Aplicación móvil. Calendario de eventos localizados

Javier Bravo Domínguez

Grado en Tecnologías de telecomunicación

Consultor: Francesc Puigvert Pell

Fecha de entrega: 05/06/2018

Resumen del trabajo

En la actualidad, cada vez es más frecuente realizar muchos de nuestros hábitos a través de aplicaciones instaladas en los dispositivos inteligentes. Además, es habitual recibir información personalizada basada en nuestra localización. Por este motivo, el presente proyecto pretende diseñar e implementar una aplicación móvil que aproveche el potencial de la geolocalización.

El objetivo del proyecto es desarrollar una aplicación nativa para el sistema operativo Android que ofrezca información basándose en la localización del usuario. Para ello, se ha utilizado el lenguaje de programación Java y el IDE (*Integrated Development Environment*) Android Studio el cual facilita la depuración y compilación del proyecto.

Además, se utiliza la API (*Application Programming Interface*) de localización de Android que permite localizar a un usuario y crear y monitorear *geofences.* También, se utiliza la API de Google Maps que posibilita el almacenamiento y gestión de localizaciones.

Como resultado, se ha obtenido una aplicación que utiliza las *geofences* de los lugares de interés y la localización para mostrar los museos, galerías de arte, actividades, restaurantes y cargadores más cercanos. Además, los datos mostrados no solo se basan en la localización sino también en la ventana temporal.

Palabras clave: Android, Geolocalización, Geofences.

Abstract

Currently, it is increasingly common to make many of our daily habits through applications installed on our smart devices. Besides, it is customary to receive personalized information based on our location. Therefore, the present project aims to design and implement a mobile application that leverages the potential of geolocation.

The aim of this project is to develop a native application for the Android operating system that offers information based on the user's location. To do this, the Java programming language and the Android Studio IDE (Integrated Development Environment) which facilitates the debugging and compilation of the project have been used.

Furthermore, the API (Application Programming Interface) of Android localization is used to locate a user and to create and monitor geofences. Also, the Google Maps API which allows the storage and the localization management is used.

As a result, an application, which uses the geofences of the places of interest and the location to show the nearest museums, art galleries, activities, restaurants and battery chargers, has been obtained. In addition, the displayed data is not only based on the location but also on the time window.

Keywords: Android, Geolocation, Geofences

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

1. Introducción

El objetivo de este capítulo es contextualizar y definir el plan de trabajo a seguir para el desarrollo del proyecto “Aplicación móvil. Calendario de eventos localizados”.

Para ello, a continuación, se especifica tanto el objetivo principal como los objetivos específicos, las tareas a realizar y una planificación por días donde se indican las fechas de inicio y fin de cada actividad.

Además, se incluye un plan de contingencia para evitar retrasos en las entregas en caso de que ocurra alguna incidencia durante el desarrollo.

También, recoge el material necesario para llevar a cabo el proyecto y realizar las pruebas necesarias, así como un breve resumen del resto de capítulos.

1.1. Contexto

En los últimos años el uso de los *smartphones* ha experimentado un crecimiento espectacular en todo el mundo. Según un estudio de Google junto a Kantar TNS el número de españoles que poseen un teléfono inteligente es del 81%, el doble que hace 6 años. [1]

Además, el uso de las aplicaciones también ha aumentado a la vez que el uso de teléfonos inteligentes, pues según el estudio realizado por INTSI en 2017, el 40% de la población lo utiliza de forma cotidiana y el 14.1% lo hace de forma semanal. [2]

Estos hechos, han cambiado nuestros hábitos como la forma de comprar, consumir contenidos, informarse sobre un producto o incluso la forma de relacionarnos. Por este motivo, las empresas e instituciones se suman a esta demanda y han transformado la forma de comunicarse con los consumidores.

La geolocalización es una herramienta muy utilizada para conseguir una comunicación eficaz, pues permite segmentar intereses (gastronomía, entretenimiento, etc.) y características de consumo basándose en las localizaciones de los usuarios.

Debido a esto, las aplicaciones que utilizan la geolocalización para ofrecer contenidos son muy habituales en los *smartphones*.

El objetivo de este trabajo de fin de grado es diseñar e implementar una aplicación para dispositivos inteligentes Android que aproveche el potencial de las *geofences* y la geolocalización para promocionar lugares.

1.1.1. Formulación del problema

Una ciudad celebra durante el fin de semana un festival en el cual hay programados diferentes espectáculos. Cada espectáculo tiene una hora y una localización determinada.

Esta localidad, quiere aprovechar el evento para promocionar los puntos de interés del lugar y los restaurantes locales.

Para ello, se quiere proporcionar a los visitantes una aplicación móvil que facilite el acceso a la agenda de los acontecimientos del festival, así como los puntos de interés del municipio.

1.1.2. Definición del proyecto

Calendario de eventos es una aplicación para el sistema operativo Android que permite ver la agenda de acontecimientos de un festival, así como los puntos de interés de una localidad. Además, este desarrollo, también muestra los puntos de carga móvil de la localidad.

Para lograrlo, la aplicación está configurada para monitorear los perímetros de los lugares de las actividades y de los puntos de interés y detecta con ayuda del GPS (*Global Position System*) cuando una persona entra o sale de la zona delimitada.

Los requisitos funcionales de los que consta este proyecto son:

- Proponer actividades en función de la localización del usuario y la ventana temporal.
- Proponer restaurantes entre las 12:00 y las 16:00 y las 19:00 y 22:00.
- Proponer puntos de carga móvil cuando la batería necesite recargarse.

1.2. Objetivos

1.2.1. Objetivo principal

- Desarrollar una aplicación móvil para el sistema operativo Android que ofrezca información en función de la localización del usuario.
- Utilizar una plataforma web para almacenar localizaciones.

1.2.2. Objetivos específicos

- Aprender el lenguaje de programación JAVA.
- Aprender el funcionamiento del programa Android Studio.
- Aprender a localizar a un usuario.
- Aprender a utilizar la localización de un usuario.
- Aprender a crear y monitorizar *geofences*.
- Aprender a almacenar localizaciones en un servicio web.

1.3. Enfoque y método seguido

Para realizar este proyecto se ha decidido desarrollar un nuevo producto que cumpla con todos los requisitos necesarios para solucionar el problema planteado.

Para ello, primero se analizarán las numerosas alternativas existentes en Play Store, tienda de aplicaciones de Google, que utilizan la geolocalización y las *geofences*. Estas se tomarán como referencia y a partir de ahí se desarrollará una nueva que incluirá tanto características de dichas aplicaciones como nuevas funcionalidades.

Para facilitar la programación, y como es habitual entre desarrolladores, se reutilizará parte del código proporcionado por Google en su página Android Developers. Dicho código será adaptado al proyecto, pues habrá partes que no sean interesantes y otras que sea necesario ampliar y/o modificar.

Siguiendo esta metodología, se podrán lograr todos los objetivos correctamente y, además, mejorar algunas funcionalidades. Asimismo, con la reutilización de código se podrá programar de forma más fácil todas las partes relacionadas con la API (*Application Programming Interface*) de localización.

1.4. Planificación del proyecto

1.4.1. Tareas

Para el desarrollo de este proyecto se han definido una serie de tareas que se enumeran a continuación:

Tarea 0	**Definición del proyecto**
Definición y planificación del proyecto	1. Leer la documentación del proyecto. 2. Establecer plan de trabajo
Tarea 1	**Definición del entorno**
Definición del origen de los datos y selección de tecnología móvil	1. Definir objetivos del proyecto 2. Realizar planificación 3. Elaborar plan de contingencia.
Tarea 2	**Análisis funcional de la aplicación**
Análisis funcional del sistema	1. Definición de la plataforma adoptada y estudio de las diferentes opciones de desarrollo disponibles. 2. Configuración del entorno de trabajo. 3. Redactar memoria. 4. Descripción funcional de los algoritmos a programar. 5. Creación del prototipo de la interfaz de usuario. 6. Revisión de la PEC 2. 7. Entrega de la PEC 2.
Tarea 3	**Diseño de la aplicación**
Diseño del sistema	1. Análisis de bibliotecas y objetos que se incluirán. 2. Análisis de clases y funciones a desarrollar. 3. Creación de servicios web. 4. Redactar memoria. 5. Revisión PEC 3. 6. Entrega PEC 3.
Tarea 4	**Construcción del sistema**
Programación de la aplicación	1. Programación de la aplicación. 2. Ejecución de pruebas. 3. Redacción de memoria 4. Corrección de posibles errores 5. Elaboración de la presentación 6. Revisión PEC 4 7. Entrega PEC 4

Tabla 1: Tareas

1.4.2. Planificación

Para realizar la planificación de este proyecto se ha tenido en cuenta el calendario de entregas y la disponibilidad diaria. Por este motivo, para conseguir completar con éxito el proyecto cumpliendo los plazos de entrega se ha estimado una dedicación de 3 horas diarias durante 99 días.

En la siguiente tabla se desglosan las tareas y se establecen los hitos que han de cumplirse. Además, se indica el tiempo de dedicación y las fechas de inicio y fin de las tareas.

Nº	Tarea	Duración (Días)	Fecha inicio	Fecha fin
0	Leer la documentación del proyecto	1	26/02/18	26/02/18
1	**PEC 1 - Elaboración del plan de trabajo**	12	26/02/18	09/02/18
1.1	Definir objetivos del proyecto	2	26/02/18	27/02/18
1.2	Realizar planificación	6	28/02/18	05/03/18
1.3	Elaborar plan de contingencia	3	05/02/18	08/03/18
1.4	Entregar PEC 1	1	08/02/18	09/03/18
2	**PEC 2 – Análisis funcional de la aplicación**	31	10/04/18	08/04/18
2.1	Definición de la plataforma adoptada y estudio de las diferentes opciones de desarrollo disponibles.	5	10/03/18	14/03/18
2.2	Configuración del entorno de trabajo	1	15/03/18	15/03/18
2.3	Redactar memoria	8	15/03/18	22/03/18
2.4	Descripción funcional de los algoritmos a programar	11	15/03/18	25/03/18
2.5	Creación de prototipo de la interfaz	9	26/03/18	03/04/18
2.6	Revisión PEC 2	4	31/03/18	03/04/18
2.7	Entregar PEC 2	1	08/04/18	08/04/18
3	**PEC 3 – Diseño del sistema**	31	09/04/18	08/05/18
3.1	Análisis de bibliotecas y objetos	11	09/04/18	19/04/18
3.2	Análisis de clases y funciones a desarrollar	11	20/04/18	30/04/18
3.3	Creación de servicios web	5	30/04/18	05/05/18
3.4	Redactar memoria	6	30/04/18	05/05/18
3.5	Revisión PEC 3	2	05/05/18	07/05/18
3.6	Entrega PEC 3	1	08/05/18	08/05/18
4	**PEC 4 – Construcción del sistema**	29	09/05/18	05/06/18
4.1	Programación de la aplicación	20	09/05/18	28/05/18
4.2	Ejecución de pruebas	2	28/05/18	30/05/18
4.3	Redacción de memoria y resumen del TFG	8	28/05/18	04/06/18

4.4	Corrección de posibles errores	6	30/05/18	03/06/18
4.5	Elaboración de la presentación y vídeo	3	01/06/18	05/05/18
4.6	Revisión de la PEC 4	2	04/05/18	05/05/18
4.7	Entrega de la PEC 4	1	05/06/18	05/06/18

Tabla 2: Planificación

Es importante destacar que la planificación anterior no es rígida y, por lo tanto, según la evolución de las tareas puede ser revisada y adaptada.

1.4.3. Productos y resultados

Para el desarrollo de este proyecto se han establecido 4 hitos en los cuales se entregará lo siguiente:

1.4 - PEC 1: Entrega del plan de trabajo.

2.7 - PEC 2: Entrega del borrador de la memoria redactada parcialmente y del prototipo de la interfaz.

3.6 - PEC 3: Entrega del borrado II de la memoria y de las pruebas de los servicios web.

4.7 - PEC 4: Entrega de la memoria completa, presentación virtual y código de la aplicación.

Por lo tanto, al finalizar el desarrollo de este proyecto se obtendrán los siguientes documentos:

- Plan de trabajo: en este documento se definen los objetivos del proyecto, la planificación y el plan de contingencia.
- Análisis funcional de la aplicación: en este documento se analiza el estado del arte, se explica la configuración del entorno de trabajo, se define el prototipo y los casos de uso.
- Diseño del sistema: este escrito analiza las bibliotecas y objetos que se utilizarán en el desarrollo de la aplicación. Además, analiza las clases y funciones a desarrollar.

Asimismo, como resultado del desarrollo se ha obtenido el siguiente producto final:

- Aplicación para Android “Calendario de eventos”

1.4.4. Diagrama de Gantt

A continuación, con el objetivo de mostrar el tiempo de dedicación a las tareas y la relación entre ellas de forma clara se adjunta el diagrama de Gantt general del proyecto junto con los de cada PEC.

Planificación general del proyecto:

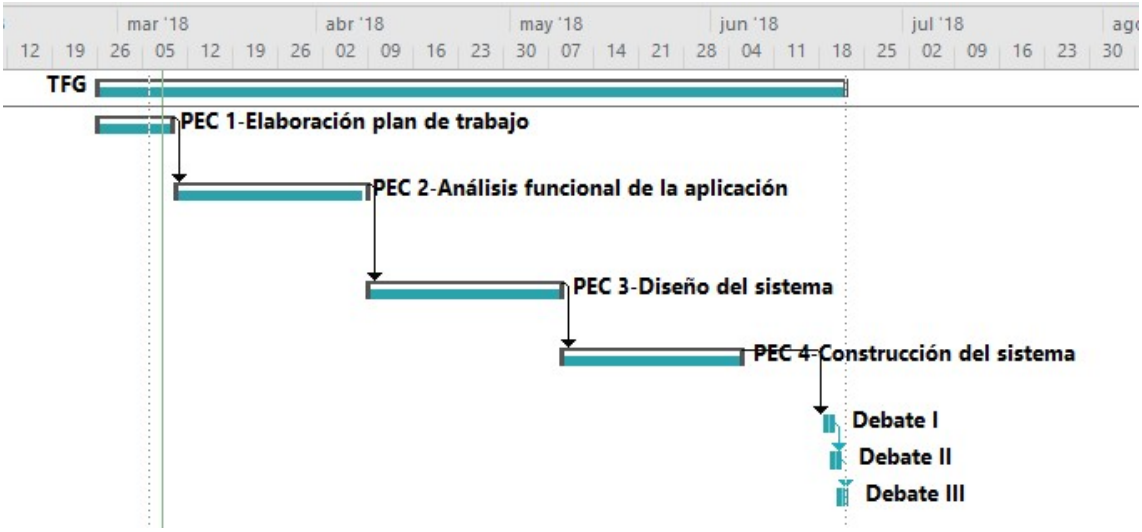

Figura 1: Diagrama de Gantt general

Planificación PEC 1:

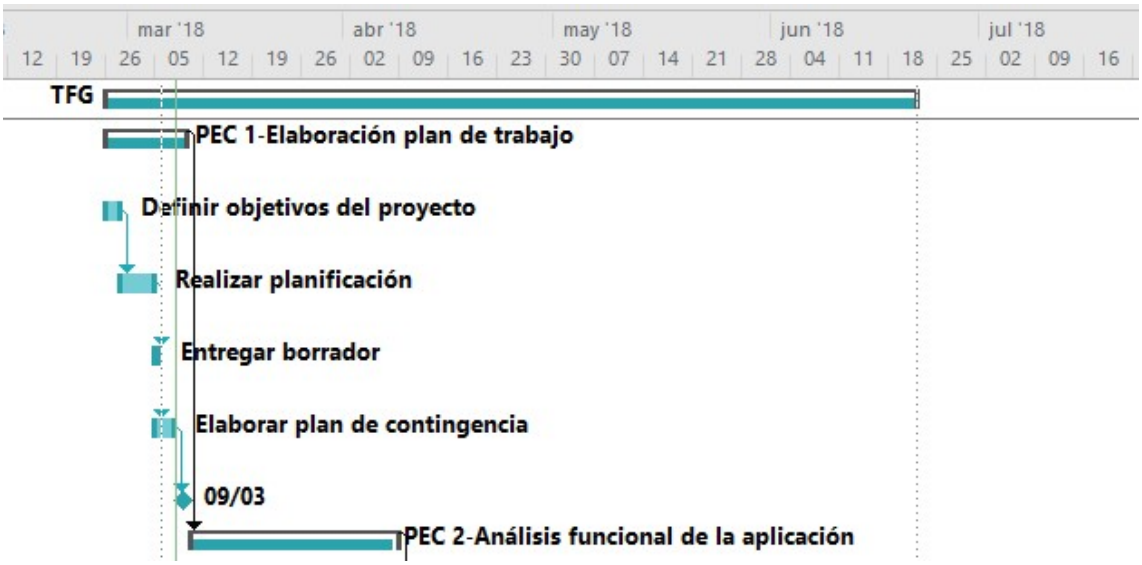

Figura 2: Diagrama de Gantt PEC 1

Planificación PEC 2:

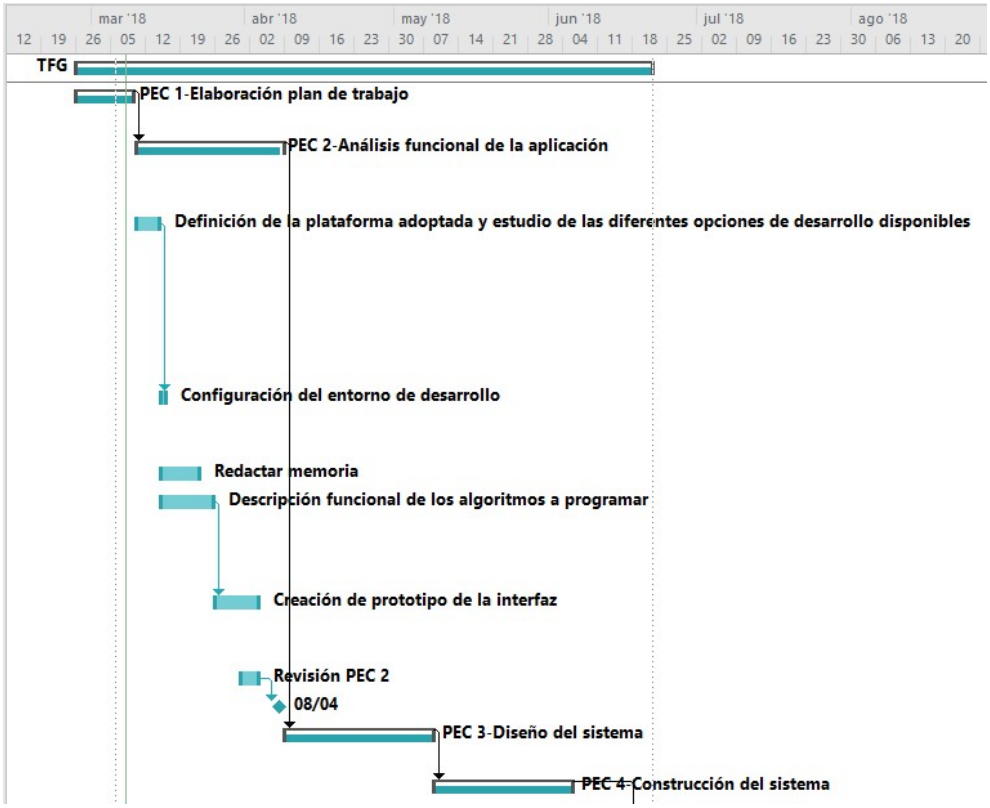

Figura 3: Diagrama de Gantt PEC 2

Planificación PEC 3:

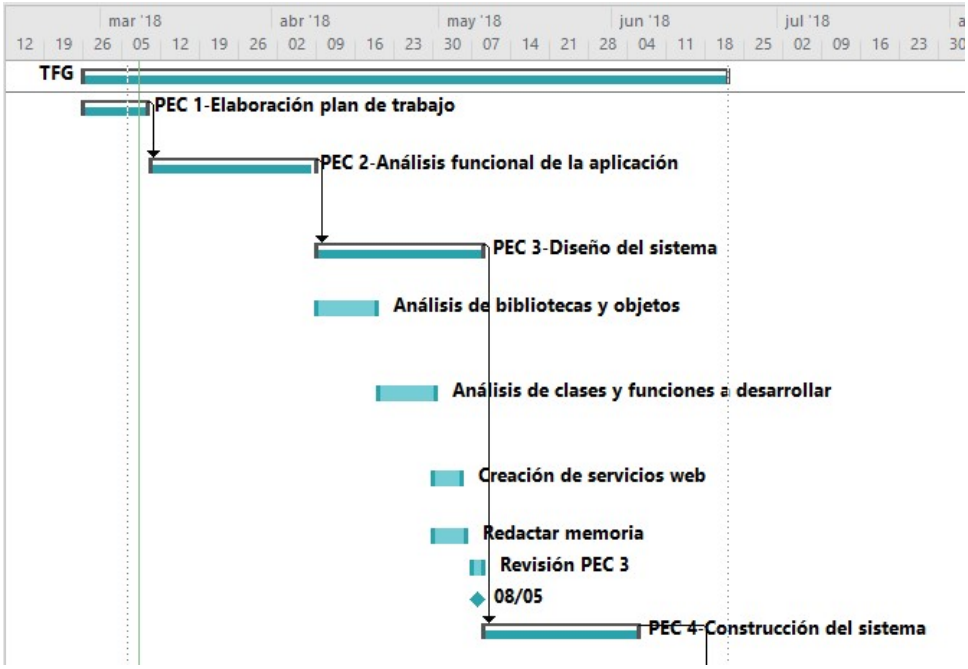

Figura 4: Diagrama de Gantt PEC 3

Planificación PEC 4:

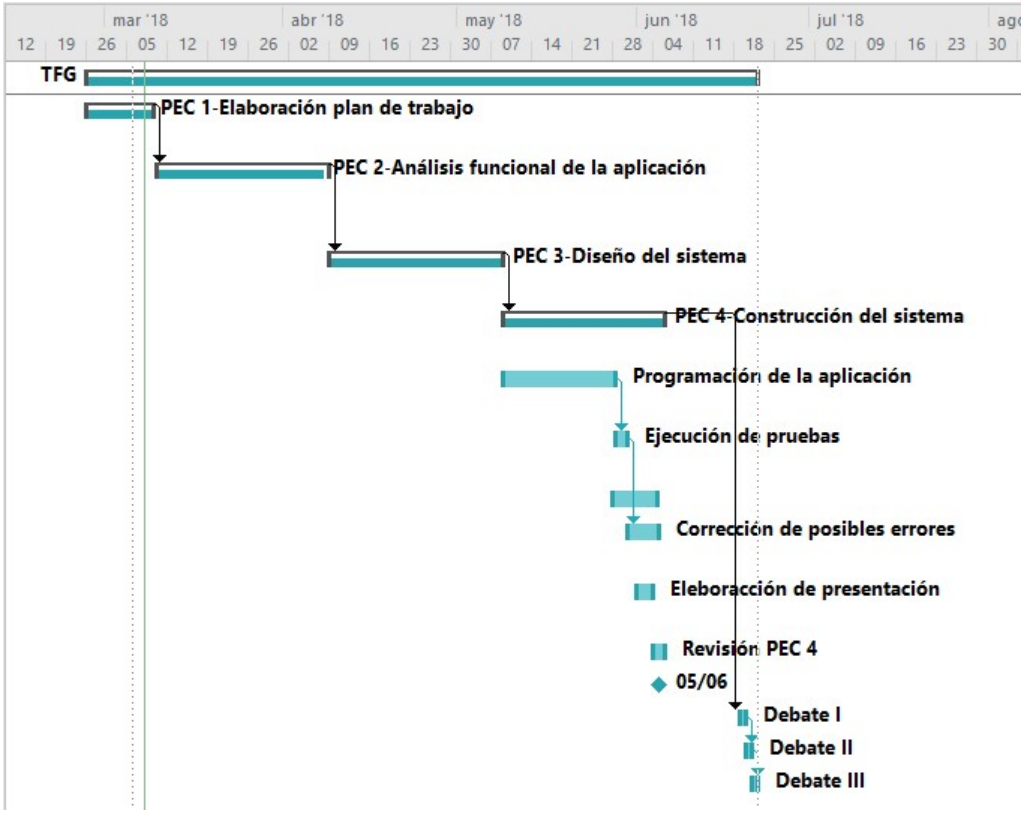

Figura 5: Diagrama de Gantt PEC 4

1.4.5. Incidencias y riesgos

En este apartado se analizan los posibles riesgos e incidencias que puede haber durante el desarrollo del proyecto y se presenta un plan de contingencia para minimizar las consecuencias derivadas de dichos problemas.

Descripción del problema	Plan de contingencia
Avería del equipo con el que se trabaja	Todo el software necesario para realizar este proyecto estará instalado en un segundo equipo que podrá utilizarse en caso de fallo del primero. Además, se realizarán copias de seguridad diarias del proyecto para evitar pérdidas de información.
Impacto	Bajo

Descripción del problema	Plan de contingencia
Enfermedad	Revisión del plan de trabajo. Aumento de horas de dedicación una vez recuperado.
Impacto	Alto

Descripción del problema	Plan de contingencia
Incremento de carga horaria en el trabajo	Revisión del plan de trabajo. Aumento de horas de dedicación los días de descanso.
Impacto	Medio

Descripción del problema	Plan de contingencia
Imprevistos que impidan cumplir temporalmente el plan de trabajo del proyecto (viajes de trabajo, aumento de horas de estudio para otras asignaturas, etc.)	Aumentar las horas de dedicación en cuanto sea posible hasta recuperar el ritmo. Si es necesario se pedirán vacaciones.
Impacto	Medio

Descripción del problema	Plan de contingencia
Vacaciones y días festivos	Aumentar las horas de dedicación en cuanto sea posible.
Impacto	Bajo

Tabla 3: Incidencias, riesgos y plan de contingencia

1.4.6. Recursos y materiales

1.4.6.1 Hardware

- Ordenador personal desde el cual se realizará tanto la memoria como el desarrollo de la aplicación.
- Dispositivos GPS Android en el cual se realizarán las pruebas.
    - Xiaomi Redmi Note 3
        - Android 6.0 Marshmallow
        - Pantalla 5.5 pulgadas

1.4.6.2 Software

- Microsoft Word: se utilizará para realizar la memoria.
- Microsoft Project: se utilizará para realizar la planificación del proyecto.
- Android Studio: se utilizará para desarrollar la aplicación.
- JustMind: se utilizará para realizar el prototipo de la aplicación.
- Microsoft PowerPoint: se utilizará para realizar la presentación final.
- Avidemux: se utilizará para realizar tareas de edición de video.

1.5. Breve descripción de los otros capítulos

El capítulo 2 se corresponde con el estado del arte. En dicho capítulo se explican los conceptos más importantes que se utilizan en el desarrollo de este proyecto. Además, se analizan otras aplicaciones disponibles en el mercado que utilizan las *geofences* y la localización.

En el capítulo 3, análisis funcional de la aplicación, se detalla la elección de la tecnología y la configuración del entorno de trabajo. Asimismo, se especifican las funcionalidades de la aplicación, los casos de uso y se presenta el prototipo.

En el capítulo 4, diseño del sistema, se analizan las clases y librerías necesarias para la implementación. Además, se detallan las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación

Por último, tenemos las conclusiones del proyecto y las líneas de trabajo futuras, el glosario y la bibliografía.

Capítulo 2

2. Estado del arte

2.1. Introducción

Para realizar este proyecto es necesario analizar la tecnología y el software existente relacionado con el campo de la localización y los *smartphones.* Además, se deben conocer algunos conceptos relacionados. Por este motivo, en este capítulo, se analizan y explican los más importantes.

2.2. ¿Qué es una API?

Una API (*Application Programming Interface*) es un conjunto de protocolos y funciones que permiten la comunicación entre dos *softwares* para que intercambien información de manera segura. De esta manera, los desarrolladores pueden acceder al *software* y *hardware* de los dispositivos inteligentes usando funciones predefinidas y sin tener que programar desde cero. [3] [4]

2.3. ¿Qué es Android?

Android es un sistema operativo multiplataforma, libre y gratuito basado en el núcleo de Linux. Fue diseñado para dispositivos inteligentes como, por ejemplo, *smartphones*, tabletas, televisiones, relojes, etc. Actualmente es el sistema operativo más extendido a nivel mundial, pues posee una cuota de mercado aproximada del 81.7% según un estudio realizado por Gatner en 2017. [5] [6] [7]

Además, posee numerosas ventajas que benefician tanto a los usuarios como a los desarrolladores. Entre estas virtudes destacan:

- Está liberado bajo la licencia Apache, una licencia libre y de código abierto que permite a los desarrolladores mejorar su código sin depender de fabricantes y detectar errores de forma más rápida.
- Es completamente personalizable.
- Es un sistema multitarea, pues gestiona varios procesos de forma simultánea.

- Utiliza aplicaciones desarrolladas en JAVA, lenguaje muy popular en el mundo de los desarrolladores.
- Incluye numerosas API (*Application Programming Interface*) que permiten el acceso al *hardware* del dispositivo, a las comunicaciones por NFC (*Near Field Communication*) o bluetooth, ubicación por GPS (*Global Positioning System*), etc.
- Posee una gran comunidad de desarrolladores que facilitan el desarrollo de aplicaciones. [8] [9] [10]

2.4. ¿Qué es la localización?

La localización permite ubicar un objeto o persona en un determinado espacio. Dicho espacio necesita coordenadas que aporten puntos de referencia y permitan saber los elementos que tiene cerca como, por ejemplo, carreteras, ríos, calles, etc.

Para conocer la localización es necesario hacer uso de los datos geográficos de referencia y de los sistemas de posicionamiento como el GPS, Galileo, etc. [11]

2.5. Sistemas de posicionamiento en *smartphones*

Los *smartphones* más modernos incorporan varias tecnologías de posicionamiento como Beidu, GLONASS, Galileo, GPS, A-GPS, y sistemas de posicionamiento terrestres basados en redes de telecomunicaciones.

En este documento solo se van a tratar las más utilizadas actualmente en Europa que son GPS, A-GPS y los sistemas de posicionamiento terrestres basados en redes de telecomunicaciones. [12]

2.5.1. GPS

El GPS (*Global Position System*) es un sistema de radionavegación por satélite que permite el posicionamiento y navegación de forma fiable. Está compuesto por 30 satélites entre los operativos y los de reserva y su diseño permite que desde cualquier punto de la tierra se pueda recibir señal de al menos 4 satélites.

Para determinar la posición, se utiliza el método de *trilateración* que consiste en determinar las distancias desde el dispositivo hasta los satélites. Para calcular estas distancias, el receptor GPS calcula el tiempo que tardan en llegar las señales comparando el momento de la transmisión con el momento de recepción y por último

multiplica el resultado por la velocidad de la luz. Una vez obtenida la distancia desde como mínimo 3 satélites se determina la longitud y la latitud. Un cuarto satélite permitiría calcular la altura. [11] [13] [14]

2.5.2. A-GPS

A-GPS *(Assisted Global Positioning System)* consigue el posicionamiento de forma más rápida que el GPS. Para lograrlo, combina los datos obtenidos de un servidor externo con los que recibe a través de los satélites de la siguiente manera:

En primer lugar, el *smartphone* envía a un servidor externo la identificación de la celda a la que está conectado. A continuación, el servidor le devuelve la lista de los satélites que tiene encima dicha antena. Por último, el receptor combina esta información con la recibida mediante el GPS y consigue el posicionamiento. [11] [15] [16]

2.5.3. Sistemas de posicionamiento terrestres basados en redes de telecomunicaciones

Este tipo de sistema consigue el posicionamiento utilizando cualquier tecnología de comunicación como, por ejemplo, el bluetooth, WLAN, la telefonía móvil, etc. Para ello, utiliza principalmente los siguientes métodos:

- Identificación de la estación base a la que el dispositivo está conectado.
- Medición de la fuerza con la que se recibe una señal.
- Medición del tiempo que tardan en llegar señales provenientes de tres o más estaciones.
- Medición del ángulo de llegada de la señal de dos estaciones base. [11]

2.6. ¿Qué son las *geofences*?

Una *geofence* es un perímetro virtual de un área geográfica real. Este perímetro se puede monitorear y cuando un usuario entra en la zona se genera una alerta.

El funcionamiento es sencillo, pues en primer lugar se define el área geográfica que se quiere controlar de manera virtual. A continuación, se vincula con una aplicación que notificará cuando un dispositivo entra o sale del perímetro. Por último, se realiza el seguimiento a través de los sistemas de posicionamiento como, por ejemplo, el GPS. [17]

Hoy en día, el *geofencing* es muy utilizado por las empresas, pues les permite realizar campañas publicitarias muy efectivas y ofrecer servicios adicionales que las diferencien de la competencia. También es muy utilizado como método de control para dar seguimiento a las zonas por las que se mueven los trabajadores. [18] [19] [20]

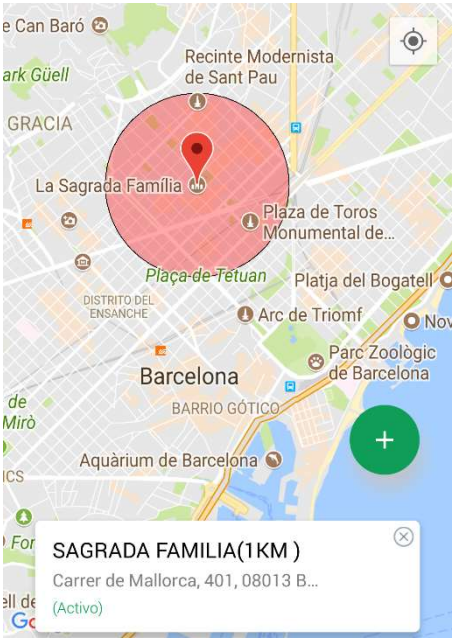

Figura 6: Ejemplo geofence

2.7. Aplicaciones que utilizan *geofences*

En Google Play Store, tienda oficial de Android, existen numerosas aplicaciones que utilizan la tecnología de localización y *geofences*. En este aparatado se van a analizar algunas de ellas para conocer con detalle su funcionamiento.

- **Geofence memo**: se trata de una aplicación que permite configurar alertas que se mostrarán al entrar o salir del área elegida.
  Su funcionamiento es sencillo, pues simplemente hay que elegir el lugar y el radio de la *geofence* y el mensaje que queremos que muestre al cruzar los límites del área predefinida.

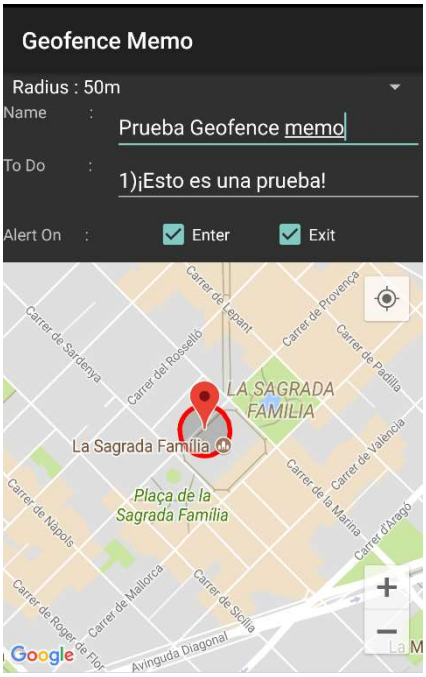

Figura 7: Configuración de la aplicación Geofence memo

- **GeoAlert**: Esta aplicación permite configurar *geofences* y emite una señal de aviso cuando se entra o sale de alguna de ellas. Además, da la posibilidad de cambiar automáticamente el perfil de sonido al cruzar la barrera virtual. Para que funcione, solo hay que configurar el radio y el lugar donde queremos crear la *geofence* y si queremos cambiar de perfil o no. Asimismo, ofrece la posibilidad de notificar con sonido.

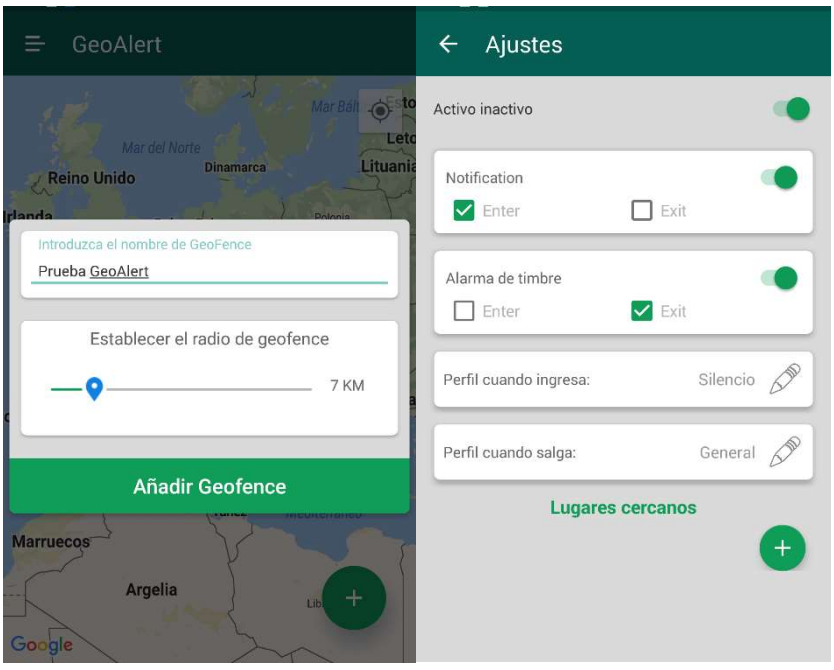

Figura 8: Configuración de geofence en GeoAlert

2.8. ¿Qué es un IDE?

Un IDE (*Integrated Development Environment*) es una aplicación que facilita a los programadores el desarrollo de *software*. Estos programas ayudan en la edición, depuración y compilación de código a los desarrolladores.

Las características más importantes de estos programas son:

- Son multiplataforma.
- Tiene reconocimiento de sintaxis.
- Poseen un depurador de código.
- Soporte para múltiples lenguajes de programación.
- Permiten la instalación de complementos y extensiones.

En el mercado existen numerosas versiones de este tipo de *software* entre los que destacan:

- Eclipse: es uno de los más extendidos para programar en JAVA, aunque permite programar en otros lenguajes. Además, es software libre.
- Android Studio: muy extendido entre programadores de aplicaciones Android.
- NetBeans: también está muy extendido entre los desarrolladores de JAVA. Asimismo, es software libre. [21]

2.9. Aplicaciones nativas, web o híbridas

Existen tres tipos de aplicaciones dependiendo de la tecnología de desarrollo utilizada:

- Nativa: son las aplicaciones desarrolladas exclusivamente para cada sistema operativo. Además, son programadas en el lenguaje oficial de dichos sistemas. Consiguen una buena experiencia de uso y son más costosas de desarrollar.
- Web: se programan en lenguajes como HTML5, JavaScript, CSS o HTML y se adaptan a cualquier sistema operativo. Son más económicas a la hora de ser desarrolladas, pero ofrecen una peor experiencia de uso y requieren el uso de internet.

- Híbridas: combinan aspectos de las aplicaciones nativas y de las aplicaciones web. Son desarrolladas en lenguajes como HTML5, JavaScript, CSS o HTML con lo cual se adaptan a cualquier sistema operativo. Asimismo, son capaces de acceder a las funcionalidades del dispositivo tal como lo hacen las nativas. [22] [23] [24]

Capítulo 3

3. Análisis funcional de la aplicación

3.1. Introducción

En este capítulo se explica la tecnología y el software que se va a emplear para el desarrollo del proyecto. Además, se detalla la configuración del entorno de trabajo y la descripción funcional. Por último, se incluye el prototipo de la aplicación y los casos de uso.

3.2. Elección de tecnología

3.2.1. Versión de Android

Para realizar este proyecto se ha elegido desarrollar una aplicación nativa para la plataforma Android. Esta elección tiene numerosas ventajas frente a aplicaciones web o aplicaciones híbridas como:

Al estar desarrollada y optimizada específicamente para Android se adapta al 100% con las características y funcionalidades del dispositivo.

- Mejor experiencia de uso
- Uso de memoria optimizado
- Mayor velocidad
- Mejor rendimiento [24]

Según el sitio web de Google, Android Developers, a fecha de 8 de enero de 2018 el 82.3% de los usuarios utilizan una versión igual o superior a Android 5.0. Por este motivo, la aplicación se desarrollará para ser compatible con dispositivos que ejecuten la versión Android 5.0 (Lollipop) o superior. De esta forma, se pueden aprovechar las características de la API 21, incluida en Lollipop y llegar a la gran mayoría del público. [25]

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.6%
4.3		18	0.7%
4.4	KitKat	19	12.0%
5.0	Lollipop	21	5.4%
5.1		22	19.2%
6.0	Marshmallow	23	28.1%
7.0	Nougat	24	22.3%
7.1		25	6.2%
8.0	Oreo	26	0.8%
8.1		27	0.3%

Datos recopilados durante un período de 7 días hasta 8/1/2018.
No se muestran versiones con una distribución inferior al 0,1%.

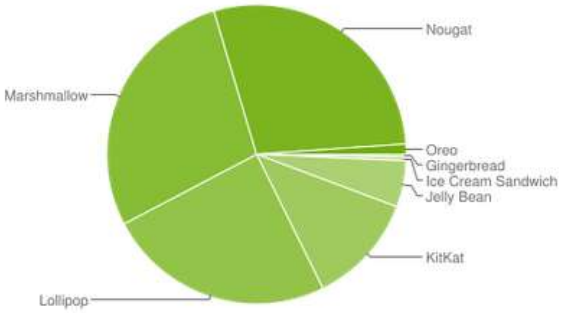

Figura 9: Cuota de las diferentes versiones de Android

Fuente: Android Developers [25]

3.2.2. IDE

Además, se utilizará el IDE (Entorno de Desarrollo Integrado o Entorno de Desarrollo Interactivo) Android Studio 3.1. Este IDE ha sido creado para programar en Android y, por lo tanto, posee mayor rendimiento que otros IDE similares.

Asimismo, posee una serie de ventajas que otros no tienen como, por ejemplo, el código está más ordenado y estructurado, posee mayor facilidad para diseñar interfaces y exporta más fácilmente los archivos APK. [25]

3.2.3. API Google Maps

Para mostrar las *geofences* es necesario que se incluya en la aplicación un servidor de mapas que permita utilizar su base de datos. Para ello, se ha elegido la API de Google Maps, pues permite conectar fácilmente la aplicación desarrollada con la base de datos de Google Maps y de esta forma acceder a sus funcionalidades.

3.2.4. Localización

Para obtener la localización es necesario utilizar el sistema de permisos que incluye Android. Por este motivo, se utiliza el permiso "android.permission.ACCESS_FINE_LOCATION" que permite acceder a la ubicación precisa del dispositivo y que hace uso de las tecnologías GPS, A-GPS y localización mediante el uso de redes de telecomunicaciones. [25]

3.3. Configuración del entorno

En este apartado se resume brevemente la configuración del entorno de trabajo en los diferentes dispositivos.

En primer lugar, se ha instalado Android Studio 3.1 en un ordenador personal. Este software se puede descargar gratuitamente desde la página de desarrolladores de Android donde también se puede consultar una guía de instalación.

Una vez instalado, y con ayuda del asistente para crear un nuevo proyecto, se han seleccionado los dispositivos a los que va dirigida la aplicación (*phone y tablet*) y la API que se utilizará (API 21 Android 5.0 Lollipop). [25]

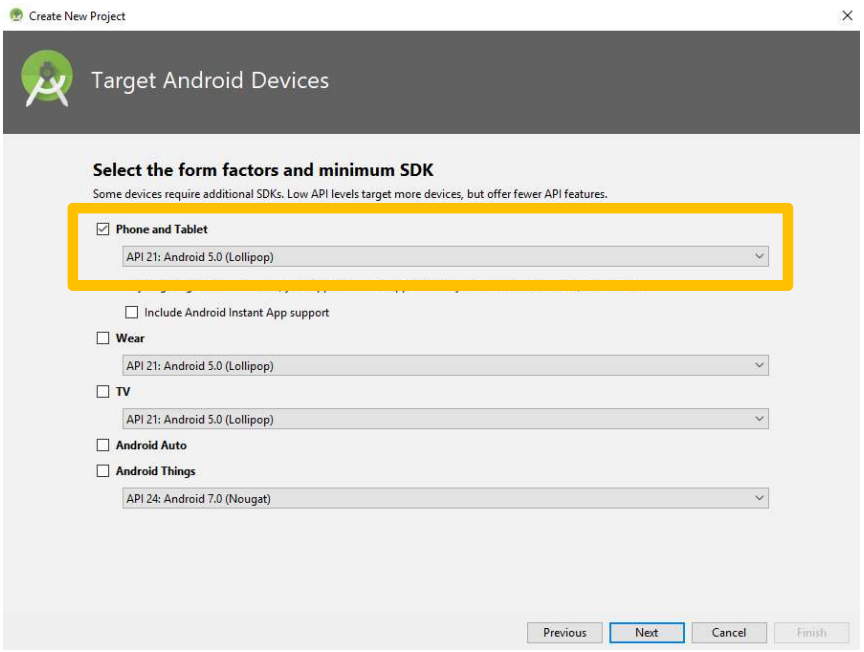

Figura 10: Configuración de Android Studio

En segundo lugar, se ha configurado el dispositivo Android con el cual se realizarán las pruebas. Se trata de un *smartphone* Xiaomi Redmi Note 3 con Android 6.0. Para configurarlo se han habilitado las opciones de desarrollador y se ha activado el modo de depuración. Además, se ha habilitado la opción "Instalar vía USB" que permite la instalación en el dispositivo de archivos APK desde Android Studio.

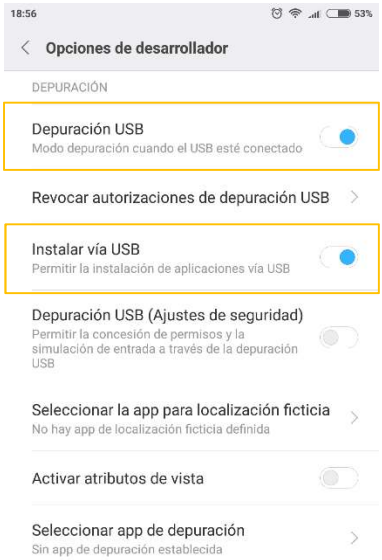

Figura 11: Opciones de desarrollador de Android 6.0

Por último, para utilizar la API de Google Maps, es necesario obtener una clave que más tarde se insertará en el código de la aplicación. Para obtenerla, basta con registrarse en la página de desarrolladores de Google (sección API Google Maps para Android). [26]

3.4. Funcionalidades

Para cumplir con los objetivos la aplicación debe contar con una serie de funcionalidades que se enumeran a continuación:

- Mostrar lugares cercanos disponibles: tiene que permitir ver los lugares de interés cercanos a la ubicación del usuario como, por ejemplo, museos, galerías de arte, actividades, etc.
- Mostrar restaurantes cercanos a la ubicación del usuario entre las 12:00 y las 16:00 y las 19:00 y las 22:00.
- Mostar puntos de recarga cercanos si la batería del dispositivo es baja.

3.5. Casos de uso

A continuación, se presentan los casos de uso que se han tenido en cuenta para desarrollar el proyecto. En ellos, se describen los pasos a seguir por un actor para realizar algún proceso dentro de la aplicación.

Diagrama de casos de uso:

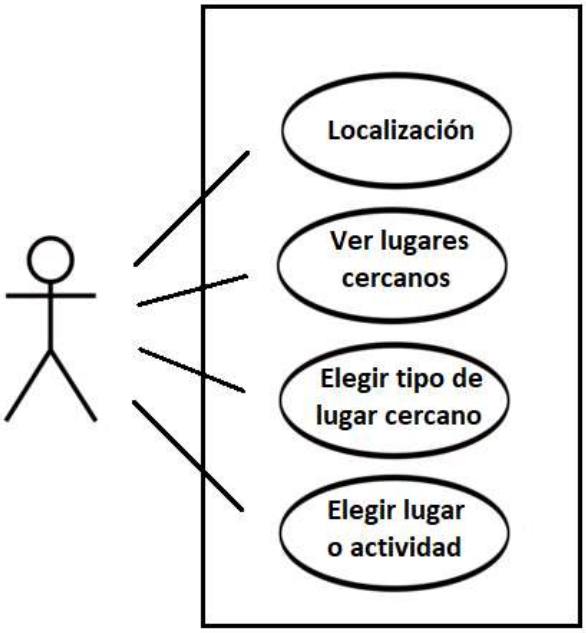

Figura 12: Diagrama de casos de uso

Código	CDU-001
Nombre	Localización
Actor	Usuario
Descripción	Permite localizar al usuario en tiempo real.
Precondición	GPS activado. Conexión a internet.
Secuencia normal	El usuario abre la aplicación. El sistema localiza al usuario.
Escenario	Muestra en pantalla un mapa con la localización del usuario en tiempo real.
Escenario alternativo	Si no hay permisos de ubicación concedidos el sistema los pedirá.

Código	CDU-002
Nombre	Ver tipos de lugares
Actor	Usuario
Descripción	Permite ver los tipos de lugares cercanos a la ubicación del usuario (Ejemplo: Museos, restaurantes, puntos de recarga, actividades, etc.)
Precondición	GPS activado Conexión a internet CU-001
Secuencia normal	El usuario pulsa el botón “Ver lugares cercanos”
Escenario	El sistema muestra los diferentes tipos de lugares cercanos disponibles.
Escenario alternativo	Ninguno.

Código	CDU-003
Nombre	Elegir tipo de lugar cercano
Actor	Usuario
Descripción	Permite elegir los tipos de lugares cercanos a la ubicación del usuario.
Precondición	GPS activado Conexión a internet CU-001 (Localización) CU-002 (Ver tipos de lugares)
Secuencia normal	El usuario pulsa sobre el tipo de lugar deseado.
Escenario	El sistema muestra la lista de actividades o lugares del tipo elegido.
Escenario alternativo	Si el sistema detecta que no hay lugares cercanos disponibles mostrará un aviso. Si el sistema detecta que no hay lugares disponibles en ese horario mostrará un aviso. Si el sistema detecta que el nivel de batería es superior al 20% mostrará un aviso (en la opción “cargadores”).

Código	CDU-004
Nombre	Elegir lugar o actividad
Actor	Usuario
Descripción	Permite elegir el lugar o actividad deseada y obtener información.
Precondición	GPS activado Conexión a internet CU-001 (Localización) CU-002 (Ver tipos de lugares) CU-003 (Elegir tipo de lugar cercano)
Secuencia normal	El usuario pulsa sobre el lugar o actividad elegida. El usuario pulsa sobre el marcador naranja en el mapa.
Escenario	El sistema señala con un marcador el lugar escogido en el mapa y al pulsar sobre él se muestra información sobre el evento.

Tabla 4: Casos de uso

3.6. Prototipo

El diseño de la aplicación se ha realizado con el software JustinMind Prototyper. Esta herramienta incluye una gran variedad de imágenes e iconos de Android que permiten diseñar un prototipo de alta fidelidad. Asimismo, ofrece la posibilidad de simular el resultado tanto en un ordenador como en un dispositivo móvil. De esta forma, se puede valorar de mejor manera los aspectos funcionales de la interfaz de la aplicación. [27]

A continuación, se muestra el resultado del prototipo de la aplicación y una descripción del funcionamiento de cada pantalla:

Pantalla inicial: se trata de la primera pantalla que se visualiza nada más abrir la aplicación. En ella se visualiza un mapa con la ubicación del usuario. Además, incluye el botón de buscar “Lugares cercanos”.

Categorías: En esta sección se muestran los botones de los diferentes tipos de lugares.

Lugares y actividades cercanos: esta pantalla muestra el listado de lugares o actividades cercanas dependiendo de la categoría elegida. Por ejemplo, si el usuario elige museos esta sección mostrará la lista de los más cercanos.

Mostrar lugar: esta sección, una vez que se ha elegido el lugar o actividad que desea, lo muestra en un mapa junto con la ubicación actual del usuario. Además, ofrece la posibilidad de abrir la navegación de Google Maps para recibir indicaciones de cómo llegar.

Batería baja: esta pantalla se muestra cuando el sistema detecta que el nivel de batería del dispositivo es bajo. Ofrece la posibilidad de ver los puntos de recarga más cercanos.

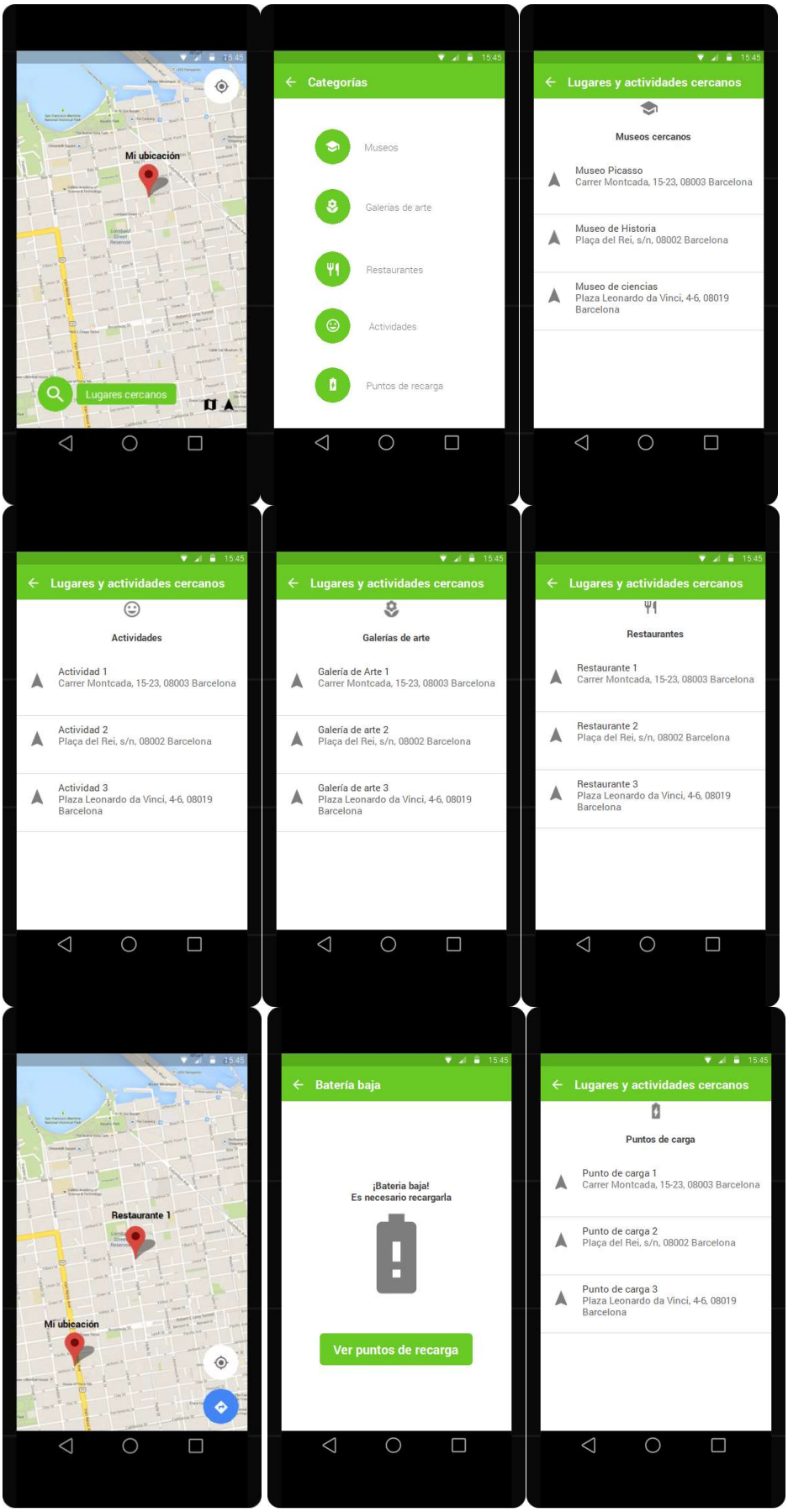

Figura 13: Prototipo de la aplicación

3.6.1. Evaluación del prototipo

Para realizar el diseño del prototipo se ha tenido en cuenta el *feedback* proporcionado por un grupo de usuarios que han probado la interfaz simulándolo con JustinMind Prototyper. Esto, ha permitido simplificar el diseño, valorar la experiencia de uso y mejorarla.

Tras varias pruebas, y teniendo en cuenta la retroalimentación recibida, se ha decidido incluir la pantalla de “Categorías” que permite filtrar los lugares dependiendo del tipo que sean en vez de mostrar todos los lugares disponibles en la misma pantalla. De esta forma, el contenido de la aplicación se muestra de forma más clara y ordenada.

Capítulo 4

4. Diseño del sistema

4.1. Introducción

En este capítulo se analizan las clases que se van a desarrollar para llevar a cabo la implementación del proyecto. Además, se analizan las librerías que se importarán al desarrollo y que facilitarán la programación.

4.2. Análisis de clases

A continuación, se enumeran las clases que componen el proyecto acompañadas de una breve descripción.

Clase "MapsActivity"

Se trata de la clase que contiene el código del "Activity" principal, es decir, los métodos necesarios para interactuar con la primera pantalla que aparece nada más abrir la aplicación. Además, está formada por los métodos necesarios para mostrar el mapa, obtener la ubicación, añadir un marcador y algunos métodos relacionados con las *geofences*.

Atributos de la clase:

Nombre	Tipo
mMap	GoogleMap
Marcador	Marker
Lat	double
Lng	double
mGeofencingClient	GeofencingClient
mGeofenceList	ArrayList<Geofence>
mGeofencePendingIntent	PendingIntent
locListener	LocationListener
notificationId	int
CHANNEL_ID	String
REQUEST_ACCESS_FINE	int

Tabla 5: Atributos de la clase "MapsActivity"

Métodos de la clase:

- onCreate (Bundle savedInstanceState): en este método se inicializa el *Activity*.
- requestPermissions (): comprueba si el permiso de ubicación está aprobado y en caso de no estarlo lo solicita.
- onRequestPermissionsResult (int requestCode, String permissions[], int[] grantResults): este método es invocado cuando el usuario responde al cuadro de dialogo mostrado para pedir permisos y muestra un mensaje emergente dependiendo de la respuesta elegida.
- OnResume (): permite reanudar la actividad de un *activity* que ha sido pausado para ver otro.
- onMapReady (GoogleMap googleMap): permite administrar el objeto GoogleMap. Se llama a este método cuando el mapa está listo para utilizarse.
- agregarMarcador (double lat, double lng): recibe por parámetros la longitud y la latitud para agregar un marcador en el mapa.
- actualizarUbicacion (Location location): actualiza la localización en la que se encuentra el dispositivo en caso de que haya cambiado.
- miUbicacion (): hace uso del GPS para obtener la ubicación del dispositivo.
- openActivityCategories (View view): permite abrir el *activity* "CategoriesActivity".
- createObjectGeofence (): crea un objeto de tipo *Geofence* (longitud, latitud, radio, tiempo de expiración y tipo de transición).
- getGeofencePendingIntent (): define un *PendingIntent* para iniciar un *IntentService* y de esta manera poder lanzar una notificación o realizar un trabajo en segundo plano.
- addGeofences (): añade las *geofences* creadas a la lista para ser monitoreadas.
- CheckHour (): comprueba la hora y lanza una notificación (entre las 12:00 y las 16:00 y las 19:00 y la 22:00) que permite abrir el *activity* restaurantes.

Clase “ActivitiesActivity”

Esta clase contiene todo el código relacionado con la pantalla de actividades. Se utiliza para mostrar la lista de actividades cercanas.

Atributos de la clase:

Nombre	Tipo
mRecyclerView	RecyclerView
mAdapter	RecyclerView.Adapter
mLayoutManager	RecyclerView.LayoutManager
listDatos	ArrayList<String>
listCategorias	ArrayList<String>
listCategoriasPorHora	ArrayList<String>

Tabla 6: Atributos de la clase “ActivitiesActivity”

Métodos de la clase:

- onCreate (Bundle savedInstanceState): inicializa el *Activity.*
- recuperarJson (): recupera el archivo *JSON* con los nombres de las g*eofences* en las que el dispositivo se encuentra dentro de su área.
- elegirCategoria (): compara la lista de *geofences* en las que se encuentra el usuario con la lista de actividades y guarda en un ArrayList las coincidencias.
- CheckHour (): compara la hora de las actividades con la hora del sistema y guarda en una lista solamente aquellas cuya hora sea 30 minutos superior o inferior a la hora del sistema.
- listarLugaresRecyclerView (): lista los lugares cercanos dónde hay actividades programadas o muestra un mensaje en caso de no haberlas.

Clase “MuseumsActivity”

Está formada por los métodos necesarios para listar los museos cercanos y por el código necesario para interactuar con el *activity* correspondiente. Esta clase es muy similar a “ActivitiesActivity” por lo que tanto los atributos como los métodos son muy parecidos.

Atributos de la clase:

Nombre	Tipo
mRecyclerView	RecyclerView
mAdapter	RecyclerView.Adapter
mLayoutManager	RecyclerView.LayoutManager
listDatos	ArrayList<String>
listCategorias	ArrayList<String>

Tabla 7:Atributos de clase "MuseumsActivity"

Métodos de la clase:

- onCreate (Bundle savedInstanceState): inicializa el *Activity.*
- recuperarJson (): recupera el archivo *JSON* con los nombres de las g*eofences* en las que el dispositivo se encuentra dentro de su área.
- elegirCategoria (): compara la lista de *geofences* en las que se encuentra el usuario con la lista de museos y guarda en un ArrayList las coincidencias.
- CheckHour (): compara el horario de los museos con la hora del sistema. En caso de que la hora del sistema este comprendida entre las 10:00 y las 19:00 horas mostrará la lista de los más cercanos (si los hay). En caso contrario, mostrará un mensaje con su horario.

Clase "ArtGaleries"

Al igual que la clase anterior, se utiliza para mostrar la lista de lugares cercanos. Por lo tanto, esta clase contiene los mismo atributos y métodos que la clase "MuseumsActivity" pero en este caso relacionados con las galerías de arte.

Clase "BatteryLevelReceiver"

Almacena el código necesario para recibir y responder al evento "batería baja" enviado por el sistema. Contiene los métodos básicos para enviar una notificación que permite abrir el *activity* "BatteryLow" [28]

Atributos de la clase:

Nombre	Tipo
CHANNEL_ID	String
notificationId	int

Tabla 8: Atributos de la clase" BatteryLevelReceiver"

Métodos de la clase:

- onReceive(Context context, Intent intent): es llamado cuando se recibe un *Intent Brodcast*, es decir, en este caso cuando se recibe la alerta de batería baja enviada por el sistema. El método envía una notificación que permite abrir un *activity.*

Clase "BatteryLow"

Contiene el código necesario para interactuar cuando se recibe el evento batería baja enviado por el sistema. Esta clase está formada por el método que permite abrir el *activity* "BatteryChargersActivity".

Métodos de la clase:

- onCreate (Bundle savedInstanceState): inicializa el *Activity "BatteryLow"*
- openActivityBatteryChargers (View view): permite abrir el *activity* "BatteryChargersActivity"

Clase "BatteryChargersActivity"

Incluye los métodos necesarios para listar los puntos de carga próximos a la ubicación donde se encuentra el usuario. Esta clase es muy parecida a "MuseumsActivity" por lo que su contenido es similar.

Atributos de la clase:

Nombre	Tipo
mRecyclerView	RecyclerView
mAdapter	RecyclerView.Adapter
mLayoutManager	RecyclerView.LayoutManager
listDatos	ArrayList<String>
listCategorias	ArrayList<String>

Tabla 9: Atributos de la clase "BatteryChargersActivity"

Métodos de la clase:

- onCreate (Bundle savedInstanceState): inicializa el *Activity.*
- recuperarJson (): recupera el archivo *JSON* con los nombres de las *geofences* en las que el dispositivo se encuentra dentro de su área.
- elegirCategoria (): compara la lista de *geofences* en las que se encuentra el usuario con la lista de cargadores y guarda en un ArrayList las coincidencias.
- CheckBatteryLevel (): se encarga de comprobar si el nivel de batería es inferior al 20%. En caso de que lo sea, muestra una lista con los cargadores cercanos si los hay. Por el contrario, si el nivel de batería es superior al 20% muestra un mensaje diciendo que no es necesario realizar una carga.

Clase "CategoriesActivity"

Esta clase está formada por el código necesario para abrir los diferentes *activities*, es decir, permite abrir las pantallas de las diferentes categorías.

Métodos de la clase:

- onCreate (Bundle savedInstanceState): inicializa el *activity "*CategoriesActivity*".*
- openActivityArtGaleries (View view): permite abrir el *activity "*ArtGaleries*"* desde el botón correspondiente.
- openActivityMuseums (View view): permite abrir el *activity* "MuseumsActivity" desde el botón del menú.
- openActivityActivities (View view): permite abrir el *activity* "ActivitiesActivity" desde el botón asociado.
- openActivityRestaurants (View view): permite abrir el *activity* "RestaurantsActivity" desde el botón del menú categorías.
- openActivityBatteryChargers (View view): permite abrir el *activity* "BatteryChargersActivity".

Clase “Constants”

En esta clase se almacenan constantes estáticas como, por ejemplo, la lista de geofences y sus parámetros. De este modo, es mucho más fácil editarlas en caso de que sea necesario.

Atributos de la clase:

Nombre	Tipo
GEOFENCE_RADIUS_IN_METERS	float
GEOFENCE_EXPIRATION_IN_MILLISECONDS	long
GEOFENCES_LIST	HashMap<String, LatLng>
GEOFENCES_LIST_MUSEOS	HashMap<String, LatLng>
GEOFENCES_LIST_ACTIVIDADES	HashMap<String, LatLng>
GEOFENCES_LIST_RESTAURANTS	HashMap<String, LatLng>
GEOFENCES_LIST_CHARGERS	HashMap<String, LatLng>
GEOFENCES_LIST_GALERIES	HashMap<String, LatLng>
GEOFENCES_LIST_ACTIVIDADES_CON_HORA	HashMap<String, String>

Tabla 10: Atributos de la clase “Constants”

Clase “RestaurantsActivity”

Esta clase contiene todo lo relacionado con los restaurantes como, por ejemplo, el método de comprobar la hora. Además, contiene todo lo necesario para el funcionamiento del *activity* “Restaurants”. También es muy similar a la clase “MuseumsActivity” por lo que tanto sus atributos como sus métodos son muy parecidos.

Atributos de la clase:

Nombre	Tipo
mRecyclerView	RecyclerView
mAdapter	RecyclerView.Adapter
mLayoutManager	RecyclerView.LayoutManager
listDatos	ArrayList<String>
listCategorias	ArrayList<String>

Tabla 11: Atributos de la clase “RestaurantsActivity”

Métodos de la clase:

- onCreate (Bundle savedInstanceState): inicializa el a*ctivity.*
- recuperarJson (): recupera el archivo *JSON* con los nombres de las *geofences* en las que el dispositivo se encuentra dentro de su área.
- elegirCategoria (): compara la lista de las *geofences* en las que se encuentra el usuario con la lista de restaurantes y guarda en un ArrayList las coincidencias.
- CheckHour (): este método comprueba si la hora del sistema está comprendida entre las 12:00 y 16:00 y las 19:00 y las 22:00. En caso de que lo esté, se muestra la lista de restaurantes cercanos. En caso contrario, se muestra el horario.

Clase "GeofenceTransitionsIntentService"

Esta clase contiene los procedimientos necesarios para iniciar una tarea en segundo plano cuando se produce una transición en las *geofences*.

Atributos de la clase:

Nombre	Tipo
triggeringGeofencesIdsList	ArrayList<String>

Tabla 12: Atributos de la clase "GeofenceTransitionsIntentService"

Métodos de la clase:

- onHandleIntent (Intent intent): este método es llamado cuando se inicia el IntentService. Comprueba el tipo de transición y realiza tareas dependiendo de si es de entrada o salida.
- getNamesGeofencesInArray (List<Geofence> triggeringGeofences): guarda en un *ArrayList* los nombres de las *geofences* en las que se ha entrado en su perímetro.
- createJson (ArrayList<String> triggeringGeofencesIdsList): crea un archivo JSON con el nombre de las *geofences* en las que se ha entrado en su perímetro.

Clase "GeofencesErrorMessages"

Esta clase incluye el código con los posibles errores que se pueden producir con el servicio de *geofences*.

Atributos de la clase:

Nombre	Tipo
mResource	Resource

Tabla 13: Atributos de la clase "GeofencesErrorMessages"

Métodos de la clase:

- getErrorString(Context context, int errorCode): este método devuelve mensajes de error dependiendo del código que reciba.

Clase "MyAdapter"

Esta clase contiene los métodos necesarios que indican cómo mostrar la información en los *recyclerView,* es decir, se encarga de crear las vistas necesarias para cada *recyclerView.*

Atributos de la clase:

Nombre	Tipo
listDatos	ArrayList<String>

Tabla 14: Atributos de la clase "MyAdapter"

Métodos de la clase:

- onCreateViewHolder (ViewGroup parent, int viewType): se encarga de inicializar el *ViewHolder*.
- onBindViewHolder (ViewHolderDatos holder, int position): este método configura el contenido de los *View*.
- getItemCount (): devuelve el tamaño del *ArrayList*.
- setOnclickListener (): escucha los eventos de clics realizados sobre un botón.
- Asignardatos (String datos): asigna el texto recibido en el *ArrayList* a cada botón *del recyclerView.*
- onClick (View view): asigna un evento a los botones del *recyclerView*.

Clase "DestinationActivity"

Está formada por los métodos necesarios para mostrar el mapa, obtener la ubicación y añadir un marcador.

Atributos de la clase:

Nombre	Tipo
mMap	GoogleMap
Marcador	Marker
Lat	double
Lng	double
name	String
Destino	LatLng

Tabla 15: Atributos de la clase "DestinationActivity"

Métodos de la clase:

- onCreate (Bundle savedInstanceState): en este método se inicializa el *Activity*.
- onMapReady (GoogleMap googleMap): permite administrar el objeto GoogleMap. En esta clase permite añadir el marcador de destino y el de la posición del dispositivo.
- recibirDatos (): recibe el nombre de las *geofences* enviado desde la clase "MyAdapter".
- getCoordenadas (): compara el nombre recibido en el método "recibirDatos" con la lista de *geofences* para obtener la longitud y la latitud.
- agregarMarcador (double lat, double lng): recibe por parámetros la longitud y la latitud para agregar un marcador en el mapa.
- actualizarUbicacion (Location location): actualiza la localización en la que se encuentra el dispositivo en caso de que haya cambiado.
- miUbicacion (): hace uso del GPS para obtener la ubicación del dispositivo.

4.3. Análisis de librerías

Las librerías están formadas por un conjunto de clases que contienen métodos que se pueden reutilizar con el fin de evitar tener que programar desde cero. [29]

A continuación, se analizan las librerías que se han incorporado al proyecto:

Librería "com.google.android.gms:play-services-maps:15.0.0": esta librería permite utilizar la API de Google Maps para Android. Debe ser incorporada en el apartado dependencias del archivo built.gradle del proyecto.

Librería "com.google.android.gms:play-services-location:15.0.0": esta librería permite utilizar los servicios de localización en Android. También debe ser incorporada en el apartado dependencias del archivo built.gradle.

Librería "com.android.support:recyclerview-v7:26.1.0": proporciona la clase "RecyclerView". Esta clase permite mostrar conjuntos de datos en pantalla. Como las librerías anteriores, también se incorpora en las dependencias de built.gradle.

Librería "com.google.code.gson:gson:2.8.0": proporciona la clase Gson que permite la serialización y deserialización entre objetos JAVA, es decir, permite la codificación de un objeto en un medio de almacenamiento para poder transmitirlo. Esta librería se añade en built.gradle. [30]

Además, en cada clase se importarán las librerías necesarias para utilizar algunas funciones predefinidas y, de esta manera, reutilizar código. Entre estas bibliotecas destacan:

Librería com.google.android.gms.location.Geofence: permite utilizar la clase "Geofence" para crear perímetros geográficos que pueden ser monitoreados.

Librería com.google.android.gms.location.GeofencingClient: permite interactuar con la API de *geofencing*.

Librería com.google.android.gms.location.GeofencingRequest: permite especificar la lista de geofences a monitorear y elegir cuándo notificar (al entrar, salir o permanecer un tiempo dentro de una geofence) .

Librería import com.google.android.gms.maps.GoogleMap: contiene las clases de la API de Google Maps.

Librería android.location.LocationListener: ofrece métodos que permiten saber cuándo cambia la ubicación.

Librería com.google.android.gms.maps.model.LatLng: permite almacenar la longitud y latitud de un punto (en grados).

Librería com.google.android.gms.maps.model.Marker: permite colocar marcadores en un punto concreto en el mapa.

Librería android.os.BatteryManager: proporciona métodos para consultar el nivel de batería.

Librería android.widget.Toast: permite crear y mostrar mensajes flotantes para el usuario.

Librería java.util.Calendar: proporciona los métodos necesarios para poder operar con los campos de un calendario.

Librería java.util.GregorianCalendar: esta subclase de *Calendar* proporciona el sistema de calendario más extendido a nivel mundial.

4.4. Pruebas de la aplicación

Para garantizar el correcto funcionamiento de todas las funcionalidades exigidas se han realizado una serie de pruebas con varios dispositivos que poseen diferentes versiones de Android. En concreto se han utilizado los siguientes dispositivos:

- Xiaomi Redmi note 3 con Android 6.0.1 (API 23)
- Xiaomi Redmi note 4 con Android 7.0 (API 24)
- Motorola Moto G2 con Android 6.0 (API 23)
- Samsung Galaxy SIV con Android 5.0.1 (API 21)

Además, estos dispositivos también tienen diferentes tamaños de pantalla (4.8’, 5’ y 5.5’) lo que ha permitido poner a prueba la capacidad de adaptación de la interfaz.

En todos ellos se han probado los diferentes casos de uso descritos en el apartado 3.5:

CDU-001: Localización

Al abrir la aplicación, el sistema automáticamente localiza la posición del usuario. Además, comprueba si hay permisos de ubicación y en caso de que no los haya los pide.

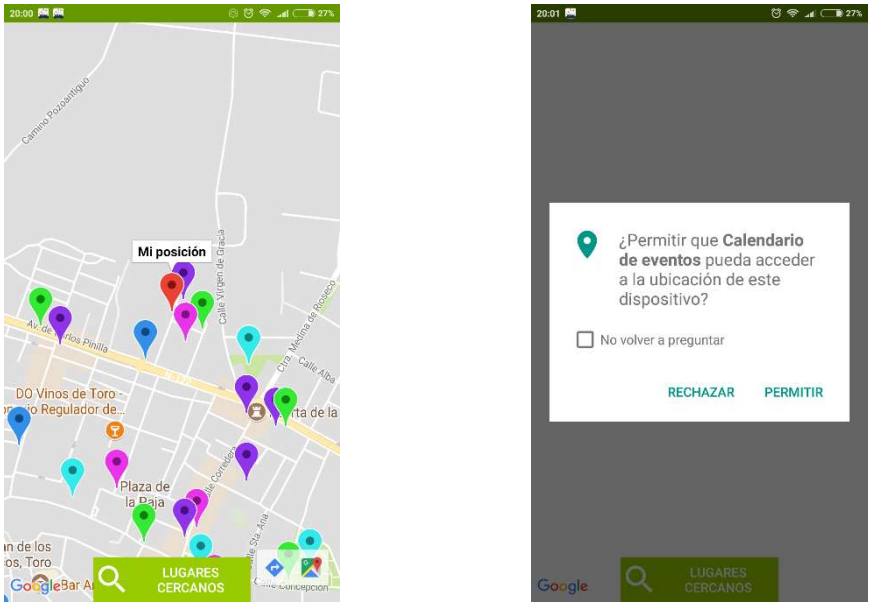

Figura 14: Prueba CDU-001

CDU-002: Ver tipos de lugares

Al pulsar el botón lugares cercanos se muestra la lista de botones con los diferentes tipos de lugares.

Figura 15: Prueba CDU-002

CDU-003: Elegir tipo de lugar cercano

Una vez elegido el tipo de lugar que queremos la aplicación muestra la lista de lugares o actividades cercanas. En caso de que no haya muestra un mensaje.

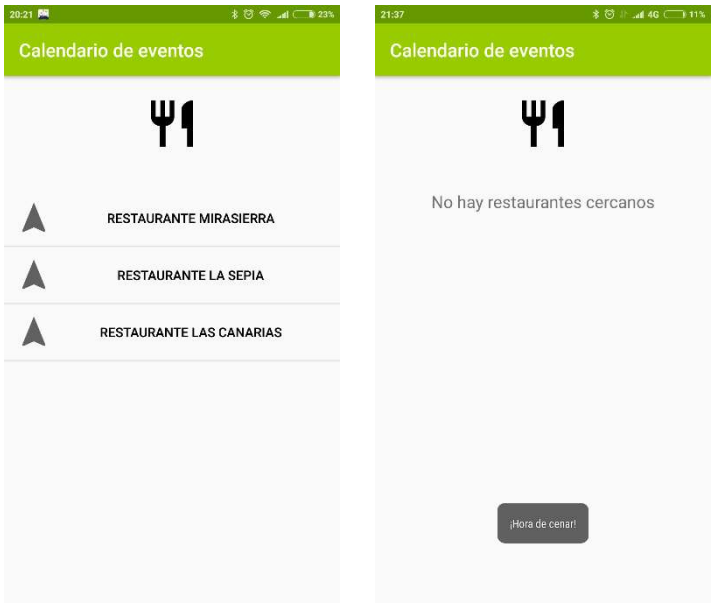

Figura 16: Prueba CDU-003 (dentro de horario)

Además, la aplicación comprueba la hora y si la categoría elegida no está disponible a la hora de la consulta muestra un mensaje. En concreto, las categorías museos y galerías de arte solo muestran contenido entre las 10:00 y las 19:00 y los restaurantes entre las 12:00 y las 16:00 y las 19:00 y las 22:00.

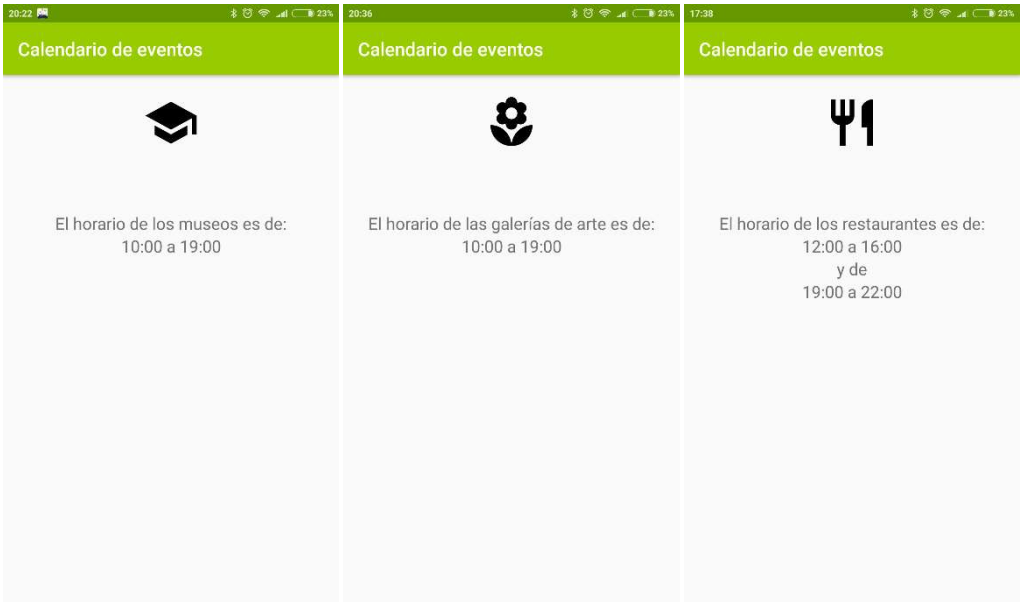

Figura 17: Prueba CDU-003 (fuera de horario)

Las actividades se muestran 30 minutos antes y 30 minutos después de la hora a la que estén programadas. En caso de que no haya actividades cercanas en tiempo y lugar se muestra un mensaje de aviso.

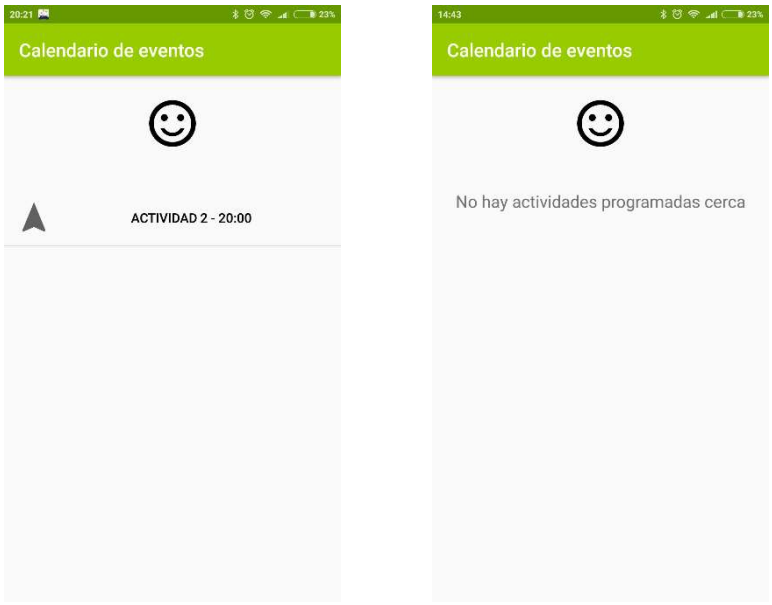

Figura 18: Prueba CDU-003 (Actividades)

Por último, el sistema también comprueba el nivel de batería y si es inferior al 20% muestra la lista de cargadores cercanos. En caso contrario, la aplicación muestra un mensaje informativo. Asimismo, cuando el sistema envía la alerta de batería baja la aplicación ofrece ver puntos cercanos de recarga.

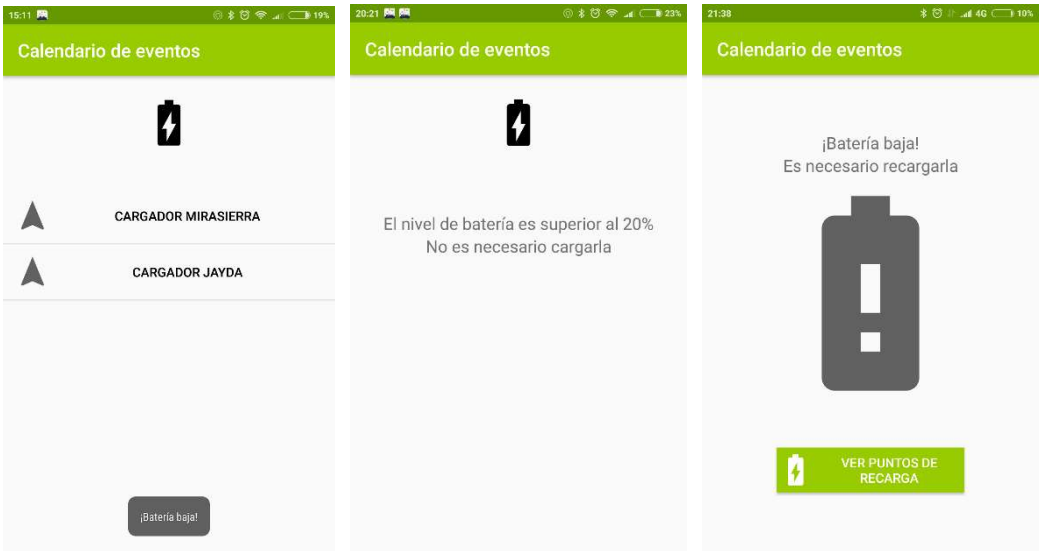

Figura 19: Prueba CDU-003 (Batería baja)

CDU-004: Elegir lugar o actividad

Al elegir el lugar deseado, el sistema muestra un mapa con un marcador de color naranja que indica la localización. Además, al pulsar sobre dicho marcador se obtienen indicaciones de cómo llegar hasta él.

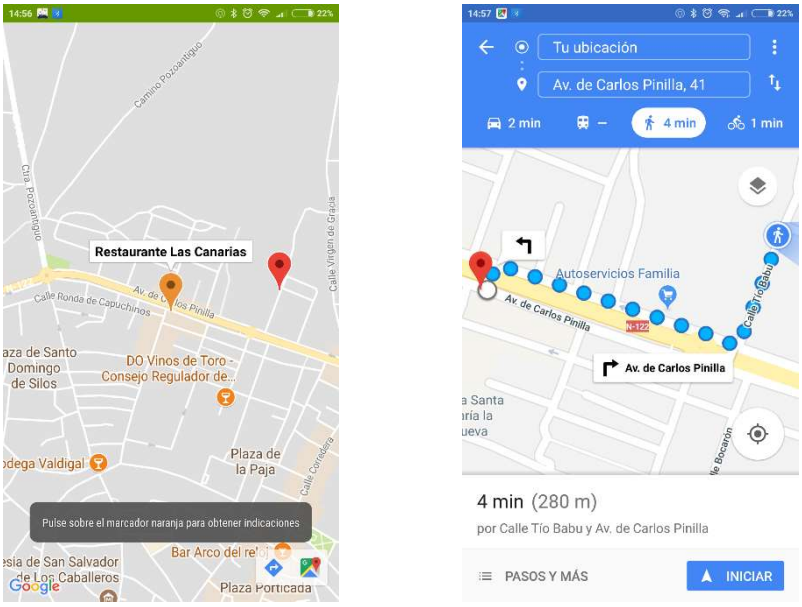

Figura 20: Prueba CDU-004

5. Conclusiones

El desarrollo de este trabajo de final de grado me ha permitido completar mi formación, ya que he podido afianzar y ampliar los conocimientos y competencias que he adquirido durante mis estudios de grado de Tecnologías de Telecomunicaciones.

Para su desarrollo ha sido necesario poner en práctica muchas de esas competencias adquiridas, como el diseño y programación orientada a objetos y la gestión de proyectos. Además, se han aplicado los conocimientos adquiridos en asignaturas como aplicaciones y servicios multimedia, SIG y geotelemática y competencia comunicativa para profesionales de las TIC.

Tanto el objetivo principal como los objetivos específicos han sido superados con éxito, pues he aprendido a desarrollar una aplicación móvil para Android, a utilizar la localización, crear y monitorear *geofences*, a almacenar localizaciones y a utilizar el IDE Android Studio. Además, el producto final, la aplicación Calendario de eventos, cumple todos los requisitos funcionales exigidos.

La metodología elegida creo que ha sido la correcta, pues ha ayudado a completar todos los objetivos y a desarrollar el proyecto de forma satisfactoria. Sin embargo, la planificación establecida no fue la correcta y se ha tenido que aplicar el plan de contingencia para minimizar las consecuencias.

En conclusión, el desarrollo de este proyecto no solo me ha permitido afianzar y aumentar mis conocimientos sino también aprender a planificar un proyecto de forma más eficiente. Además, me ha ayudado a descubrir la rama profesional de desarrollo de aplicaciones la cual me ha parecido muy interesante.

5.1. Líneas de trabajo futuras

Debido a los límites temporales que están fijados para desarrollar este proyecto no es posible implementar todas las funcionalidades extra que podrían hacer más interesante aún la aplicación. Por este motivo, a continuación, se enumeran una serie de funcionalidades que se podrían implementar en futuras actualizaciones.

- Añadir un método que ordene por horas las actividades
- Añadir más información sobre los lugares como, por ejemplo, fotografías, textos informativos, etc.

- Integrar redes sociales para conocer las opiniones de otros usuarios que hayan visitado esos lugares.
- Permitir realizar reservas en los restaurantes o comprar entradas para los museos y galerías de arte.
- Mejorar la interfaz para hacerla más atractiva.
- Añadir un apartado de lugares favoritos que permita guardar los sitios visitados que más hayan gustado.
- Añadir un método para pedir la activación del GPS en caso de que esté apagado.

6. Glosario

- ***Activity*:** es cada una de las pantallas o vistas que forman las aplicaciones en Android.
- **Android:** sistema operativo propiedad de Google para dispositivos inteligentes basado en Linux.
- **Android Studio:** es un IDE especializado en la programación para Android.
- **API (*Application Programming Interface*):** conjunto de protocolos y funciones que permiten la comunicación entre dos softwares.
- **BrodcastReceiver:** es un componente que recibe y responde a eventos generados por el sistema como, por ejemplo, una alarma de batería baja.
- **Geofence:** perímetro virtual de un área geográfica que puede ser monitoreado.
- **Google Play:** tienda oficial de aplicaciones de Android.
- **IDE (Integrated Development Environment):** aplicación que facilita el desarrollo de software.
- **Intent:** sirve para invocar componentes como, por ejemplo, *activities,* aplicaciones externas, lanzar eventos, alarmas o código en segundo plano, etc.
- **IntentService:** es un tipo de servicio en Android que se utiliza para tareas de larga duración que no dependen del hilo principal y que se detienen una vez finalizada su misión.
- **Java:** es un lenguaje de programación orientado a objetos.
- **PendingIntent:** especifican una acción futura, es decir, permite pasar una intención futura a otra aplicación con los mismos permisos que posee la que lo envía.
- **RecyclerView:** contenedor de elementos en forma de lista.

7. Bibliografía

[1] P. M. Martori, «La Vanguardia,» 28 02 2017. [En línea]. Available: http://www.lavanguardia.com/tecnologia/20170228/42379146092/uso-smartphone-duplica-cinco-anos-mwc.html. [Último acceso: 20 05 2018].

[2] A. Clemente, «Atrevia,» 26 12 2017. [En línea]. Available: https://atrevia.com/blog/smartphone-ha-cambiado-habitos-consumo/. [Último acceso: 20 05 2018].

[3] «ABC TEC,» 2015. [En línea]. Available: http://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html.

[4] El Android Libre, «El Android Libre,» 2018. [En línea]. Available: https://elandroidelibre.elespanol.com/2017/09/que-es-api-software-android.html. [Último acceso: 2018 marzo 19].

[5] Edgar, «Edgar D'Andrea,» 2015. [En línea]. Available: https://www.edgardandrea.com/ventajas-de-ser-desarrollador-android/.

[6] Gartner, 2017. [En línea]. Available: https://www.gartner.com/newsroom/id/3609817.

[7] M. Hernadez, 2017. [En línea]. Available: https://voltaico.lavozdegalicia.es/2017/11/cuota-mercado-android-2017/.

[8] A. Nieto, «Xataka,» [En línea]. Available: https://www.xatakandroid.com/sistema-operativo/que-es-android.

[9] «Webgenio,» [En línea]. Available: https://webgenio.com/blog/que-es-android-y-que-es-un-telefono-movil-android/.

[10] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Android.

[11] J. Rovira Jofre, «Geotelemática. Posicionamiento y navegación,» 2016.

[12] ABC Tecnología, «ABC,» 2014. [En línea]. Available: http://www.abc.es/tecnologia/moviles-telefonia/20140320/abci-localizacion-movil-201403192024.html.

[13] Wikipedia, «Wikipedia,» 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global.

[14] «gps.gov,» [En línea]. Available: https://www.gps.gov/spanish.php.

[15] J. Penalva, «Xataka,» 2007. [En línea]. Available: https://www.xataka.com/moviles/que-es-el-a-gps.

[16] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/GPS_Asistido . [Último acceso: 2018].

[17] Rebeca, «APP.NET,» 2018. [En línea]. Available: https://www.tu-app.net/blog/geolocalizacion/.

[18] M. Rouse, «Whatls.com,» 2016. [En línea]. Available: http://whatis.techtarget.com/definition/geofencing. [Último acceso: 10 marzo 2018].

[19] B. Thumar, «Android KT,» 2017. [En línea]. Available: http://androidkt.com/android-geofence/. [Último acceso: 10 marzo 2018].

[20] Wikipedia, «Wikipedia,» 2017. [En línea]. Available: https://en.wikipedia.org/wiki/Geo-fence. [Último acceso: 10 marzo 2018].

[21] F. Garcia, «fergarciac,» 2013. [En línea]. Available: https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/. [Último acceso: 2018 marzo 20].

[22] «Solbyte,» 2014. [En línea]. Available: https://www.solbyte.com/blog/2014/07/21/tipos-de-aplicaciones-moviles-nativas-webs-hibridas/. [Último acceso: 2018 Marzo 25].

[23] Raona, «Raona,» 2017. [En línea]. Available: http://www.raona.com/aplicacion-nativa-web-hibrida/. [Último acceso: 2018 Marzo 25].

[24] Nextu, «Nextu,» 2018. [En línea]. Available: https://www.nextu.com/blog/apps-nativas-vs-apps-hibridas/. [Último acceso: 20 03 2018].

[25] Android Developers, «Android Developers,» 2018. [En línea]. Available: https://developer.android.com/about/dashboards/index.html. [Último acceso: 20 03 2018].

[26] Developers Google, «Developers Google,» 2018. [En línea]. Available: https://developers.google.com/maps/documentation/android-api/?refresh=1&pli=1. [Último acceso: 12 marzo 2018].

[27] JustMind Prototyper, «JustMind Prototyper,» 2018. [En línea]. Available: https://www.justinmind.com/. [Último acceso: 30 Marzo 2018].

[28] Google, «Android Developers,» [En línea]. Available: https://developer.android.com/training/monitoring-device-state/battery-monitoring?hl=es-419. [Último acceso: 15 04 2018].

[29] J. D. Meza, «Programar ¡ya!,» 2016. [En línea]. Available: https://www.programarya.com/Cursos/Java/Librerias. [Último acceso: 20 04 2018].

[30] Wikipedia, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Serializaci%C3%B3n. [Último acceso: 30 04 2018].

[31] Developers Android, «Developers Android,» 2018. [En línea]. Available: https://developer.android.com/studio/index.html?hl=es-419. [Último acceso: 11 marzo 2018].

[32] Google, «Android Developers,» [En línea]. Available: https://developer.android.com/training/location/geofencing. [Último acceso: 15 04 2018].

8. Anexos

Anexo I - Manual de instalación de la aplicación: este anexo contiene una guía de instalación de la aplicación.