

```
let canvas: any = document.getElementById('myChart');
let ctx = canvas.getContext('2d');
this.myChart = new Chart(ctx, {
  type: 'bar',
  data: <ChartData>{···
  },
  options: <ChartOptions>{···
  }
});
```

Ilustración 58 - Ejemplo creación gráfico

Como puede verse en la ilustración anterior, lo principal que debe definirse en un gráfico de *Chart.js* es el tipo, los datos y las opciones.

En el atributo ***data*** se incluyen los datos a representar en el gráfico. Estos datos pueden formar parte todos de un mismo *dataset* (conjunto de datos) o, por el contrario, si se quieren realizar comparaciones (como en este caso se ha hecho), se deben definir varios *dataset* y en cada uno de estos asignarle los datos correspondientes. Cada *dataset* es personalizable, en cuanto a nombre del *dataset* y estilo (colores, grosor de líneas, opacidad, etc.).

En cuanto al atributo ***options***, en este se definen las configuraciones del gráfico. En este atributo, se decide las escalas de los ejes; se decide si el gráfico debe adaptarse automáticamente a la interfaz gráfica o no; se elige si debe aparecer una leyenda y dónde debe aparecer; entre otras tantas opciones disponibles.

A continuación, se muestra un ejemplo de los gráficos empleados en este desarrollo (Ilustración 59):

[17] https://www.chartjs.org/

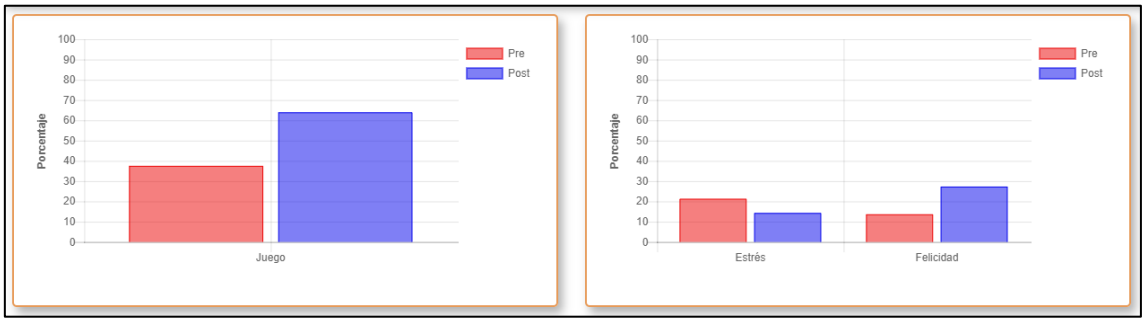


Ilustración 59 - Ejemplo de gráficos

Multiidioma

Debido al pensamiento de que este proyecto pueda expandirse, y con vista de futuro, se ha aportado la característica de ser multiidoma.

Para ello, se ha utilizado la internacionalización *i18n*[18], que permite, de manera sencilla, definir las traducciones de los textos que se muestran en la aplicación web.

Para comenzar, se ha importado el módulo *LOCALE_ID* (que incluye Angular en *@angular/core*) en el módulo principal de la aplicación (*app.module.ts*). En este se define el idioma local. A continuación, se presenta un ejemplo, en el que se define como idioma local el castellano (Ilustración 60):

```
import { LOCALE_ID } from '@angular/core';

@NgModule({
  // ...
  providers: [ { provide: LOCALE_ID, useValue: 'es' } ],
  // ...
})
export class AppModule { }
```

Ilustración 60 - Ejemplo definición i18n

La forma de trabajar, es a través de definición de traducciones por medio de identificadores, esto es, en los ficheros *HTML* se referencian, a través de los identificadores definidos en el fichero “messages.xml”, los textos que deben ser traducidos en cada caso. Se debe crear un fichero “messages.xml” por cada idioma a traducir y las traducciones correspondientes se encuentran en su interior.

[18] https://www.i18next.com/

A continuación, se muestra un ejemplo de cómo funciona (Ilustración 61). En este puede observarse cómo se define tanto el identificador como la traducción correspondiente en el fichero "messages.xml", en este caso, en inglés.

```
<trans-unit id="myId" datatype="html">
  <source>Hola</source>
  <target state="new">Hello</target>
</trans-unit>
```

Ilustración 61 - Ejemplo definición traducción

A continuación, en el fichero *HTML* que se desee, para que se realice la traducción, se indica el identificador al que hace referencia dicho texto. A continuación, se muestra un ejemplo (Ilustración 62), donde se ve cómo se desea traducir la palabra "Hola" del castellano al inglés, usando el identificador "myId" definido en la ilustración anterior:

```
<h3 i18n="@@myId">Hola</h3>
```

Ilustración 62 - Ejemplo de traducción con i18n

Estilo gráfico

En cuanto al estilo de la aplicación, se ha optado por utilizar la librería de estilos *ng-Bootstrap*[19].

Esta librería está basada en la librería *Bootstrap*[20], pero lo que las diferencia, básicamente, es que *ng-Bootstrap* está adaptada y optimizada para su uso en Angular.

Ng-Bootstrap proporciona elementos gráficos básicos con un aspecto profesional. Además, utilizando sus estilos, se ha conseguido que la aplicación web y sus interfaces gráficas se adapten a diferentes dispositivos y resoluciones de pantalla.

Lo primero que se ha hecho es instalar la librería a través del gestor de paquetes *NPM*.

El uso que se le ha querido dar a esta librería es a nivel de toda la aplicación, es por ello que, una vez completada la instalación, se ha importado e inyectado el módulo principal de la librería (*NgbModule*) en el módulo principal de la aplicación *(app.module.ts*).

Una vez incluido *ng-Bootstrap* en la aplicación, se pueden hacer uso de sus estilos. Por ejemplo, se ha hecho especial uso de sus ventanas emergentes (*modals)*

[19] https://ng-bootstrap.github.io/#/home
[20] https://getbootstrap.com/

para la muestra de avisos y al investigador. A continuación, se muestran algunos ejemplos de las ventanas emergentes creadas a través de esta librería (Ilustración 63 e Ilustración 64):

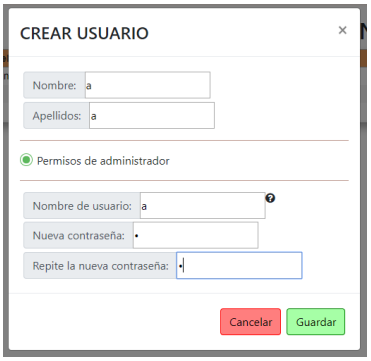


Ilustración 63 - Ejemplo ventana emergente usando ng-Bootstrap (I)

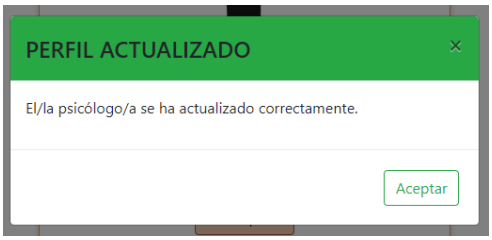


Ilustración 64 - Ejemplo ventana emergente usando ng-Bootstrap (II)

Además, para disminuir el tamaño de la aplicación y mejorar su rendimiento, se ha incluido el paquete de iconos *Font-Awesome*[21]. Este paquete incluye iconos propios, con lo cual, evita el tener que añadir imágenes externas que convierten a la aplicación más pesada y con más carga de procesamiento.

Factoría de ventanas

Para favorecer la mantenibilidad, modularidad y escalabilidad de la aplicación, se ha desarrollado una factoría de ventanas, la cual se encarga de construir las ventanas emergentes (*modals*) de la aplicación. De esta manera, para cambiar alguna característica de alguna ventana común, como por ejemplo la ventana de éxito cuando una operación se ha realizado correctamente, basta con modificarlo una sola vez en esta factoría y el cambio se hará efectivo en toda la aplicación.

La factoría de ventanas es un *service* que se ha creado en la carpeta *"generalServices"*. A continuación, se muestra un ejemplo de la construcción de una ventana, en este caso, de una ventana de éxito, donde se puede ver como se hace uso de la librería *ng-Bootstrap* (detallada en el apartado anterior), para abrir en una ventana emergente el componente "OkModalComponent" (Ilustración 65):

[21] https://fontawesome.com/

```
constructor(private modalService: NgbModal) { }

openOkModal(title: string, message: string) {
  const modalRef = this.modalService.open(OkModalComponent, { centered: true });
  modalRef.componentInstance.message = message;
  modalRef.componentInstance.title = title;
  return modalRef;
}
```

Ilustración 65 - Ejemplo creación ventana emergente

Finalmente, para crear una ventana de éxito, bastaría con llamar a la función mostrada en la ilustración anterior, desde el componente que se quiera abrir (Ilustración 66):

```
this.modalFactoryService.openOkModal("TITULO", "MENSAJE");
```

Ilustración 66 - Ejemplo haciendo uso de la factoría de ventanas

6.3. Aplicación móvil

Antes de comenzar con el detalle de este desarrollo, se procederá explicar el proceso de instalación y preparación de las herramientas necesarias que se necesitan para llevarlo a cabo.

6.3.1. Instalación y preparación

Como se ha mencionado anteriormente, la aplicación móvil se realizará para el sistema operativo Android y usando el entorno de desarrollo Android Studio.

En este caso, se han utilizado las siguientes versiones:

- **Android:** 8.1.0 (Oreo)
- **Android Studio:** Se ha ido actualizando durante todo el desarrollo (última versión).

Una vez instalado Android Studio, basta con generar un proyecto nuevo, siguiendo los pasos del asistente de creación.

Antes de proceder al desarrollo, es importante destacar el fichero "build.gradle". En este fichero se definen tanto las dependencias del proyecto, así como también datos importantes relativos a la aplicación. Estos datos importantes son los siguientes:

- **compileSdkVersion:** Este atributo indica respecto a las especificaciones de qué versión de Android se compila la aplicación. En este caso, se ha elegido el nivel de API número 27, que se

corresponde con la última versión de Android actualmente disponible (Android Oreo 8.1.0).

- **minSdkVersion:** Este atributo indica cuál es la versión mínima de Android requerida para poder instalar la aplicación. En este caso, se ha elegido el nivel de API número 21, que se corresponde con la versión Android Lollipop 5.0.

6.3.2. Aspectos generales

Una vez especificadas las características de la aplicación, se procede a continuación a detallar cuáles son sus aspectos generales y qué puntos son los más destacables.

Como se ha mencionado anteriormente en el capítulo de diseño (ver apartado 5.3), el núcleo de la aplicación se ha dividido en varios paquetes: *activities*, *bean*, *adapters*, *services* ,y *dialogs*.

En el paquete *activities* se encuentran las actividades de la aplicación, esto es, todas las clases que se corresponden con las interfaces gráficas que componen la aplicación. A continuación, se muestran de ejemplo las actividades correspondientes al registro en la aplicación (Ilustración 67) y a la respuesta a una pregunta del cuestionario (Ilustración 68).

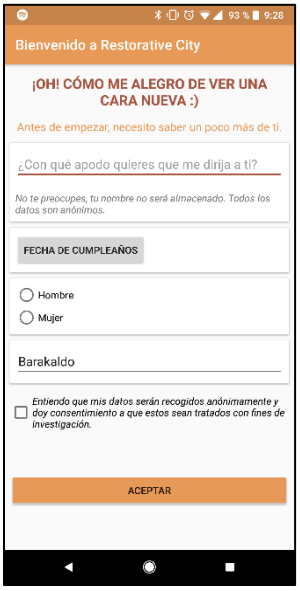


Ilustración 67 - Ejemplo de actividad (I)

Ilustración 68 - Ejemplo de actividad (II)

Las actividades, hacen uso de las clases contenidas en el resto de paquetes de la aplicación, para ejecutar diferentes tareas.

En el paquete *bean* se encuentran aquellas clases que son de utilidades o se encargan del almacenaje de información y comunicación con la API REST. Por un lado, se encuentra el gestor de la base de datos (*DataBaseManager*) que se encarga de gestionar las peticiones que se realizan a la base de datos, y la base de datos.

Por otro lado, está el gestor de peticiones a la API REST (*HttpRequestManager*) que se encarga de realizar las peticiones. Finalmente, una clase de utilidades (*Utils*) que contiene funciones de interés para toda la aplicación, como por ejemplo, funciones de tratamiento de fechas, entre otras.

Los adaptadores contenidos en el paquete *adapters* básicamente sirven para definir de qué modo deben tratarse la información que aparece en cualquier lista de la aplicación. Esto es, es el encargado de recibir los datos, y asignar a cada elemento de la interfaz gráfica, su valor correspondiente.

En el paquete *service* se encuentran tanto el servicio en segundo plano, que se encarga de comprobar si un usuario entra o sale de un PPR; como el programador de tareas, que se encarga de lanzar periódicamente el servicio en segundo plano.

Finalmente, en el paquete *dialogs* se han incluido las ventanas de diálogo de la aplicación. Estas ventanas sirven para requerir la atención del usuario o mostrar información relevante, como por ejemplo, los resultados obtenidos en la prueba de concentración. A continuación, se muestra un ejemplo de una ventana de diálogo (Ilustración 69):

Ilustración 69 - Ejemplo ventana de diálogo

6.3.3. Aspectos destacados

A continuación, se detallan los puntos más destacables que se han desarrollado para esta aplicación móvil, haciendo hincapié en aquello más curioso o en los problemas que hayan surgido durante su implementación.

Peticiones a la API REST

En este desarrollo, es necesario interactuar con la API REST. Para ello, se ha optado por utilizar la librería *Volley*[22]. Pese a poder realizarse esta implementación de manera nativa utilizando tareas asíncronas *(AsyncTask)* y el cliente de conexiones HTTP *(HttpURLConnection)*, estas no son igual de eficientes como lo es *Volley*. Además, esta librería es la recomendada por la página oficial de Android[23].

La forma de trabajar de *Volley* es a través de la creación de peticiones que, posteriormente, se añaden a la cola de peticiones de este y *Volley* se encarga de ejecutar dichas peticiones en su propio hilo de ejecución. Además, esta librería trabaja con caché HTTP, es decir, que si el servicio al que se llama indica que los datos no van a cambiar hasta dentro de un tiempo, *Volley* no volverá a hacer llamadas para recuperar los datos de nuevo, sino que guardará y usará una copia que tendrá en caché. Estas características, mejoran notablemente la eficiencia de las peticiones a la API REST.

Para comenzar a trabajar con *Volley*, ha sido necesario incluir la librería en las dependencias de la aplicación. Para ello, en el fichero "build.gradle" se ha incluido lo siguiente (Ilustración 70):

```
implementation 'com.android.volley:volley:1.1.0'
```

Ilustración 70 - Importación librería Volley

Para realizar peticiones, se utilizan objetos *Request*. En estos objetos, se definen las características de la petición que se desea realizar. Para este desarrollo, todas las peticiones serán del tipo *JsonObjectRequest*. Por ello, se ha creado una clase llamada "CustomHttpRequest" que extiende de la clase "JsonObjectRequest". Esta nueva clase, facilita la creación de las peticiones y asegura que todas ellas sean del tipo antes mencionado.

Para construir una nueva *CustomHttpRequest*, tan solo será necesario proporcionar el método HTTP que se quiera usar; la URL de la llamada a la API REST; la propia petición que se desea realizar; una función que se ejecutará al recibir la respuesta; y otra que se ejecutará en caso de producirse un error. Además, esta clase posee un atributo *Priority*, en el cual se especifica la prioridad con la que *Volley* debe tratar dicha petición. A continuación, se presenta la clase "CustomHttpRequest" (Ilustración 71):

[22] https://github.com/google/volley
[23] https://developer.android.com/training/volley/

```
public class CustomHttpRequest extends JsonObjectRequest {

    private Priority mPriority;

    public CustomHttpRequest(int method, String url, JSONObject jsonRequest,
                             Response.Listener<JSONObject> listener, Response.ErrorListener errorListener) {

        super(method, url, jsonRequest, listener, errorListener);
    }

    public void setPriority(Priority priority) { mPriority = priority; }

    @Override
    public Priority getPriority() { return mPriority == null ? Priority.NORMAL : mPriority; }

}
```

Ilustración 71 - CustomHttpRequest

Además, también se ha creado un gestor de peticiones al que se le ha llamado "HttpRequestManager". Este gestor, contiene la cola de peticiones *(RequestQueue)* que se van creando, y es este el encargado de ejecutar cada una de estas. Es importante detallar que la ejecución de las peticiones se hace de manera automática, por ello, tan solo es necesario implementar la función que permita añadir nuevas peticiones a la cola, como se muestra a continuación (Ilustración 72):

```
public <T> void add(Request<T> req) {
    req.setTag(reqTag);
    getRequestQueue().add(req);
}
```

Ilustración 72 - Ejemplo añadir peticiones a la cola

Asimismo, se muestra a continuación un ejemplo de creación de una petición *CustomHttpRequest* (Ilustración 73):

```
CustomHttpRequest request = new CustomHttpRequest(
        Request.Method.GET,
        url: HttpRequestManager.BASE_API_URL + "/activePoints",
        jsonRequest: null,
        new Response.Listener<JSONObject>() {

            @Override
            public void onResponse(JSONObject response) {
                // Ha ido bien
                // Obtener datos de response
            }
        },
        new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                // Se ha producido un error
            }
        });

request.setPriority(Request.Priority.HIGH); // Prioridad deseada
HttpRequestManager.getInstance().add(request); // Añadir peticion a la cola
```

Ilustración 73 - Creación de petición

Mapa

Cuando el usuario entra en la aplicación, se muestra un mapa indicándole su ubicación en tiempo real. Además, se realiza una petición a la API REST para obtener los PPR que se encuentren activos en ese momento, y estos se muestran sobre el mapa. La forma de representar cada punto en estudio activo es a través de objetos *Marker*, propios de *Google Maps*, de color verde.

Para la creación del mapa se ha utilizado los servicios propios de *Google Maps*, incluyéndolos en las dependencias de la aplicación.

Además, para poder crear un mapa, se ha tenido que pedir las credenciales en la *Google Cloud Platform*, de la misma manera que se han solicitado para el desarrollo anterior (ver desarrollo 6.2).

Una vez obtenidas las credenciales de uso de mapas, estas deben ser incluidas en la aplicación. Para ello, en el fichero “AndroidManifest.xml” se han incluido de la siguiente manera (Ilustración 74):

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

Ilustración 74 - Credenciales para uso del mapa

En cuanto a la vista, el mapa está contenido en un elemento <*fragment*> dentro de la interfaz gráfica general.

La creación del mapa se ha hecho en la actividad que lo muestra ("MapActivity.java"). Para ello, dicha actividad implementa la interfaz *OnMapReadyCallback* que le proporciona las funciones necesarias para trabajar con el mapa.

La función más importante para la creación del mapa es la *onMapReady*, que determina cuando el mapa está listo y creado. En esta función, se guarda el objeto del mapa creado y se le asigna el tipo de mapa que se quiere mostrar. A continuación, se muestra un ejemplo (Ilustración 75):

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mGoogleMap = googleMap;
    mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
}
```

Ilustración 75 - Ejemplo creación del mapa

Una vez creado el mapa, se puede trabajar con las funciones que *Google Maps* ofrece para este tipo de objeto.

A continuación, se muestran unos ejemplos del mapa que se ha utilizado (Ilustración 76 e Ilustración 77):

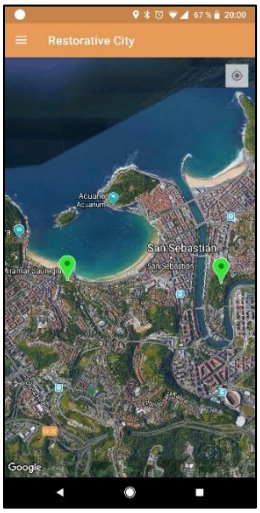


Ilustración 76 - Ejemplo de mapa (I)

Ilustración 77 - Ejemplo de mapa (II)

Servicios en segundo plano

Con la llegada de la versión 8.0 de Android (Android Oreo), se introdujeron algunos cambios de comportamiento y limitaciones de los servicios que se ejecutan en segundo plano. Estas modificaciones han afectado totalmente al desarrollo de esta aplicación, ya que, en esta, se utilizan servicios en segundo plano para comprobar cuándo un usuario está entrando o saliendo de un PPR. Por ello, a continuación se exponen los problemas y dificultades encontrados durante el desarrollo, y de qué forma se han solventado.

Uno de los mayores problemas que han surgido debido a las nuevas limitaciones que se han añadido por parte de Google en la versión Oreo, es la durabilidad de la ejecución de los servicios en segundo plano. En esta versión, cuando un servicio se encuentre en segundo plano, el sistema operativo podrá finalizar dicho servicio cuando lo considere necesario.

No obstante, si un servicio en segundo plano le indica en todo momento al usuario por medio de un icono permanente en la barra de tareas que se está ejecutando, el sistema operativo no finaliza este servicio puesto que considera que el usuario es consciente de que este se está ejecutando, y el usuario puede decidir libremente si mantener la ejecución o no.

Por ello, se ha aplicado esa solución al problema: añadirle, a través de una notificación, un icono permanente en la barra de tareas mientras el servicio en segundo plano se esté ejecutando (Ilustración 78).

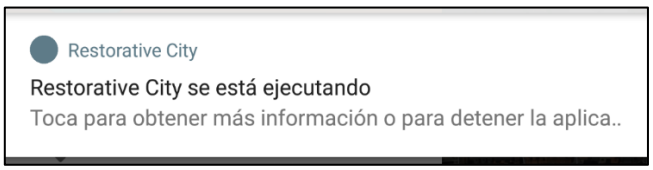


Ilustración 78 - Notificación servicio en segundo plano

Sin embargo, tras solucionar este problema, surgieron dos más.

Por un lado, resultaba para el usuario muy incómodo tener constantemente una notificación en el dispositivo. Además, mantener un proceso ejecutándose indefinidamente resultaba una idea preocupante, ya que era evidente que podría afectar al rendimiento general del dispositivo.

Por otro lado, una de las limitaciones que se han incorporado en Android Oreo es la frecuencia con la que se puede pedir la geolocalización del dispositivo. La geolocalización del dispositivo puede solicitarse tantas veces se quiera, pero solo se atenderá la petición de una misma aplicación cada 15 minutos, lo cual es un gran inconveniente para esta aplicación, que necesita un tiempo menor de comprobación de la localización para la detección de los PPR.

Vistos y entendidos los dos principales problemas que se presentan en uno de los aspectos esenciales para este desarrollo, a continuación se detalla cómo se ha logrado resolver el problema de la ejecución del servicio en segundo plano, y

en el siguiente apartado (ver apartado de Geolocalización), se detalla la resolución del problema de la limitación de las peticiones de geolocalización.

Para evitar la ejecución constante del servicio en segundo plano que se encarga de obtener la ubicación del dispositivo, se ha utilizado un programador de tareas *(JobScheduler)*. Al programador de tareas se le añade una tarea la primera vez que el usuario entra en la aplicación, que será la encargada de iniciar el servicio en segundo plano.

A continuación, se muestra cómo se añade la tarea al *JobScheduler* (Ilustración 79):

```
JobInfo jobInfo = new JobInfo.Builder( jobId: 1, new ComponentName( pkg: this, JobSchedulerService.class))
        .setRequiredNetworkType(JobInfo.NETWORK_TYPE_ANY)
        .setMinimumLatency(TimeUnit.SECONDS.toMillis( duration: 3))
        .setPersisted(true)
        .build();

JobScheduler js = (JobScheduler) getSystemService(Context.JOB_SCHEDULER_SERVICE);
js.schedule(jobInfo);
```

Ilustración 79 - Ejemplo de tarea programada

Algo a tener en cuenta de las tareas programadas, es que se pueden configurar para que solo sean ejecutadas cuando se cumplan unas ciertas condiciones. En este caso, como puede observarse en la ilustración anterior, las condiciones que deben cumplirse son dos:

- Al menos tener alguna fuente de conexión a internet *(setRequiredNetworkType)*.
- Mantenerse programada aunque el dispositivo se reinicie *(setPersisted)*.

Esta última condición, exige que se conceda un permiso extra en el fichero de manifiesto, como se muestra a continuación (Ilustración 80):

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

Ilustración 80 - Permiso extra para tarea programada

Al ejecutarse la tarea anteriormente especificada, lanza el servicio en segundo plano (Ilustración 81).

```
@Override
public boolean onStartJob(JobParameters jobParameters) {
    Intent myService = new Intent( packageContext: this, LocationBackgroundService.class);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        startForegroundService(myService);
    } else {
        startService(myService);
    }

    jobFinished(jobParameters, needsReschedule: true);

    return false;
}
```

Ilustración 81 - Lanzamiento del servicio en segundo plano

El servicio en segundo plano por su parte, comprueba la ubicación del dispositivo y obtiene los PPR activos haciendo una petición a la API REST. Una vez obtenga los puntos activos, comprueba si alguno de esos puntos está cerca de la posición del usuario, y en caso de haber algún punto cercano, se mantiene en ejecución comprobando frecuentemente la ubicación; en el caso contrario, esto es, de no haber ningún punto cercano, el servicio finalizará, y será la tarea programada la que vuelva a iniciarlo pasado un tiempo.

Geolocalización

Antes de proceder al detalle de la resolución del problema de la limitación de peticiones de geolocalización, es importante detallar que, si la petición de geolocalización no es aprobada por la API de Google, se puede obtener la última ubicación registrada en el dispositivo, aunque haya sido solicitada por otra aplicación. Esta ha sido la solución empleada, ya que permite obtener la ubicación del dispositivo igualmente, aunque no de manera precisa.

Para las peticiones de geolocalización, se ha utilizado la API *FusedLocationProviderClient* [24]. Primero, se obtiene el objeto a través del cual se realizarán las peticiones como se muestra a continuación (Ilustración 82):

```
mFusedLocationClient = LocationServices.getFusedLocationProviderClient( context: this);
```

Ilustración 82 - Obtención API de geolocalización

[24] https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient

A continuación, se define la petición que se desea realizar (Ilustración 83):

```
mLocationRequest = new LocationRequest();
mLocationRequest.setInterval(10000);
mLocationRequest.setFastestInterval(5000);
mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
```

Ilustración 83 - Definición de petición de geolocalización

Es importante destacar algunos detalles de la petición representada en la ilustración anterior:

- **setInterval:** Cada cuánto tiempo se realiza la petición.
- **setFastestInterval:** Cuál es el tiempo mínimo que necesita la aplicación entre cada respuesta para poder gestionarlas correctamente.
- **setPriority:** Prioridad asignada a la petición.

Finalmente, se realiza la petición, indicándole además una función *callback* que se ejecutará cuando se reciba una respuesta. A continuación, puede observarse la ejecución de la petición (Ilustración 84):

```
mFusedLocationClient.requestLocationUpdates(mLocationRequest, mLocationCallback, Looper.myLooper());
```

Ilustración 84 - Ejecución de petición de geolocalización

Y la función *callback* correspondiente es (Ilustración 85):

```
mLocationCallback = new LocationCallback() {
    @Override
    public void onLocationResult(LocationResult locationResult) {
        super.onLocationResult(locationResult);
        mLocation = locationResult.getLastLocation();
    }
};
```

Ilustración 85 - Función callback de petición de geolocalización

Como puede observarse en la ilustración anterior, se obtiene la última ubicación registrada a través de la función *getLastLocation()*, sin saber si esta se ha obtenido a través de la petición realizada, o si por el contrario, ya estaba registrada en el dispositivo por otra aplicación.

Base de datos local

Para este desarrollo, se ha utilizado una base de datos local. Esta base de datos, en gran parte, es una base de datos temporal, ya que su función en este desarrollo no es más que almacenar datos temporales. En esta se almacenan resultados de cada participación que al finalizar la participación o al abandonarla, los datos son eliminados. Exceptuando dos tablas, que almacenan información necesaria como se ha explicado anteriormente (ver apartado 4.6.2).

Para favorecer la modularidad de la aplicación y la correcta gestión de la base de datos, se ha creado una clase que sigue el patrón de diseño *Singleton* (Welicki, s.f.), y que se encarga de la gestión de la base de datos. De esta manera, las clases no accederán directamente a la base de datos, sino que realizarán llamadas al gestor de base de datos, y será este quien realice las peticiones a la base de datos.

La clase que contiene la base de datos, extiende de *SQLiteOpenHelper*. La creación de las tablas es muy sencilla, como se muestra a continuación (Ilustración 86):

```
db.execSQL("CREATE TABLE User (" +
        "'id' INTEGER NOT NULL," +
        "'name' VARCHAR(50) NOT NULL)"
);
```

Ilustración 86 - Ejemplo de creación de tabla SQLite

La forma de realizar consultas a la base de datos se asemeja bastante a las consultas que se realizan en *MySQL*, con lo cual, resulta muy sencillo realizarlas (Ilustración 87):

```
SQLiteDatabase db = getReadableDatabase();
Cursor cursor = db.rawQuery( sql: "SELECT * FROM User",  selectionArgs: null);
```

Ilustración 87 - Ejemplo de consulta SQLite

En este desarrollo, para proporcionar una mejor manejabilidad de los datos devueltos por la base de datos, se ha implementado una función que, transforma el resultado de la consulta a formato *JSONObject* (Ilustración 88):

```
private JSONArray convertCursorToJSONArray(Cursor cursor) {

    JSONArray resultSet = new JSONArray();
    cursor.moveToFirst();
    while (!cursor.isAfterLast()) {
        int totalColumn = cursor.getColumnCount();
        JSONObject rowObject = new JSONObject();
        for (int i = 0; i < totalColumn; i++) {
            if (cursor.getColumnName(i) != null) {
                try {
                    rowObject.put(cursor.getColumnName(i), cursor.getString(i));
                } catch (Exception e) {
                    Log.d(TAG, e.getMessage());
                }
            }
        }
        Log.i(Utils.TAG, rowObject.toString());
        resultSet.put(rowObject);
        cursor.moveToNext();
    }

    return resultSet;

}
```

Ilustración 88 - Función de transformación a JSONObject

Multiidioma

Para favorecer la adaptabilidad y personalización de la aplicación a todos sus usuarios, se le ha dotado a esta la característica de ser multiidioma. En este caso, los idiomas implementados han sido el inglés (como idioma por defecto), el castellano y el euskera.

Para desarrollar esta característica, se han creado 3 ficheros (uno por cada idioma) que contienen los textos de la aplicación, traducidos a la lengua correspondiente. A continuación, se muestra un ejemplo, donde se puede ver los 3 ficheros creados, con su correspondiente idioma asignado (Ilustración 89):

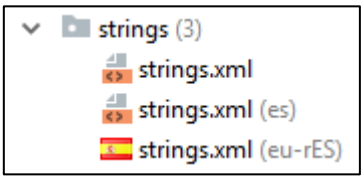


Ilustración 89 - Ejemplo ficheros multiidioma

La estructura que contiene cada uno de los ficheros de idiomas, se basa en *clave-valor*, esto es, se asigna un identificador único y su correspondiente

traducción. A continuación, se muestra un fragmento del fichero de idioma en inglés, donde se puede observar cada identificador (propiedad *name*) con su correspondiente valor (Ilustración 90):

```
<string name="app_name">Restorative City</string>
<string name="hello_world">Hello World!</string>
<string name="more_about_you">Oh! It\'s always nice to see new faces :)</string>
<string name="before_start">Before start, I need to know a little more about you.</string>
<string name="name">what nickname do you want me to refer to you?</string>
```

Ilustración 90 - Ejemplo fichero de idioma

Los dos ficheros restantes, el de castellano y euskera, poseen los mismos identificadores, pero con su correspondiente traducción. Sin embargo, si en alguno de los ficheros no se introdujera algún identificador ni su traducción, la aplicación pondrá el texto de dicho identificador en el idioma por defecto.

La forma de aplicar los textos definidos en los ficheros de idiomas es referenciando cada clave en el lugar donde se quiera introducir el texto, en vez de introducir el texto de manera manual (ya que si no, no tendrían sentido los pasos anteriores). A continuación, se muestra un ejemplo donde se define una caja de texto, que contiene la traducción correspondiente al identificador *"more_about_you"* (Ilustración 91):

```
android:text="@string/more_about_you"
```

Ilustración 91 - Ejemplo de uso de fichero de idioma

7. VERIFICACIÓN Y EVALUACIÓN

En este capítulo, se detallará cuáles han sido las pruebas realizadas para cada uno de los desarrollos y los resultados obtenidos en cada caso.

7.1. Pruebas de la API REST

Debido a la cantidad de servicios que expone la API REST, se ha optado por automatizar las pruebas para agilizar así el proceso de pruebas. Para ello, se ha utilizado la herramienta *Postman*.

Como se ha comentado en el apartado de herramientas (ver apartado 2.4), *Postman* es una plataforma que permite la creación de solicitudes a una API. A través de estas solicitudes, se puede comprobar si el servicio que se ha solicitado funciona como se esperaba o no. Además, *Postman* permite la creación de variables para que estas puedan ser usadas en otras pruebas, por ejemplo, en la prueba de inicio de sesión, el *token* es recogido en una variable, para posteriormente ser añadido automáticamente en las pruebas restantes que lo requieran.

Para mantener un cierto orden dentro del conjunto de pruebas realizadas, los servicios han sido agrupados por temática o funcionalidad, esto es, las pruebas relacionadas con los PPR, no están en el mismo bloque de pruebas que las de los cuestionarios. Por ello, los bloques de pruebas resultantes han sido ocho: *auth*, *psychologist*, *point*, *question*, *questionnaire*, *answer*, *suggestion* y *user*. De esta manera, cada bloque de pruebas puede ejecutarse individualmente, comprobando así el comportamiento de un grupo de funcionalidades concretas.

A continuación, se muestra la interfaz gráfica de *Postman* (Ilustración 92) donde puede verse la división de los bloques de pruebas:

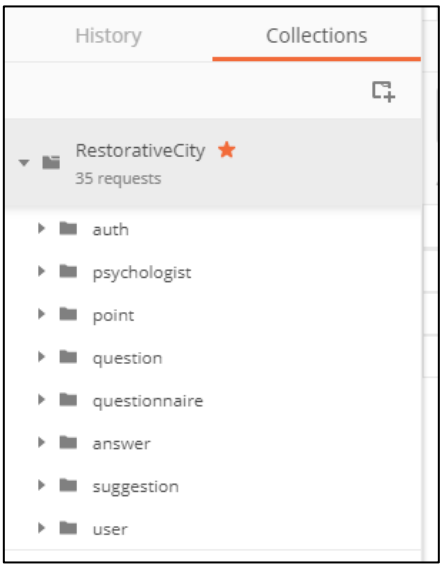


Ilustración 92 – Bloques de prueba en Postman

En la para realizar peticiones en *Postman*, se debe elegir el método *HTTP* que se quiera utilizar en cada caso, e introducir la llamada a la API REST (*endpoint*) que se quiera realizar. Además, se han utilizado los apartados "Headers", para introducir la cabecera *Authorization* con el *token*; "Body", para introducir el cuerpo de la petición; y "Test" para realizar pruebas automáticas una vez recibida la respuesta. A continuación, se muestra un ejemplo de petición en *Postman* (Ilustración 93), donde se puede ver que el método usado es el *GET*, y se realiza la petición de obtener todas las preguntas, pasándole además, el *token* en la cabecera *Authorization*. Además, se observa a la derecha la respuesta obtenida:

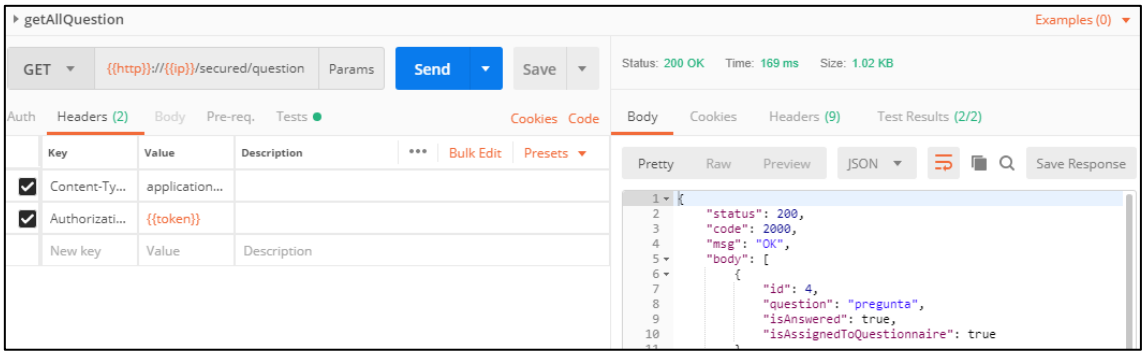


Ilustración 93 - Ejemplo petición en Postman

En las siguientes tablas (Tabla 39, Tabla 40, Tabla 41, Tabla 42, Tabla 43, Tabla 44, Tabla 45 y Tabla 46) se presentan las pruebas realizadas para cada uno de los conjuntos de pruebas anteriormente definidos.

Tabla 39 - Pruebas del bloque Auth

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Iniciar sesión con un nombre de usuario que no está registrado en el sistema.	Mensaje de error notificando de que ese usuario no está registrado en el sistema.	Se ha recibido un mensaje de error notificando de que ese usuario no está registrado en el sistema.	SÍ	
2	Iniciar sesión con un nombre de usuario existente, pero con una contraseña incorrecta.	Mensaje de error notificando de que la contraseña no es correcta.	Se ha recibido un mensaje de error notificando de que la contraseña no es correcta.	SÍ	
3	Iniciar sesión con un nombre de usuario y contraseña correctos.	Mensaje de OK y la información del usuario y su token viene en el cuerpo de la respuesta.	Se ha recibido un mensaje de OK y la información del usuario y su token viene en el cuerpo de la respuesta.	SÍ	
4	Iniciar sesión sin haber introducido el nombre de usuario ni contraseña.	Mensaje de error notificando de que los datos de la petición no son válidos.	Se ha recibido un mensaje de error notificando de que los datos de la petición no son válidos.	SÍ	
5	Cerrar sesión.	Mensaje de OK.	Se ha recibido un mensaje de OK.	SÍ	

Tabla 40 - Pruebas del bloque Psychologist

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Obtener todos los miembros investigadores dados de alta en el sistema.	Mensaje de OK y en el cuerpo de la respuesta viene la lista de los miembros investigadores dados de alta.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene la lista de los miembros investigadores dados de alta.	SÍ	
2	Añadir un nuevo miembro investigador que no está dado de alta.	El nuevo investigador se guarda. Mensaje de OK y en el cuerpo de la respuesta viene el nuevo investigador almacenado.	El nuevo investigador se ha guardado. Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene el nuevo investigador almacenado.	SÍ / NO	Algunos de los atributos del investigador no se han devuelto en el formato correcto.
2.1			Ha surgido un error en el servidor.	NO	
2.2			El nuevo investigador se ha guardado. Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene el nuevo investigador almacenado.	SÍ	Se producía una excepción no controlada.
3	Añadir un nuevo miembro investigador que ya está dado de alta.	Investigador no añadido. Mensaje de error informando que el miembro investigador ya existe.	El investigador no se ha añadido. Se ha mostrado un mensaje de error informando que el miembro investigador ya existe.	SÍ	
4	Actualizar un miembro investigador diferente al que ha iniciado la sesión.	Investigador no actualizado. Mensaje de error informando de que cada investigador puede actualizar su propio perfil.	El investigador se ha actualizado. Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene el nuevo investigador actualizado.	NO	

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
4.1			El investigador no se ha actualizado. Se ha recibido un mensaje de error informando de que cada investigador puede actualizar su propio perfil.	SÍ	No se estaba controlando la identidad del investigador que realizaba la petición.
5	Actualizar el miembro investigador que ha iniciado sesión.	Investigador actualizado. Mensaje de OK.	El investigador se ha actualizado. Se ha recibido mensaje de OK.	SÍ	
6	Ascender a un investigador.	Investigador actualizado. Mensaje de OK.	El investigador se ha actualizado. Se ha recibido mensaje de OK.	SÍ	
7	Degradar a un investigador administrador (pero no el último).	Investigador actualizado. Mensaje de OK.	El investigador se ha actualizado. Se ha recibido mensaje de OK.	SÍ	
8	Degradar al último investigador administrador.	Investigador no actualizado. Mensaje de error informando que debe permanecer al menos un administrador.	El investigador no se ha actualizado. Se ha recibido mensaje de error informando que debe permanecer al menos un administrador.	SÍ	
9	Eliminar a un investigador	Investigador eliminado. Mensaje de OK.	El investigador se ha eliminado. Se ha recibido mensaje de OK.	SÍ	
10	Las pruebas número 1, 2, 3, 4, 5, 6, 7, 8 y 9, pero sin haber iniciado sesión o con un token no válido.	Mensaje de error.	Se ha recibido mensaje de OK.	NO	
10.1			Se ha recibido mensaje de error.	SÍ	No se estaba comprobando el token.
11	Las pruebas número 1, 2, 3, 4, 6, 7, 8 y 9, pero sin ser administrador.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	

Tabla 41 - Pruebas del bloque Point

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Obtener todos los PPR.	Mensaje de OK y en el cuerpo de la respuesta viene la lista de los PPR.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene la lista de los PPR.	SÍ	
2	Añadir un nuevo PPR.	PPR añadido. Mensaje de OK y en el cuerpo de la respuesta viene el nuevo PPR almacenado.	El PPR se ha añadido. Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene el nuevo PPR almacenado.	SÍ	
3	Actualizar un PPR existente.	PPR actualizado. Mensaje de OK.	Se ha actualizado el PPR. Se ha recibido mensaje de OK.	SÍ	
4	Actualizar un PPR no existente.	Mensaje de error.	Error del servidor.	NO	
4.1			Se ha recibido mensaje de error.	SÍ	No se controlaba el error.
5	Activar un PPR existente.	PPR activado. Mensaje de OK.	Se ha activado el PPR. Se ha recibido mensaje de OK.	SÍ	
6	Activar un PPR no existente.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
7	Desactivar un PPR existente.	PPR desactivado. Mensaje de OK.	Se ha desactivado el PPR. Se ha recibido mensaje de OK.	SÍ	
8	Desactivar un PPR no existente.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
9	Eliminar un PPR existente.	PPR eliminado. Mensaje de OK.	Se ha eliminado el PPR. Se ha recibido mensaje de OK.	SÍ	
10	Eliminar un PPR no existente.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
11	Las pruebas número 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10 pero sin haber iniciado sesión o con un token no válido.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
12	Las pruebas número 2, 3, 4, 5, 6, 7, 8, 9 y 10, pero sin ser administrador.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
13	Obtener los PPR activos.	Mensaje de OK y en el cuerpo de la respuesta viene la lista de los PPR activos.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene la lista de los PPR activos.	SÍ	

Tabla 42 - Pruebas del bloque Question

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Obtener todas las preguntas.	Mensaje de OK y en el cuerpo de la respuesta viene la lista de las respuestas.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene la lista de las respuestas.	SÍ	
2	Obtener una pregunta concreta dado su identificador.	Mensaje de OK y en el cuerpo de la respuesta viene la pregunta.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene la pregunta.	SÍ	
3	Actualizar una pregunta existente.	Pregunta actualizada. Mensaje de OK.	Se ha actualizado la pregunta. Se ha recibido mensaje de OK.	SÍ	
4	Actualizar una pregunta no existente.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
5	Eliminar una pregunta existente.	Pregunta eliminada. Mensaje de OK.	Se ha eliminado la pregunta. Se ha recibido mensaje de OK.	SÍ	
6	Eliminar una pregunta no existente.	Mensaje de error.	Error del servidor.	NO	
6.1			Se ha recibido mensaje de error.	SÍ	No se estaba controlando el error.
7	Las pruebas número 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10 pero sin haber iniciado sesión o con un token no válido.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
8	Las pruebas número 2, 3, 4, 5, 6, 7, 8, 9 y 10, pero sin ser administrador.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	

Tabla 43 - Pruebas del bloque Questionnaire

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Obtener todos los cuestionarios.	Mensaje de OK y en el cuerpo de la respuesta viene la lista de los cuestionarios.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene la lista de los cuestionarios.	SÍ	
2	Obtener un cuestionario concreto dado su identificador.	Mensaje de OK y en el cuerpo de la respuesta viene el cuestionario.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene el cuestionario.	SÍ	
3	Actualizar un cuestionario que ha sido respondido.	Cuestionario creado. Mensaje de OK.	Se ha creado el cuestionario. Se ha recibido mensaje de OK.	SÍ	
4	Actualizar un cuestionario que no ha sido respondido.	Cuestionario actualizado. Mensaje de OK.	Se ha actualizado el cuestionario. Se ha recibido mensaje de OK.	SÍ	
5	Actualizar un cuestionario no existente.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
6	Eliminar un cuestionario que ha sido respondido.	Cuestionario no eliminado. Mensaje de error.	No se ha eliminado el cuestionario. Se ha recibido mensaje de error.	SÍ	
7	Eliminar un cuestionario que no ha sido respondido.	Cuestionario eliminado. Mensaje de OK.	Cuestionario eliminado. Mensaje de error.	NO	
7.1			Cuestionario eliminado. Mensaje de OK.	SÍ	No se devolvía el código de respuesta correcto.
8	Eliminar un cuestionario no existente.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
9	Las pruebas número 1, 2, 3, 4, 5, 6, 7 y 8 pero sin haber iniciado sesión o con un token no válido.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
10	Las pruebas número 2, 3, 4, 5, 6, 7 y 8, pero sin ser administrador.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	

Cod.	Descripción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
11	Obtener el cuestionario activo.	Mensaje de OK y en el cuerpo de la respuesta viene el cuestionario activo.	Se ha recibido mensaje de OK y en el cuerpo de la respuesta viene el cuestionario activo.	SÍ	

Tabla 44 - Pruebas del bloque Answer

Cod.	Descripción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
1	Añadir respuestas de participación.	Respuestas añadidas. Mensaje de OK.	Respuestas no añadidas. Mensaje de error no controlado.	NO	
1.1			Las respuestas del cuestionario no se han almacenado correctamente.	NO	
1.2			Respuestas añadidas correctamente. Mensaje de OK.	SÍ	No se estaba controlando el orden de inserción de las respuestas. Además, no se estaba asociando el cuestionario correctamente.

Tabla 45 - Pruebas del bloque Suggestion

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Obtener todas las sugerencias.	Mensaje de OK y en el cuerpo de la respuesta viene la lista de las sugerencias.	Se ha recibido un mensaje de OK y en el cuerpo de la respuesta viene la lista de las sugerencias.	SÍ	
2	Aceptar una sugerencia.	PPR añadido. Sugerencia eliminada. Mensaje de OK.	Se ha añadido un PPR en el lugar que indicaba la sugerencia. Sin embargo, la sugerencia no se ha eliminado. Se ha obtenido un mensaje de OK.	NO	
2.1			Se ha añadido un PPR en el lugar que indicaba la sugerencia. Además, la sugerencia se ha eliminado. Se ha obtenido un mensaje de OK.	SÍ	No se estaba borrando de la tabla de sugerencias.
3	Rechazar una sugerencia.	Sugerencia eliminada. Mensaje de OK.	La sugerencia se ha eliminado. Se ha obtenido un mensaje de OK.	SÍ	
4	Posponer una sugerencia.	Sugerencia actualizada y pospuesta. Mensaje de OK.	La sugerencia se ha pospuesto. Se ha obtenido un mensaje de OK.	SÍ	
5	Las pruebas número 1, 2, 3, y 4, pero sin haber iniciado sesión o con un token no válido.	Mensaje de error.	Se ha recibido mensaje de error.	SÍ	
6	Las pruebas número 1, 2, 3 y 4, pero sin ser administrador.	Mensaje de error.	Se ha recibido mensaje de OK.	NO	
6.1			Se ha recibido mensaje de error.	SÍ	No se estaba controlando el nivel de permisos.
7	Proponer sugerencia.	Sugerencia añadida. Mensaje de OK.	Se ha añadido la sugerencia. Se ha recibido mensaje de OK.		

Tabla 46 - Pruebas del bloque User

Cod.	***Descripción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Un usuario se registra.	Usuario añadido. Mensaje de OK.	Se ha añadido el usuario. Se ha recibido mensaje de OK.	SÍ	

7.2. Pruebas de la aplicación web

Para la realización de estas pruebas, se han diseñado ocho conjuntos de pruebas (*Auth*, *Puntos*, *Preguntas*, *Cuestionarios*, *Sugerencias*, *Gestión de investigadores* y *Perfil de usuario*) que se presentan en las siguientes tablas (Tabla 47, Tabla 48, Tabla 49, Tabla 50, Tabla 51, Tabla 52 y Tabla 53), donde se detalla en qué consiste cada prueba, cuál es el resultado esperado de la misma, si el resultado obtenido era el esperado o no, y en algunos casos, algún comentario sobre la prueba realizada o sobre cómo se ha solucionado el error (en caso de haberlo).

Tabla 47 - Pruebas del conjunto Auth

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	El investigador intenta iniciar sesión con un nombre de usuario que no está registrado en el sistema.	El usuario introduce un nombre de usuario no existente y una contraseña existente.	Se muestra un aviso en la interfaz gráfica notificando de que ese usuario no está registrado en el sistema.	Al intentar iniciar sesión, se ha mostrado un aviso informando al usuario de que el usuario con el que intenta acceder, no existe.	SÍ	
2	El investigador intenta iniciar sesión con un nombre de usuario existente, pero con una contraseña incorrecta.	El usuario introduce un nombre de usuario existente y una contraseña incorrecta.	Se muestra un aviso en la interfaz gráfica notificando de que la contraseña no es correcta.	Al intentar iniciar sesión, se ha mostrado un aviso informando al usuario de que la contraseña no es correcta.	SÍ	
3	El investigador intenta iniciar sesión con un nombre de usuario y contraseña correctos.	El usuario introduce un nombre de usuario existente y una contraseña correcta.	Se muestra la interfaz gráfica principal de la aplicación.	Al iniciar sesión, la aplicación muestra la interfaz gráfica principal.	SÍ	
4	El investigador intenta iniciar sesión sin haber introducido el nombre de usuario ni contraseña.	El usuario no rellena ningún campo del formulario e intenta iniciar sesión.	Se muestra un aviso en la interfaz gráfica notificando de que los campos son obligatorios.	Al intentar iniciar sesión, se ha mostrado un aviso informando de que los campos son obligatorios.	SÍ	

Tabla 48 - Pruebas del conjunto Puntos

Cod.	Descripción	Acción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
1	Se visualizan los PPR sobre un mapa.	Ninguna.	Aparece un mapa en la interfaz gráfica y sobre este, aparecen los PPR (en verde los activos y rojo los desactivados).	No se visualiza ningún PPR sobre el mapa.	NO	
1.1				Se visualizan los PPR correctamente, con los colores adecuados.	SÍ	Las imágenes que representan los PPR no se estaban cargando bien y por ello, los PPR no se veían.
2	El investigador administrador desea añadir un PPR.	Pincha sobre el mapa.	Aparece un nuevo marcador sobre el mapa de color azul y un radio alrededor de este del mismo color. Además, aparecen las opciones de configuración del punto.	Al pinchar en el mapa, aparece el marcador azul y las opciones de configuración, pero no el radio.	NO	
2.1				Al pinchar en el mapa, aparece el marcador azul, el radio y las opciones de configuración.	SÍ	La creación de la circunferencia que dibujaba el radio no era la correcta.
3	El investigador administrador desea añadir un PPR sin introducir ningún dato.	Pincha sobre el mapa y no introduce ningún dato.	No aparece la opción de guardar.	No aparece el botón que permite guardar el PPR.	SÍ	
4	El investigador administrador desea ver los resultados y estadísticas de un PPR.	El investigador administrador pincha sobre un PPR ya definido.	Sobre el PPR aparece una ventana de información con los datos de hombres y mujeres que han participado en dicho punto. Además, aparece un botón que permite ver las respuestas y estadísticas del punto seleccionado.	No aparece información sobre el número de participaciones. Además, el botón no hace nada.	NO	

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
4.1				Aparece información sobre el número de participaciones, pero, el botón sigue sin hacer nada.	NO	El número de participantes no se estaba obteniendo correctamente.
4.2				Aparece información sobre el número de participaciones, y, además, el botón redirige a la vista que permite ver las respuestas y estadísticas del punto.	SÍ	Al estar el botón en un elemento del mapa, los eventos del botón no estaban siendo recogidos por el controlador del mismo.
5	El investigador administrador desea eliminar un PPR.	El investigador administrador pincha sobre un PPR ya definido y pulsa sobre el botón de borrar.	Aparecen las características del punto, y además un botón que permite borrar dicho punto. Al pulsar este botón, se pide confirmación al investigador y en caso de aceptar, el punto es eliminado informando al investigador. Además, el punto eliminado desaparece del mapa.	Las características del punto han aparecido, y al pulsar en el botón de eliminar, la ventana de confirmación ha aparecido. Al confirmar la acción, el punto ha sido eliminado correctamente, y se ha notificado al investigador. Además, el punto ha desaparecido del mapa.	SÍ / NO	El PPR se ha eliminado. Sin embargo, al cargar de nuevo los PPR sobre el mapa, estos han aparecido sobre los anteriores.
5.1					SÍ	Antes de cargar de nuevo los PPR, se limpia el mapa.
6	El investigador administrador desea editar un PPR.	El investigador administrador pincha sobre un PPR ya definido y pulsa sobre el botón de editar.	Aparecen las características del punto, y además un botón que permite editar dicho punto. Al pulsar en editar, los campos del punto no modificables aparecen deshabilitados. Además, en caso de producirse algún cambio, aparece un botón que permite guardar los cambios.	Las características del punto han aparecido, y al pulsar en el botón de editar, los campos se han convertido en editables (excepto los no modificables). El investigador ha modificado los datos y ha pulsado en guardar. La acción ha sido notificada al investigador.	SÍ	

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
7	El investigador administrador desea desactivar un PPR activo.	El investigador administrador pincha sobre un PPR ya definido y pulsa sobre el botón de editar. Después, desactiva el campo de las características que permite desactivar o activar un PPR.	Aparecen las características del punto, y además un botón que permite editar dicho punto. Al pulsar en editar, los campos del punto no modificables aparecen deshabilitados. Al pinchar sobre el campo que permite modificar el estado de los PPR, este se desactiva. Finalmente, el PPR ha sido desactivado y el mapa recarga todos los PPR de nuevo, mostrando el modificado como deshabilitado.	El PPR se ha deshabilitado, y la acción ha sido notificada al investigador. Además, se han cargado todos los PPR de nuevo sobre el mapa y el modificado aparece en estado deshabilitado (color rojo).	SÍ	
8	El investigador administrador desea activar un PPR desactivado.	El investigador administrador pincha sobre un PPR ya definido y pulsa sobre el botón de editar. Después, desactiva el campo de las características que permite desactivar o activar un PPR.	Aparecen las características del punto, y además un botón que permite editar dicho punto. Al pulsar en editar, los campos del punto no modificables aparecen deshabilitados. Al pinchar sobre el campo que permite modificar el estado de los PPR, este se desactiva. Finalmente, el PPR ha sido desactivado y el mapa recarga todos los PPR de nuevo, mostrando el modificado como deshabilitado.	El PPR se ha deshabilitado, y la acción ha sido notificada al investigador. Además, se han cargado todos los PPR de nuevo sobre el mapa y el modificado aparece en estado deshabilitado (color rojo).	SÍ	
9	El investigador no administrador desea añadir un PPR.	El investigador no administrador pincha sobre el mapa.	No ocurre nada.	No ha ocurrido nada.	SÍ	No tiene permiso.
10	El investigador no administrador desea editar un PPR.	El investigador no administrador pincha sobre un PPR del mapa.	Se muestran los datos del PPR seleccionado, pero no permite editarlos.	Se han mostrado los datos del PPR seleccionado, pero no permite editarlos.	SÍ	No tiene permiso.

Cod.	Descripción	Acción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
11	El investigador no administrador desea eliminar un PPR.	El investigador no administrador pincha sobre un PPR del mapa.	Se muestran los datos del PPR seleccionado, pero no permite eliminar el PPR.	Se han mostrado los datos del PPR seleccionado, pero no permite eliminar el PPR.	SÍ	No tiene permiso.

Tabla 49 - Pruebas del conjunto Preguntas

Cod.	Descripción	Acción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
1	Visualizar las preguntas.	Ninguna.	Se muestra una tabla con las preguntas almacenadas en el sistema.	Se ha mostrado una tabla, pero algunas columnas están vacías.	NO	
1.2				Se ha mostrado una tabla con las preguntas almacenadas en el sistema.	SÍ	Algunos campos de las preguntas no se correspondían con los campos definidos en la cabecera de la tabla.
2	El investigador administrador desea añadir una pregunta.	Introduce el texto de la pregunta en la caja de texto y pulsa en el botón de aceptar.	La pregunta es añadida en el sistema y, además, aparece en la tabla de preguntas.	La pregunta ha sido añadida al sistema y, además, ha aparecido en la tabla de preguntas.	SÍ	
3	El investigador administrador desea eliminar una pregunta.	Selecciona en la tabla la pregunta que desea editar y pulsa en el botón de eliminar.	Aparece una ventana pidiendo confirmación para eliminar la pregunta. Si el investigador confirma, la pregunta es eliminada del sistema y, además, desaparece de la tabla de preguntas.	Ha aparecido una ventana pidiendo confirmación de la acción. En investigador ha confirmado y la pregunta ha sido eliminada del sistema, pero no ha desaparecido de la tabla de preguntas.	NO	

Cod.	*Descripción*	*Acción*	*Resultado esperado*	*Resultado obtenido*	*¿OK?*	*Comentarios*
3.1				Ha aparecido una ventana pidiendo confirmación de la acción. En investigador ha confirmado y la pregunta ha sido eliminada del sistema y, además, ha desaparecido de la tabla de preguntas.	SÍ	El evento de refresco de la tabla cuando se elimina un elemento no se estaba realizando correctamente.
4	El investigador administrador desea editar una pregunta.	Selecciona en la tabla la pregunta que desea editar y pulsa en el botón de editar.	Aparece una ventana pidiendo el nuevo título de la pregunta. Al rellenar el campo y pulsar en guardar, la pregunta se actualiza en el sistema y la tabla de preguntas se actualiza también.	Ha aparecido una ventana pidiendo un nuevo título para la pregunta. El investigador ha introducido el texto y ha pulsado en guardar. La pregunta ha sido actualizada en el sistema y la tabla también se ha actualizado.	SÍ	
5	El investigador administrador desea editar o eliminar una pregunta que haya sido respondida por algún usuario o esta se encuentre asignada a algún cuestionario.	Selecciona de la tabla la pregunta que se desea editar o eliminar.	Aparecen las acciones de editar y eliminar deshabilitadas y se muestra un aviso indicando el motivo.	Han aparecido los botones de las acciones deshabilitados y, además, se ha mostrado un aviso indicando el motivo.	SÍ	
6	El investigador no administrador desea añadir una pregunta.	Introduce el texto de la pregunta en la caja de texto y pulsa en el botón de aceptar.	No aparece la caja de texto.	No ha aparecido la caja de texto.	SÍ	No tiene permiso.
7	El investigador no administrador desea eliminar una pregunta.	Selecciona de la tabla la pregunta que desea eliminar.	No aparece ninguna acción disponible.	No ha aparecido ninguna acción disponible.	SÍ	No tiene permiso.
8	El investigador no administrador desea editar una pregunta.	Selecciona de la tabla la pregunta que desea editar.	No aparece ninguna acción disponible.	No ha aparecido ninguna acción disponible.	SÍ	No tiene permiso.

Tabla 50 - Pruebas del conjunto Cuestionarios

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Visualizar los cuestionarios.	Ninguna.	Se muestra una tabla con los cuestionarios almacenados en el sistema. Además, por cada cuestionario, se muestra un icono verde en el caso que este se encuentre activo; y un icono rojo en caso contrario. Además, el cuestionario activo está seleccionado por defecto.	Se ha mostrado una tabla con los cuestionarios. El cuestionario activo está seleccionado por defecto. Sin embargo, los iconos no se han cargado correctamente.	NO	
1.2				Se ha mostrado una tabla con los cuestionarios. El cuestionario activo está seleccionado por defecto. Además, los iconos se han cargado correctamente.	SÍ	El sistema no era capaz de encontrar los iconos.
2	El investigador administrador desea activar un cuestionario.	Selecciona de la tabla el cuestionario que desea activar, y pulsa sobre el botón de activar.	El cuestionario seleccionado es activado y las fechas de activación y desactivación de este cambian. Además, el cuestionario que se encontraba activo, es desactivado y su fecha de desactivación cambia. Seguidamente, se muestra un aviso en la interfaz gráfica de la acción realizada. Finalmente, se selecciona automáticamente el cuestionario activo.	El cuestionario seleccionado se ha activado y las fechas de activación y desactivación de este han cambiado. Además, el cuestionario que se encontraba activo, se ha desactivado y su fecha de desactivación han cambiado. Seguidamente, se ha mostrado un aviso en la interfaz gráfica de la acción realizada. Sin embargo, no se ha seleccionado automáticamente el cuestionario activo.	NO	

Cod.	*Descripción*	*Acción*	*Resultado esperado*	*Resultado obtenido*	*¿OK?*	*Comentarios*
2.1					SÍ	Pese a actualizar la tabla, no se estaba controlando la detección del activo.
3	El investigador administrador desea desactivar un cuestionario.	Selecciona de la tabla el cuestionario que desea desactivar.	El botón de desactivar no se encuentra disponible.	El botón que permite desactivar el cuestionario no está habilitado.	SÍ	Siempre debe de haber un cuestionario activo. Por ello, no se puede desactivar un cuestionario, este será desactivado cuando otro se active.
4	El investigador administrador desea modificar el título del cuestionario.	Selecciona de la tabla el cuestionario que desea modificar. Introduce un nuevo título para el cuestionario y se aceptan los cambios.	El cuestionario es actualizado y se muestra un aviso indicando la acción realizada.	El cuestionario se ha actualizado y se ha mostrado un aviso indicando la acción realizada.	SÍ	
5	El investigador administrador desea crear un cuestionario.	Selecciona el cuestionario que quiera tomar como plantilla, añade o elimina alguna pregunta, y acepta los cambios.	Se crea un nuevo cuestionario con las preguntas que se hayan elegido.	No se ha creado el nuevo cuestionario. Ha dado un error por consola.	NO	
5.1				Se ha creado el cuestionario. Sin embargo, las preguntas no están en el orden correcto.	NO	La petición a la API REST no se estaba realizando correctamente.

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
5.2				Se ha creado el cuestionario y las preguntas están en el orden correcto.	SÍ	No se estaba manteniendo el orden de las preguntas al guardar el cuestionario.
6	El investigador administrador desea eliminar un cuestionario.	Selecciona de la tabla el cuestionario que desea eliminar y elige la opción de borrar.	Aparece un aviso de confirmación de la acción. Si el investigador acepta, el cuestionario es eliminado. Además, todas las respuestas asociadas a dicho cuestionario también son eliminadas. Finalmente, sale un aviso indicando la acción realizada.	El aviso de confirmación ha aparecido, y al aceptar, el cuestionario y sus respuestas asociadas se han eliminado. Finalmente, ha salido un aviso indicando la acción realizada.	SÍ	
7	El investigador no administrador desea editar un cuestionario.	Selecciona de la tabla el cuestionario que desea editar.	No se muestra ninguna acción que permita editar el cuestionario.	No se ha mostrado ninguna acción que permita editar el cuestionario.	SÍ	No tiene permiso.
8	El investigador no administrador desea eliminar un cuestionario.	Selecciona de la tabla el cuestionario que desea eliminar.	No se muestra ninguna acción que permita eliminar el cuestionario.	No se ha mostrado ninguna acción que permita eliminar el cuestionario.	SÍ	No tiene permiso.

Tabla 51 - Pruebas del conjunto Sugerencias

Cod.	Descripción	Acción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
1	Visualizar las sugerencias.	Ninguna.	Se muestra una lista de las sugerencias propuestas por los usuarios. Además, en cada sugerencia se detalla la fecha en la que ha sido realizada.	Se ha mostrado la lista de las sugerencias, pero no la fecha.	NO	
1.2				Se ha mostrado la lista de las sugerencias y en cada una de ella la fecha en la que ha sido realizada.	SÍ	No se estaba realizando bien la conversión de las fechas.
2	El investigador administrador desea ver los detalles de una sugerencia.	Pulsa sobre “Ver detalles” en la sugerencia que desea ver.	Se muestra un mapa indicando con un punto azul el punto geográfico de dicha sugerencia. Además, aparecen 3 botones para aceptarla, rechazarla o posponerla.	La sugerencia se muestra correctamente sobre el mapa, además, aparecen los botones para aceptarla, rechazarla o posponerla.	SÍ	
3	El investigador administrador desea aceptar una sugerencia.	Pulsa sobre “Ver detalles” en la sugerencia que desea aceptar, y posteriormente pulsa “Aceptar”. Rellena los campos requeridos y pulsa en “Guardar”.	La sugerencia se almacena como un PPR en el sistema. Además, la sugerencia desaparece de la lista de sugerencias y el mapa se limpia. Finalmente, se le notifica al investigador de la acción realizada.	Se ha añadido un nuevo PPR. El mapa se ha limpiado. Se ha mostrado el aviso al investigador. Sin embargo, la sugerencia no ha desaparecido de la lista de sugerencias.	NO	
3.1					SÍ	Faltaba refrescar la información de la pantalla
4	El investigador administrador desea rechazar una sugerencia.	Pulsa sobre “Ver detalles” en la sugerencia que desea eliminar, y posteriormente pulsa en “Rechazar”.	La sugerencia se elimina del sistema y de la lista de sugerencias.	La sugerencia se ha eliminado del sistema y de la lista de sugerencias.	SÍ	

Cod.	Descripción	Acción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
5	El investigador administrador desea posponer una sugerencia.	Pulsa sobre "Ver detalles" en la sugerencia que desea posponer, y posteriormente pulsa en "Posponer".	La sugerencia desaparece de lista de sugerencias. La sugerencia ahora aparece en la carpeta de sugerencias pospuestas.	La sugerencia ha desaparecido de la lista de sugerencias y se ha creado una carpeta, que pinchándola, muestra la sugerencia pospuesta.	SÍ	
6	El investigador administrador desea traer a la vista principal una sugerencia pospuesta.	Accede a la carpeta de sugerencias pospuestas. Selecciona la sugerencia y pulsa "Enviar al frente".	La sugerencia desaparece de la lista de sugerencias pospuestas y aparece en la lista de sugerencias.	La sugerencia ha desaparecido de la lista de sugerencias pospuestas y ha aparecido en la lista de sugerencias.	SÍ	
7	Icono de notificación de nuevas sugerencias.	Ninguna.	Aparece un icono rojo en el menú de la aplicación, indicando que hay nuevas sugerencias.	El icono rojo ha aparecido y realmente había sugerencias.	SÍ	

Tabla 52 - Pruebas del conjunto Gestión de investigadores

Cod.	Descripción	Acción	Resultado esperado	Resultado obtenido	¿OK?	Comentarios
1	Visualizar los miembros investigadores.	Ninguna.	Se muestran en tablas separadas los miembros investigadores dados de alta en el sistema. En una de las tablas se muestran los administradores y en la otra los no administradores.	Los miembros investigadores se muestran en dos tablas separadas, diferenciados por sus permisos.	SÍ	
2	Degradar los permisos de un investigador administrador.	Selecciona en la tabla el miembro investigador que se desea degradar y pulsa sobre la opción de degradar.	Los permisos del miembro investigador se han actualizado en el sistema. Además, el miembro investigador degradado desaparece de la tabla "Administradores", y aparece en la tabla "No Administradores".	Los permisos del investigador se han actualizado en el sistema y este ahora ha aparecido en la tabla de "No Administradores".	SÍ	

3	Ascender los permisos de un investigador no administrador.	Selecciona en la tabla el miembro investigador que se desea ascender y pulsa sobre la opción de ascender.	Los permisos del miembro investigador se han actualizado en el sistema. Además, el miembro investigador ascendido desaparece de la tabla "No Administradores", y aparece en la tabla "Administradores".	Los permisos del investigador se han actualizado en el sistema y este ahora ha aparecido en la tabla de "Administradores".	SÍ	
4	Añadir un miembro investigador.	Pulsa sobre el botón de añadir un nuevo investigador, rellena los campos requeridos y pulsa en "Guardar".	Se almacena el nuevo investigador en el sistema. Este aparece en la tabla correspondiente a su nivel de permisos.	Se ha añadido correctamente el nuevo investigador. Además, ha aparecido en la tabla correspondiente a su nivel de permisos.	SÍ	
5	Eliminar un miembro investigador.	Selecciona en la tabla el miembro investigador que se desea ascender y pulsa sobre la opción de eliminar.	El miembro se elimina del sistema y además desaparece de la tabla.	El miembro se ha eliminado del sistema y además ha desaparecido de la tabla.	SÍ	

Tabla 53 - Pruebas del conjunto Perfil de usuario

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Visualizar los datos del perfil.	Ninguna.	Se muestran los datos del perfil.	Los datos del perfil se han mostrado.	SÍ	
2	Editar perfil.	Pulsa sobre "Editar", modifica algún dato y pulsa "Guardar".	Los datos del miembro investigador se actualizan en el sistema.	Los datos del miembro investigador se han actualizado en el sistema y en la interfaz gráfica.	SÍ	

7.3. Pruebas de la aplicación móvil

A continuación, se presenta en la siguiente tabla (Tabla 54) las pruebas realizadas, donde se detalla en qué consiste cada prueba, cuál es el resultado esperado de la misma, si el resultado obtenido era el esperado o no, y en algunos casos, algún comentario sobre la prueba realizada o sobre cómo se ha solucionado el error (en caso de haberlo).

Tabla 54 - Pruebas de la aplicación móvil

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
1	Darse de alta rellenando todos los campos requeridos.	Rellena todos los campos y pulsa en “Aceptar”.	El usuario es registrado en el sistema y es redirigido a la pantalla principal.	El usuario se ha registrado en el sistema y se le redirige a la pantalla principal.	SÍ	
2	Darse de alta sin rellenar todos los campos requeridos.	No rellena todos los campos y pulsa en “Aceptar”.	El usuario no es registrado en el sistema y se muestra un aviso informándole que debe rellenar todos los campos.	El usuario se ha registrado en el sistema y se le redirige a la pantalla principal.	SÍ	
3	Visualizar los PPR activos.	Ninguna.	Se muestra sobre el mapa con forma de marcador verde los PPR activos. Además, se muestra también de color azul la ubicación del usuario.	No se muestra ningún PPR sobre el mapa. Sin embargo, se muestra la ubicación del usuario.	NO	
3.1					SÍ	No se estaba realizando la petición correcta a la API REST.
4	Creación de la tarea programada encargada de lanzar el servicio en segundo plano.	Entrar a la aplicación.	La tarea se ha creado y se ha programado. Además, cada cierto tiempo lanza el servicio en segundo plano.	La tarea se ha creado, pero no se ha programado, ni se ejecuta.	NO	
4.1				La tarea se ha creado, y además se ha programado. Sin embargo, no lanza el servicio en segundo plano.	NO	No se estaba configurando el programador de tareas de la manera correcta.
4.2				La tarea se ha creado, y además se ha programado. Intenta lanzar el servicio en segundo plano, pero surge un error.	NO	No se estaba teniendo en cuenta la nueva forma de lanzar los servicios en Android 8.0.
4.3				La tarea se ha creado y programado. Además, lanza el servicio en segundo plano.	SÍ	Surgía un error no controlado.

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
5	Detección de la ubicación a través del servicio en segundo plano.	Lanzar el servicio en segundo plano.	Se obtiene periódicamente la ubicación del dispositivo.	Se obtiene una única vez la ubicación.	NO	
5.1				Se obtiene periódicamente la ubicación del dispositivo.	SÍ	No se estaba aprovechando la ubicación obtenida por otras aplicaciones para aumentar la frecuencia de comprobación.
6	Detección de PPR cercanos.	Acercarse a un PPR activo.	Se detecta que el usuario se encuentra cerca del PPR.	Se ha detectado que el usuario se encuentra cerca del PPR.	SÍ	
7	Notificación de que el usuario está entrando en un PPR.	Entrar en el área de un PPR activo.	Se lanza una notificación indicándole al usuario que se encuentra entrando de un PPR. En la notificación aparecen los detalles del PPR y la opción de participar o silenciarlo.	No se ha lanzado la notificación.	NO	
7.1				Se ha lanzado la notificación, indicando los detalles del PPR y aparecen las opciones de participar o silenciar el PPR.	SÍ	No se había creado el canal a través del cual se lanza la notificación en Android 8.0.

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
8	Realizar la prueba del juego de atención.	Decidir participar en un PPR.	Se muestran los símbolos de la prueba, de manera aleatoria junto con un número. Además, aparecen 9 botones, numerados del 1 al 9. Cuando el usuario pulsa en un botón, el tiempo empieza a ir cuenta atrás durante 90 segundos. Los símbolos van cambiando al completarse. Una vez finalizado el tiempo, se muestran los resultados en una ventana de información.	Se muestran los símbolos de la prueba, de manera aleatoria junto con un número. Además, aparecen 9 botones, numerados del 1 al 9. Cuando el usuario pulsa en un botón, el tiempo empieza a ir cuenta atrás durante 90 segundos. Los símbolos van cambiando al completarse. Una vez finalizado el tiempo, no se muestran los resultados y se detiene la aplicación.	NO	
8.1				Se muestran los símbolos de la prueba, de manera aleatoria junto con un número. Además, aparecen 9 botones, numerados del 1 al 9. Cuando el usuario pulsa en un botón, el tiempo empieza a ir cuenta atrás durante 90 segundos. Los símbolos van cambiando al completarse. Una vez finalizado el tiempo, se muestran los resultados en una ventana de información.	SÍ	
9	Guardar resultados de la prueba del juego.	Finalizar la prueba del juego.	Los resultados son almacenados en la base de datos.	Los resultados han sido almacenados en la base de datos.	SÍ	
10	Realizar la prueba del termómetro de estrés.	Finalizar la prueba del juego.	Aparece un termómetro vertical. Se puede seleccionar un valor para indicar el nivel de estrés.	Aparece un termómetro vertical. Se puede seleccionar un valor para indicar el nivel de estrés.	SÍ	
11	Guardar resultados del termómetro de estrés.	Finalizar la prueba del termómetro de estrés.	Los resultados son almacenados en la base de datos.	Los resultados han sido almacenados en la base de datos.	SÍ	

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
12	Realizar la prueba del termómetro de felicidad.	Finalizar la prueba del termómetro de estrés.	Aparece un termómetro vertical. Se puede seleccionar un valor para indicar el nivel de felicidad.	Aparece un termómetro vertical. Se puede seleccionar un valor para indicar el nivel de felicidad.	SÍ	
13	Guardar resultados del termómetro de felicidad.	Finalizar la prueba del termómetro de felicidad.	Los resultados son almacenados en la base de datos.	Los resultados han sido almacenados en la base de datos.	SÍ	
14	Realizar el cuestionario.	Finalizar la prueba del termómetro de felicidad y estar saliendo de un PPR.	Aparecen las preguntas que deben ser contestadas. Además, aparecen 5 botones con los que contestar a las preguntas. Al contestar una pregunta, aparece la siguiente pregunta.	Aparecen las preguntas que deben ser contestadas. Además, aparecen 5 botones con los que contestar a las preguntas. Al contestar una pregunta, no aparece la siguiente pregunta.	NO	
14.1				Aparecen las preguntas que deben ser contestadas. Además, aparecen 5 botones con los que contestar a las preguntas. Al contestar una pregunta, aparece la siguiente pregunta, pero sigue la respuesta anterior marcada.	NO	
14.2				Aparecen las preguntas que deben ser contestadas. Además, aparecen 5 botones con los que contestar a las preguntas. Al contestar una pregunta, aparece la siguiente pregunta.	SÍ	
15	Enviar los resultados al sistema haciendo uso de la API REST.	Finalizar el cuestionario final.	Los resultados obtenidos en el PPR son enviados al sistema por medio de una petición a la API REST.	Los resultados obtenidos en el PPR se han enviado al sistema por medio de una petición a la API REST.	SÍ	
16	Limpiar los resultados almacenados al salir de un PPR.	Salir de un PPR donde se estuviera participando.	Los resultados obtenidos en el PPR se eliminan de la base de datos del móvil.	Los resultados obtenidos en el PPR se han eliminado de la base de datos del móvil.	SÍ	

Cod.	***Descripción***	***Acción***	***Resultado esperado***	***Resultado obtenido***	***¿OK?***	***Comentarios***
17	Enviar una sugerencia de un PPR.	Rellenar los campos necesarios para enviar una sugerencia y pulsar “Enviar”.	La sugerencia se envía y almacena en el sistema.	La sugerencia se ha enviado y se ha almacenado en el sistema.	SÍ	
18	Silenciar un PPR.	Entrar en el área de un PPR activo y sobre la notificación pulsar en “Silenciar”.	El PPR se silencia y este no vuelve a avisar cuando el usuario entre en su área.	El PPR se ha silenciado, y no avisa al entrar de nuevo en su área.	SÍ	
19	Activar sonido de un PPR.	En la pantalla de gestión de PPR silenciados, desplazar hacia la derecha o izquierda.	El PPR desaparece de la lista de PPR silenciados y este vuelve a notificar cuando el usuario entre en su área.	Se ha detenido la aplicación.	NO	
19.1				El PPR ha desaparecido de la lista de PPR silenciados y este vuelve a notificar cuando el usuario entre en su área.	SÍ	
20	El servicio en segundo plano se detiene automáticamente.	Ninguna.	El servicio en segundo plano se detiene automáticamente cuando el usuario no se encuentra cerca de ningún PPR activo.	El servicio en segundo plano se ha detenido automáticamente cuando el usuario no se encuentra cerca de ningún PPR activo.	SÍ	

8. CONCLUSIONES Y TRABAJO FUTURO

Este capítulo pretende recoger las conclusiones (tanto técnicas como personales) obtenidas tras realizar este proyecto, así como mencionar las posibles líneas de futuro que existen para *Restorative City*.

8.1. Objetivos cumplidos

Al comienzo de esta memoria, en el apartado de objetivos (ver apartado 2.1), se listaban aquellos objetivos que se pretendía cumplir, tanto a nivel de requerimiento por parte del cliente como a nivel personal, con la realización de este Trabajo de Fin de Grado.

A continuación, se presentan de nuevo los objetivos establecidos, argumentando por qué se consideran cumplidos:

Objetivo 1

"Realización de una aplicación web que permita a los integrantes del grupo CRIM-AP definir qué puntos urbanos desean examinar para comprobar si son restauradores o no, y analizar los resultados obtenidos".

Prueba del cumplimiento de este objetivo es la aplicación web resultante. Dicha web, cumple con todos los requerimientos importantes que mostró el cliente al comienzo del proyecto, además de algunos de los que han surgido durante el mismo.

En la aplicación web, los miembros investigadores requerían poder definir puntos urbanos cuya capacidad de restauración deseen examinar. Para ello, se ha diseñado e implementado una interfaz gráfica que permite a los miembros investigadores, a través de un mapa, gestionar (añadir, activar, desactivar, editar, eliminar, etc.) los puntos geográficos que deseen examinar o realizar el estudio.

Otro de los aspectos importantes que el cliente requería era poder analizar los resultados obtenidos en cada punto examinado. Para ello, se dispone en la aplicación de una interfaz gráfica donde los miembros investigadores pueden visualizar los resultados de las pruebas realizadas por cada usuario en un PPR concreto. Estos resultados, son acompañados además por unas pequeñas estadísticas generales. Para la facilitación de la interpretación de dichos resultados, se han implementado unos gráficos, que aportan información útil a los investigadores de una manera mucho más visual y clara. Además, para facilitar el trabajo de análisis de los investigadores, se ha implementado un filtro a través del cual pueden filtrar los resultados obtenidos, para así poder realizar un estudio más focalizado en aquellos aspectos que deseen. Finalmente, para acabar con este requerimiento, se ha proporcionado una opción de descarga de datos, a través de la cual los miembros investigadores podrán descargar los resultados para así poder tratarlos más a fondo con herramientas de análisis de datos.

Por otro lado, para realizar los exámenes a los usuarios que participen en el estudio de cada PPR, los miembros investigadores del grupo CRIM-AP requerían una interfaz gráfica donde pudieran gestionar el cuestionario que deben rellenar los usuarios al abandonar el PPR. Un cuestionario, está compuesto de preguntas, por ello, se necesitaba además otra interfaz gráfica que permitiera a los miembros investigadores gestionar las preguntas que luego podrían formar parte de los cuestionarios. Estas dos interfaces gráficas mencionadas han sido implementadas. Además, en la interfaz gráfica de la gestión de cuestionarios, se ha incorporado información que permite a los investigadores llevar un control de los cuestionarios de manera más organizada, como por ejemplo, se indica las fechas de activación y desactivación de cada cuestionario.

Además, se ha desarrollado una interfaz gráfica que permite a los miembros investigadores gestionar las sugerencias de PPR que envían los usuarios desde la aplicación móvil. Pese a no ser este un requerimiento imprescindible por parte del cliente, desde el primer momento del proyecto se ha tratado como tal, puesto que esta funcionalidad aporta gran valor a la aplicación y facilita la labor de los investigadores.

Por último, se requería una gestión de permisos de los investigadores, diferenciando entre investigadores administradores y no administradores. Para ello, se ha desarrollado un sistema de control de permisos de usuario, que permite o no a ciertos investigadores a realizar diferentes acciones dentro de la aplicación. Además, se ha implementado una interfaz gráfica que permite gestionar los usuarios de los investigadores y realizar cambios de permisos. Asimismo, cada investigador dispone de un apartado de modificación del perfil donde puede editar sus datos.

Cabe destacar que, puesto que la intención de este proyecto es usarlo en un entorno real, la aplicación es multiidioma.

Para finalizar con este objetivo, cabe destacar que la aplicación web ya ha sido mostrada y probada por el cliente, y este ha transmitido su aceptación y conformidad con todos los aspectos anteriormente mencionados. Además, la aplicación ha sido presentada por el cliente, en el principal congreso internacional sobre interacción persona-ambiente[25], celebrado en el mes de julio de este año 2018 en Roma, y ha transmitido que la acogida de la aplicación web de *Restorative City*, y del proyecto en general, entre los asistentes ha sido muy buena.

El cliente afirma que, desde su perspectiva como desde la de otros equipos de investigación de otros países que ya han mostrado su interés en utilizar *Restorative City*, esta aplicación les permitirá avanzar en la comprensión de las dinámicas de restauración en la vida diaria. Además, les ayudará a detectar características clave de los lugares más restauradores, que puedan servir de guía para el diseño de barrios y ciudades que promuevan el bienestar de sus habitantes.

[25] http://iaps2018.com/

Objetivo 2

"Realización de una aplicación móvil que facilite al grupo CRIM-AP la recogida de datos sobre las personas en los puntos urbanos que hayan definido. Además, dichos datos deben permitir a las personas medir su nivel de estrés y la capacidad de atención dirigida y comprobar qué lugares les ayudan a reducir el estrés y mejorar su atención".

Prueba del cumplimiento de este objetivo es la aplicación móvil resultante. Dicha aplicación móvil, cumple con todos los requerimientos importantes que mostró el cliente al comienzo del proyecto, además de algunos adicionales que han surgido durante el mismo.

Para la aplicación móvil, los investigadores necesitaban que permitiera la recogida de datos sobre cada usuario en aquellos PPR en los que participase. Para ello, se ha desarrollado una interfaz gráfica que permite al usuario visualizar los PPR activos sobre un mapa y su propia ubicación. Cuando un usuario está entrando en el área de un PPR, la aplicación móvil muestra de manera automática un aviso, preguntándole si desea participar en el estudio de dicho PPR, o no. En caso afirmativo, a través de la aplicación móvil se realizan unas pruebas para medir el nivel de estrés y la capacidad de atención dirigida del usuario en ese momento. Cuando el usuario decide salir del área del PPR, la aplicación, de nuevo de manera automática, pregunta si desea completar las pruebas finales, y en caso afirmativo, estas se realizan de la misma manera que las anteriores. Finalmente, los resultados de ese usuario en ese PPR, son almacenados en el sistema.

Con la intención de mejorar la experiencia de usuario y optimizar el consumo de batería y memoria de su dispositivo, se han incorporado servicios en segundo plano en la aplicación. Con el uso de estos servicios, se ha conseguido avisar de manera automática al usuario cuándo se encuentra cerca de un PPR y preguntarle si desea participar. Asimismo, se detecta de la misma manera cuándo está abandonando un PPR y se le propone finalizar su participación en el estudio. Esta implementación, evita que el usuario tenga que estar obligatoriamente interactuando en la aplicación para poder detectar cuándo entra o sale de un PPR, lo cual aporta valor a la aplicación y se ajusta muchísimo más a lo que el cliente quería.

Otro de los puntos importantes para el cliente es que los usuarios pudieran realizar las pruebas necesarias para evaluar el poder restaurador de un punto en la propia aplicación y que, además, pudieran ver sus resultados. Por ello, todas las pruebas que el cliente ha especificado se han incorporado en la aplicación, siguiendo las especificaciones dadas para cada prueba.

Además, la aplicación incorpora una interfaz gráfica a través de la cual cada usuario puede configurar qué PPR desea que se le notifique y cuál no. Pese a no ser este un requerimiento imprescindible por parte del cliente, desde el primer momento del proyecto se ha tratado como tal, puesto que esta funcionalidad aporta gran valor a la aplicación respecto al usuario, ya que le otorga cierto control sobre esta y se evita, por ejemplo, que se le esté notificando continuamente si reside cerca de uno de los puntos a estudiar.

Por último, puesto que la intención de este proyecto es usarlo en un entorno real, la aplicación es multiidioma.

Para finalizar, cabe decir que la aplicación móvil, al igual que se ha comentado anteriormente con la aplicación web, también ha sido presentada en el mismo congreso en Roma, y el cliente ha transmitido su conformidad con la aplicación. Además, ha declarado que la aceptación que la aplicación móvil de *Restorative City* ha tenido ha sido muy buena entre los asistentes.

Objetivo 3

"Aprendizaje de nuevas tecnologías para cada desarrollo, tanto para las partes front-end como back-end".

Hasta la realización de este proyecto, nunca se habían realizado dos aplicaciones para un uso tan real y que requirieran cumplir unas funcionalidades tan completas, siendo algunas de ellas requerimientos directos de un cliente real.

La realización de este Trabajo de Fin de Grado, ha requerido aprendizaje de nuevas tecnologías para los desarrollos como, por ejemplo, Angular para la aplicación web, o NodeJS para la API REST.

El resultado obtenido por parte de ambas aplicaciones (web y móvil), puede considerarse la prueba del cumplimiento de este objetivo.

Objetivo 4

"Estudio y análisis de las arquitecturas y modelos de trabajo que se utilizan actualmente para cada desarrollo, y aplicarlos".

Con la realización de este proyecto, es la primera vez que se ha afrontado la creación, empezando de cero, de tres desarrollos distintos (API REST, web y móvil), que como resultado final, deben trabajar conjuntamente. Por ello, para la implementación de cada uno de los desarrollos, se ha estudiado y analizado cada tipo de arquitectura que podría usarse, y finalmente se ha escogido la que se ha considerado más oportuna para este proyecto.

Por ello, puede considerarse también que este objetivo ha sido cumplido.

8.2. Planificación final

En este apartado se presenta la planificación final resultante de este proyecto. Cabe decir que pese a haber conseguido los objetivos propuestos para este Trabajo Fin de Grado, no se ha conseguido ajustar a la planificación inicial. Por ello, a continuación se presenta un gráfico (Ilustración 94) que refleja la variación temporal que ha sufrido este proyecto en cada uno de sus paquetes de trabajo.

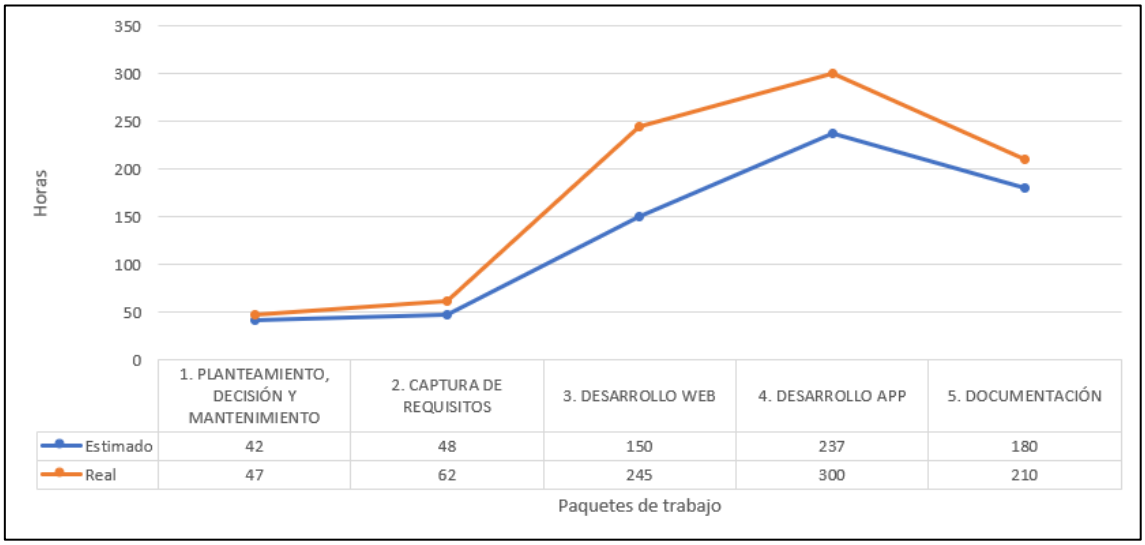


Ilustración 94 – Diferencias en la planificación

Como puede verse en la ilustración anterior, en todos los paquetes de trabajo se ha producido un desfase temporal respecto a la estimación realizada para cada uno de ellos.

La planificación temporal inicial estimaba un total de 657 horas de trabajo para este proyecto. Sin embargo, la estimación real indica que el número de horas de trabajo dedicadas a la realización de este proyecto han sido 864. Por lo tanto, el desfase ha sido de 207 horas, esto es, casi un incremento del 32 % respecto a la planificación inicial.

Sin duda, los paquetes que más variación han sufrido han sido los paquetes 3 y 4, que se corresponden a los desarrollos de la aplicación web y móvil (y sus correspondientes llamadas en la API REST).

Principalmente, los mayores retrasos han venido derivados de problemas propios del desarrollo de cualquier aplicación como, por ejemplo, la búsqueda de información para implementar una funcionalidad concreta, la resolución de problemas no esperados o no identificados, cambios en las funcionalidades, etc.

Además, el desconocimiento previo de las tecnologías y arquitecturas que se habían elegido utilizar para este proyecto, ha conllevado un periodo más largo de aprendizaje y mejora continua del que se había estimado inicialmente.

Adicionalmente, durante el desarrollo de las aplicaciones, se ha intentado dotar a estas de un aspecto profesional, como por ejemplo, que la aplicación se adapte a cada tipo de pantalla. Este aspecto también aumentó ligeramente el número de horas requeridas.

Por último, las funcionalidades nuevas que fueron surgiendo durante el desarrollo, como por ejemplo el filtro en la aplicación web, han supuesto un ligero retraso en la planificación.

En resumen, la suma de cada uno de los obstáculos encontrados durante el desarrollo de *Restorative City*, han resultado el motivo del desfase temporal resultante.

8.3. Evaluación económica final

Como se ha detallado en el apartado anterior, el desarrollo del proyecto ha durado más de lo estimado inicialmente. A consecuencia de ello, cabe rehacer la evaluación económica para ver los gastos reales del proyecto.

Empleando los mismos razonamientos que en el apartado de evaluación económica del capítulo de planteamiento inicial (ver apartado 2.8), se presenta a continuación la tabla final de gastos del proyecto (Tabla 55).

Tabla 55 - Gastos finales del proyecto

Gastos finales del proyecto	
Descripción	***Coste***
Coste de sueldo	10.120 €
Coste de licencias	231,18 €
Coste de hardware	180 €
Gastos indirectos	526,56 €
Coste total:	***11.057,74 €***

En conclusión, el proyecto supondría un coste de 2.656,56 € por encima de lo estimado inicialmente.

Como se ha comentado en el apartado de la evaluación económica inicial (ver apartado 2.8) dado que este proyecto está desarrollado para un bien social, y por tanto no lucrativo, y para dar soporte al grupo de investigación CRIM-AP, no aplica hablar sobre beneficios económicos.

Sin embargo, cambiando la temática del proyecto, y enfocándolo de manera comercial, podrían realizarse algunos cambios en el proyecto y convertirlo en un juego. El juego podría consistir en que los usuarios deben acudir a ciertos puntos urbanos para completar allí una serie de retos, pruebas, adivinanzas e incluso que deban interactuar con el entorno de dicho punto. Este juego podría tener su parte web para los administradores del mismo, que se encargarían de definir las pruebas a realizar en cada punto; y la parte móvil, para los usuarios que quieran disfrutar del juego.

De esta manera, cambiando el enfoque del proyecto y convirtiéndolo en un producto con afán de lucro, podrían conseguirse beneficios.

8.4. Líneas futuras

Como en todo proyecto, no todo se ha realizado exactamente de la manera en la que se desearía. Por ello, en este apartado se mencionarán aquellas partes que podrían mejorarse o incluirse en una siguiente versión de *Restorative City*, de manera que este cada vez fuese más completo y útil.

1. Mientras un usuario está participando en un PPR, la aplicación móvil, cada cierto tiempo, está recogiendo la ruta que dicho usuario realiza durante el tiempo que pasa dentro del punto. Dicha ruta podría ser útil para los miembros investigadores para, por ejemplo, descartar aquellas respuestas de los usuarios que hayan estado quietos durante todo el tiempo. O en aquellos PPR que abarquen una gran extensión, poder analizar los resultados en función de las zonas por donde se han movido los usuarios. Sin embargo, pese a que *Restorative City* actualmente almacena dichas rutas, estas no son mostradas a los miembros investigadores. Sería interesante dibujar en un mapa, por cada respuesta, la ruta realizada, y que el investigador pudiera analizarla.

2. Una idea que surgió al comienzo del proyecto y que se tuvo en cuenta, fue la de poder añadir fotografías a los PPR, tanto en la web por parte de los investigadores, así como también por parte de los usuarios al sugerir nuevos puntos desde la aplicación móvil. Sin embargo, por falta de tiempo y ya que este no era un aspecto fundamental para el proyecto, solo se ha podido dejar el sistema preparado para almacenar dichas fotografías asociadas a cada PPR. Por lo tanto, lo que faltaría para completar esta funcionalidad, sería habilitar la subida y visualización de las fotografías en ambos *front-end* (web y móvil).

3. En la aplicación web, se ha comentado anteriormente que se ha implementado un filtro, para que los investigadores puedan hacer uso de este, y focalizar en algunos aspectos de las respuestas obtenidas de cada PPR. Sin embargo, en ese filtro no se ha proporcionado la opción de filtrar por municipio, esto es, que eligiendo un municipio, únicamente muestre las respuestas de los usuarios que pertenezcan al municipio escogido. Por ello, podría ser interesante para el cliente poder filtrar por municipios para realizar estudios más concretos en un mismo PPR. Por ejemplo, le podría dar una especial importancia a las respuestas que pertenezcan al municipio donde se ha definido el PPR.

4. Como se ha comentado anteriormente, se han empleado gráficos para facilitar a los investigadores la visualización y el análisis de las respuestas obtenidas en cada PPR. Sin embargo, podría ser interesante analizar qué tipos de gráficos adicionales resultarían útiles para permitir a los investigadores realizar un análisis visual más completo desde la propia aplicación web de *Restorative City*.

5. Cuando un usuario se registra en la aplicación móvil, uno de los datos que se le pide es el municipio. En este proyecto, el municipio se obtiene a través de la geolocalización del dispositivo y se le muestra al usuario para que, si desea, pueda cambiarlo. Esta manera de actuar, puede provocar que los municipios se introduzcan de manera errónea o incluso siendo el mismo municipio, este sea escrito de dos maneras distintas. Con lo cual, este aspecto podría influir sobre el filtro de municipios que se mencionaba en el punto anterior. Por ello, una mejora podría ser añadir a la aplicación móvil una librería de municipios donde el usuario tuviera que elegir de una lista su municipio. De esa manera, se evitarían los errores de escritura al introducir el municipio.

6. Una de las motivaciones a la hora de elegir este proyecto era el pensamiento de poder realizar una posible ampliación de *Restorative City* integrándolo con otros sistemas de temáticas similares. En el caso de la aplicación web de este proyecto, esta podría integrarse con cierta facilidad con el proyecto *Walkability Analyzer*, ya que la tecnología y la arquitectura empleada para ambos desarrollos es similar. De la misma manera, la aplicación móvil, podría integrarse con el proyecto *Walkability Capturer*, aunque dicha integración posiblemente requeriría más trabajo puesto que la tecnología empleada es algo diferente. Finalmente, como resultado de estas integraciones, surgiría una aplicación mucho más completa, útil y beneficiosa, tanto para los miembros investigadores, como para la ciudadanía en general.

Para concluir, teniendo en cuenta el tiempo que conllevaría aplicar cada una de estas mejoras y el beneficio que aportaría cada una de ellas al proyecto, en mi opinión, se debería comenzar por la primera y la segunda, ya que son las más sencillas de integrar con la estructura interna actual del sistema. Después, realizaría la quinta, y posteriormente, la tercera, ya que considero que para lograr un buen funcionamiento del filtro, la funcionalidad quinta debe ser cumplida. Posteriormente, realizaría las integraciones, primero con *Walkability Analyzer*, ya que la tecnología usada es la misma y su integración no sería costosa y sí muy beneficiosa para los miembros investigadores; y posteriormente con *Walkability Capturer*, que conllevaría más trabajo puesto que las tecnologías usadas no son las mismas. Finalmente, incorporaría más gráficos.

8.5. Reflexión personal

Una vez visto el trabajo terminado y funcionando, solo puedo sentir felicidad por haber conseguido acabarlo, y mucha ilusión porque este se vaya a usar para un beneficio social común y pueda llegar a mejorar los espacios de nuestras ciudades, pueblos y municipios con el trabajo imprescindible, sin duda, de los miembros investigadores del grupo CRIM-AP.

Con la realización de este Trabajo Fin de Grado, siento que he aprendido mucho, no solo a nivel técnico y profesional, sino también a nivel personal, ya que hasta ahora, no me había enfrentado a un proyecto de esta magnitud, tanto en tamaño como en importancia (al menos para mí).

En cuanto a la gestión del tiempo, este proyecto me ha hecho reflexionar y darme cuenta de que quizá mis estimaciones eran algo optimistas. Sin duda, me ha hecho ver mis errores en cuanto a la organización del tiempo y me ha enseñado a realizar una mejor planificación.

En lo referente a los desarrollos realizados, los tres, han supuesto un reto para mí puesto que, de la API REST y de la aplicación web, no conocía en absoluto la tecnología, y de la aplicación móvil conocía algunos aspectos (puesto que ya había trabajado anteriormente con Android), pero no al nivel que este proyecto requería.

Nunca antes había realizado una API REST y, es más, apenas sabía realmente cómo funcionaba y cuáles eran sus características a nivel teórico siquiera. Con lo cual, considero que el haber desarrollado una API REST desde cero me ha beneficiado enormemente para mi futuro profesional. Es cierto que, al partir de una base prácticamente nula en este aspecto, al principio daba un poco de miedo enfrentarse a ello por lo que pudiera pasar. Sin embargo, viéndolo ahora, pienso que no ha sido tan complicado, ni de entender, ni de desarrollar. Además, las tecnologías empleadas para su implementación, han sido totalmente nuevas para mí y me han sorprendido gratamente, puesto que han resultado ser considerablemente sencillas de usar y de aprender. Añadir, que, ahora que entiendo lo que es una API REST y su arquitectura, me doy cuenta de la importancia que esta tiene hoy en día en el desarrollo de aplicaciones, además de haberme aportado una visión más amplia de las alternativas que existen.

Con la realización de la API REST, además, he mejorado mis habilidades con el lenguaje de programación *JavaScript*.

El desarrollo de la página web me ha resultado ser muy entretenido y, de nuevo, muy beneficioso para mi futuro puesto que, usando Angular, me he dado cuenta de la importancia y la relevancia que tiene hoy en día en el desarrollo de páginas web. El desarrollo de la página web ha estado lleno de cambios hasta lograr finalmente estar a gusto con su estructura y funcionamiento. Al no conocer Angular, al principio me resultó algo complejo situarme y saber qué estaba bien hacer y qué no, puesto que en Internet aparecían muchas maneras de hacer la misma tarea (como suele ser habitual en la programación). Sin embargo, tras muchas horas de lectura y a base de pruebas, conseguí tener cierta habilidad con Angular, y eso me permitió ser autocrítico con el desarrollo que estaba realizando. Por ello, considero que este desarrollo ha estado en un proceso de mejora continua hasta lograr el objetivo deseado: hacer la web de la mejor manera posible y siguiendo los consejos que Angular proporciona. Sin embargo, esto afectó a la planificación temporal que tenía preparada.

Además, con este desarrollo he podido mejorar notablemente mis habilidades con *HTML* y *CSS*, así como también aprender un lenguaje de programación nuevo (para mí): *TypeScript*.

Finalmente, realizando la aplicación móvil, he podido reforzar los conocimientos básicos que disponía de Android, además de haber aprendido muchísimos conceptos nuevos y nuevas formas de trabajar de algunos componentes de las aplicaciones Android. Por su puesto, me ha aportado también la puesta al día en las novedades que este sistema operativo incluye en sus nuevas APIs, funcionalidades, etc. En este desarrollo en especial, he tenido momentos de gran preocupación por los cambios de comportamiento que Google ha incorporado a Android 8.0 Oreo en cuanto a servicios en segundo plano y limitaciones de estos. Sin embargo, estoy satisfecho con las soluciones empleadas y he aprendido mucho de ello para futuros proyectos.

Durante el tiempo que ha durado el proyecto, ha habido momentos muy buenos por ver cómo este iba creciendo, pero también momentos de agobio y estrés cuando surgían los problemas. Sin embargo, considero que esta experiencia me ha aportado fortaleza a la hora de enfrentarme a nuevos retos y problemas.

Hasta antes de realizar este proyecto, no era muy "amigo" de las librerías externas, ya que no estaba familiarizado con su instalación ni manipulación. Sin embargo, el haber trabajado con tantas y tan diferentes en este proyecto, me ha aportado confianza en el uso de estas, además de hacerme ver de primera mano lo útiles que pueden llegar a ser.

Por último, me siento realmente feliz por saber que el cliente está satisfecho y contento con el trabajo realizado y, además, por saber que este ha presentado *Restorative City* en el extranjero como muestra de su satisfacción, y que la aceptación ha sido muy buena, llegando incluso a llamar la atención de otros grupos de investigación de diferentes países. Siento que este ha sido el grano de arena que he podido aportar al interesante tema de los espacios restauradores, pero espero que *Restorative City* siga creciendo mucho más y se convierta en una herramienta más completa y eficaz para los miembros investigadores, y en consecuencia, para la sociedad.

Antes de despedirme, quisiera dar las gracias a mis Directores de proyecto, Mikel Villamañe y Maider Azanza, por haber confiado en mí para la realización de este proyecto, y sobre todo, por toda la gran ayuda prestada durante el desarrollo del mismo. Además, me gustaría agradecer también al grupo de investigación en criminología aplicada, CRIM-AP, de la UPV/EHU *(Universidad del País Vasco/Euskal Herriko Unibertsitatea)* por todo el interés mostrado en el proyecto y el apoyo recibido por su parte. Finalmente, también quisiera agradecer a mi familia y amigos que me han acompañado durante estos meses y me han demostrado su apoyo.

Espero que esta memoria haya resultado clara e interesante. Muchas gracias.

9. BIBLIOGRAFÍA

Alligator. (2018). *Alligator*. Obtenido de https://alligator.io/js/introduction-localstorage-sessionstorage/

Android Developers. (2018). *Android Developers*. Recuperado el 2018, de https://developer.android.com/guide/components/services?hl=es-419

Android Developers. (2018). *Android Developers*. Recuperado el 2018, de https://developer.android.com/guide/components/activities?hl=es-419

Android Developers. (2018). *Android Developers*. Recuperado el 2018, de https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419

Harting, T. (2004). Restorative Environments. (C. Spielberger, Ed.) *Encyclopedia of applied psychology, Vol. 3*, 273-279.

Hinton-Bayre, Anton, Geffen, & Gina. (2005). Comparability, Reliability, and Practice Effects on Alternate Forms of the Digit Symbol Substitution and Symbol Digit Modalities Tests. *APAC PsycNET*.

Jenson, J. (2017). *tandemseven*. Recuperado el 2018, de https://www.tandemseven.com/blog/performance-java-vs-node/

Kaplan, R., & Kaplan, S. (1989). *The experience of nature: a psychological perspective.* Cambridge: Cambridge University Press. Recuperado el Enero de 2018

Kaplan, S. (1995). THE RESTORATIVE BENEFITS OF NATURE: TOWARD AN INTEGRATIVE FRAMEWORK. *Journal of Environmental Psychology*, 169-182. Recuperado el Enero de 2018

Porto, J. P., & Merino, M. (2015). *Definición.DE*. Recuperado el Enero de 2018, de https://definicion.de/android/

RECURSOS ENPROJECTMANAGEMENT. (2017). Recuperado el Enero de 2018, de https://www.recursosenprojectmanagement.com/definicion-del-alcance-del-proyecto/

Rouse, M. (Diciembre de 2016). *TechTarget*. Recuperado el Enero de 2018, de http://searchdatacenter.techtarget.com/es/definicion/Framework

San Juan, C., Subiza-Pérez, M., & Vozmediano, L. (7 de Diciembre de 2017). *Frontiers in Psychology*. Obtenido de https://www.frontiersin.org/articles/10.3389/fpsyg.2017.02093/full

Tyrväinen, L., Ojala, A., Korpela, K., Lanki, T., Tsunetsugu, Y., & Kagawa, T. (2014). The influence of urban green environments on stress relief measures: A field experiment. *Journal of Environmental Psychology*, 1-9. Recuperado el Enero de 2018, de https://www.journals.elsevier.com/journal-of-environmental-psychology

Ulrich, R. S. (1993). Biophilia, Biophobia, and Natural Landscapes. (E. Wilson, & S. E. Kellert, Edits.) *The Biophilia Hypothesis*, 73-173.

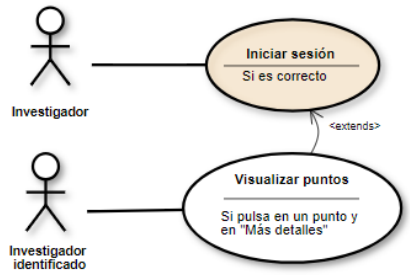
Welicki, L. (s.f.). *Microsoft Developer Network*. Recuperado el 2018, de https://msdn.microsoft.com/es-es/library/bb972272.aspx

10. ANEXO I: CASOS DE USO EXTENDIDOS

En este anexo, se presentan los casos de uso extendidos representados en el capítulo de casos de uso (ver apartado 4.5). Para facilitar la representación del caso de uso que se esté extendiendo en cada caso, este se representa de color marrón.

10.1. Aplicación web

10.1.1. Iniciar sesión



Nombre: Iniciar sesión.
Descripción: Permite al investigador identificarse en el sistema.
Actores: Investigador.
Precondiciones: Estar dado de alta en el sistema.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. El investigador introduce sus credenciales y pulsa en "Iniciar sesión" (Ilustración 95). *[Si el nombre de usuario no existe]* 2a. Se muestra un mensaje de error informando de que ese investigador no existe en el sistema (Ilustración 96). *[Si la contraseña es incorrecta]* 2b. Se muestra un mensaje de error informando de que la contraseña no es correcta (Ilustración 97). *[Si los datos introducidos son correctos]* 2c. Se redirigirá a la pantalla principal de la aplicación (Ilustración 98).
Postcondiciones: El investigador habrá iniciado sesión.

Interfaz gráfica:

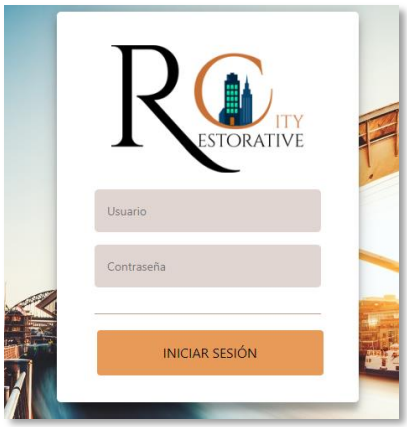


Ilustración 95 - Pantalla de inicio

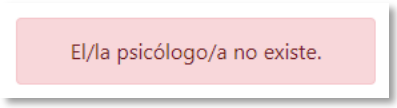


Ilustración 96 - Mensaje de error de investigador no existente

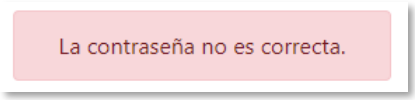


Ilustración 97 - Mensaje de error de contraseña incorrecta

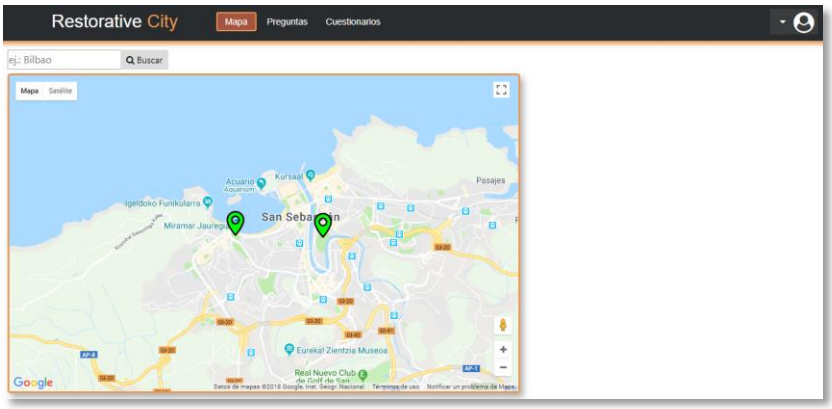
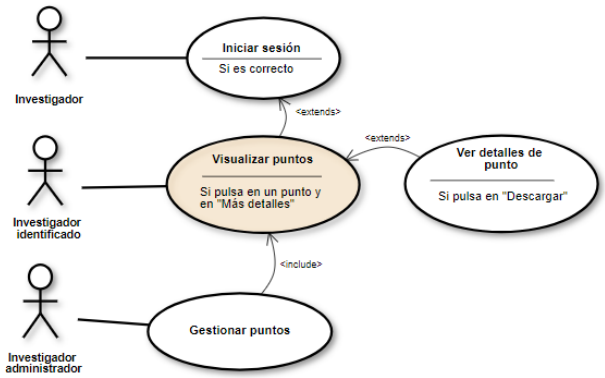


Ilustración 98 - Pantalla principal

10.1.2. Visualizar puntos



Nombre: Visualizar puntos.

Descripción: Permite al investigador identificado visualizar los PPR almacenados en el sistema.

Actores: Investigador identificado.

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El investigador identificado visualiza los PPR dibujados en la pantalla principal (Ilustración 98).

[Si pincha sobre un PPR]

2. Se muestra una ventana de información sobre el PPR (Ilustración 99).

[Si pincha en "Más detalles"]

3. Se redirigirá a la pantalla de ver los resultados del PPR (Ilustración 100).

Postcondiciones: El investigador identificado visualizará los PPR.

Interfaz gráfica:

Ilustración 99 - Ventana de información del PPR

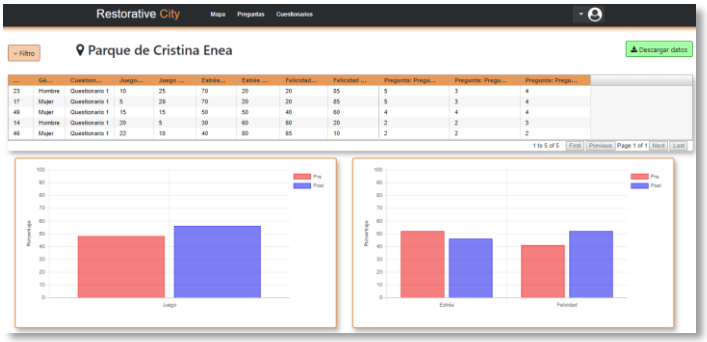
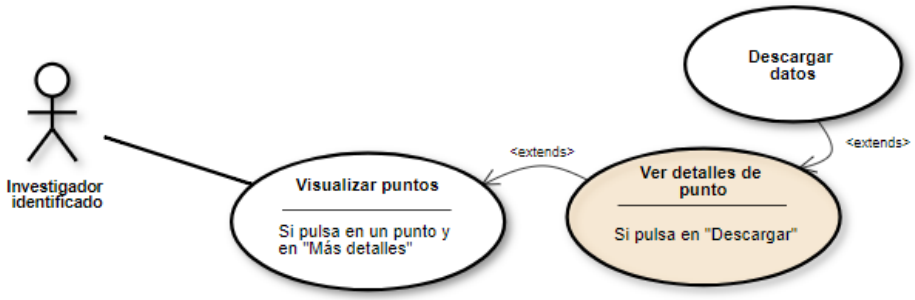


Ilustración 100 - Pantalla de ver los resultados de un PPR

10.1.3. Ver detalles de punto



Nombre: Ver detalles de punto.

Descripción: Permite al investigador identificado visualizar las respuestas obtenidas en un PPR concreto.

Actores: Investigador identificado.

Precondiciones: Haber entrado a la vista en detalle de un PPR.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El investigador identificado visualiza las respuestas obtenidas en el PPR, acompañado de unos gráficos (Ilustración 100).

[Si pincha en "Descargar datos"]

2. Se descargan los datos que estuviera visualizando en ese momento.

*[Si despliega el filtro, introduce los parámetros de filtrado y pulsa en "Filtrar" (*Ilustración 101*)]*

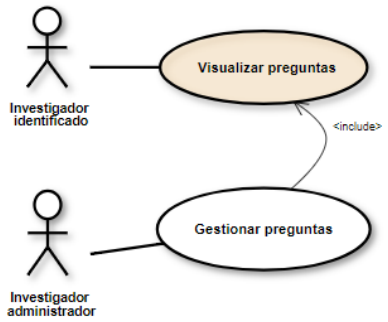
3. Se realiza una búsqueda filtrada de los resultados.

Postcondiciones: El investigador identificado visualizará el detalle del PPR.

Interfaz gráfica:

Ilustración 101 - Filtro

10.1.4. Visualizar preguntas



Nombre: Visualizar preguntas.

Descripción: Permite al investigador identificado visualizar las preguntas almacenadas en el sistema.

Actores: Investigador identificado.

Precondiciones: Haber entrado a la interfaz de visualización de preguntas.

Requisitos no funcionales: Ninguno.

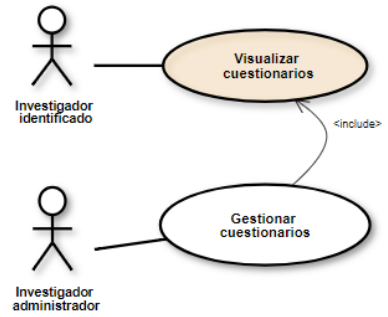
Flujo de eventos:
1. El investigador identificado visualiza las preguntas almacenadas en el sistema (Ilustración 102).

Postcondiciones: El investigador identificado visualizará las preguntas.

Interfaz gráfica:

Ilustración 102 – Pantalla de visualización de preguntas

10.1.5. Visualizar cuestionarios



Nombre: Visualizar cuestionarios.

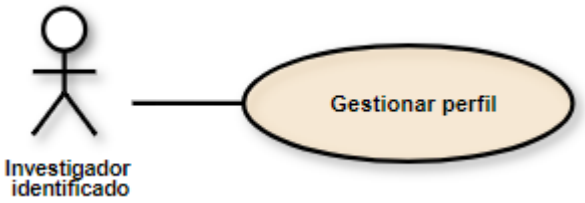
Descripción: Permite al investigador identificado visualizar los cuestionarios definidos en el sistema.

Actores: Investigador identificado.

Precondiciones: Haber entrado a la interfaz de visualización de cuestionarios.

Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. El investigador identificado visualiza los cuestionarios definidos en el sistema (Ilustración 103).
Postcondiciones: El investigador identificado visualizará los cuestionarios.
Interfaz gráfica:  *Ilustración 103 – Pantalla de visualización de cuestionarios*

10.1.6. Gestionar perfil


Nombre: Gestionar perfil.
Descripción: Permite al investigador identificado visualizar los datos de su perfil y, si desea, además, modificarlos.
Actores: Investigador identificado.
Precondiciones: Haber entrado a la interfaz de gestión del perfil.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. El investigador identificado visualiza los datos de su perfil (Ilustración 104). 2. Pulsa en “Editar perfil” 2a. Edita algún dato y pulsa “Guardar” (Ilustración 105). *[Si se ha actualizado correctamente]* 2aa. Se muestra un mensaje de éxito (Ilustración 107). *[Si no]* 2ab. Se muestra un mensaje de error (Ilustración 108). 2b. Pulsa en “Editar contraseña”. 2ba. Edita algún dato y pulsa “Guardar” (Ilustración 106). *[Si se ha actualizado correctamente]* 2baa. Se muestra un mensaje de éxito (Ilustración 107). *[Si no]* 2bab. Se muestra un mensaje de error (Ilustración 108). 2bb. Pulsa en “No editar contraseña”.
Postcondiciones: El investigador identificado habrá visualizado, y si desea, modificado su perfil.

Interfaz gráfica:

Ilustración 104 – Pantalla de gestión del perfil

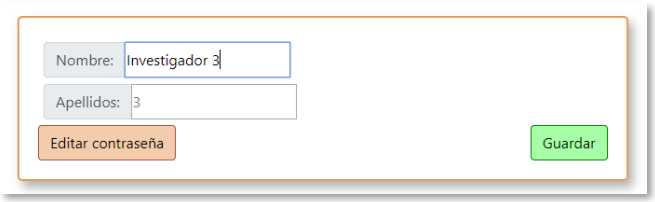


Ilustración 105 - Editar datos

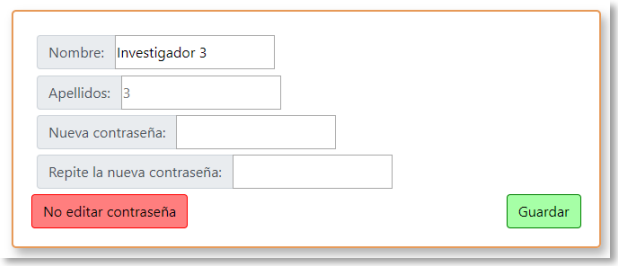


Ilustración 106 - Editar contraseña

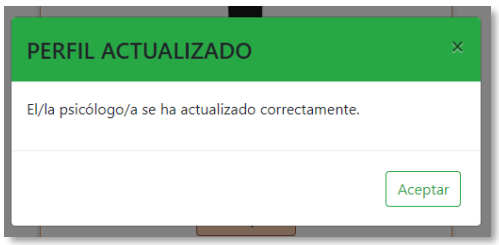


Ilustración 107 - Mensaje de éxito

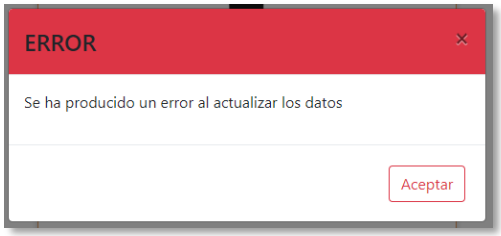
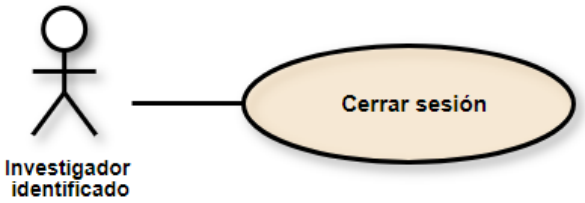


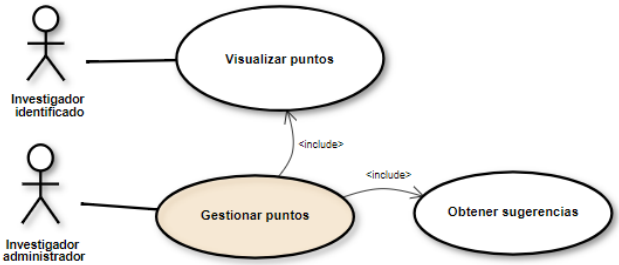
Ilustración 108 - Mensaje de error

10.1.7. Cerrar sesión



Nombre: Cerrar sesión.
Descripción: Permite al investigador identificado cerrar la sesión.
Actores: Investigador identificado.
Precondiciones: Ninguna.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. El investigador identificado pulsa en "Cerrar sesión" en la cabecera de la web. 2. Se redirige a la pantalla de inicio (Ilustración 95).
Postcondiciones: El investigador identificado habrá cerrado la sesión.
Interfaz gráfica:

10.1.8. Gestionar puntos



Nombre: Gestionar puntos.
Descripción: Permite al investigador administrador visualizar, añadir, editar y eliminar los PPR.
Actores: Investigador administrador.
Precondiciones: Ninguna.
Requisitos no funcionales: Ninguno.

Flujo de eventos:
[Si desea añadir un punto]
1. Pincha sobre el mapa en el lugar que desea añadir el punto.
2. Configura el PPR y pulsa “Guardar” (Ilustración 109).
   [Si se ha guardado correctamente]
      3a. Se muestra un mensaje de éxito (Ilustración 107).
   [Si no]
      3b. Se muestra un mensaje de error (Ilustración 108).

[Si desea editar un punto]
4. Pincha sobre el PPR que desea editar y pulsa “Editar” (Ilustración 110).
5. Modifica los datos y pulsa “Guardar cambios”.
   [Si se ha actualizado correctamente]
      6a. Se muestra un mensaje de éxito (Ilustración 107).
   [Si no]
      6b. Se muestra un mensaje de error (Ilustración 108).

[Si desea eliminar un punto]
7. Pincha sobre el PPR que desea eliminar y pulsa “Eliminar” (Ilustración 110).
   [Si se ha eliminado correctamente]
      8a. Se muestra un mensaje de éxito (Ilustración 107).
   [Si no]
      8b. Se muestra un mensaje de error (Ilustración 108).

Postcondiciones: El PPR se ha añadido, editado o eliminado.

Interfaz gráfica:

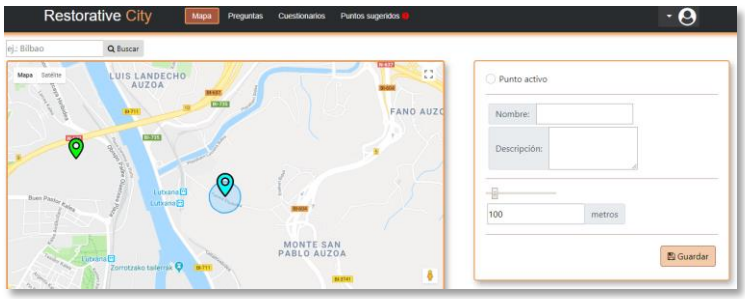


Ilustración 109 - Añadir punto

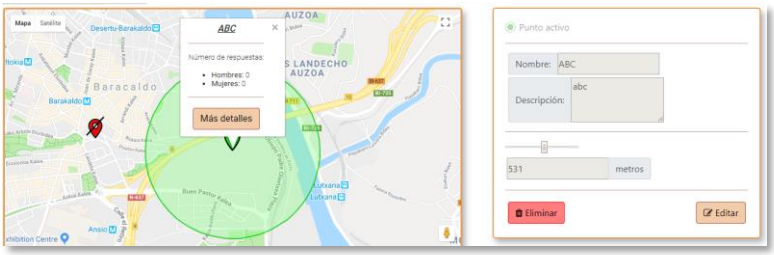
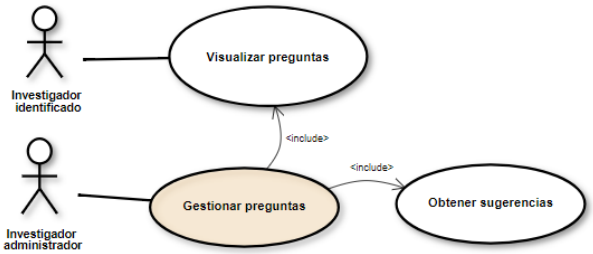


Ilustración 110 - Editar o eliminar punto

10.1.9. Gestionar preguntas



Nombre: Gestionar preguntas.

Descripción: Permite al investigador administrador visualizar, añadir, editar y eliminar preguntas.

Actores: Investigador administrador.

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

[Si desea añadir una pregunta]

1. Escribe el texto de la pregunta y pulsa “Añadir” (Ilustración 111).
   [Si se ha guardado correctamente]
      2a. Se muestra un mensaje de éxito (Ilustración 107).
   [Si no]
      2b. Se muestra un mensaje de error (Ilustración 108).

[Si desea editar una pregunta]

3. Pincha sobre la pregunta que desea editar y pulsa “Editar” (Ilustración 111).
4. Modifica el texto de la pregunta y pulsa “Aceptar”.
   [Si se ha actualizado correctamente]
      5a. Se muestra un mensaje de éxito (Ilustración 107).
   [Si no]
      5b. Se muestra un mensaje de error (Ilustración 108).

[Si desea eliminar una pregunta]

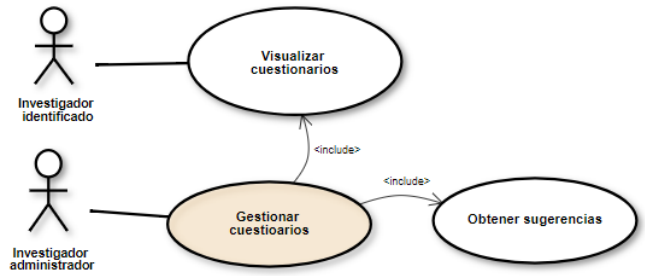
6. Pincha sobre la pregunta que desea eliminar y pulsa “Eliminar” (Ilustración 111).
   [Si se ha eliminado correctamente]
      7a. Se muestra un mensaje de éxito (Ilustración 107).
   [Si no]
      7b. Se muestra un mensaje de error (Ilustración 108).

Postcondiciones: La pregunta se ha añadido, editado o eliminado.

Interfaz gráfica:

Ilustración 111 - Añadir, editar o eliminar pregunta

10.1.10. Gestionar cuestionarios



Nombre: Gestionar cuestionarios.
Descripción: Permite al investigador administrador visualizar, añadir, editar y eliminar cuestionarios.
Actores: Investigador administrador.
Precondiciones: Ninguna.
Requisitos no funcionales: Ninguno.

Flujo de eventos:
[Si desea añadir o editar un cuestionario]
1. Selecciona el cuestionario que quiere tomar como referencia.
2. Edita el cuestionario y pulsa "Guardar" (Ilustración 112).
[Si el cuestionario no ha sido respondido, se actualiza]
[Si se ha actualizado correctamente]
3a. Se muestra un mensaje de éxito (Ilustración 107).
[Si no]
3b. Se muestra un mensaje de error (Ilustración 108).

[Si no, se crea uno nuevo]
[Si se ha guardado correctamente]
4a. Se muestra un mensaje de éxito (Ilustración 107).
[Si no]
4b. Se muestra un mensaje de error (Ilustración 108).

[Si desea eliminar un cuestionario]
5. Pincha sobre el cuestionario que desea eliminar y pulsa "Eliminar" (Ilustración 112).
[Si se ha eliminado correctamente]
7a. Se muestra un mensaje de éxito (Ilustración 107).
[Si no]
7b. Se muestra un mensaje de error (Ilustración 108).

Postcondiciones: El cuestionario se ha añadido, editado o eliminado.

Interfaz gráfica:

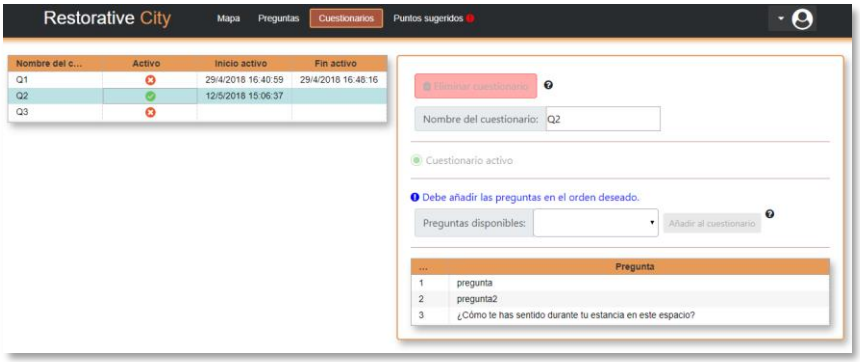


Ilustración 112 - Añadir, editar o eliminar cuestionarios

10.1.11. Gestionar sugerencias

Nombre: Gestionar sugerencias.

Descripción: Permite al investigador administrador aceptar, posponer o rechazar los PPR sugeridos por los usuarios de la aplicación.

Actores: Investigador administrador.

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Pincha en "Ver detalles" de la sugerencia que quiera gestionar (Ilustración 113).

[Si desea aceptar una sugerencia]

2. Configura los datos que faltan del PPR y pulsa "Guardar" (Ilustración 114).

[Si se ha guardado correctamente]

3a. Se muestra un mensaje de éxito (Ilustración 107).

[Si no]

3b. Se muestra un mensaje de error (Ilustración 108).

[Si desea posponer una sugerencia]

4. Pulsa en "Posponer" (Ilustración 113).

[Si desea rechazar una sugerencia]

7. Pulsa en "Rechazar" (Ilustración 113).

[Si se ha eliminado correctamente]

8a. Se muestra un mensaje de éxito (Ilustración 107).

[Si no]

8b. Se muestra un mensaje de error (Ilustración 108).

Postcondiciones: La sugerencia se ha añadido, pospuesto o rechazado.

Interfaz gráfica:

Ilustración 113 - Aceptar, posponer o rechazar sugerencias

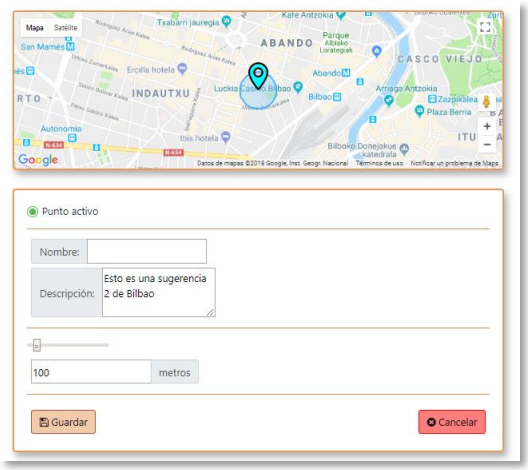


Ilustración 114 - Aceptar una sugerencia

10.1.12. Gestionar investigadores

Nombre: Gestionar investigadores.

Descripción: Permite al investigador administrador dar de alta nuevos miembros investigadores y visualizar, eliminar o cambiar los permisos a los miembros investigadores ya existentes en el sistema.

Actores: Investigador administrador.

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

[Si desea dar de alta un nuevo miembro investigador]
1. Pulsa en el botón de añadir (Ilustración 115).
2. Rellena los datos del nuevo investigador y pulsa “Guardar” (Ilustración 116).
   [Si se ha guardado correctamente]
      3a. Se muestra un mensaje de éxito (Ilustración 107).
   [Si no]
      3b. Se muestra un mensaje de error (Ilustración 108).

[Si desea cambiar de permisos a un investigador]
4. Selecciona el investigador que desea cambiar.
5. Pulsa sobre los botones con flechas para realizar el cambio de permisos (Ilustración 115).

[Si desea eliminar miembro investigador]
6. Selecciona el investigador que desea eliminar.
7. Pulsa en el botón de eliminar (Ilustración 115).
   [Si se ha eliminado correctamente]
      8a. Se muestra un mensaje de éxito (Ilustración 107).

[Si no]
8b. Se muestra un mensaje de error (Ilustración 108).

Postcondiciones: La sugerencia se ha añadido, pospuesto o rechazado.

Interfaz gráfica:

Ilustración 115 - Gestionar investigadores

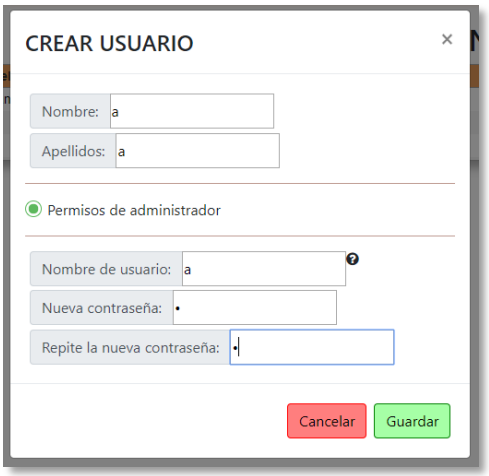
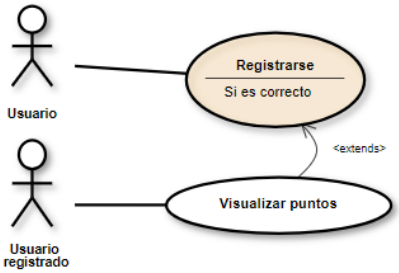


Ilustración 116 - Dar de alta un nuevo investigador

10.2. Aplicación móvil

10.2.1. Registrarse



Nombre: Registrarse.

Descripción: Permite al usuario registrarse en la aplicación.

Actores: Usuario.

Precondiciones: Tener la aplicación instalada.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario rellena los datos requeridos y pulsa en “Aceptar” (Ilustración 117).
[Si el registro ha sido satisfactorio]
2. Se muestra la interfaz gráfica principal de la aplicación móvil, donde puede ver los PPR activos (Ilustración 118).

Postcondiciones: El usuario se habrá dado de alta en el sistema.

Interfaz gráfica:

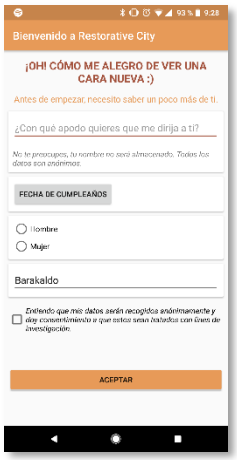
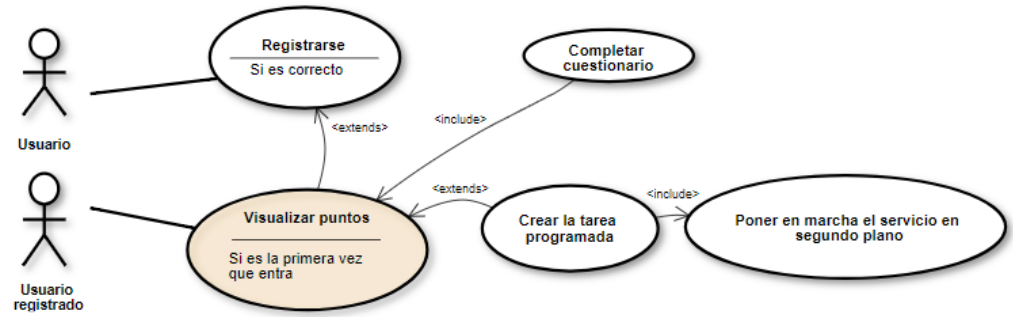


Ilustración 117 - Registro en la aplicación

Ilustración 118 - Interfaz gráfica principal de la aplicación

10.2.2. Visualizar puntos



Nombre: Visualizar puntos.

Descripción: Permite al usuario registrado visualizar sobre un mapa los PPR activos.

Actores: Usuario registrado.

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:
1. El usuario registrado visualiza sobre un mapa los PPR activos (Ilustración 118).

[Si es la primera vez que el usuario accede a la aplicación]
2. Se crea la tarea programada.

3. Se lanza el servicio en segundo plano (Ilustración 119).

Postcondiciones: El usuario registrado visualiza los PPR y la tarea programada ha sido creada.

Interfaz gráfica:

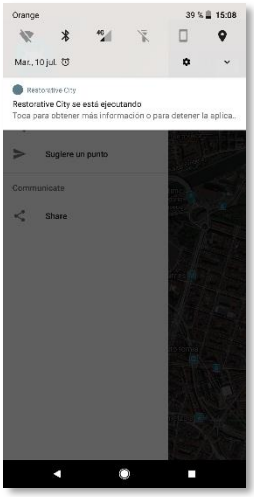
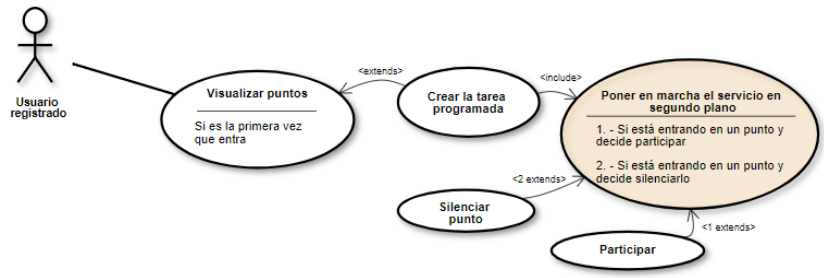


Ilustración 119 - Notificación ejecución del servicio en segundo plano

10.2.3. Poner en marcha el servicio en segundo plano



Nombre: Poner en marcha el servicio en segundo plano.

Descripción: Se pone en marcha el servicio en segundo plano, encargado de comprobar la ubicación del usuario para detectar cuando este entra o sale de un PPR.

Actores: Usuario registrado.

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se lanza el servicio en segundo plano (Ilustración 119).

[Si el usuario está entrando en el área de un PPR]

2. Se le avisa y pregunta si desea participar en el estudio del PPR (Ilustración 120).

[Si decide participar]

3a. Se muestran las pruebas que debe realizar.

[Si decide silenciarlo]

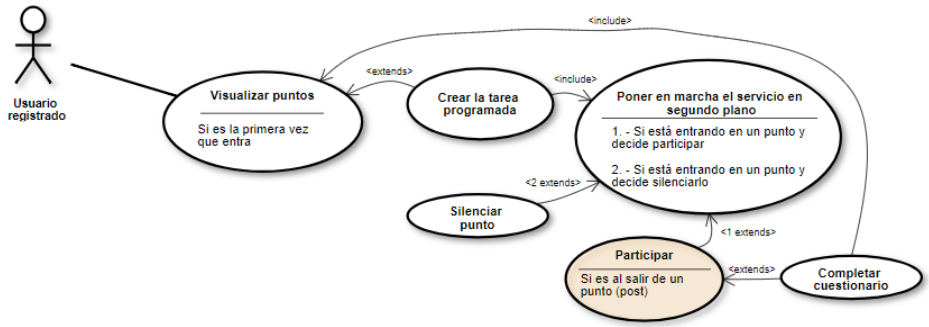
3b. Se silencia el PPR.

Postcondiciones: El servicio en segundo plano ha sido puesto en marcha.

Interfaz gráfica:

Ilustración 120 - Notificación de entrada al punto

10.2.4. Participar



Nombre: Participar.

Descripción: Realizar las pruebas correspondientes a la entrada o salida de un PPR.

Actores: Usuario registrado.

Precondiciones: Haber aceptado a participar en el PPR.

Requisitos no funcionales: Ninguno.

Flujo de eventos:
1. Se realiza la prueba de concentración (Ilustración 121).
2. Se muestra los resultados obtenidos en dicha prueba (Ilustración 122).
3. Se realiza la prueba del termómetro de estrés (Ilustración 123).
4. Se realiza la prueba del termómetro de felicidad (Ilustración 124).

[Si la participación está ocurriendo al salir de un PPR]
5. Se realiza el cuestionario (Ilustración 125).

6. Se muestra la interfaz principal de la aplicación (Ilustración 118).

Postcondiciones: Se ha completado la participación.

Interfaz gráfica:

Ilustración 121 - Prueba de concentración

Ilustración 122 - Resultados obtenidos en la prueba de concentración

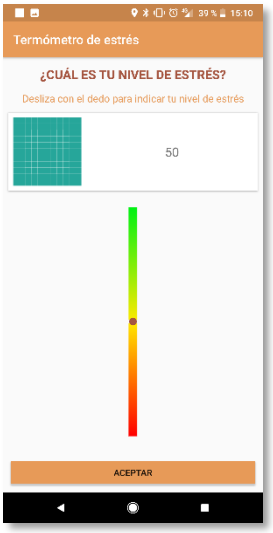


Ilustración 123 - Prueba del termómetro de estrés

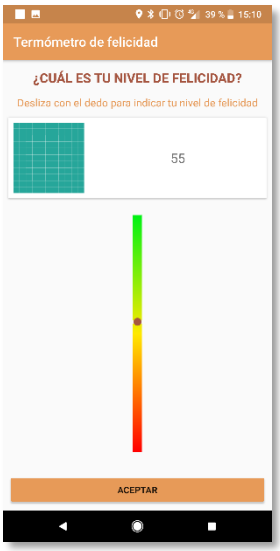


Ilustración 124 - Prueba del termómetro de felicidad

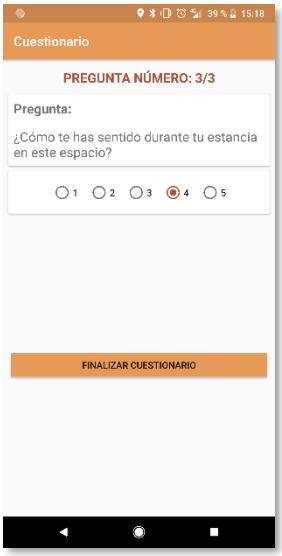
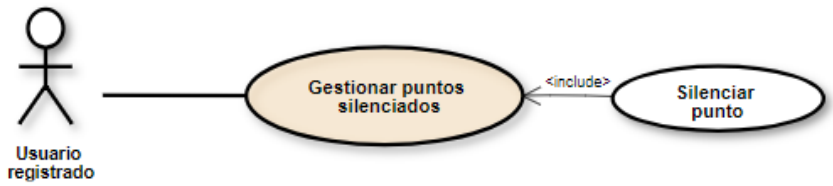


Ilustración 125 - Cuestionario

10.2.5. Gestionar puntos silenciados



Nombre: Gestionar puntos silenciados.
Descripción: Permite al usuario registrado gestionar los PPR que ha silenciado.
Actores: Usuario registrado.
Precondiciones: Ninguna.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. Se visualizan los PPR silenciados para ese usuario (Ilustración 126). *[Si el usuario desplaza la tarjeta que contiene la información de un PPR]* 2. Se elimina el silencio que tenía el PPR. 3. Se muestra, durante unos segundos, la opción para revertir la acción. *[Si pincha en "Deshacer"]* 4. Se volverá a silenciar el PPR.
Postcondiciones: Se han gestionado los puntos silenciados.

Interfaz gráfica:

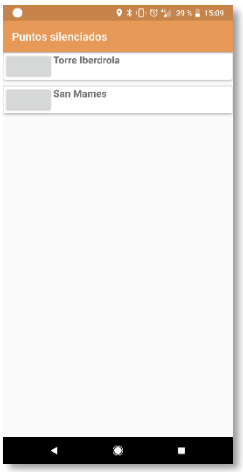
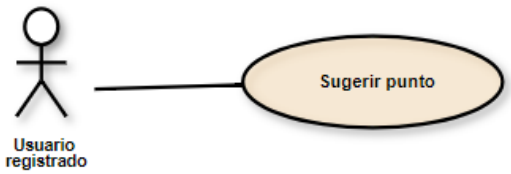


Ilustración 126 - Gestión de puntos silenciados

10.2.6. Sugerir punto



Nombre: Sugerir punto.
Descripción: Permite al usuario registrado sugerir PPRs.
Actores: Usuario registrado.
Precondiciones: Ninguna.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. Se rellenan los campos requeridos para sugerir un PPR (Ilustración 127). 2. Pulsa en “Enviar”
Postcondiciones: Se ha sugerido un PPR.

Interfaz gráfica:

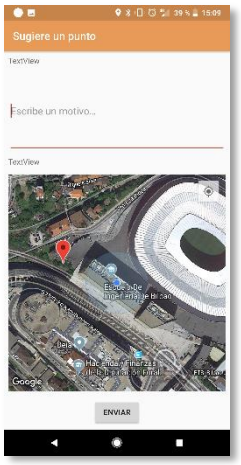


Ilustración 127 - Interfaz gráfica para sugerir un PPR

11. ANEXO II: DIAGRAMAS DE SECUENCIA

En este anexo, se presentan los diagramas de secuencia de cada caso de uso extendido, detallados en el Anexo I (Anexo I: Casos de uso extendidos).

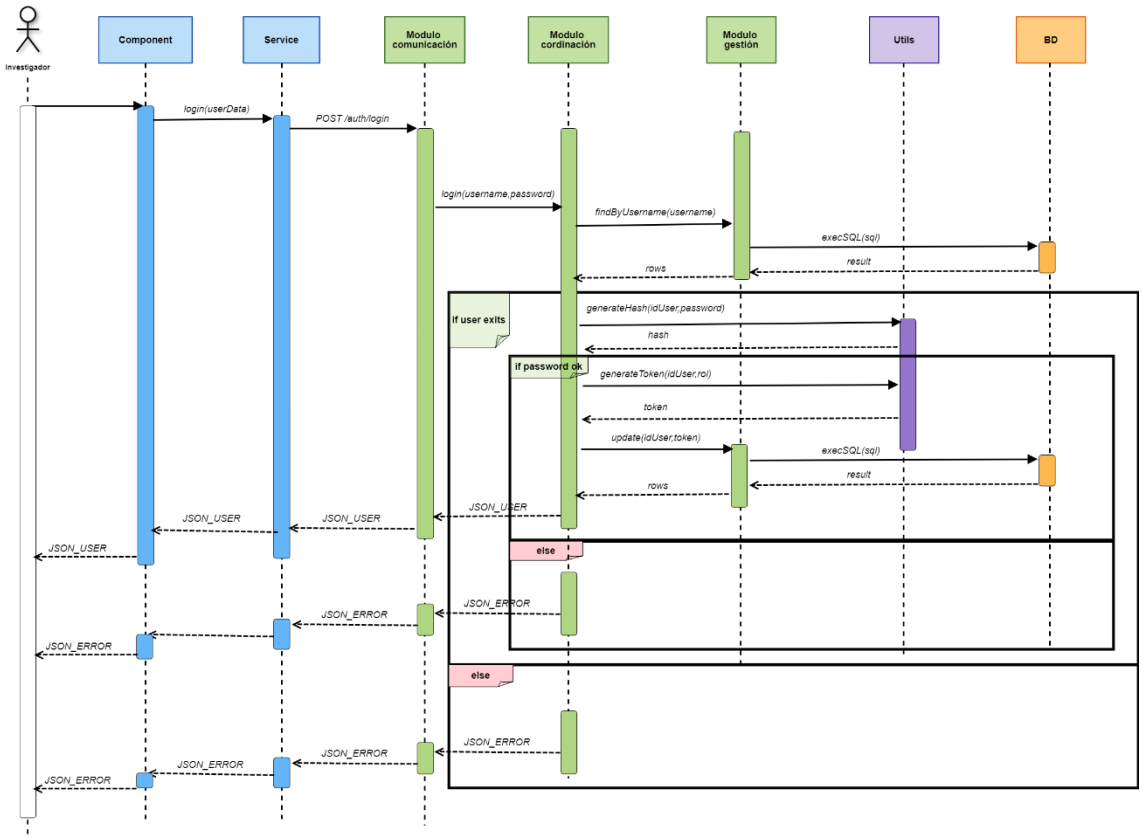
11.1. Aplicación web

Con el fin de simplificar los diagramas que se muestran a continuación, se ha supuesto que en aquellas llamadas que requieran autenticación (*token*), esto es, todas menos la de inicio de sesión, el *token* es válido. En caso de no ser válido, como ya se ha comentado anteriormente, la API REST devolverá un error *HTTP 401 UNAUTHORIZED*, en todos los casos.

El formato del objeto JSON_ERROR es el siguiente:

```
{
    "status": 401,
    "code": 40101,
    "msg": "Token caducado/no válido.",
    "body": null
}
```

11.1.1. Iniciar sesión



La SQL correspondiente a la función *findByUsername()* es la siguiente:

```
'SELECT * FROM psychologist WHERE username = %username%'
```

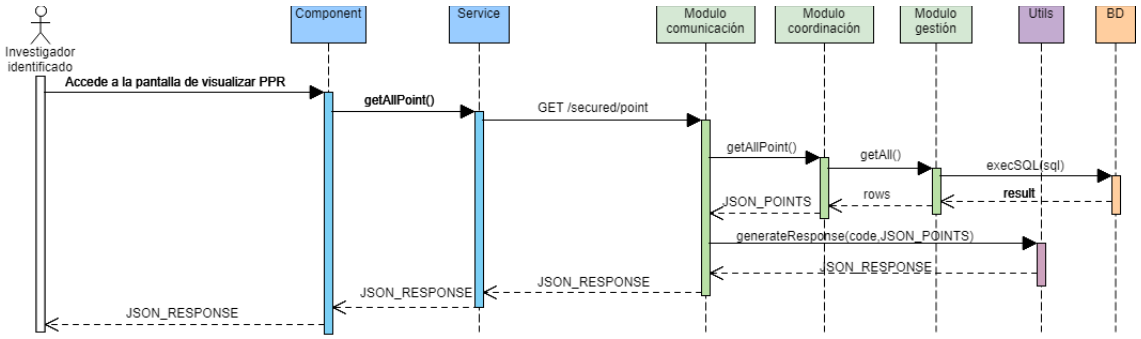
La SQL correspondiente a la función *update(idUser,token)* es la siguiente:

```
'UPDATE psychologist SET token=%token% WHERE id = %id%'
```

El objeto JSON_USER sigue el siguiente formato:


```
"id": 15,
"admin": true,
"username": "ohr",
"name": "Oscar",
"surnames": "Hernández Rivas",
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
    .eyJpZCI6MTUsImFkbWluIjp0cnVlLCJ0b2tlbnZhbGlkaXR5Ijo
    iMjAxOC0wNy0yNFQxNjoyMjozMiswMjowMCIsImlhdCI6MTUzMjQ
    0MDM1Mn0.0kbN2leB9in8ENPMZXy-ywrjx
    -x7DV6X2sCVYclaLYM"
```


11.1.2. Visualizar puntos



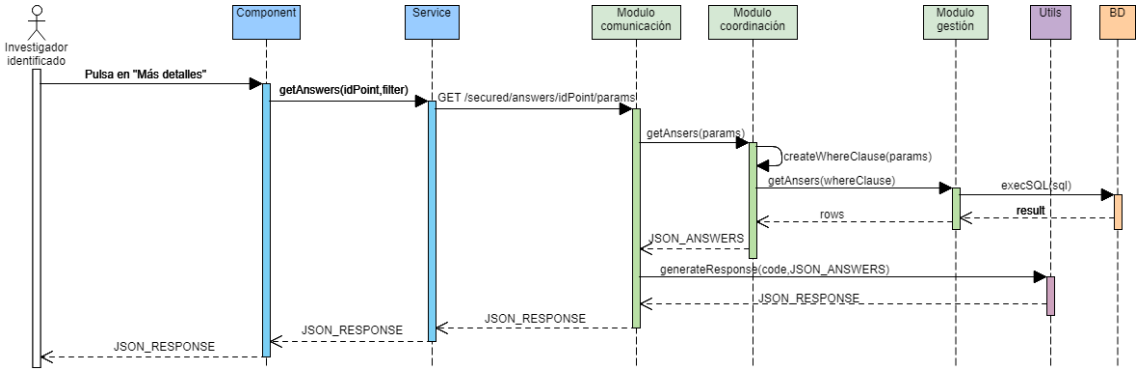
La SQL correspondiente a la función *getAll()* es la siguiente:

```
'SELECT * FROM point'
```

El objeto JSON_POINTS es una lista que contiene objetos con el siguiente formato:


```
{
    "id": 55,
    "name": "Palacio de Miramar",
    "description": "Punto en estudio del Palacio de
        Miramar",
    "ratio": 100,
    "lat": 43.314536240546495,
    "lng": -1.9988794169619268,
    "active": true,
    "menCount": 4,
    "womenCount": 3,
    "imagepaths": []
},
```

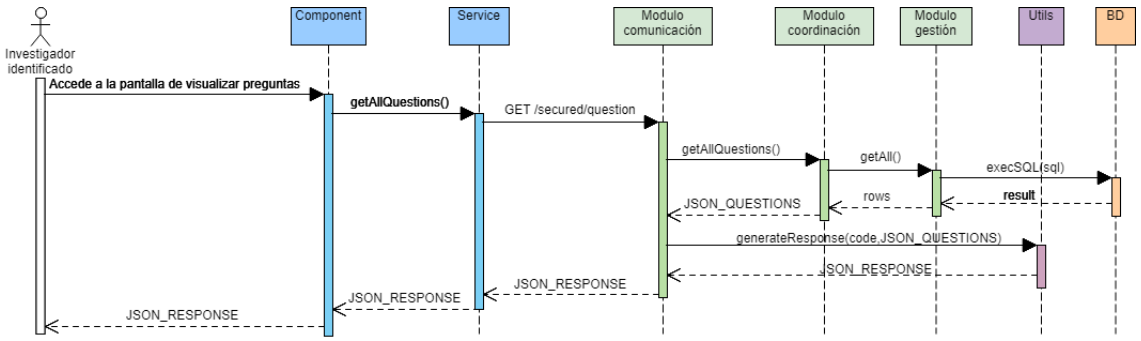
11.1.3. Ver detalles de punto



La SQL correspondiente a la función *getAnswers(whereClause)* es la siguiente:

```
'SELECT A.id, A.iduser, A.idpoint, A.datetime, A.attemptspre, A.correctspre, A.errorspre, A.attemptspost, A.correctspost, A.errorspost, A.stresslevelpre, A.stresslevelpost, A.happinesslevelpre, A.happinesslevelpost, QQ.name, Q.question, AQ.answer, U.gender, U.birthday, U.municipality FROM answer A INNER JOIN answer_question AQ ON A.id = AQ.idanswer INNER JOIN user U ON U.id = A.iduser INNER JOIN questionnaire QQ ON QQ.id = AQ.idquestionnaire INNER JOIN question Q ON Q.id = AQ.idquestion' + whereClause
```

11.1.4. Visualizar preguntas



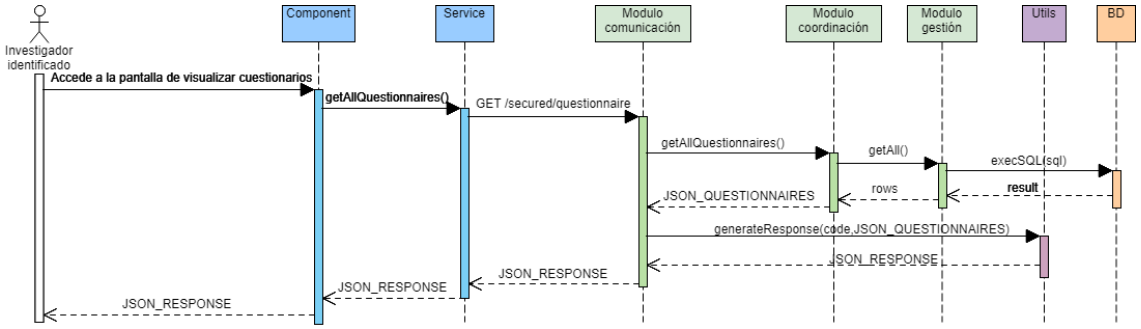
La SQL correspondiente a la función *getAll()* es la siguiente:

```
'SELECT * FROM question'
```

El objeto JSON_QUESTIONS es una lista que contiene objetos con el siguiente formato:

```
{
    "id": 39,
    "question": "Pregunta 1",
    "isAnswered": true,
    "isAssignedToQuestionnaire": true
},
```

11.1.5. Visualizar cuestionarios



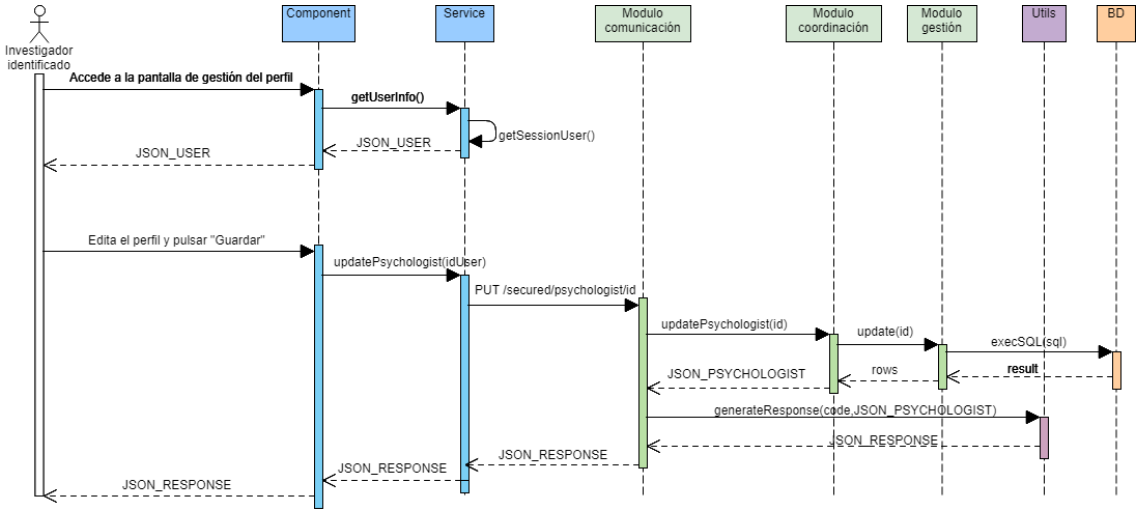
La SQL correspondiente a la función *getAll()* es la siguiente:

```
'SELECT * FROM questionnaire'
```

El objeto JSON_QUESTIONNAIRES es una lista que contiene objetos con el siguiente formato:


```
{
    "id": 77,
    "name": "Questionario 1",
    "active": true,
    "activationdate": "2018-06-26T15:07:28.000Z",
    "deactivationdate": null,
    "anyQuestionIsAnswered": true,
    "questions": [
        {
            "id": 39,
            "question": "Pregunta 1",
            "order": 1
        },
        {
            "id": 40,
            "question": "Pregunta 2",
            "order": 2
        },
        {
            "id": 41,
            "question": "Pregunta 3",
            "order": 3
        }
    ]
}
```

11.1.6. Gestionar perfil



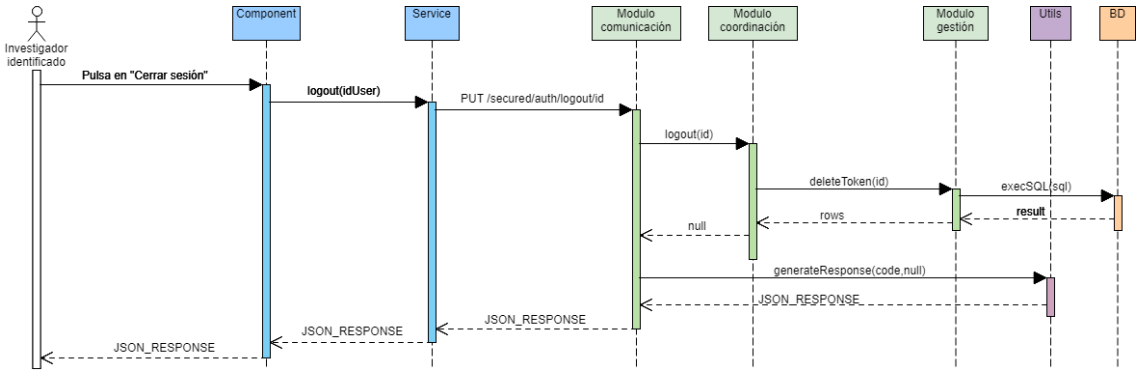
La SQL correspondiente a la función *update(id)* es la siguiente:

```
'UPDATE psychologist SET name=%name%, surnames=%surnames%, password=%password% WHERE id = %id%'
```

El objeto JSON_PSYCHOLOGIST sigue el siguiente formato:


```
"id": 15,
"admin": true,
"username": "ohr",
"name": "Oscar",
"surnames": "Hernández Rivas",
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
    .eyJpZCI6MTUsImFkbWluIjp0cnVlLCJ0b2tlbnZhbGlkaXR5Ijo
    iMjAxOC0wNy0yNFQxNjoyMjozMiswMjowMCIsImlhdCI6MTUzMjQ
    0MDM1Mn0.0kbN2leB9in8ENPMZXy-ywrjx
    -x7DV6X2sCVYclaLYM"
```

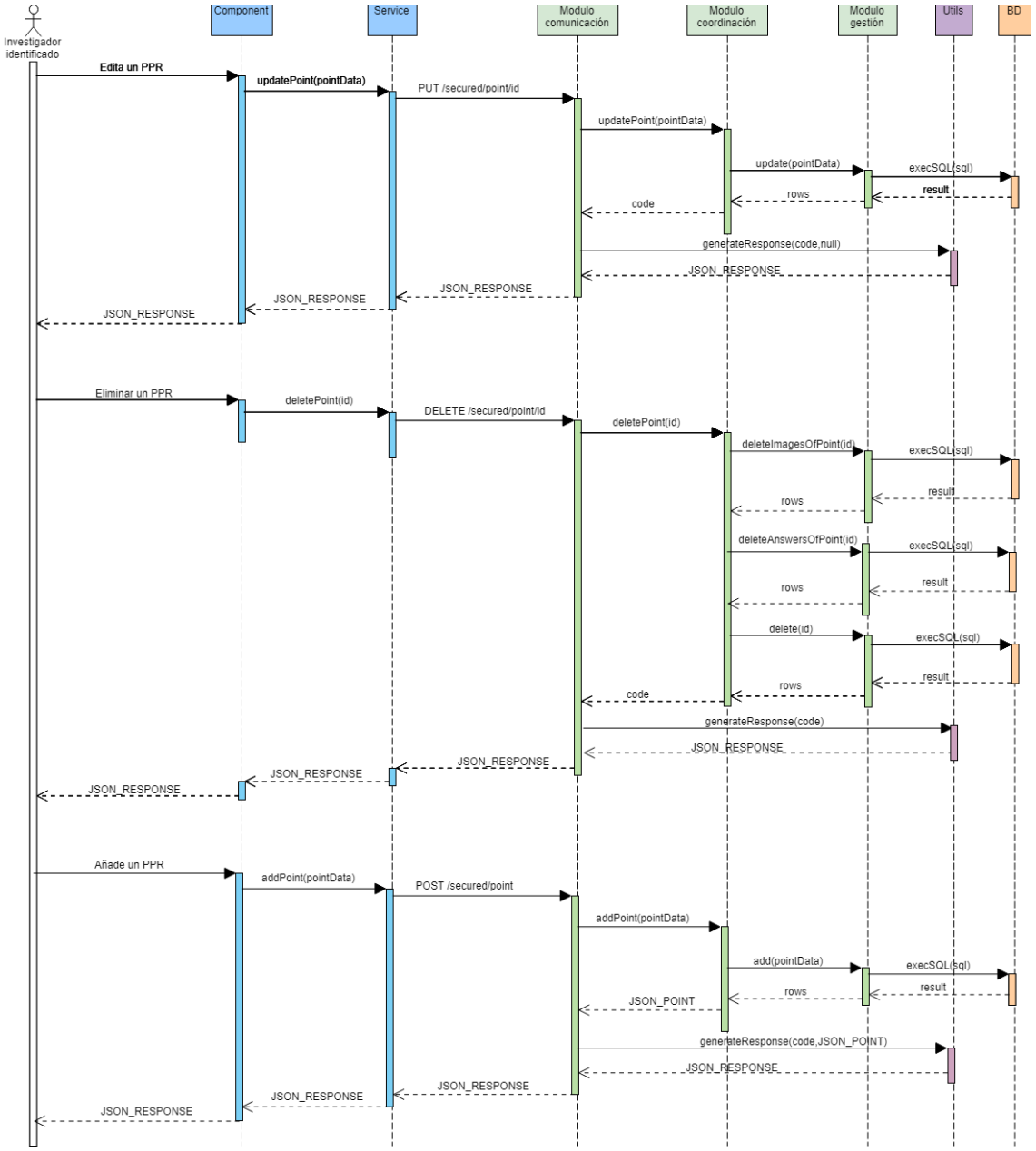

11.1.7. Cerrar sesión



La SQL correspondiente a la función *deleteToken(id)* es la siguiente:

```
'UPDATE psychologist SET token=null WHERE id = %id%'
```

11.1.8. Gestionar puntos



La SQL correspondiente a la función *update(pointData)* es la siguiente:

```
'UPDATE point SET name=%name%, description=%description%,
active=%active% WHERE id = ?'
```

La SQL correspondiente a la función *deleteImagesOfPoint(id)* es la siguiente:

```
'DELETE FROM image_point WHERE idpoint = %id%'
```

La SQL correspondiente a la función *deleteAnswersOfPoint(id)* es la siguiente:

```
'DELETE FROM answer WHERE idpoint = %id%'
```

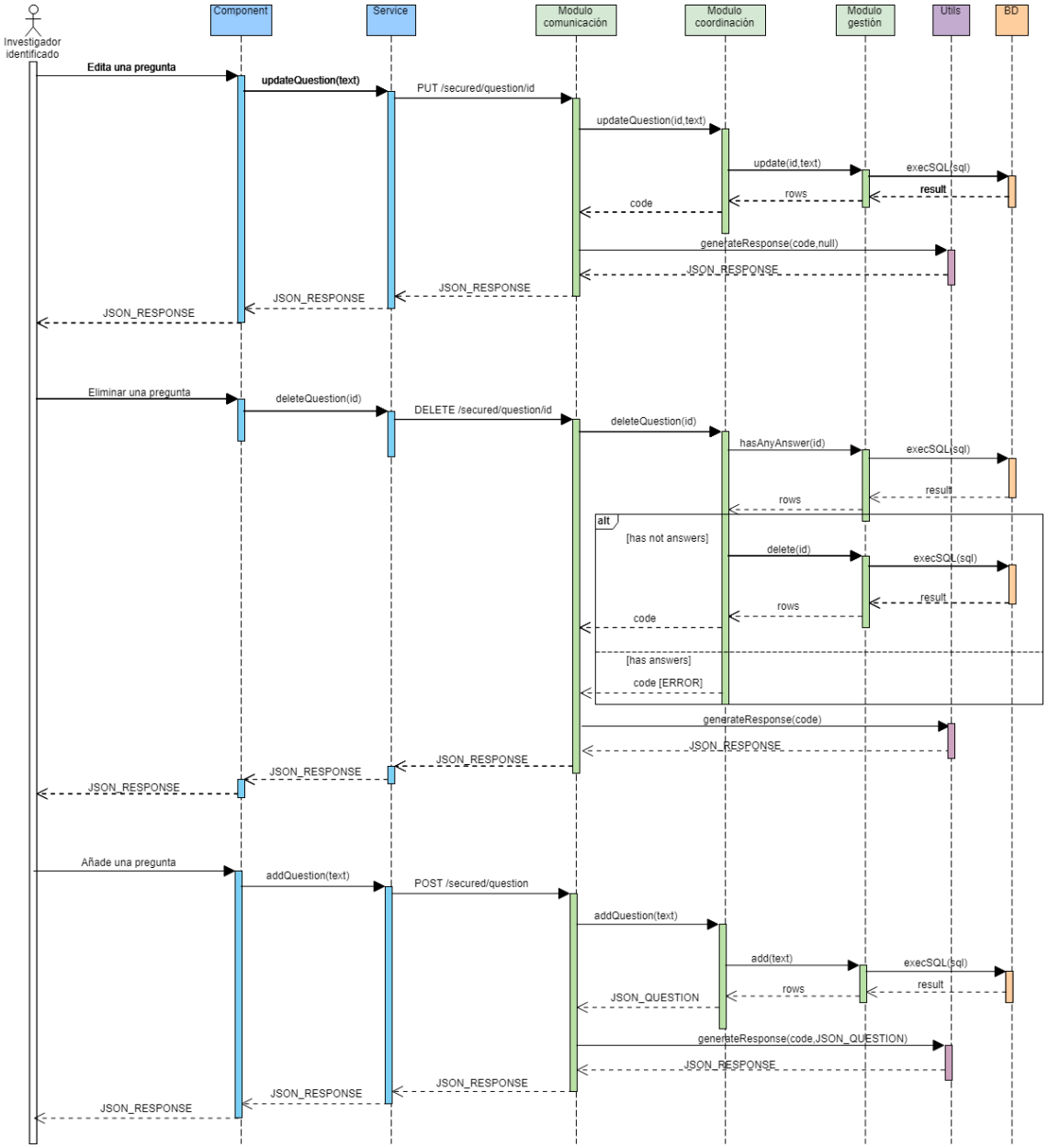
La SQL correspondiente a la función *delete(id)* es la siguiente:

```
'DELETE FROM point WHERE id = %id%'
```

La SQL correspondiente a la función *add(pointData)* es la siguiente:

```
'INSERT INTO point SET name=%name%, description=%description%,
ratio=%ratio%, lat=%lat%, lng=%lng%, active=%active%'
```

11.1.9. Gestionar preguntas



La SQL correspondiente a la función *update(id,text)* es la siguiente:

```
'UPDATE question SET question=%text% WHERE id = %id%'
```

La SQL correspondiente a la función *hasAnyAnswer(id)* es la siguiente:

```
'SELECT * FROM answer_question WHERE idquestion = %id%'
```

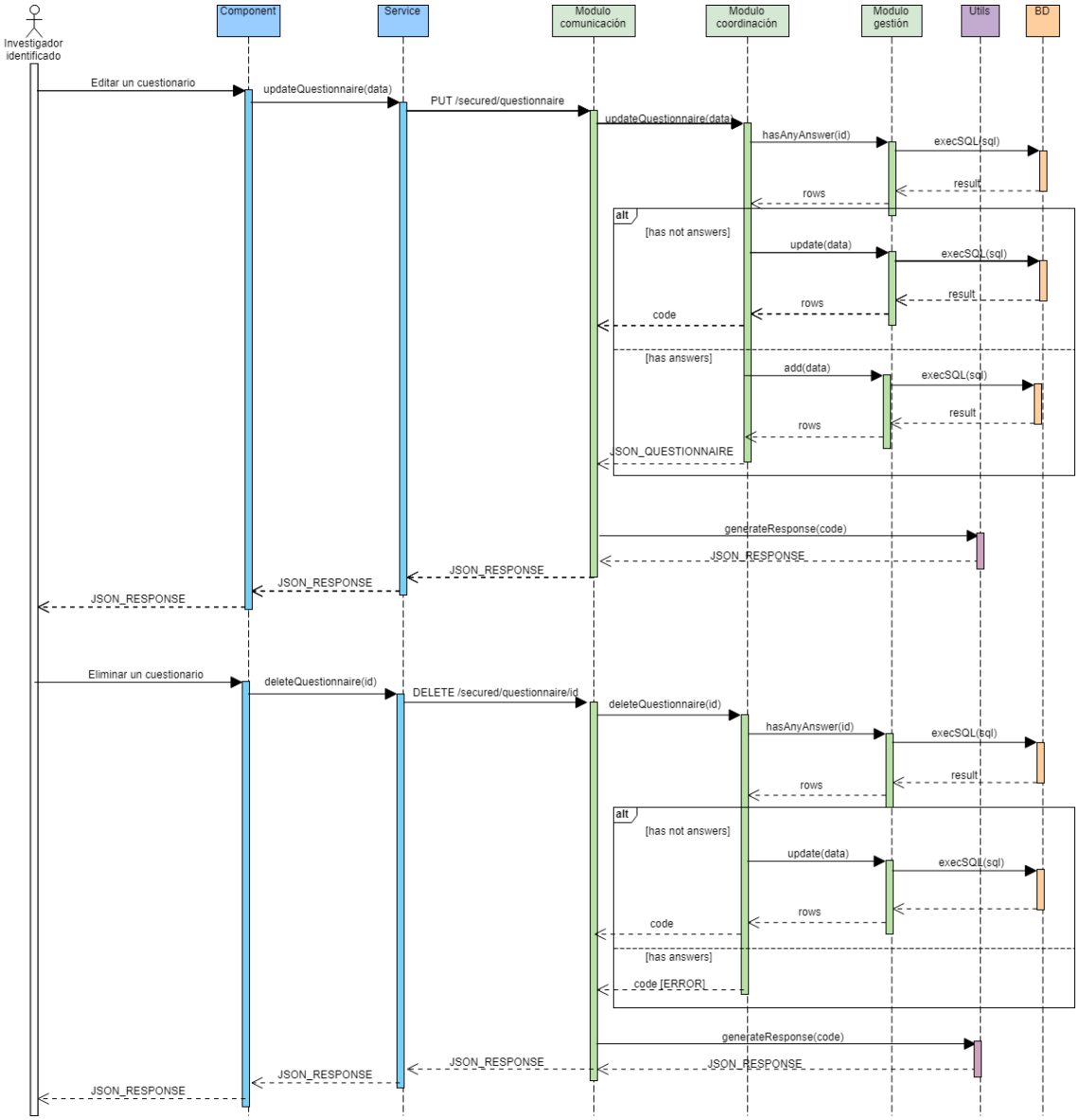
La SQL correspondiente a la función *delete(id)* es la siguiente:

```
'DELETE FROM question WHERE id = %id%'
```

La SQL correspondiente a la función *add(text)* es la siguiente:

```
'INSERT INTO question SET question=%text%'
```

11.1.10. Gestionar cuestionarios



La SQL correspondiente a la función *hasAnyAnswer (id)* es la siguiente:

```
'SELECT * FROM answer_question WHERE idquestionnaire = %id%'
```

La SQL correspondiente a la función *add(data)* es la siguiente:

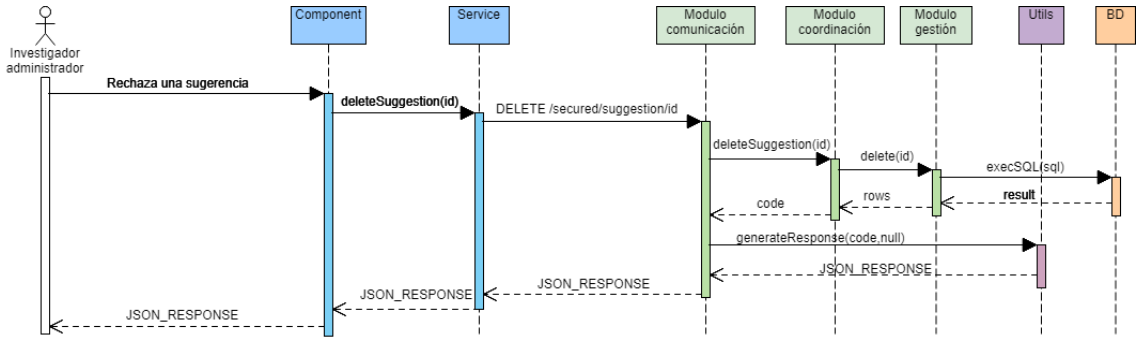
```
'INSERT INTO questionnaire SET name=%name%, active=%active%'
```

La SQL correspondiente a la función *delete(id)* es la siguiente:

```
'DELETE FROM questionnaire WHERE id = %id%'
```

11.1.11. Gestionar sugerencias

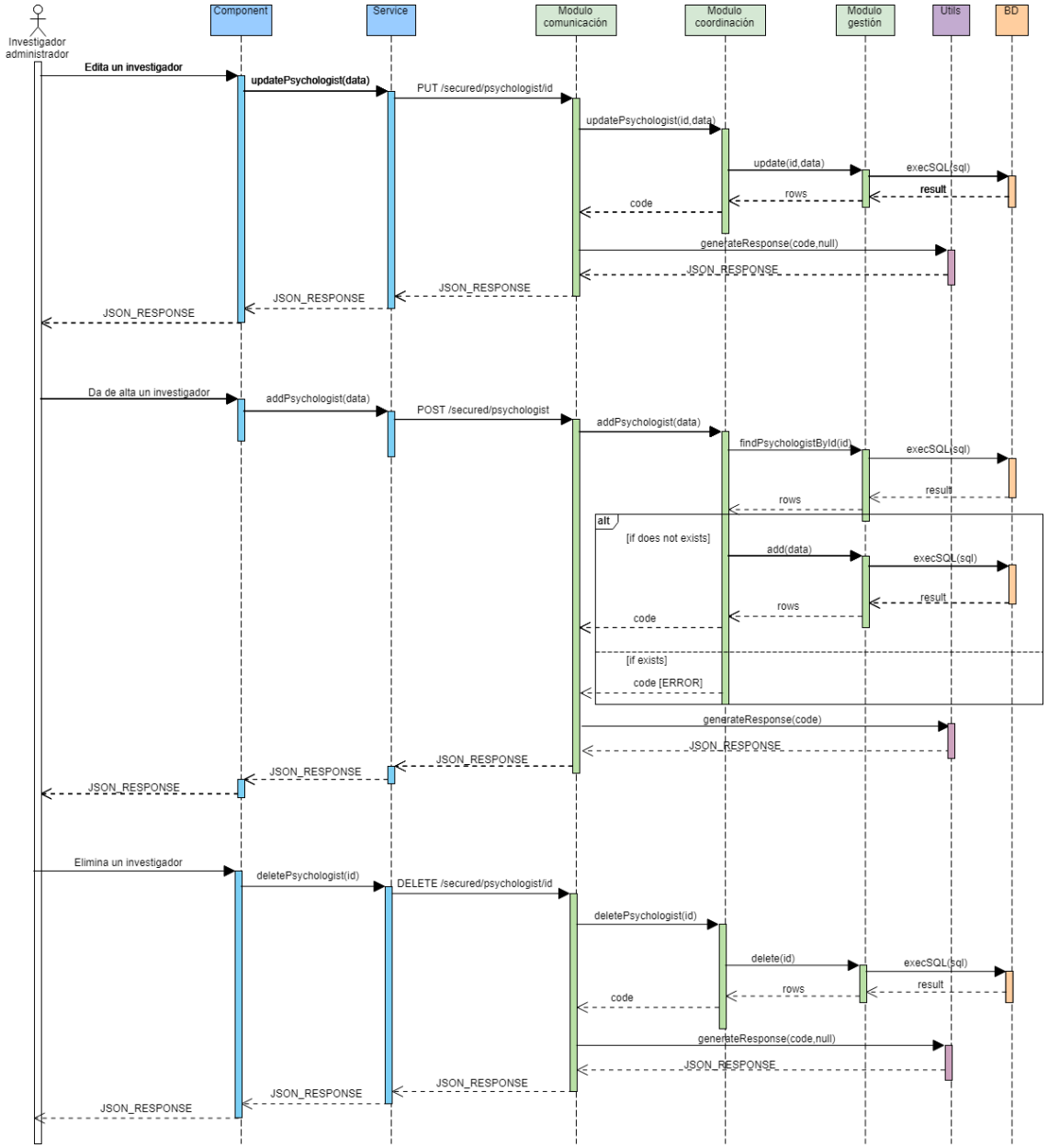
La funcionalidad de “aceptar una sugerencia” comparte el mismo flujo de secuencia que la funcionalidad “añadir punto”, representada anteriormente (Gestionar puntos). Por este motivo, la primera funcionalidad mencionada no se refleja en el siguiente diagrama.



La SQL correspondiente a la función *delete (id)* es la siguiente:

```
'DELETE FROM suggestion WHERE id = %id%'
```

11.1.12. Gestionar investigadores



La SQL correspondiente a la función *update(id,data)* es la siguiente:

```
'UPDATE psychologist SET name=%name%, surnames=%surnames%, password=%password% WHERE id = %id%'
```

La SQL correspondiente a la función *findPsychologistById(id)* es la siguiente:

```
'SELECT * FROM psychologist WHERE id = %id%'
```

La SQL correspondiente a la función *add(data)* es la siguiente:

```
'INSERT INTO psychologist SET username=%username%, password=%password%, admin=%admin%,name=%name%, surnames=%surnames%'
```

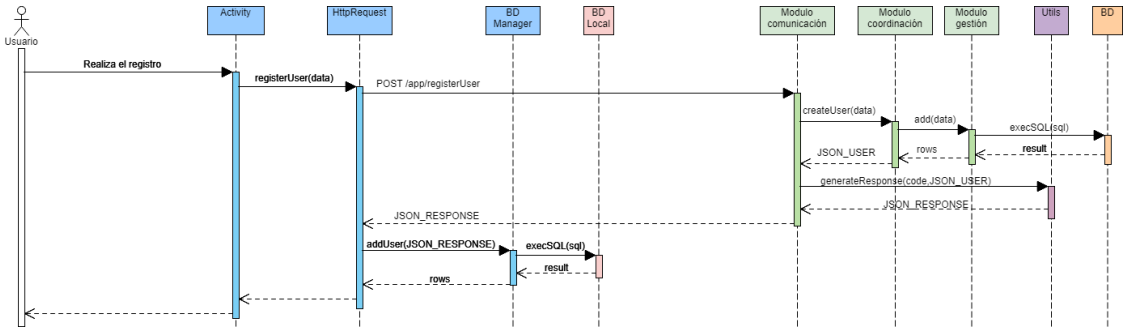
La SQL correspondiente a la función *delete(id)* es la siguiente:

```
'DELETE FROM psychologist WHERE id = %id%'
```

11.2. Aplicación móvil

A continuación, se presentan los diagramas de secuencia referentes a los casos de uso de la aplicación móvil.

11.2.1. Registrarse



La SQL correspondiente a la función *add(data)* es la siguiente:

```
'INSERT INTO user SET gender=%gender%, birthday=%birthday%,
municipality=%municipality%'
```

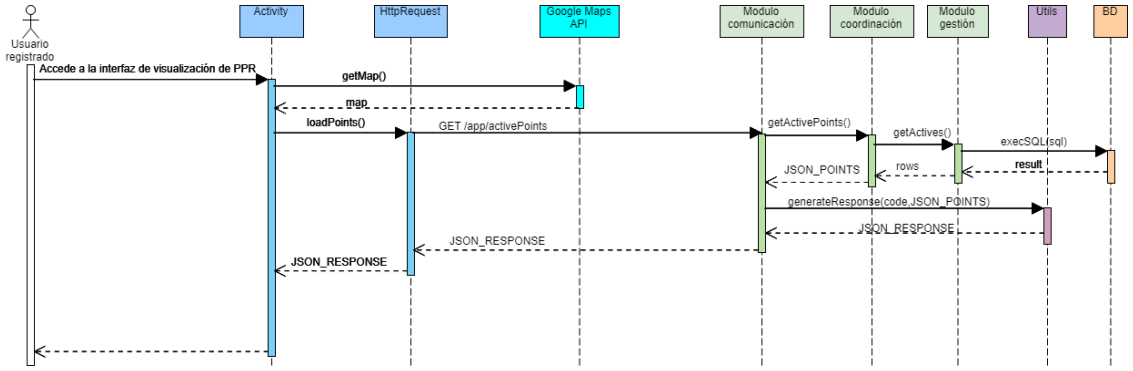
El objeto JSON_USER sigue el siguiente formato:

```
{
    "gender": 0,
    "birthday": "1995/01/01",
    "municipality": "Barakaldo"
}
```

La SQL correspondiente a la función *addUser(JSON_USER)* es la siguiente:

```
'INSERT INTO user SET id=%id%, name=%name%;
```

11.2.2. Visualizar puntos



La SQL correspondiente a la función *getActives()* es la siguiente:

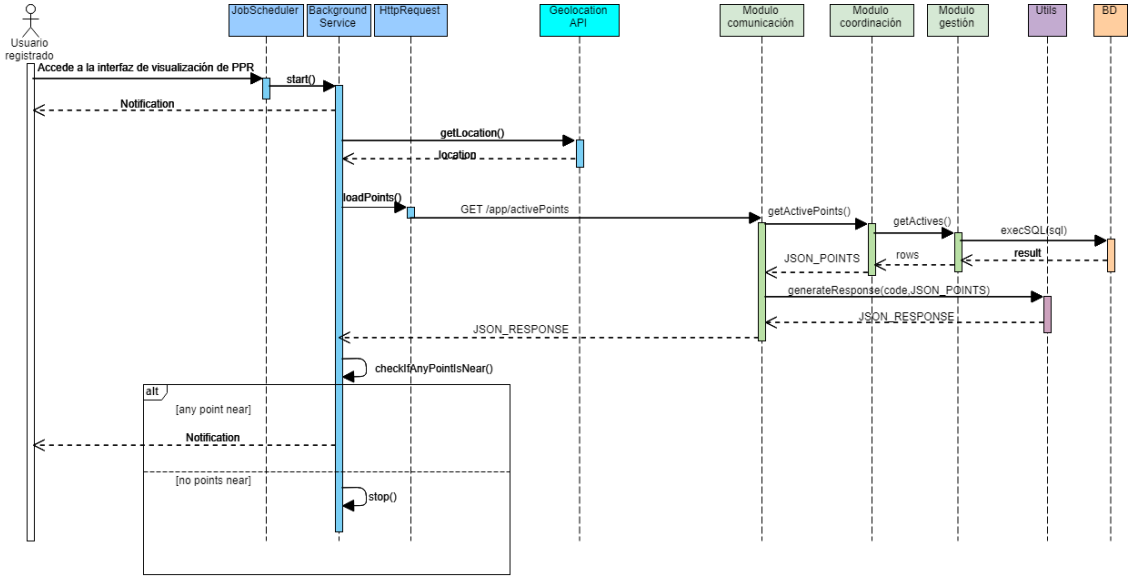
```
'SELECT * FROM point WHERE active = 1'
```

El objeto JSON_POINTS es una lista que contiene objetos con el siguiente formato:

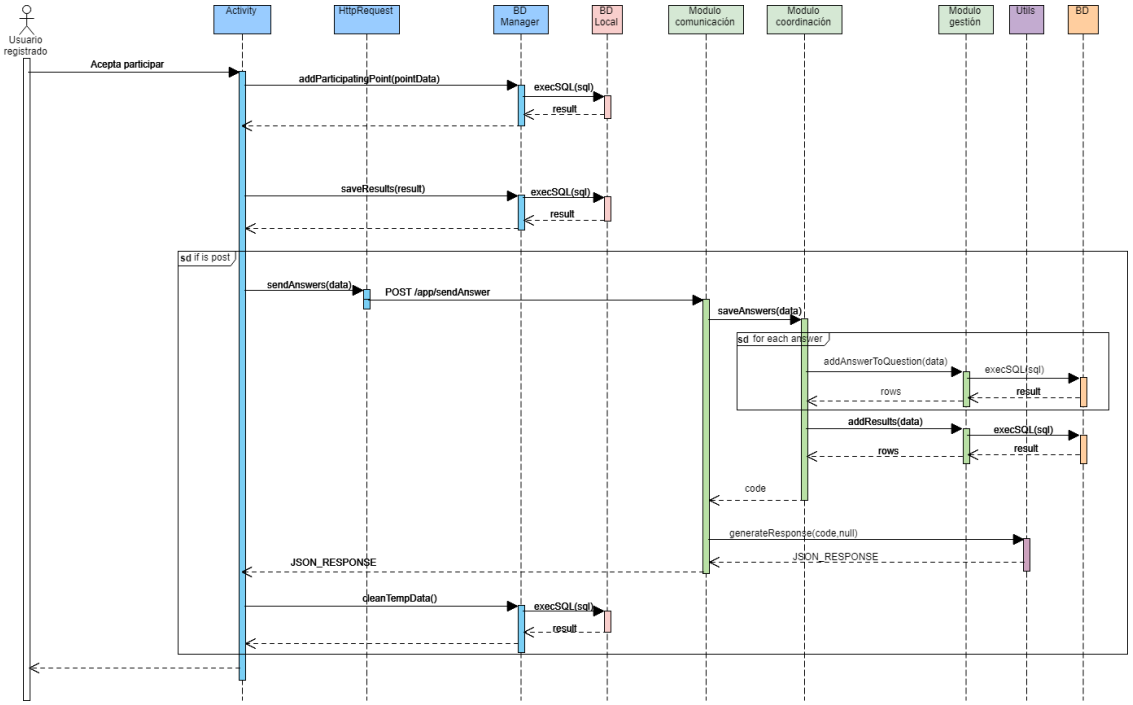
```
{
    "id": 55,
    "name": "Palacio de Miramar",
    "description": "Punto en estudio del Palacio de
        Miramar",
    "ratio": 100,
    "lat": 43.314536240546495,
    "lng": -1.9988794169619268,
    "active": true,
    "menCount": 4,
    "womenCount": 3,
    "imagepaths": []
},
```

11.2.3. Poner en marcha el servicio en segundo plano

Las SQL y los tipos de objetos utilizados en este diagrama, coinciden con las utilizadas en el anterior.



11.2.4. Participar



La SQL correspondiente a la función *addParticipatingPoint(pointData)* es la siguiente:

```
'INSERT INTO ParticipatingPoint SET id=%id%, name=%name%,
description=%description%, ratio=%ratio%, lat=%lat%, lng=%lng%'
```

La función *saveResults(results)* es genérica para cada tipo de prueba que se realiza en el examen. La SQL genérica es:

```
'INSERT INTO %TestTable% SET isOnPre=%isOnPre%, result=%result%'
```

La SQL correspondiente a la función *addAnswerToQuestion(data)* es la siguiente:

```
'INSERT INTO answer_question SET idAnswer=%idAnswer%,
idQuestionnaire=%idQuestionnaire%, idQuestion=%idQuestion%,
answer=%answer%'
```

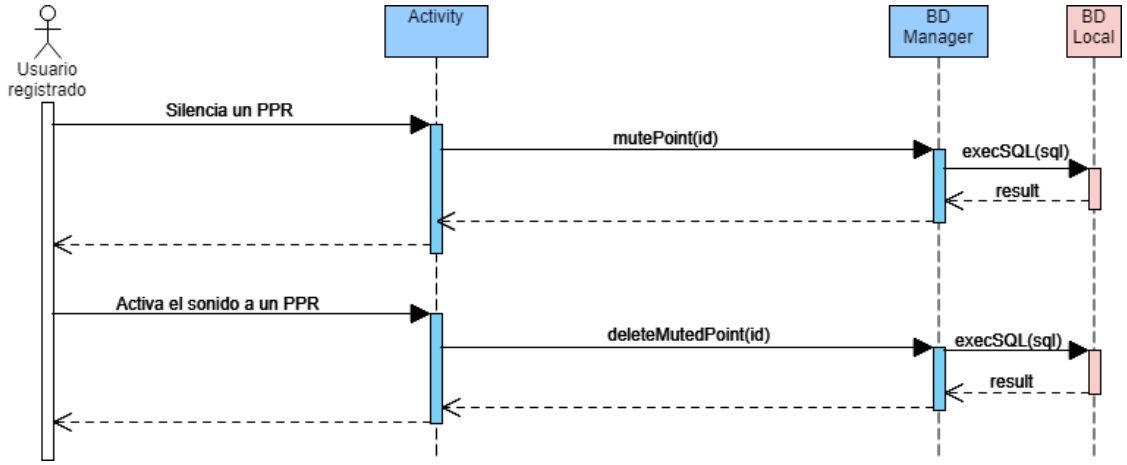
La SQL correspondiente a la función *addResults(data)* es la siguiente:

```
'INSERT INTO answer SET idUser=%idUser%, idPoint=%idPoint%,
attemptsPre=%attemptsPre%, correctsPre=%correctsPre%,
errorsPre=%errorsPre%, attemptsPost=%attemptsPost%,
correctsPost=%correctsPost%, errorsPost=%errorsPost%,
stressLevelPre=%stressLevelPre%, stressLevelPost=%stressLevelPost%,
happinessLevelPre=%happinessLevelPre%,
happinessLevelPost=%happinessLevelPost% '
```

Las SQLs correspondientes a la función *cleanTempData()* es la siguiente:

```
'DELETE FROM Game'

'DELETE FROM Stress'

'DELETE FROM Happiness'

'DELETE FROM Answer'

'DELETE FROM ParticipatingPoint'
```

11.2.5. Gestionar puntos silenciados



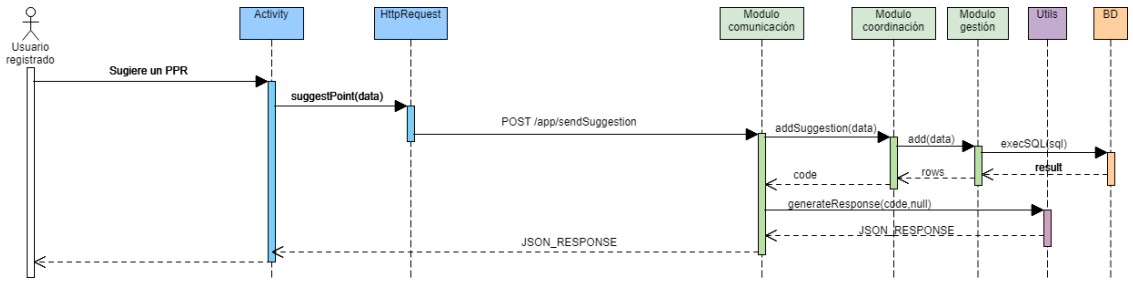
La SQL correspondiente a la función *mutePoint(id)* es la siguiente:

```
'INSERT INTO MutedPoint SET id=%id%'
```

La SQL correspondiente a la función *deleteMutedPoint(id)* es la siguiente:

```
'DELETE FROM MutedPoint WHERE id=%id%'
```

11.2.6. Sugerir punto



La SQL correspondiente a la función *add(data)* es la siguiente:

```
'INSERT INTO suggestion SET reason=%reason%, lat=%lat%, lng=%lng%,
scheduled=%scheduled%'
```