Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo de fin de grado

Grado en Ingeniería Informática
(Mención en Ingeniería de Software)

Aplicación móvil de guía turística con realidad aumentada

Autor:

Fernando Peláez Asenjo

Tutor:

Miguel Ángel Laguna Serrano

Resumen

En la actualidad, las tecnologías móviles están muy presentes en nuestro día a día. Continuamente utilizamos nuestro dispositivo móvil para todo tipo de tareas, ya que nos permiten llevarlas a cabo desde cualquier lugar. En los últimos años ha aparecido una nueva tecnología que puede cambiar sustancialmente la forma en la que usamos estos dispositivos: la realidad aumentada. Se trata de una tecnología emergente que permite superponer elementos virtuales sobre nuestra visión de la realidad, y cada vez es más demandada.

En este trabajo se aplica esta tecnología sobre una aplicación de guía turística de la ciudad de Valladolid (España), que proporciona al usuario una forma diferente de interactuar con los distintos lugares que visita.

Esta aplicación Android permite al usuario ver los atractivos turísticos más característicos de la localidad, conocer algunos datos curiosos de los mismos, ver fotografías, obtener la ruta que debe seguir para verlo en persona y, una vez allí, ver a través de la cámara de su dispositivo cómo era ese mismo lugar en el pasado, gracias a la realidad aumentada.

Abstract

Nowadays, mobile technologies are very present in our daily lives. We continuously use our mobile device for tasks of any kind, due to us being able to use them anywhere. A new technology has appeared a few years ago, and it can substantially change the way we use our devices: Augmented reality. It is an emerging technology which allows virtual elements to be superimposed on our vision of reality, and it gets more and more demanded over time.

In this project, this technology is applied to a tourist guide application of Valladolid (Spain), which provides the user a different way to interact with the different places he can visit.

This Android application enables the user to see the most characteristic tourist attractions of the city and to learn some curious information about them. It also allows the user to find the shortest route to the tourist attraction and, once there, see how that place looked like in the past through his mobile device's camera, thanks to augmented reality.

Índice general

Resumen III

Abstract V

1. Introducción 1
- 1.1. Motivación . . . 1
- 1.2. Alcance y objetivos . . . 1
- 1.3. Entregables del proyecto . . . 2
- 1.4. Estructura de la memoria . . . 2

2. Contexto 3
- 2.1. Definición . . . 3
- 2.2. Usos . . . 4

3. Tecnologías utilizadas 7
- 3.1. Android . . . 7
- 3.2. Kotlin . . . 8
- 3.3. Google Maps SDK . . . 9
- 3.4. Unity 2019 . . . 9
- 3.5. Vuforia Engine . . . 10
- 3.6. Git . . . 10

4. Plan de proyecto 13
- 4.1. Metodología . . . 13
  - 4.1.1. Desarrollo en cascada . . . 13
- 4.2. Plan de control del proyecto . . . 18
  - 4.2.1. Gestión de gastos . . . 18
  - 4.2.2. Gestión de riesgos . . . 18
- 4.3. Seguimiento del proyecto . . . 24
  - 4.3.1. Elección de tecnología . . . 24
  - 4.3.2. Configuración de build y exportación de librería . . . 24
  - 4.3.3. Implementación de librería Unity . . . 24
  - 4.3.4. Obtención de fotografías . . . 26
  - 4.3.5. Creación de base de datos de objetivos de imagen . . . 26

5. Análisis del proyecto **27**
5.1. Elicitación de requisitos . . . 27
5.1.1. Requisitos funcionales . . . 27
5.1.2. Requisitos no funcionales . . . 28
5.1.3. Reglas de negocio . . . 30
5.1.4. Requisitos de información . . . 30
5.2. Casos de uso . . . 31
5.2.1. Diagrama de casos de uso . . . 31

6. Diseño **41**
6.1. Diseño de las vistas . . . 42
6.2. Diseño de arquitectura . . . 45
6.2.1. Model-View-ViewModel . . . 45
6.3. Diseño detallado . . . 47
6.4. Patrones de diseño empleados . . . 48
6.4.1. Patrón observador . . . 48
6.5. Patrón adaptador . . . 50
6.6. Gestión de los datos . . . 50

7. Implementación **53**
7.1. Implementación de la aplicación Android . . . 53
7.1.1. MainActivity . . . 53
7.1.2. Fragmento de lugares . . . 55
7.1.3. Fragmento de detalles . . . 57
7.1.4. Fragmento de mapa . . . 61
7.2. Implementación de la librería de realidad aumentada . . . 65
7.3. Integración de la librería Unity en la aplicación Android . . . 74

8. Pruebas **77**
8.1. Pruebas de caja blanca . . . 77
8.2. Pruebas de caja negra . . . 77
8.3. Pruebas de la realidad aumentada . . . 80

9. Conclusiones y trabajo futuro **83**

A. Manual de instalación **87**

B. Manual de usuario **89**

C. Entregables **95**

Índice de figuras

2.1. Ejemplo de aplicación de realidad aumentada. 3
2.2. Imagen de ”Pokemon Go” 4
2.3. Detalle de un cuadro en realidad aumentada. 5

3.1. Comparación entre usuarios de Android e iOS a lo largo de la década. 7
3.2. Interfaz de Unity. 9
3.3. Comparación del reconocimiento de AR Core. 10

4.1. Distintas fases del desarrollo en cascada. [11] 14
4.2. Diagrama de Gantt del análisis de requisitos. 14
4.3. Diagrama de Gantt de la fase de diseño. 15
4.4. Diagrama de Gantt de la fase de implementación. 16
4.5. Diagrama de Gantt de la fase de verificación. 17
4.6. Comparación de línea base con la tarea Elección de tecnología. 24
4.7. Comparación de línea base con la tarea Configuración de build. 24
4.8. Intercambio en el orden de implementación Android y librería Unity. 25
4.9. Comparación de línea base con la tarea obtención de fotografías. 26
4.10. Comparación de línea base con la tarea creación de base de datos de objetivos de imagen. 26

5.1. Diagrama de casos de uso. 31
5.2. Diagrama secuencia de UC01. 32
5.3. Diagrama secuencia de UC02. 33
5.4. Diagrama secuencia de UC03. 34
5.5. Diagrama secuencia de UC04. 36
5.6. Diagrama secuencia de imprimir ruta en el mapa. 37
5.7. Diagrama secuencia de UC05. 38
5.8. Diagrama secuencia de UC06. 39
5.9. Diagrama secuencia de UC07. 40

6.1. Diseño preliminar de la vista de la lista de lugares. 42
6.2. Diseño preliminar de la vista de los detalles de un lugar. 43
6.3. Diseño preliminar de la vista de la navegación hacia un lugar. 44
6.4. Flujo de datos entre los componentes de MVVM [14]. 45
6.5. Ciclo de vida de ViewModel [15]. 46

6.6. Diagrama de clases preliminar del sistema. 47
6.7. Diagrama de secuencia del patrón observador. 49
6.8. Diagrama de secuencia del patrón adaptador. 50

7.1. Código de MainActivity. 53
7.2. Toolbar configurada para navegación. 54
7.3. Grafo de navegación usado por el componente Navigation. 54
7.4. Diferencia entre el funcionamiento de ListView y RecyclerView [16]. 55
7.5. Diagrama del funcionamiento de la clase Adapter [18]. 56
7.6. Ejemplo de vista de la galería de imágenes. 58
7.7. Ejemplo de vista de información de un lugar y botones de navegación. 59
7.8. Ejemplo de vista de detalles con orientación horizontal. 60
7.9. Ejemplo de vista de detalles con orientación horizontal con layout de landscape. 60
7.10. Ejemplo de vista de ruta dibujada sobre el mapa. 61
7.11. El sistema pide los permisos de ubicación al usuario. 62
7.12. El sistema muestra mensaje de aviso al usuario. 62
7.13. El sistema pide al usuario activar la ubicación. 63
7.14. El sistema muestra un mensaje de aviso para activar la ubicación al usuario. 63
7.15. El sistema muestra un mensaje de aviso para notificar que el dispositivo no tiene conexión. 64
7.16. Comparación de modelo 3D con marcador. 65
7.17. Formulario para añadir un marcador a la base de datos de Vuforia. 66
7.18. Marcador con bajo índice de reconocimiento junto a sus características. . . . 67
7.19. Marcador con alto índice de reconocimiento junto a sus características. . . . 68
7.20. Etiqueta del objeto AR Camera. 69
7.21. Estructura del proyecto Unity. 70
7.22. Ejemplo de un objetivo de imagen con su modelo 3D en la escena. 71
7.23. Ejemplo de reconocimiento y superposición de realidad aumentada en la calle. 72
7.24. Ejemplo de reconocimiento y superposición de realidad aumentada en la calle. 73
7.25. Directorio de librería local añadido. 74
7.26. Árbol de directorios con librería añadida. 75

B.1. Primera vista de la aplicación. 89
B.2. Vista de detalles de un lugar. 90
B.3. Vista de ruta en el mapa. 91
B.4. Pantalla de carga de Unity. 92
B.5. Modelo 3D sobre imagen actual. 93

Índice de cuadros

4.1. Risk01 - Modificación de los requisitos 19
4.2. Risk02 - Retraso en la fecha de fin de una tarea 19
4.3. Risk03 - Pérdida del código fuente 20
4.4. Risk04 - Escasez de recursos personales 20
4.5. Risk05 - Diseño pobre 21
4.6. Risk06 - Mala planificación inicial 21
4.7. Risk07 - Problemas con el hardware 22
4.8. Risk08 - Problemas con el software 22
4.9. Risk09 - Complicaciones externas al proyecto 23
4.10. Risk10 - Desconocimiento de las tecnologías a emplear 23

5.1. FR01 - Listado de lugares 27
5.2. FR02 - Presentación de lugar 27
5.3. FR03 - Detalles de lugar 27
5.4. FR04 - Image slider 27
5.5. FR05 - Ruta al destino 28
5.6. FR06 - Control de mapa 28
5.7. FR07 - Abrir la cámara 28
5.8. FR08 - Reconocimiento de imagen 28
5.9. FR09 - Servicio de localización 28
5.10. FR10 - Aviso de rechazo en permisos esenciales 28
5.11. NFR01 - Sistema operativo 29
5.12. NFR02 - Requisitos del dispositivo Android 29
5.13. NFR03 - API de mapas 29
5.14. NFR04 - Motor de realidad aumentada 29
5.15. NFR05 - Rendimiento 29
5.16. NFR06 - Permisos 29
5.17. NFR07 - Almacenamiento de imágenes 29
5.18. NFR08 - Almacenamiento de imágenes 30
5.19. NFR09 - Material Design 30
5.20. CR01 - Instancias de lugar singleton 30
5.21. CR02 - Navegación de la aplicación 30
5.22. IR01 - Lugar 31

5.23. UC01 - Ver lista de lugares . . . 32
5.24. UC02 - Ver detalles de un lugar . . . 33
5.25. UC03 - Ver galería de imágenes . . . 34
5.26. UC04 - Ver ruta en el mapa hacia un lugar . . . 35
5.27. UC05 - Pedir activación de ubicación . . . 38
5.28. UC06 - Abrir la cámara . . . 39
5.29. UC07 - Reconocer imagen . . . 40

8.1. P01 - Ver lista de lugares . . . 77
8.2. P02 - Ver detalles de un lugar . . . 78
8.3. P03 - Ver galería de imágenes . . . 78
8.4. P04 - Ver ruta en el mapa sin servicios . . . 78
8.5. P05 - Ver ruta en el mapa sin servicios de ubicación . . . 78
8.6. P06 - Ver ruta en el mapa sin ubicación activada . . . 79
8.7. P07 - Rechazo de petición de permisos . . . 79
8.8. P08 - Rechazo de activación de servicio de localización . . . 79
8.9. P09 - Rechazo de activación de servicio de localización . . . 79
8.10. P10 - Pérdida de servicio de ubicación mientras el mapa está activo . . . 80
8.11. P11 - Abrir cámara de realidad aumentada . . . 80
8.12. P12 - Cerrar cámara de realidad aumentada . . . 80

Capítulo 1

Introducción

1.1. Motivación

El turismo es uno de los sectores más importantes en España, aunque el turismo rural y en zonas del interior no es tan popular. Este proyecto puede fomentar la afluencia de turistas en estas zonas del país ya que permite al usuario conocer el estado de estos lugares en el siglo pasado en un formato muy llamativo para los jóvenes al tratarse de una nueva tecnología: la realidad aumentada. En algunos casos da la impresión de viajar al pasado, con lo que puede llamar la atención de los usuarios a conocer cómo era nuestro mundo y qué construyeron nuestros antepasados hace muchos años.

Si a esto le añadimos que se trata de una aplicación Android, que se trata probablemente del software más accesible para cualquier tipo de usuario (un 85 % de los usuarios en España usan un dispositivo Android), puede que se trate de una de las vías más eficaces para atraer el interés de los ciudadanos hacia la historia y el pasado de su ciudad.

1.2. Alcance y objetivos

La finalidad de este proyecto es crear una aplicación Android que permita al usuario recorrer los lugares más emblemáticos de la ciudad de Valladolid utilizando únicamente esta aplicación, sin necesidad de un guía turístico que le indique qué lugares debe visitar. Los objetivos son los siguientes:

- Mostrar un listado al usuario con todos los lugares que este puede visitar. Cada lugar cuenta con un título y una fotografía que lo hacen más fácil de reconocer.
- Mostrar información detallada de cada emplazamiento en forma de una serie de fotografías tanto actuales como antiguas y una breve descripción que permite al usuario conocer la historia y datos curiosos del mismo.
- Permitirá al usuario ver la ruta desde su localización hacia dicho lugar utilizando la API de Google Maps Directions.

- Cuando el usuario se encuentre lo suficientemente cerca de la localización tras seguir la ruta del apartado anterior, se le permitirá acceder a la aplicación de realidad aumentada. Esta aplicación, mediante el uso de la cámara del dispositivo, reconoce los patrones de imagen de dicho emplazamiento y superpone una imagen de cómo era ese lugar hace muchos años.

1.3. Entregables del proyecto

- Un documento de memoria del proyecto que contendrá toda la información acerca del diseño y desarrollo del mismo, así como las fuentes de información utilizadas durante este periodo.
- Un manual de usuario que describirá cómo debe instalarse y utilizarse la aplicación para obtener el máximo rendimiento de la misma.
- El código fuente del proyecto Android en su última versión.
- El código fuente del proyecto Unity de realidad aumentada.

1.4. Estructura de la memoria

La memoria contará con las siguientes secciones: contexto, plan del proyecto y desarrollo del mismo, análisis, diseño, implementación, pruebas, conclusiones y bibliografía.

Capítulo 2

Contexto

En este capítulo se expondrá información sobre la realidad aumentada para que el lector tenga una visión clara acerca de esta tecnología.

2.1. Definición

La realidad aumentada es un conjunto de tecnologías que permiten a los usuarios visualizar, a través de un dispositivo, el mundo real con elementos virtuales añadidos por dicho dispositivo. De esta forma los elementos virtuales se combinan con el mundo real creando, en tiempo real, una realidad aumentada.

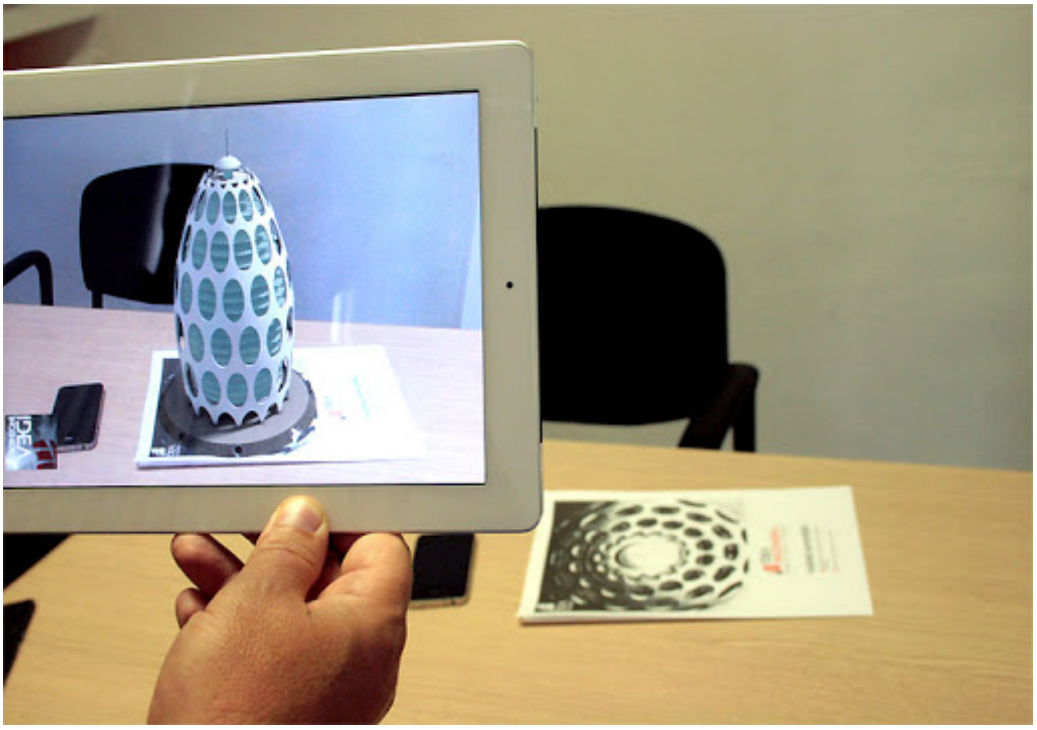


Figura 2.1: Ejemplo de aplicación de realidad aumentada.

2.2. Usos

Por lo general la realidad aumentada se utiliza para mostrar un objeto virtual en 3D que se superpone sobre un plano reconocible por el dispositivo (denominado marcador), como hemos visto en la figura anterior. De esta forma, se pueden encontrar multitud de aplicaciones para esta tecnología, entre las que destacan las siguientes:

- **Educación:** La realidad aumentada aumenta la motivación de los alumnos si se usa en el entorno educativo, y les permite aprender haciendo actividades e interactuando con el entorno virtual [1].
- **Televisión:** A menudo se utiliza la realidad aumentada en deportes como el fútbol para mostrar el resultado en el círculo central, en natación para mostrar una línea de récord frente a la situación actual, e incluso es común verlo en informativos que han optado por este nuevo lenguaje.
- **Entretenimiento:** Muchos juegos que utilizan esta tecnología han surgido estos últimos años, y algunos han alcanzado una altísima popularidad como por ejemplo el famoso juego ”Pokemon Go”.

Figura 2.2: Imagen de ”Pokemon Go”

- **Medicina:** La realidad aumentada se ha aplicado con mucho éxito a ecografías que permiten ver un modelo 3D del feto en movimiento gracias a diversas imágenes de ecografías 3D. También permite una vista interna del paciente sin necesidad de cirugía, lo que facilita el entrenamiento médico [2].
- **Turismo:** Muchos museos ofrecen una experiencia de realidad aumentada. Además diversas aplicaciones que permiten visitas turísticas también incluyen publicidad de los productos temáticos, lo que favorece el comercio.

Figura 2.3: Detalle de un cuadro en realidad aumentada.

Capítulo 3

Tecnologías utilizadas

En este capítulo se especifica acerca de las tecnologías que se van a utilizar a lo largo del desarrollo del proyecto, tanto para el desarrollo de la aplicación Android como de la librería de realidad aumentada, así como el control de versiones y lenguaje de programación.

3.1. Android

Android es un sistema operativo para dispositivos móviles desarrollado por Google, y se basa en el kernel de Linux. Es el sistema operativo más utilizado más utilizado del mundo, con una cuota de mercado superior al 80 % en 2018 [3].

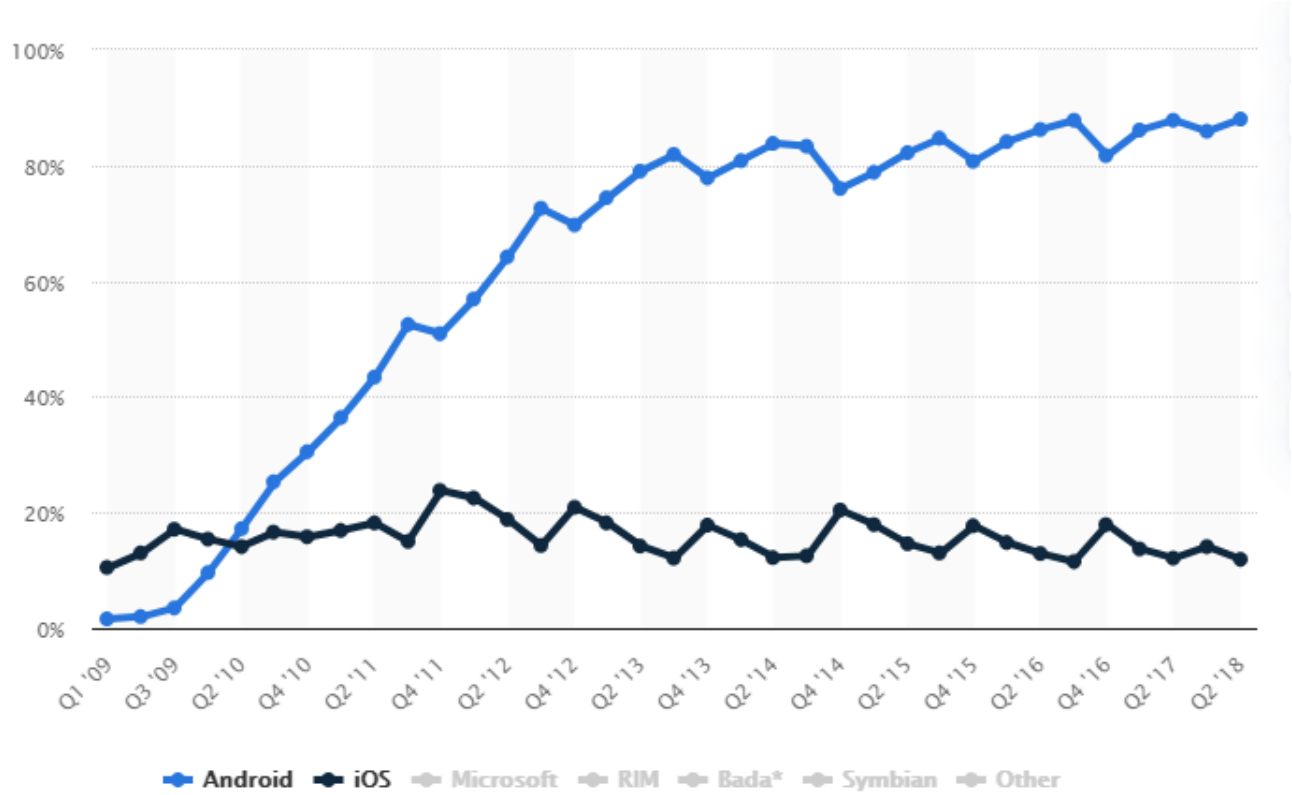


Figura 3.1: Comparación entre usuarios de Android e iOS a lo largo de la década. [3]

El IDE para desarrollo Android es Android Studio, ya que es el entorno de desarrollo oficial y gratuito que proporciona Google, sustituyendo a Eclipse como IDE más popular en el año 2013 [4]. Entre sus principales ventajas se encuentran que permite un desarrollo muy rápido debido a la fácil inclusión de librerías mediante el plugin Gradle, así como una función de auto-completar código muy inteligente. También cuenta con un emulador que podemos personalizar para comprobar el funcionamiento de nuestra aplicación sobre distintas versiones de Android y hardware emulado. Soporta varios lenguajes de programación: Java, Kotlin y C/C++.

3.2. Kotlin

El lenguaje de programación utilizado durante este proyecto es Kotlin, ya que es el lenguaje preferido por Google para el desarrollo de aplicaciones móviles desde el Google I/O de 2017 [5] y tiene una serie de ventajas que lo hacen la elección más adecuada para este tipo de proyectos. Se puede utilizar en cualquier dispositivo que ejecute la JVM (Java Virtual Machine), por lo que es completamente compatible con Java y sus librerías, y entre las ventajas que presenta sobre Java podemos encontrar:

- **Menos código:** No sólo la sintaxis de Kotlin es más fácil de entender, si no que es más concisa (por ejemplo, prescinde de la palabra clave new y de ";" después de cada declaración), lo que lleva a menos errores por parte de los programadores.
- **Sistema de tipado:** Kotlin cuenta con un sistema de inferencia de tipos que permite evitar los errores de null pointer que comúnmente se encuentran desarrollando en Java. Esto se debe a que Kotlin permite determinar variables Nullable mediante un carácter de interrogación. Este tipo de variables es especialmente útil cuando se trabaja con APIs, ya que reduce enormemente el número de comprobaciones necesarias.
- **Más seguro:** El compilador de Kotlin detecta errores en tiempo de compilación y no en tiempo de ejecución.

También existen otras ventajas que lo hacen más ligero como las corutinas o las "data classes" que permiten la creación de clases que sólo sirven para almacenar de forma mucho más simple [6]. En el siguiente snippet se puede ver la sintaxis de este lenguaje.

```
class Greeter(val name: String) {
    fun greet() {
    println("Hello, $name")
    }
}

    fun main(args: Array<String>) {
        Greeter(args[0]).greet()
}
```

3.3. Google Maps SDK

Es la API proporcionada por Google que permite a los desarrolladores incluir mapas en aplicaciones. Permite añadir marcadores, dibujar elementos sobre el mismo y también se encarga de controlar las acciones que lleve a cabo el usuario sobre el mapa.

Para poder utilizar este servicio se debe obtener una clave de uso para la API. Esto se consigue iniciando sesión con una cuenta de Google en la consola de desarrolladores de Android. A partir de 2018 este servicio dejó de ser gratuito y cada tipo de llamada a la API tiene su propia facturación. Hay que tener en cuenta que Google proporciona a los desarrolladores un crédito de 200 dólares mensuales para la API y el precio oscila entre 5 y 20 dólares por cada 100 peticiones en función del tipo de llamada que se ejecute [7].

3.4. Unity 2019

Se ha utilizado Unity como software para la creación de la librería de realidad aumentada. Es un motor de videojuego multiplataforma que puede utilizarse de forma gratuita para pequeños proyectos (menos de 100.000 dólares de ingresos) con la restricción de compilar con un splash screen con el icono y la leyenda ”Made with Unity”[8]. Unity permite añadir elementos a un espacio 3D y asignarles un comportamiento mediante un script (C#).

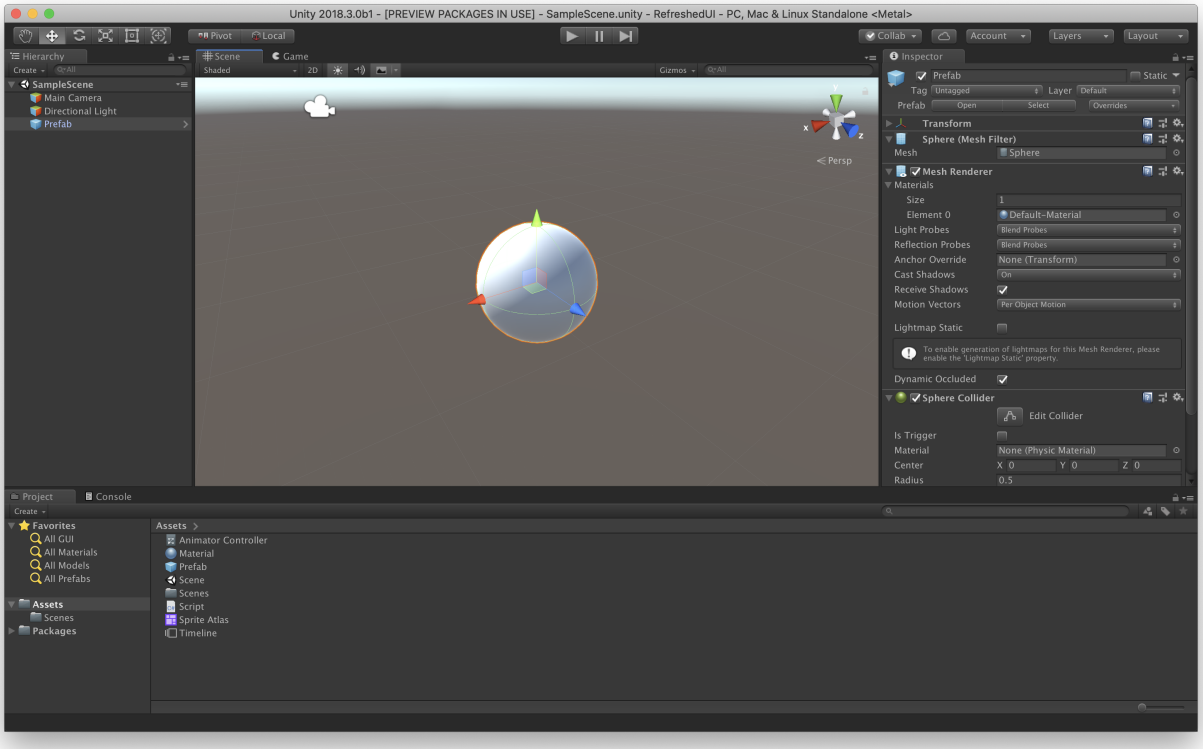


Figura 3.2: Interfaz de Unity.

Mediante el uso de distintos plugins y librerías podemos conseguir objetos con un comportamiento asignado por defecto y en este caso se utiliza el plugin de Vuforia para realidad aumentada, que se explica a continuación.

3.5. Vuforia Engine

Vuforia se trata de un kit de desarrollo software para aplicaciones de realidad aumentada que permite reconocer tanto imágenes planas como objetos 3D usando la cámara del dispositivo [9]. Usar Vuforia con Unity proporciona una serie de ventajas para este proyecto que lo hacen preferible a plataformas como AR Core de Google. AR Core no reconoce fácilmente patrones que incluyan fachadas de edificios ya que son patrones muy repetitivos [10].

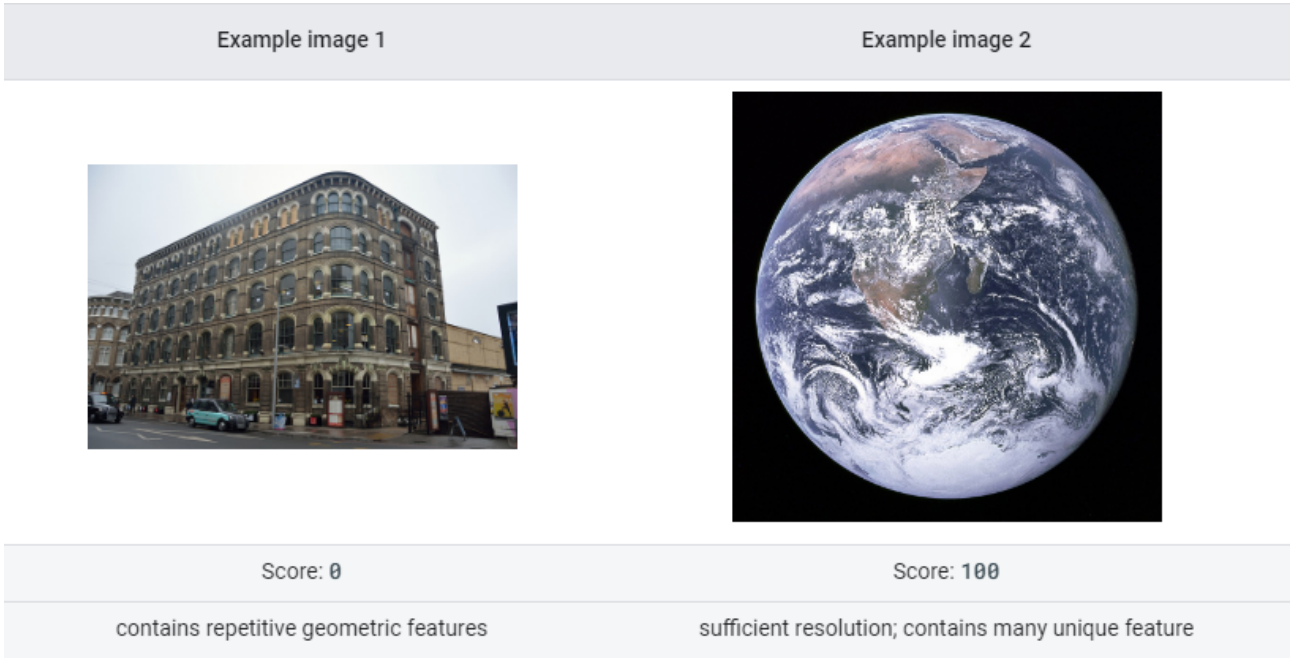


Figura 3.3: Comparación del reconocimiento de AR Core.
[10]

Además, una de las principales ventajas que proporciona AR Core es su control de los objetos y su posicionamiento en el espacio. Por ejemplo, si se coloca un objeto 3D en una superficie y se mueve el dispositivo de cualquier forma, AR Core recuerda la localización de dicho objeto 3D y al mover la cámara hacia ese punto se volverá a ver. Esto tiene mucho potencial, pero en este proyecto se necesita colocar el objeto 3D en un lugar muy concreto respecto al objeto reconocido, y para esta tarea Vuforia y Unity ofrecen un rendimiento óptimo.

3.6. Git

Para el control de versiones se utilizará Git. Es un famoso sistema gratuito y de código abierto. Como repositorio se usará GitHub, en el siguiente enlace:

https://github.com/Fer97p/ARGuide

Android Studio cuenta con una integración de control de versiones que permite enlazar un repositorio local con el repositorio en la nube, así como llevar a cabo cualquier tarea: commits, push, pull, etc.

Capítulo 4

Plan de proyecto

4.1. Metodología

El equipo que llevará a cabo el proyecto es de un único miembro, que asumirá todos los roles que se necesiten. Por este motivo, en lugar de utilizar metodología ágil, que es especialmente útil para organizar y gestionar equipos de tamaño medio-alto, se utilizará metodología de desarrollo en cascada, que utiliza una cantidad de recursos mínima, ya que se ajusta mejor a las necesidades de este proyecto.

4.1.1. Desarrollo en cascada

Se trata de un modelo lineal de desarrollo de software que emplea un proceso de diseño secuencial. Tiene varias ventajas, entres ellas se encuentran el hecho de que es un modelo ampliamente aceptado y fácil de seguir debido a su simplicidad. Permite la creación de un producto correcto y que cumple con las peticiones del cliente. La principal desventaja es la rigidez de su estructura, ya que si se altera el orden de sus fases el producto final pierde calidad. Esto puede deberse a un cambio en los requisitos o un malentendido durante las fases iniciales con el cliente. En este caso, dicha desventaja no es muy relevante debido a que no existe un cliente y los requisitos vienen dados por una única persona que también es el desarrollador. De esta forma, es difícil que los requisitos no se entiendan bien o cambien durante el desarrollo ya que todos proceden de una visión global del proyecto.
El desarrollo fluye secuencialmente desde el punto inicial hasta el punto final, con varias etapas diferentes.

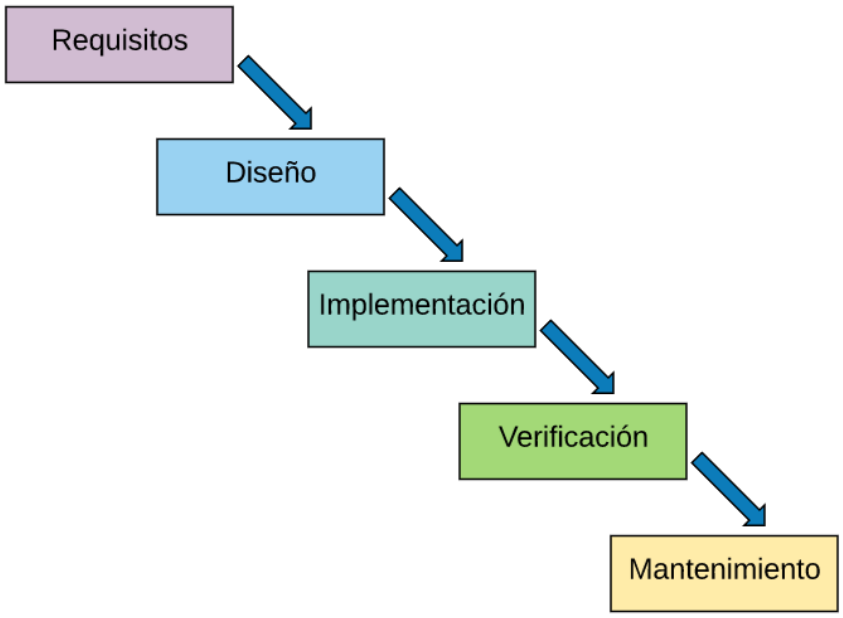


Figura 4.1: Distintas fases del desarrollo en cascada. [11]

Fase de análisis de requisitos

Esta se trata de la primera fase de este desarrollo. Consiste en obtener una lista coherente de requisitos que servirán de guía a lo largo del proyecto, determinando lo que el sistema debe ser capaz de hacer una vez el desarrollo haya concluido. Se debe elaborar una lista inicial de requisitos, para posteriormente evaluarlos conjuntamente y darles coherencia así como asignar a cada uno una prioridad. Por último, se registra toda esta información en un documento de requisitos.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Realizar proyecto	**50,25 días**	**lun 13/04/20**	**lun 22/06/20**	
Fase de análisis de requisitos	**5,75 días**	**lun 13/04/20**	**lun 20/04/20**	
Determinación de requisitos	12 horas	lun 13/04/20	mar 14/04/20	
Análisis de requisitos	12 horas	mar 14/04/20	mié 15/04/20	3
Especificación de requisitos	6 horas	jue 16/04/20	jue 16/04/20	4
Validación de requisitos	6 horas	jue 16/04/20	vie 17/04/20	5
Plan de riesgos	10 horas	vie 17/04/20	lun 20/04/20	6
Requisitos obtenidos	0 días	lun 20/04/20	lun 20/04/20	7

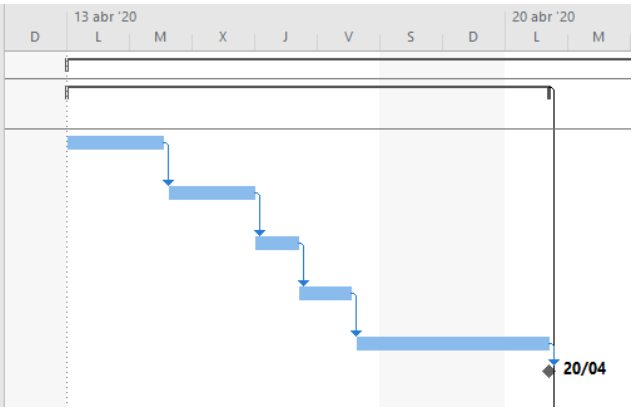
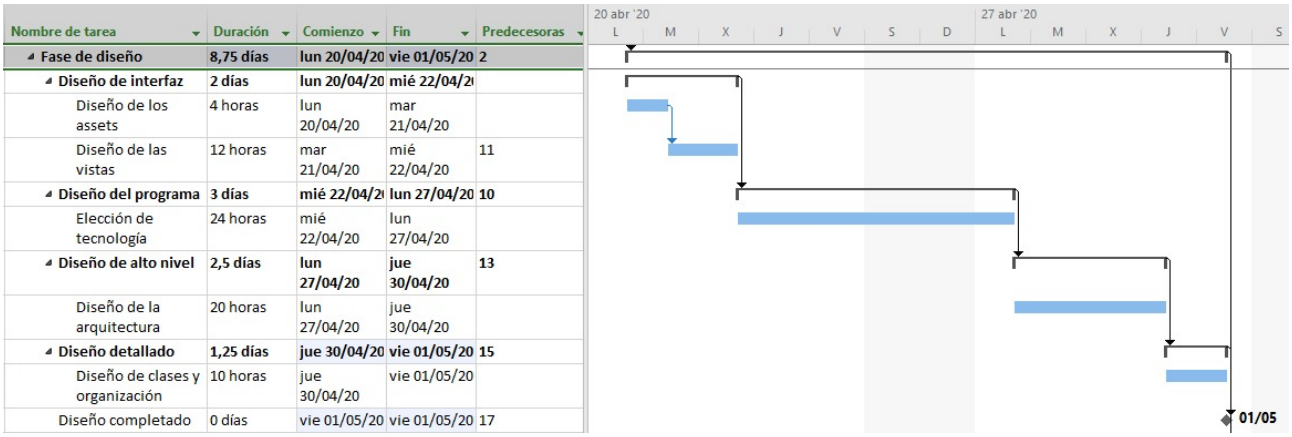


Figura 4.2: Diagrama de Gantt del análisis de requisitos.

Fase de diseño del sistema

En esta etapa se describe la estructura interna del software, las relaciones entre las entidades que lo componen y la especificación de lo que debe hacer cada una de sus partes.

Conviene distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El diseño de alto nivel define la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos y sus relaciones. Con ello se define la arquitectura de la solución elegida. El diseño detallado define los algoritmos empleados y la organización del código para comenzar la implementación. El diseño del programa comprende el análisis de las herramientas que se utilizarán posteriormente en la fase de implementación o codificación. [12]



Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase de diseño	**8,75 días**	**lun 20/04/20**	**vie 01/05/20**	**2**
Diseño de interfaz	**2 días**	**lun 20/04/20**	**mié 22/04/2**	
Diseño de los assets	4 horas	lun 20/04/20	mar 21/04/20	
Diseño de las vistas	12 horas	mar 21/04/20	mié 22/04/20	11
Diseño del programa	**3 días**	**mié 22/04/2**	**lun 27/04/20**	**10**
Elección de tecnología	24 horas	mié 22/04/20	lun 27/04/20	
Diseño de alto nivel	**2,5 días**	**lun 27/04/20**	**jue 30/04/20**	**13**
Diseño de la arquitectura	20 horas	lun 27/04/20	jue 30/04/20	
Diseño detallado	**1,25 días**	**jue 30/04/20**	**vie 01/05/20**	**15**
Diseño de clases y organización	10 horas	jue 30/04/20	vie 01/05/20	
Diseño completado	0 días	vie 01/05/20	vie 01/05/20	17

Figura 4.3: Diagrama de Gantt de la fase de diseño.

Fase de implementación

Esta es la etapa en la que se elabora el código fuente y el resto de elementos necesarios para la creación del sistema. Durante este periodo se emplean las tecnologías investigadas en la fase de diseño de programa y es la fase más costosa en el tiempo respecto al resto de fases. Esto se debe a que el desarrollador no está completamente familiarizado con la tecnología a utilizar y por ello, para agilizarlo, se deben utilizar estrategias como uso de librerías o reutilización de código. En este proyecto se divide esta fase en la implementación de la aplicación Android por un lado y la implementación de la librería Unity por otro. Son dos elementos muy distintos que deben trabajar juntos para crear un producto funcional.

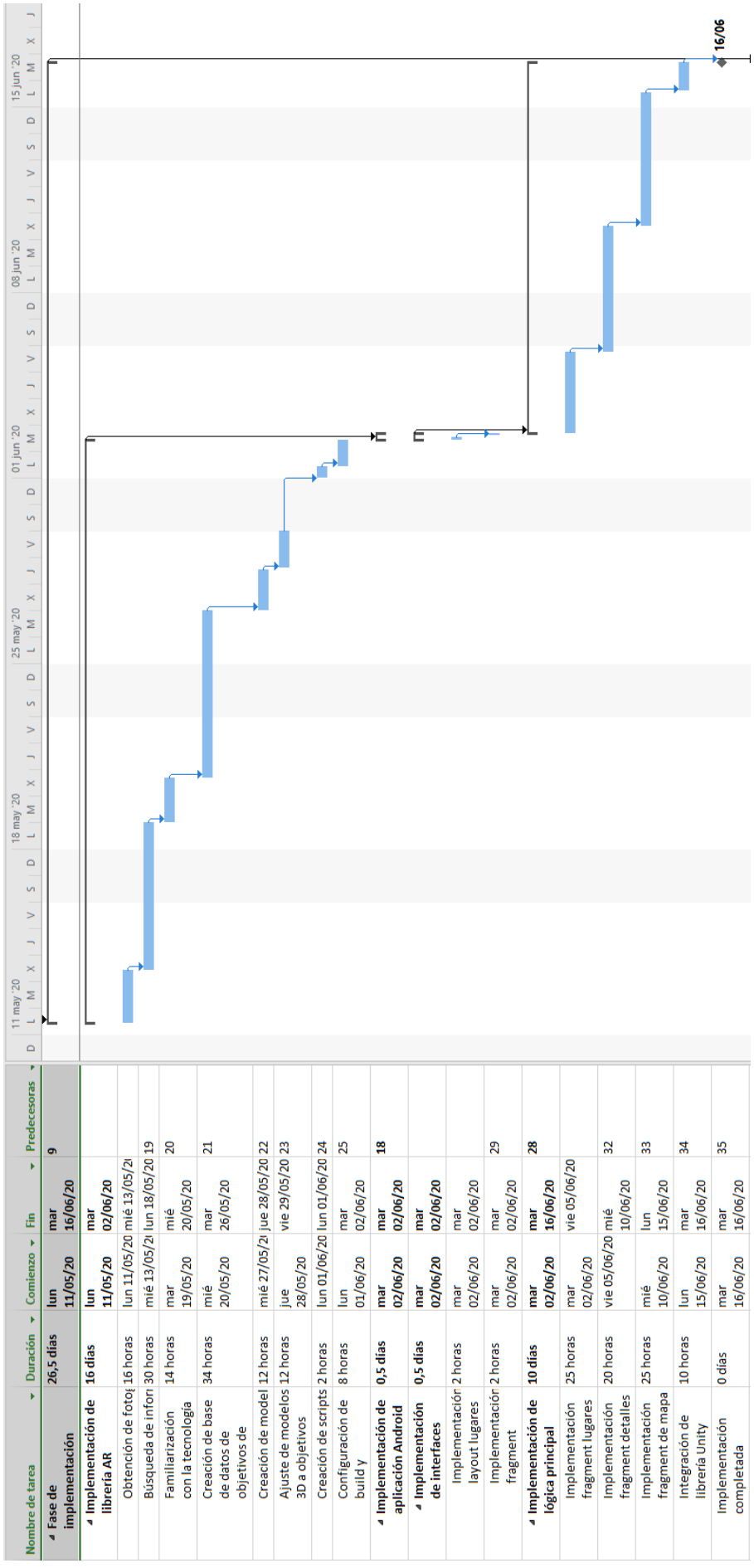


Figura 4.4: Diagrama de Gantt de la fase de implementación.

Fase de verificación

En esta fase se genera una batería de pruebas de caja negra para su posterior comprobación. Una vez todos los test tengan un resultado satisfactorio, se avanzará a la fase de despliegue y mantenimiento.

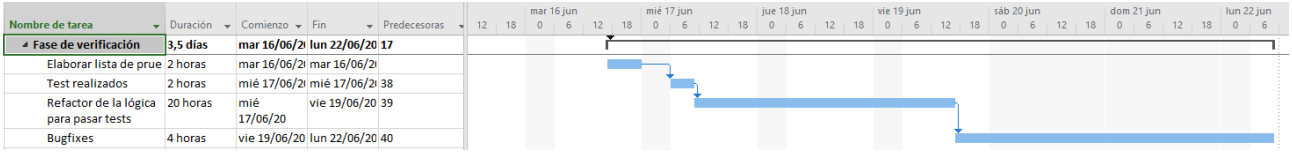


Figura 4.5: Diagrama de Gantt de la fase de verificación.

Fase de despliegue y mantenimiento

En esta etapa se preparará la aplicación para su despliegue (en este caso se generará el pack necesario para poder subirla a Google Play) y se creará un manual de usuario. Posteriormente, el desarrollador deberá mantener la aplicación (en caso de que algún objetivo de imagen cambie, por ejemplo) y la actualizará, subiendo una nueva versión a la tienda de aplicaciones.

4.2. Plan de control del proyecto

4.2.1. Gestión de gastos

Para el desarrollo del proyecto se ha empleado hardware, software y recursos personales.

- **Recursos personales:** Si estimamos el sueldo medio de un programador en 18€/hora, y el proyecto está estimado en 356 horas, se obtiene un total de 6408 €.
- **Recursos materiales:** Para el desarrollo del proyecto se ha empleado un teléfono móvil Android OnePlus 7T, con un precio de 440 €. También se ha utilizado un ordenador portátil Huawei Matebook D15 con un precio de 635 €. Entre ambos dispositivos obtenemos un total de 1075 €. Si estimamos la vida útil de estos dispositivos en 4 años (48 meses), y el desarrollo del proyecto en 4 meses, obtenemos como amortización 1075*4/48 = 89,58 €.
- **Recursos de software:** Para el desarrollo de la librería Unity necesitamos una licencia de Unity Pro. Esta licencia tiene un precio de 126 € al mes. La versión básica de Vuforia (Vuforia Basic) que permite desarrollos con reconocimientos ilimitados tiene un precio de 35 € al mes. Estimando la duración de la fase de implementación y pruebas en 2 meses, se calcula un total de 35*2+126*2 = 322 € en costes de software.

En total, el coste teórico estimado del proyecto asciende a un total de 6819,58 €, puesto que el coste real no incluye el coste de recursos de personal ni licencias comerciales.

4.2.2. Gestión de riesgos

A lo largo del desarrollo pueden producirse inconvenientes o darse ciertas situaciones que impidan continuar con el mismo, retrasarlo o hacer que el producto final pierda calidad. A continuación se expondrán los riesgos más comunes para un proyecto de este tipo, detallando sus siguientes características:

- Nombre del riesgo.
- Descripción del riesgo.
- Probabilidad de que se de dicha situación. Se categoriza en:
  - Muy Alta: Más del 50 % de probabilidad.
  - Alta: Entre el 30 y el 50 % de probabilidad.
  - Media: Entre el 10 y el 30 % de probabilidad.
  - Baja: Inferior o igual al 10 % de probabilidad.
- Impacto que tendría esta situación sobre el proyecto. Se categoriza en:
  - Crítico: Supondría un aumento de tiempo de más de un 20 % del tiempo total.

- Importante: Supondría un aumento de tiempo de un 10 a un 20 % del tiempo total.
- Tolerable: Supondría un aumento de un 5 a un 10 % del tiempo total.
- Insignificante: Supondría un aumento de tiempo menor al 5 % del tiempo total.

- Plan de protección frente al riesgo.
- Plan de contingencia.

A continuación se expondrá un listado con todos los riesgos que se han considerado relevantes para este proyecto.

Identificador	Risk01
Nombre del riesgo	Modificación de los requisitos
Descripción del riesgo	Si se un cambio en los requisitos durante la fase de implementación puede hacer que el producto final pierda calidad.
Probabilidad	Baja.
Impacto	Importante.
Plan de protección frente al riesgo	Dedicar más tiempo a la fase de análisis de requisitos para estudiar todas las posibilidades.
Plan de contingencia	Estudiar la implementación de los cambios de requisitos de forma que afecte lo menos posible al resto de la aplicación.

Cuadro 4.1: Risk01 - Modificación de los requisitos

Identificador	Risk02
Nombre del riesgo	Retraso en la fecha de fin de una tarea
Descripción del riesgo	Puede que la estimación de tiempo para cada tarea no sea correcta y en ocasiones las tareas lleven más tiempo del esperado, retrasando todo el proyecto.
Probabilidad	Media.
Impacto	Importante.
Plan de protección frente al riesgo	Intentar estimar los tiempos lo mejor posible y buscar información externa en caso de no poder avanzar con el desarrollo.
Plan de contingencia	Procurar llevar a cabo otras tareas más sencillas en menos tiempo del estimado.

Cuadro 4.2: Risk02 - Retraso en la fecha de fin de una tarea

Identificador	Risk03
Nombre del riesgo	Pérdida del código fuente
Descripción del riesgo	Debido a algún fallo en el almacenamiento o por la corrupción de ciertos archivos, se puede perder el código ya escrito y volver a repetir la implementación es muy costoso en el tiempo.
Probabilidad	Baja.
Impacto	Crítico.
Plan de protección frente al riesgo	Usar un sistema de control de versiones, tanto local como en la nube para poder recuperar el código en caso de fallo local.
Plan de contingencia	Evaluar las consecuencias y reestructurar el proyecto teniendo en cuenta cómo se desarrolló la primera vez.

Cuadro 4.3: Risk03 - Pérdida del código fuente

Identificador	Risk04
Nombre del riesgo	Escasez de recursos personales
Descripción del riesgo	Al tratarse de un proyecto de cuyo desarrollo se encarga una única persona, si esta persona enferma o no puede continuar con el desarrollo, el impacto es muy alto ya que no puede ser cubierto por ninguna otra persona e inevitablemente se detiene el avance.
Probabilidad	Baja.
Impacto	Crítico.
Plan de protección frente al riesgo	Ninguno.
Plan de contingencia	Evaluar las consecuencias y reestructurar el proyecto de acuerdo al tiempo restante para llevarlo a cabo.

Cuadro 4.4: Risk04 - Escasez de recursos personales

Identificador	Risk05
Nombre del riesgo	Diseño pobre
Descripción del riesgo	Si el diseño no es bueno, la fase de implementación se vería seriamente afectada, ya que en ciertos aspectos no se podría seguir un guión firme y el producto final sería de menor calidad.
Probabilidad	Baja.
Impacto	Importante.
Plan de protección frente al riesgo	Completar un buen diseño de la aplicación aunque el tiempo dedicado sea mayor del estimado inicialmente.
Plan de contingencia	Investigar acerca de aquellos aspectos del diseño que no han quedado claros en la fase de diseño durante la fase de implementación para intentar mitigar el impacto.

Cuadro 4.5: Risk05 - Diseño pobre

Identificador	Risk06
Nombre del riesgo	Mala planificación inicial
Descripción del riesgo	Si la planificación y estimación inicial es pobre, se puede pensar que se ha dedicado demasiado poco o demasiado tiempo a ciertas tareas y centrarse más en el tiempo que debería tardar en lugar de llevar a cabo la tarea de forma correcta.
Probabilidad	Media.
Impacto	Tolerable.
Plan de protección frente al riesgo	Intentar estimar el tiempo de las tareas de la mejor forma posible.
Plan de contingencia	No guiarse demasiado por el tiempo que se debería tardar de manera que no se dejen las tareas a medias porque el tiempo estimado es menor al tiempo empleado.

Cuadro 4.6: Risk06 - Mala planificación inicial

Identificador	Risk07
Nombre del riesgo	Problemas con el hardware
Descripción del riesgo	Puede darse una situación en la que alguno de los elementos de hardware que se utilizan para el desarrollo del proyecto dejen de funcionar o funcionen mal.
Probabilidad	Baja.
Impacto	Importante.
Plan de protección frente al riesgo	Cuidar de los elementos de trabajo.
Plan de contingencia	Usar algún otro equipo de desarrollo que tengamos disponible.

Cuadro 4.7: Risk07 - Problemas con el hardware

Identificador	Risk08
Nombre del riesgo	Problemas con el software
Descripción del riesgo	Puede que las condiciones de uso o las condiciones de las licencias del software empleado para codificar cambien durante el desarrollo del proyecto, de forma que no se puedan utilizar más o no se puedan utilizar de forma gratuita.
Probabilidad	Baja.
Impacto	Importante.
Plan de protección frente al riesgo	Usar software de desarrollo que estén muy extendidos o de código abierto.
Plan de contingencia	Buscar una alternativa software que se puede utilizar para el mismo fin.

Cuadro 4.8: Risk08 - Problemas con el software

Identificador	Risk09
Nombre del riesgo	Complicaciones externas al proyecto
Descripción del riesgo	Pueden darse situaciones externas al proyecto que impidan el desarrollo del mismo, como por ejemplo apagones, problemas con la conexión a internet, etc.
Probabilidad	Baja.
Impacto	Tolerable.
Plan de protección frente al riesgo	Ninguno.
Plan de contingencia	Intentar continuar con otra parte del proyecto que pueda ser ejecutada para perder el menor tiempo posible.

Cuadro 4.9: Risk09 - Complicaciones externas al proyecto

Identificador	Risk10
Nombre del riesgo	Desconocimiento de las tecnologías a emplear
Descripción del riesgo	Suele ser necesario algo de tiempo para familiarizarse con nuevas tecnologías que se emplean en el proyecto.
Probabilidad	Muy alta.
Impacto	Tolerable.
Plan de protección frente al riesgo	Documentarse e informarse lo máximo posible antes de comenzar con la fase de implementación.
Plan de contingencia	Buscar información sobre aquellas tareas que no sepamos implementar fácilmente.

Cuadro 4.10: Risk10 - Desconocimiento de las tecnologías a emplear

4.3. Seguimiento del proyecto

En esta sección se expondrá el seguimiento del proyecto frente al plan inicial del mismo, explicando las diferencias que se han producido. Se ilustrarán dichas diferencias en un diagrama de Gantt frente a la línea base de la planificación inicial (en color negro)

4.3.1. Elección de tecnología

Se empleó más tiempo del esperado para determinar la tecnología a emplear para el desarrollo de la librería de realidad aumentada. Fue necesario crear varias aplicaciones de prueba con las distintas API de realidad aumentada (Wikitude, AR Core, Vuforia). Por este motivo esta tarea pasó a tener una duración de 30 horas en lugar de las 20 previstas.

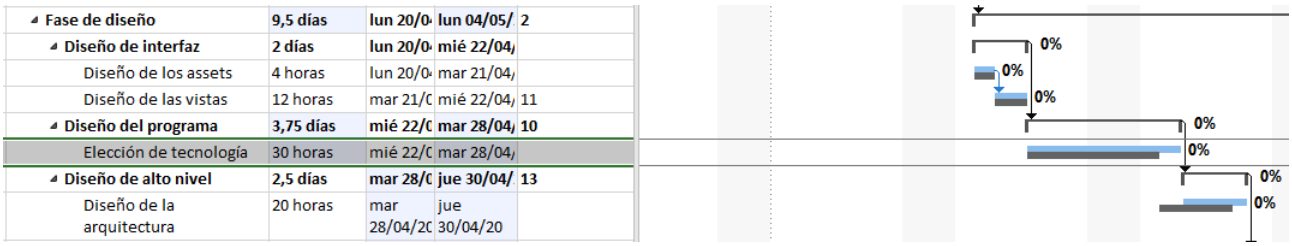


Figura 4.6: Comparación de línea base con la tarea Elección de tecnología.

4.3.2. Configuración de build y exportación de librería

En esta tarea también se dedicó más tiempo del esperado debido a una serie de errores que se producían por no limpiar de forma correcta la caché de Android Studio así como no eliminar correctamente todos los archivos presentes de una versión anterior de dicha librería. Por esto esta tarea pasó a necesitar 12 horas en lugar de las 8 horas previstas.

Figura 4.7: Comparación de línea base con la tarea Configuración de build.

4.3.3. Implementación de librería Unity

Debido al confinamiento en la ciudad de Valladolid por la enfermedad COVID-19, que ha provocado que se tomen medidas de contención en todo tipo de ámbitos, tuvo que posponerse el desarrollo de la librería de realidad aumentada ya que se necesita tomar una serie de fotografías de los lugares a reconocer y la mayoría de estos lugares se encontraban fuera del radio permitido para recorrer durante el confinamiento. Por este motivo, en lugar de comenzar la fase de implementación con esta librería, se comenzó implementando la aplicación Android.

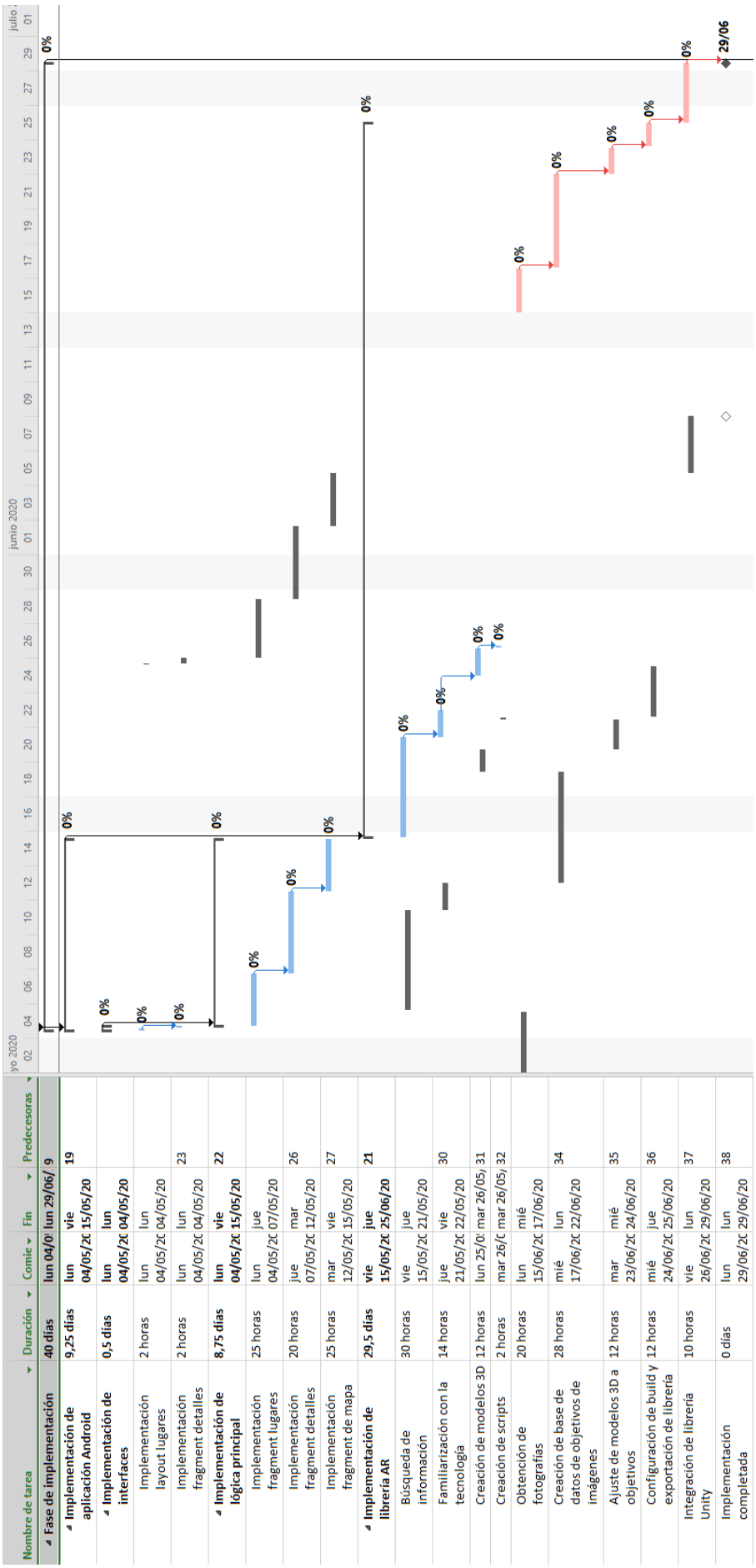


Figura 4.8: Intercambio en el orden de implementación Android y librería Unity.

4.3.4. Obtención de fotografías

Como se ha explicado en el apartado anterior, el confinamiento no permitió obtener las fotografías necesarias para la creación de la librería de realidad aumentada. No fue hasta el día 15 de junio que se pudo salir hasta las zonas más alejadas de la ciudad. Mientras tanto, sólo fue posible adelantar las tareas de búsqueda de información, familiarización con la tecnología y creación de modelos 3D con las imágenes antiguas obtenidas de internet.

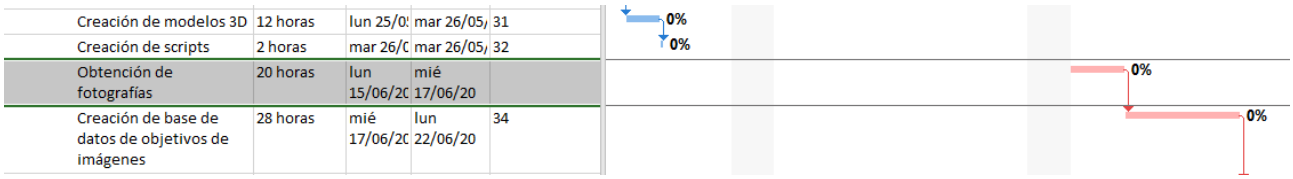


Figura 4.9: Comparación de línea base con la tarea obtención de fotografías.

4.3.5. Creación de base de datos de objetivos de imagen

A pesar de tener que esperar hasta el 15 de junio para conseguir las imágenes necesarias, se dispuso de mucho tiempo para hacer pruebas y descubrir el funcionamiento de Vuforia y Unity. Por este motivo, el tiempo de la creación de la base de datos se redujo a 28 horas en lugar de las 34 previstas, ganando algo de tiempo.

Figura 4.10: Comparación de línea base con la tarea creación de base de datos de objetivos de imagen.

A pesar de este recorte en el tiempo, el balance total es adecuado, ya que se previó como fecha de fin de proyecto el 22 de junio cuando la fecha de finalización real ha sido el 2 de julio, y un retraso de 10 días es aceptable en el mundo real. En total, el coste teórico del proyecto tras su seguimiento asciende a un total de 6963,58 €, puesto que las horas necesarias para llevarlo a cabo han aumentado ligeramente.

Capítulo 5

Análisis del proyecto

5.1. Elicitación de requisitos

5.1.1. Requisitos funcionales

En esta sección se establecen las funciones que el sistema debe ser capaz de cumplir para tener un comportamiento correcto.

Identificador	FR01
Descripción	El sistema deberá mostrar al usuario una lista con todos los lugares que se pueden visitar.

Cuadro 5.1: FR01 - Listado de lugares

Identificador	FR02
Descripción	El sistema deberá mostrar al usuario una imagen y un título para cada lugar de la lista.

Cuadro 5.2: FR02 - Presentación de lugar

Identificador	FR03
Descripción	El sistema deberá mostrar al usuario detalles de un lugar (imágenes y texto) en una nueva pantalla si selecciona un elemento de la lista.

Cuadro 5.3: FR03 - Detalles de lugar

Identificador	FR04
Descripción	El sistema deberá permitir al usuario cambiar la imagen que ve en la pantalla de detalles deslizando hacia izquierda o derecha.

Cuadro 5.4: FR04 - Image slider

Identificador	FR05
Descripción	El sistema deberá permitir al usuario abrir el mapa con una ruta hacia el destino si está viendo los detalles de un lugar.

Cuadro 5.5: FR05 - Ruta al destino

Identificador	FR06
Descripción	El sistema deberá permitir al usuario mover el mapa o hacer zoom sobre el mismo una vez la ruta se haya dibujado.

Cuadro 5.6: FR06 - Control de mapa

Identificador	FR07
Descripción	El sistema deberá permitir al usuario abrir la cámara del dispositivo para reconocer imágenes.

Cuadro 5.7: FR07 - Abrir la cámara

Identificador	FR08
Descripción	El sistema deberá mostrar al usuario un modelo 3D cuando la aplicación reconozca un objetivo de imagen.

Cuadro 5.8: FR08 - Reconocimiento de imagen

Identificador	FR09
Descripción	El sistema deberá permitir al usuario activar la localización si es necesaria mediante un aviso en pantalla.

Cuadro 5.9: FR09 - Servicio de localización

Identificador	FR10
Descripción	El sistema deberá mostrar al usuario un mensaje de aviso si rechaza alguna solicitud de permisos o de activación de servicios.

Cuadro 5.10: FR10 - Aviso de rechazo en permisos esenciales

5.1.2. Requisitos no funcionales

Estos requisitos determinan características de funcionamiento y restricciones del sistema que complementan a los requisitos funcionales.

Identificador	NFR01
Descripción	El sistema será desarrollado para dispositivos Android.

Cuadro 5.11: NFR01 - Sistema operativo

Identificador	NFR02
Descripción	El sistema podrá ser instalado en dispositivos Android con versión superior a 5.0 y con un procesador tanto ARM-v7 como ARM64, pero no x86 (en desuso).

Cuadro 5.12: NFR02 - Requisitos del dispositivo Android

Identificador	NFR03
Descripción	El sistema utilizará la API de Google Maps para la gestión de rutas hacia cada lugar.

Cuadro 5.13: NFR03 - API de mapas

Identificador	NFR04
Descripción	El sistema utilizará Vuforia como motor de reconocimiento de imágenes para la realidad aumentada.

Cuadro 5.14: NFR04 - Motor de realidad aumentada

Identificador	NFR05
Descripción	El sistema deberá tener un rendimiento adecuado en dispositivos que cumplan ciertos requisitos mínimos (2GB RAM, octa-core, versión Android 5.0).

Cuadro 5.15: NFR05 - Rendimiento

Identificador	NFR06
Descripción	El sistema deberá pedir al usuario los permisos que necesite en cada momento si no han sido concedidos hasta ese momento.

Cuadro 5.16: NFR06 - Permisos

Identificador	NFR07
Descripción	El sistema almacenará todas las imágenes que utilice de forma que no necesite consumir datos de red para obtenerlas.

Cuadro 5.17: NFR07 - Almacenamiento de imágenes

Identificador	NFR08
Descripción	El sistema requiere conexión a internet y la ubicación del dispositivo para mostrar una ruta.

Cuadro 5.18: NFR08 - Almacenamiento de imágenes

Identificador	NFR09
Descripción	El sistema usará las librerías de Material Design para un mejor diseño y aspecto.

Cuadro 5.19: NFR09 - Material Design

5.1.3. Reglas de negocio

En esta sección se expondrán algunas restricciones que la aplicación establecerá sobre sus funciones.

Identificador	CR01
Descripción	Los lugares almacenados en el sistema serán instancias únicas, no puede haber dos instancias iguales almacenadas.

Cuadro 5.20: CR01 - Instancias de lugar singleton

Identificador	CR02
Descripción	Sólo se podrá acceder a la cámara desde la vista de detalles o la vista del mapa.

Cuadro 5.21: CR02 - Navegación de la aplicación

5.1.4. Requisitos de información

La información que almacenará el sistema es la siguiente:

Identificador	IR01
Descripción	El sistema almacenará la información de los lugares que se pueden visitar con los siguientes campos para cada lugar: ▪ Un identificador. ▪ Nombre completo. ▪ Una descripción. ▪ Su imagen de portada. ▪ Latitud y longitud.

Cuadro 5.22: IR01 - Lugar

5.2. Casos de uso

5.2.1. Diagrama de casos de uso

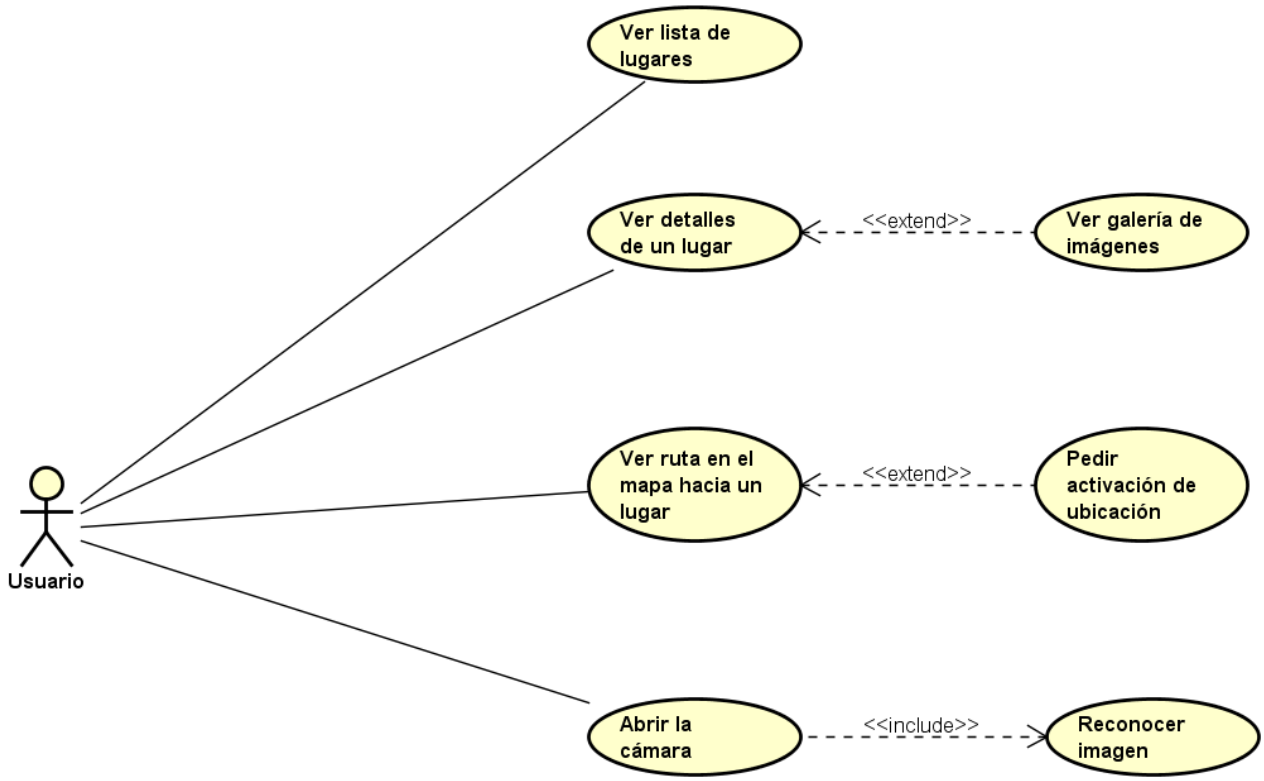


Figura 5.1: Diagrama de casos de uso.

A continuación se describirán los casos de uso de forma individual, junto a un diagrama de actividad que muestra una secuencia de acciones que componen dicho caso de uso.

Identificador	UC01
Nombre	Ver lista de lugares
Dependencias	Ninguna
Descripción	El usuario podrá ver una lista de lugares que puede visitar cuando inicie la aplicación.
Precondición	Ninguna.
Secuencia normal	1. El usuario inicia la aplicación. 2. El sistema muestra la pantalla de carga (splash screen). 3. El sistema muestra la lista de lugares.
Frecuencia	Siempre que el usuario inicie el sistema o vuelva de la pantalla de detalles.

Cuadro 5.23: UC01 - Ver lista de lugares

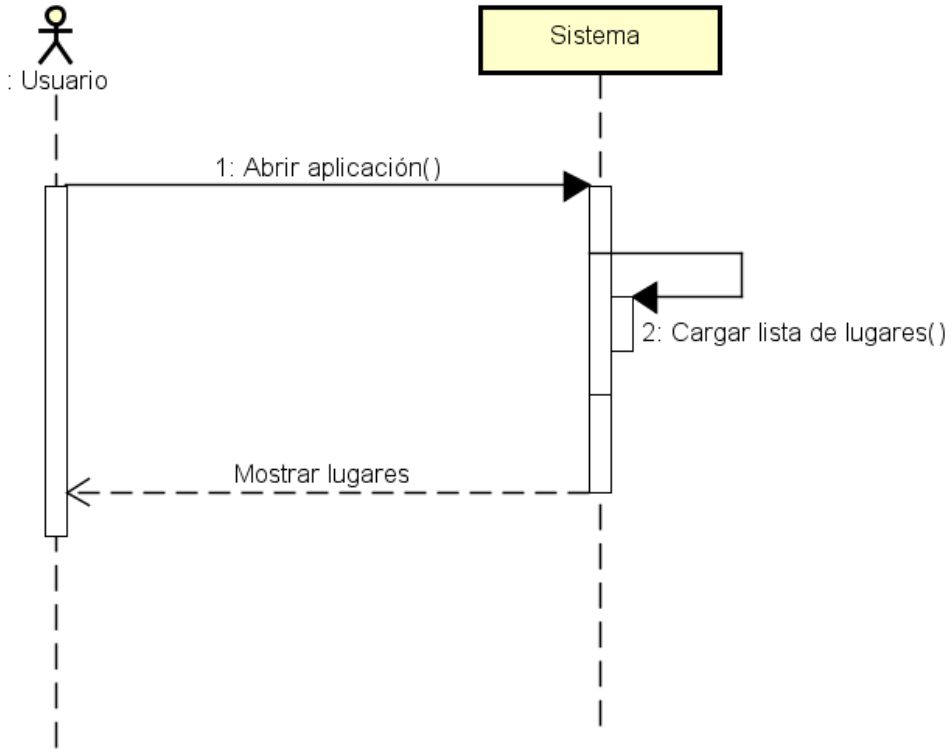


Figura 5.2: Diagrama secuencia de UC01.

Identificador	UC02
Nombre	Ver detalles de un lugar
Dependencias	UC03 - Ver galería de imágenes
Descripción	El usuario podrá ver una serie de detalles de un lugar que haya seleccionado (galería de imágenes y descripción).
Precondición	Estar en la pantalla de la lista de lugares.
Secuencia normal	1. El usuario selecciona un lugar de la lista. 2. El sistema muestra la pantalla de detalles de dicho lugar.
Frecuencia	Siempre que el usuario seleccione un lugar de la lista o vuelva de la pantalla de mapas.

Cuadro 5.24: UC02 - Ver detalles de un lugar

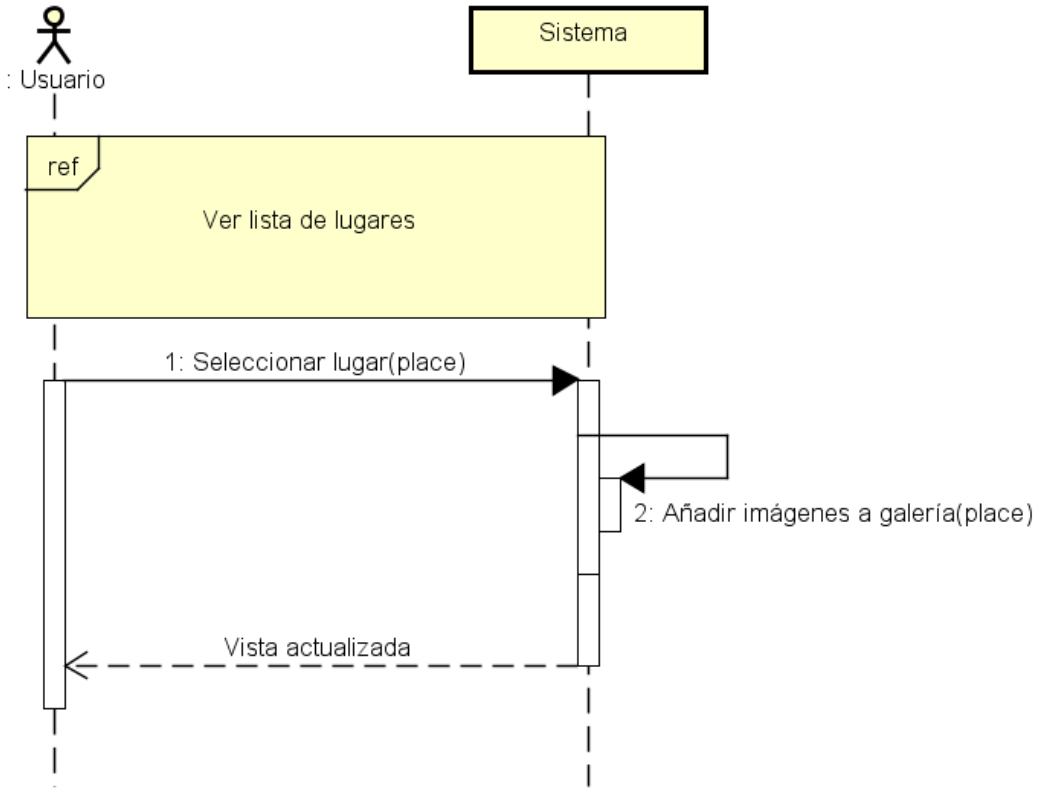


Figura 5.3: Diagrama secuencia de UC02.

Identificador	UC03
Nombre	Ver galería de imágenes
Dependencias	UC02 - Ver detalles de un lugar
Descripción	El usuario puede deslizar a izquierda y derecha sobre las imágenes de la galería para ver otras nuevas.
Precondición	Estar en la pantalla de detalles de un lugar.
Secuencia normal	1. El usuario desliza la imagen del slider. 2. El sistema cambia la imagen que muestra y actualiza el contador de imágenes.
Frecuencia	Entre 3 y 4 veces por sesión.

Cuadro 5.25: UC03 - Ver galería de imágenes

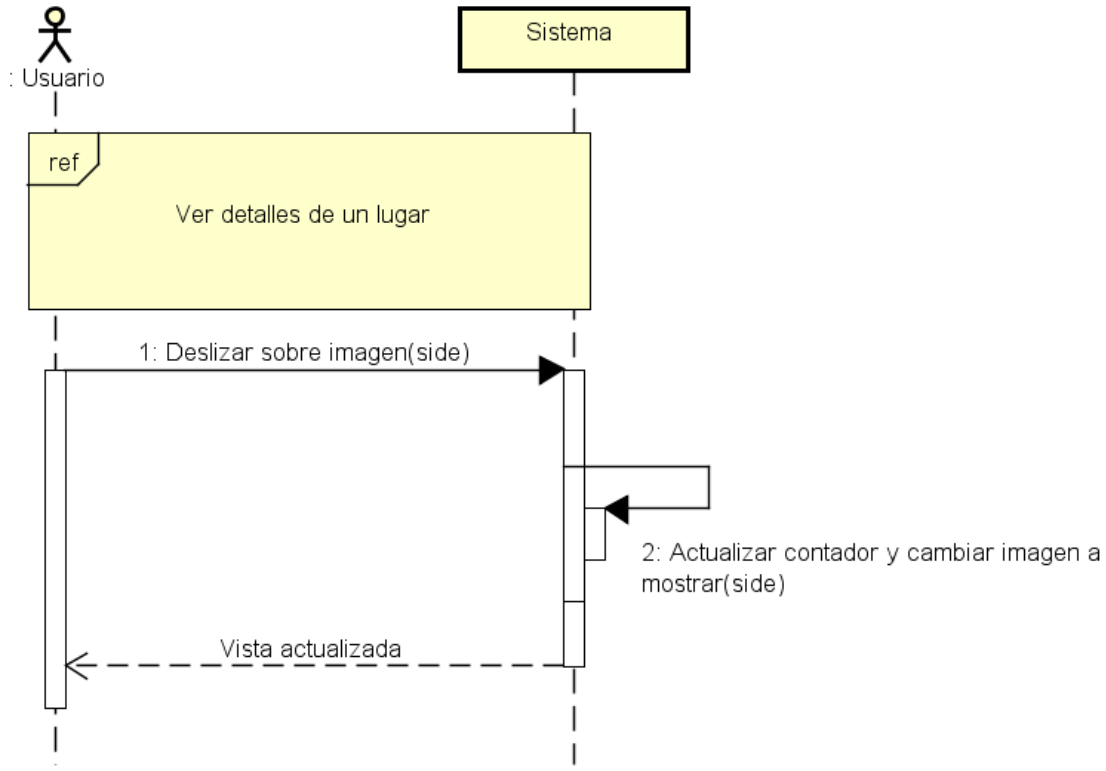


Figura 5.4: Diagrama secuencia de UC03.

Identificador	UC04
Nombre	Ver ruta en el mapa hacia un lugar
Dependencias	UC05 - Activar la ubicación desde la aplicación
Descripción	El usuario puede conocer la ruta hacia el lugar cuyos detalles está viendo dibujada en el mapa.
Precondición	Estar en la pantalla de detalles de un lugar.
Secuencia normal	1. El usuario pulsa el botón de navegación. 2. Si la ubicación está activada, se muestra la ruta. 3. Si la ubicación no está activada, el sistema pide al usuario que la active a través de un mensaje. 4. Si el usuario lo rechaza, se le muestra un mensaje de aviso. 5. Si el usuario lo activa, se muestra la ruta.
Secuencia excepcional	1. El usuario pulsa el botón de navegación. 2. Si la ubicación no está activada, el sistema pide al usuario que la active a través de un mensaje. 3. Si el usuario lo rechaza, se le muestra un mensaje de aviso. 4. Si el usuario lo ignora, el caso de uso queda sin efecto.
Frecuencia	Cada vez que el usuario pulse el botón de navegación o vuelva de la pantalla de cámara.

Cuadro 5.26: UC04 - Ver ruta en el mapa hacia un lugar

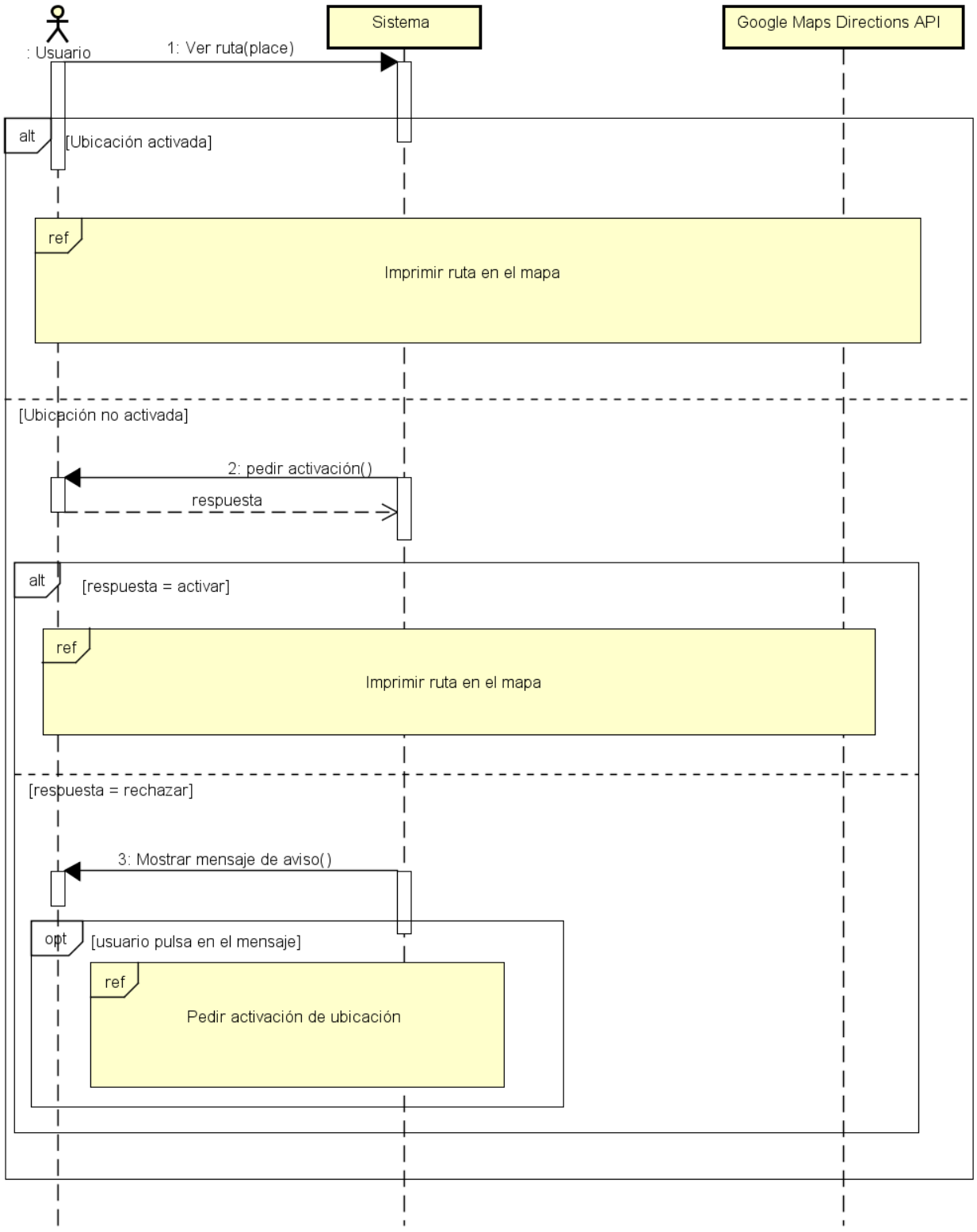


Figura 5.5: Diagrama secuencia de UC04.

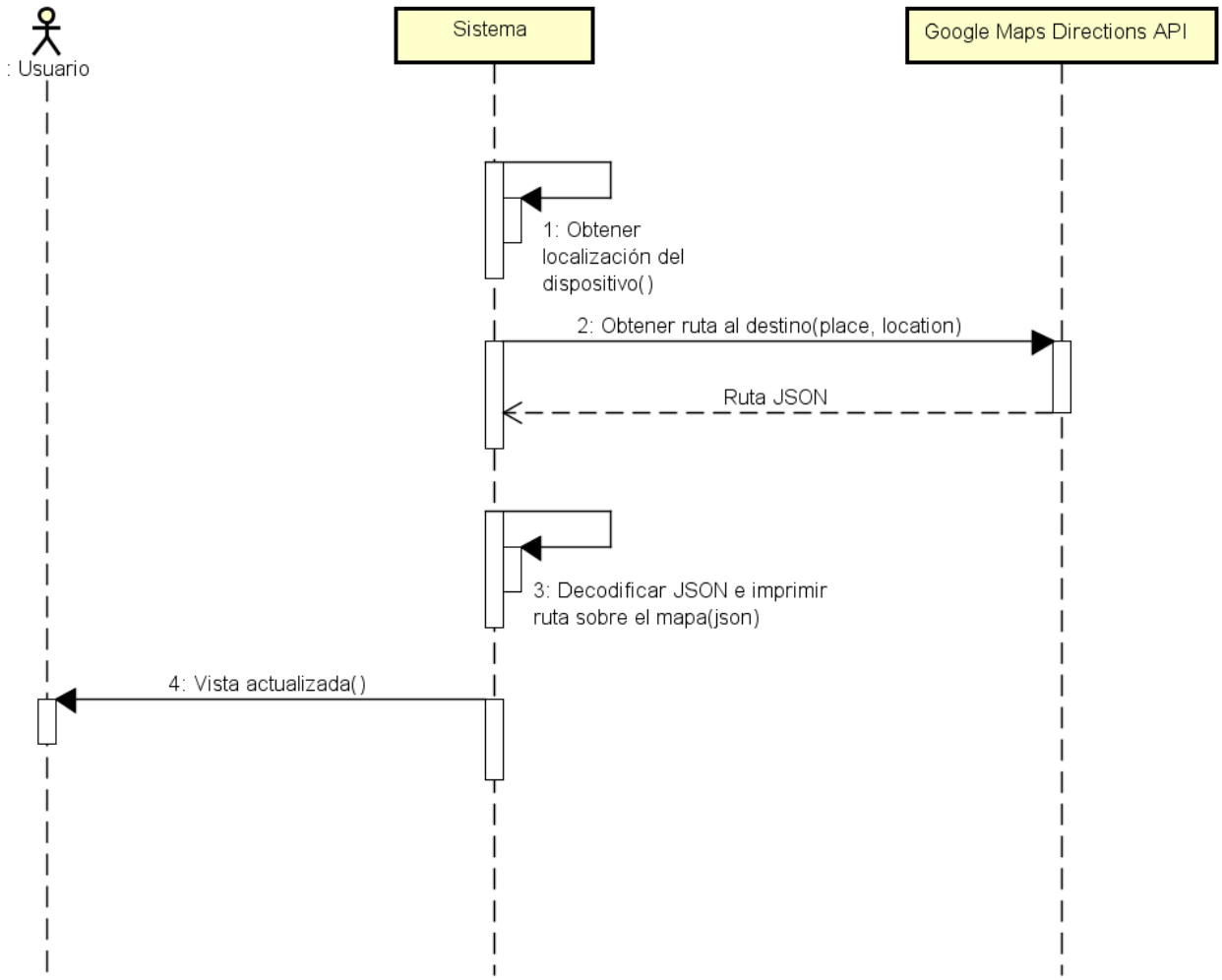


Figura 5.6: Diagrama secuencia de imprimir ruta en el mapa.

Identificador	UC05
Nombre	Pedir activación de ubicación
Dependencias	UC04 - Ver ruta en el mapa hacia un lugar
Descripción	Si el usuario rechaza la petición de activar la ubicación, se le mostrará un mensaje que puede pulsar para volver a ver el mensaje con el que puede activar este servicio.
Precondición	Tener la ubicación apagada y haber rechazado la petición de activación de la misma.
Secuencia normal	1. El usuario rechaza la solicitud de activación de ubicación. 2. El sistema muestra el mensaje al usuario. 3. El usuario pulsa el mensaje. 4. El sistema pide al usuario que active la ubicación.
Frecuencia	Tantas veces como el usuario rechace la petición.

Cuadro 5.27: UC05 - Pedir activación de ubicación

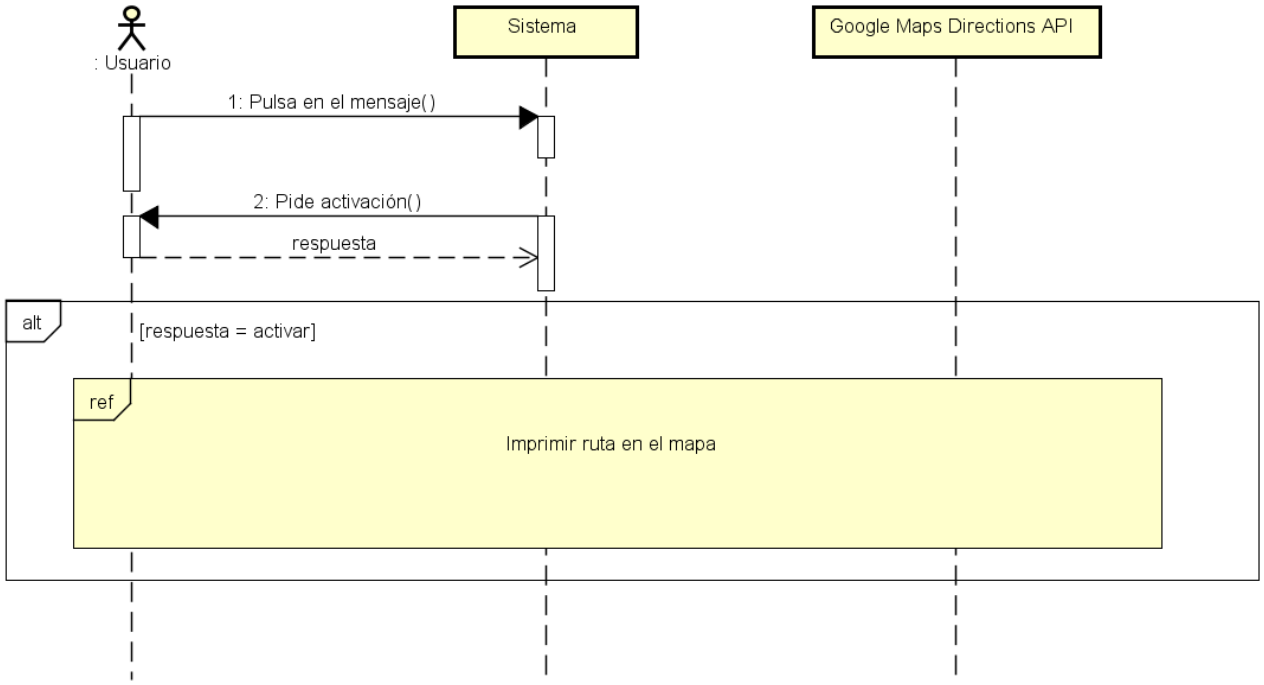


Figura 5.7: Diagrama secuencia de UC05.

Identificador	UC06
Nombre	Abrir la cámara
Dependencias	UC07 - Reconocer imagen
Descripción	El usuario puede abrir la cámara para reconocer un lugar a través de la misma.
Precondición	Estar en la pantalla de detalles o de navegación.
Secuencia normal	1. El usuario pulsa el botón de la cámara en la aplicación. 2. El sistema abre la cámara del dispositivo.
Frecuencia	1-2 veces por sesión.

Cuadro 5.28: UC06 - Abrir la cámara

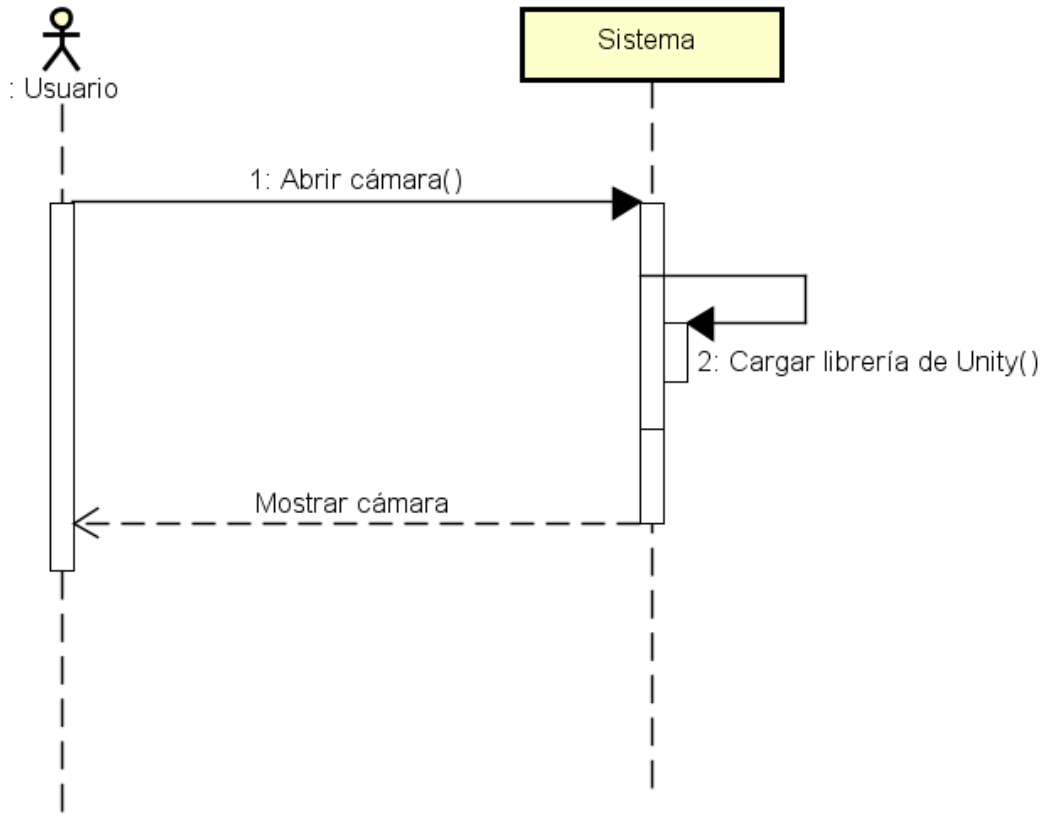


Figura 5.8: Diagrama secuencia de UC06.

Identificador	UC07
Nombre	Reconocer imagen
Dependencias	UC06 - Abrir la cámara
Descripción	El usuario puede reconocer imágenes usando la cámara del dispositivo y ver un modelo 3D sobre el objeto reconocido.
Precondición	La aplicación tiene permisos para abrir la cámara y ésta está funcionando correctamente.
Secuencia normal	1. El sistema reconoce una imagen del mundo real. 2. El sistema muestra un modelo 3D sobre la imagen reconocida.
Frecuencia	Tantas como imágenes sean reconocidas por el sistema.

Cuadro 5.29: UC07 - Reconocer imagen

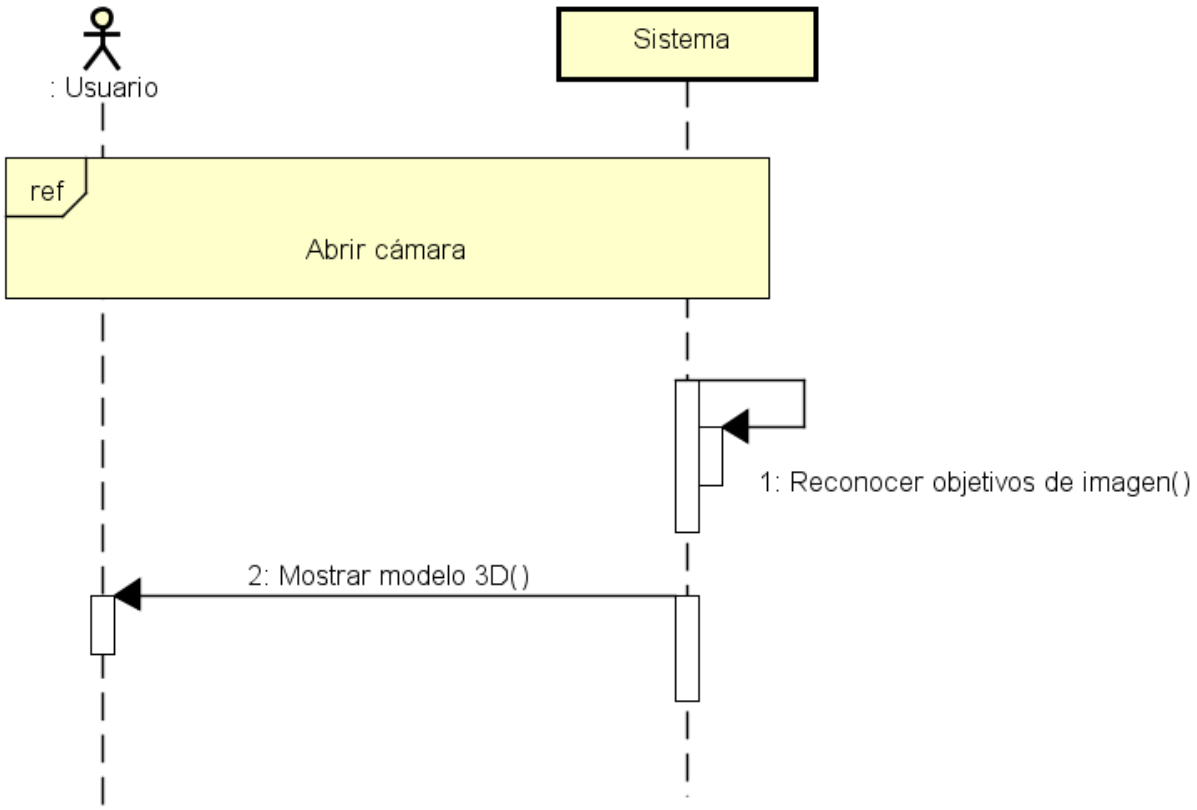


Figura 5.9: Diagrama secuencia de UC07.

Capítulo 6

Diseño

6.1. Diseño de las vistas

Vista de lugares

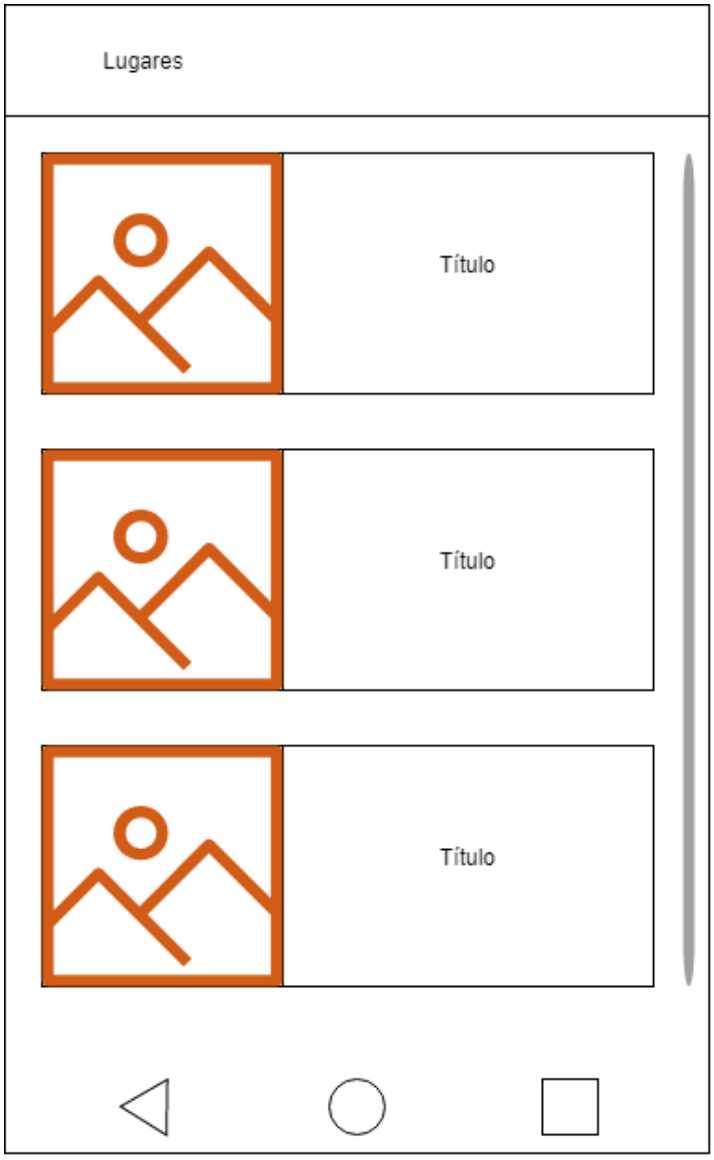


Figura 6.1: Diseño preliminar de la vista de la lista de lugares.

Vista de detalles

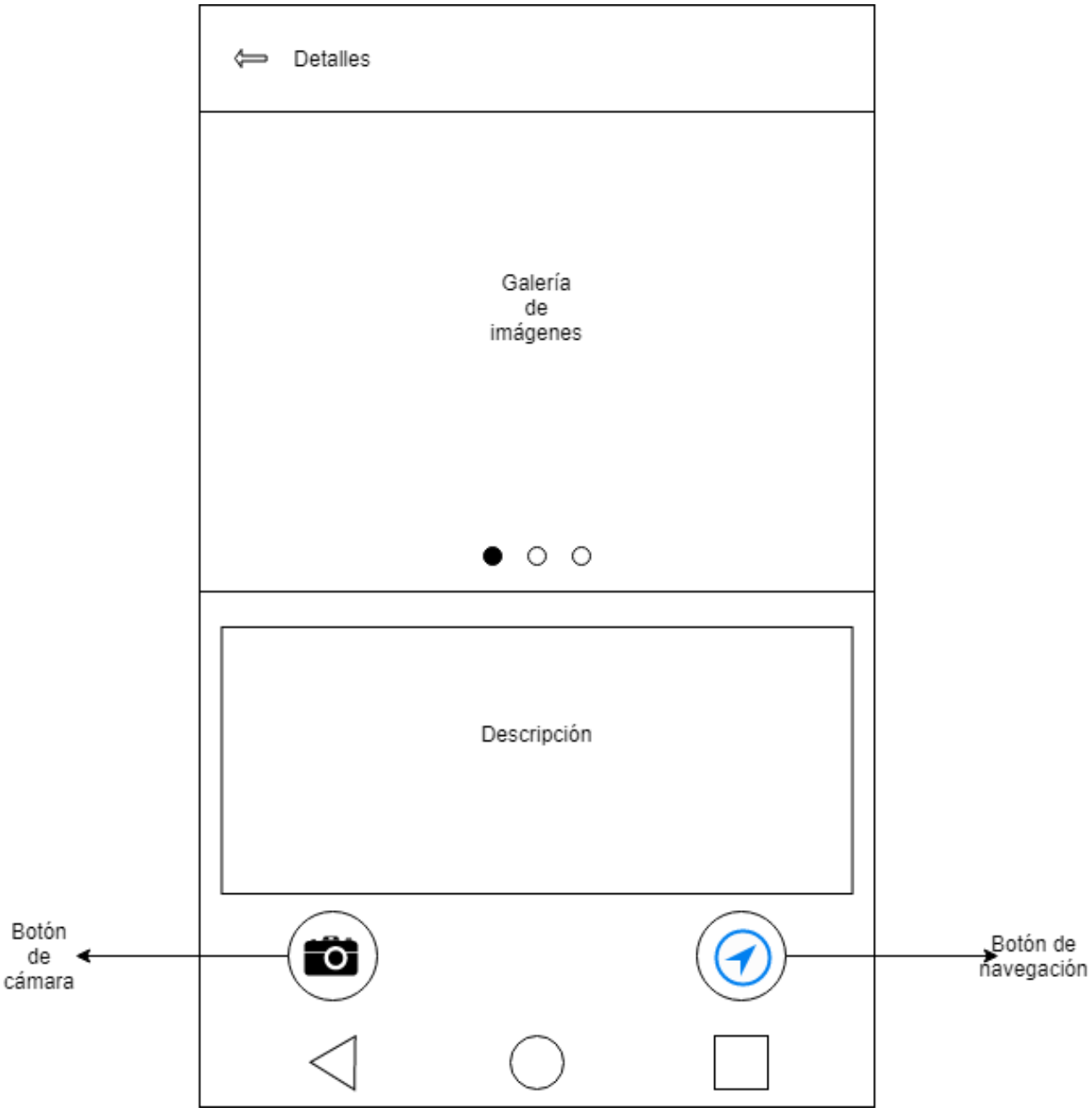


Figura 6.2: Diseño preliminar de la vista de los detalles de un lugar.

Vista de mapa

Figura 6.3: Diseño preliminar de la vista de la navegación hacia un lugar.

6.2. Diseño de arquitectura

La arquitectura que se usará para crear la aplicación es MVVM (Model-View-ViewModel). No es necesaria para un proyecto que carece de base de datos, pero es el estándar para implementar aplicaciones Android modernas según Google [13] y resulta más conocida, además proporciona una serie de ventajas que se explicarán en esta sección.

6.2.1. Model-View-ViewModel

Como su propio nombre indica, este patrón tiene 3 componentes principales:

- **View:** Se encarga de comunicar al ViewModel las acciones que lleva a cabo el usuario en la vista.
- **ViewModel:** Actúa como enlace entre la vista (View) y el modelo (Model). Transforma los datos de la vista para el modelo como si se tratase de un controlador, y pasa los datos del modelo a la vista cuando ésta se los pide.
- **Model:** Almacena los datos que la aplicación necesita. No puede comunicarse con la vista de forma directa, únicamente a través del ViewModel. Los datos que queremos que comunique al ViewModel se suelen categorizar como Observable.

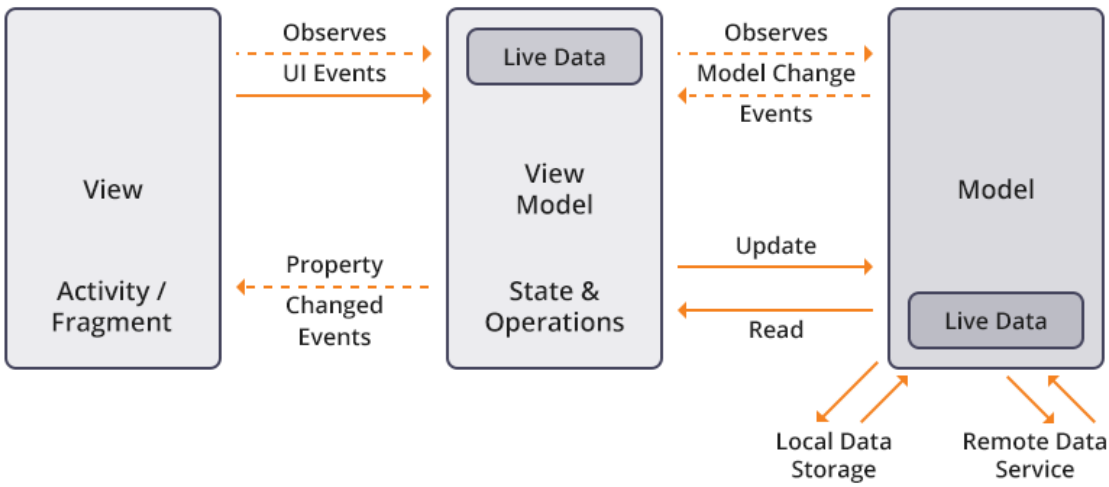


Figura 6.4: Flujo de datos entre los componentes de MVVM [14].

La gran ventaja de la clase ViewModel es que permite que los datos y estados sobrevivan a los cambios de configuración (como por ejemplo rotaciones de pantalla, que llaman a onCreate()). En la siguiente imagen se ilustra el ciclo de vida de las clases de este tipo respecto al de las actividades o fragmentos.

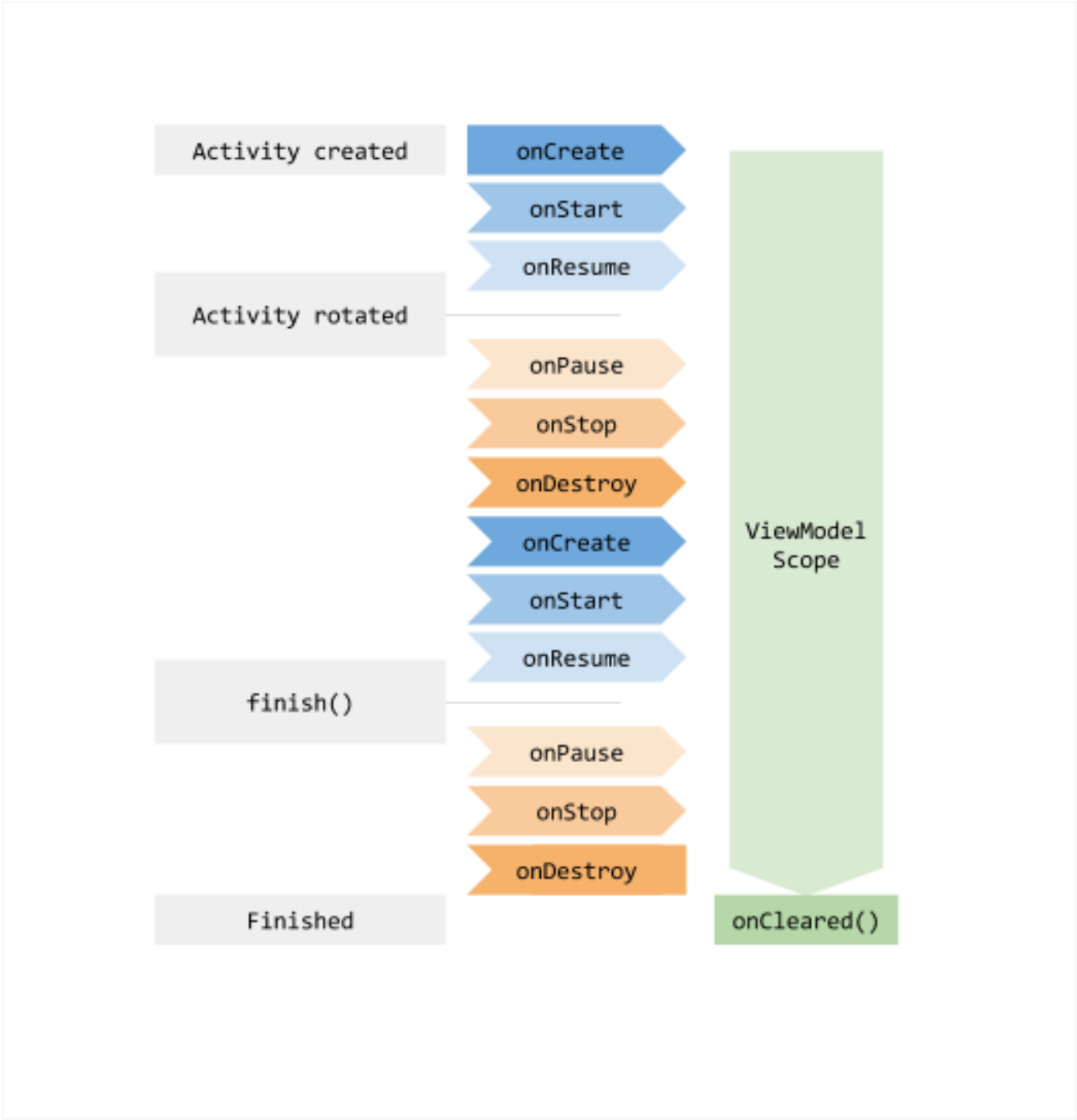


Figura 6.5: Ciclo de vida de ViewModel [15].

Entre otras grandes ventajas encontramos que esta arquitectura permite separar por completo la lógica de la vista de la lógica de negocio y de los datos. El objetivo es que la vista sólo tenga código relacionado con la vista y su comportamiento, mientras que el ViewModel se encarga del tratamiento de los datos y estados de la vista, y el modelo se ocupa del almacenamiento de los datos y creación de los mismos. Esto es muy útil tanto para el testeo de la aplicación como para la legibilidad del código.

Este patrón arquitectónico también nos permite programar de forma reactiva, es decir, que el sistema reaccione a los cambios en los datos en tiempo real (como un callback). Para esto se emplea el patrón observador y los MutableLiveObjects. Estos objetos están

especialmente diseñados para ser utilizados con patrón observador, y contienen un objeto de cualquier tipo. En la vista se observa un MutableLiveObject que guarda el modelo, y cuando este objeto se modifica (una llamada a una API, un cambio por parte de otro usuario, etc) la vista puede actualizarse en tiempo real sin necesidad de una recarga o de volver a crearla.

ViewModel también puede emplearse para compartir información entre fragmentos con un ciclo de vida distinto entre ellos. Se puede emplear un ViewModel compartido para dos fragmentos al cual ambos pueden acceder, y si un fragmento desaparece, el ViewModel, cuyo ciclo de vida es independiente, continúa funcionando normalmente y el otro fragmento puede seguir usándolo sin problemas.

6.3. Diseño detallado

A continuación se mostrará un diagrama con la organización de las clases principales y paquetes que se va a utilizar en la implementación de la aplicación.

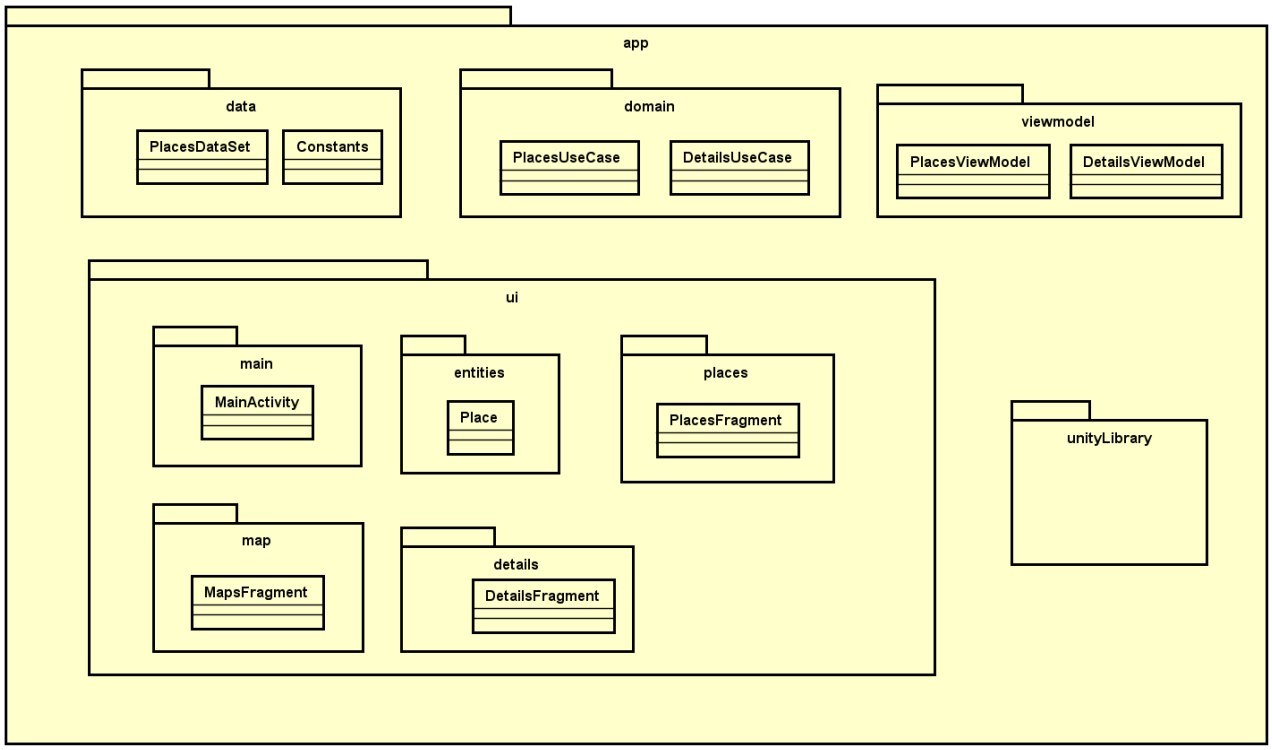


Figura 6.6: Diagrama de clases preliminar del sistema.

En el paquete data se almacenará la clase PlacesDataSet, que se encargará de inicializar la lista de lugares que usará la aplicación. La clase Constants contendrá algunas constantes. En el paquete domain se almacenarán las clases UseCase que serán las que llamen a las clases del paquete data. En viewmodel se guardarán las clases ViewModel que serán llamadas por la vista para que empleen los UseCase y almacenen los MutableLiveData. El paquete ui almacenará todas las clases de la vista y contendrá paquetes dividirán estas

clases por categorías. En main estará MainActivity (primera clase que se llama al iniciar la aplicación). En entities estará la única entidad que utilizamos, Place (los lugares que se pueden visitar). También encontraremos en ui los fragmentos de la aplicación organizados en paquetes. Por último, la librería Unity de realidad aumentada estará fuera de estos paquetes mencionados anteriormente, ya que es independiente a la aplicación Android nativa.

6.4. Patrones de diseño empleados

6.4.1. Patrón observador

Este patrón se emplea para implementar la arquitectura MVVM. Se observarán los cambios que se producen sobre la lista de lugares y se actualizará la vista notificando al fragment. Seguidamente se expone un diagrama de secuencia de cómo se implementará este patrón para este caso en concreto.

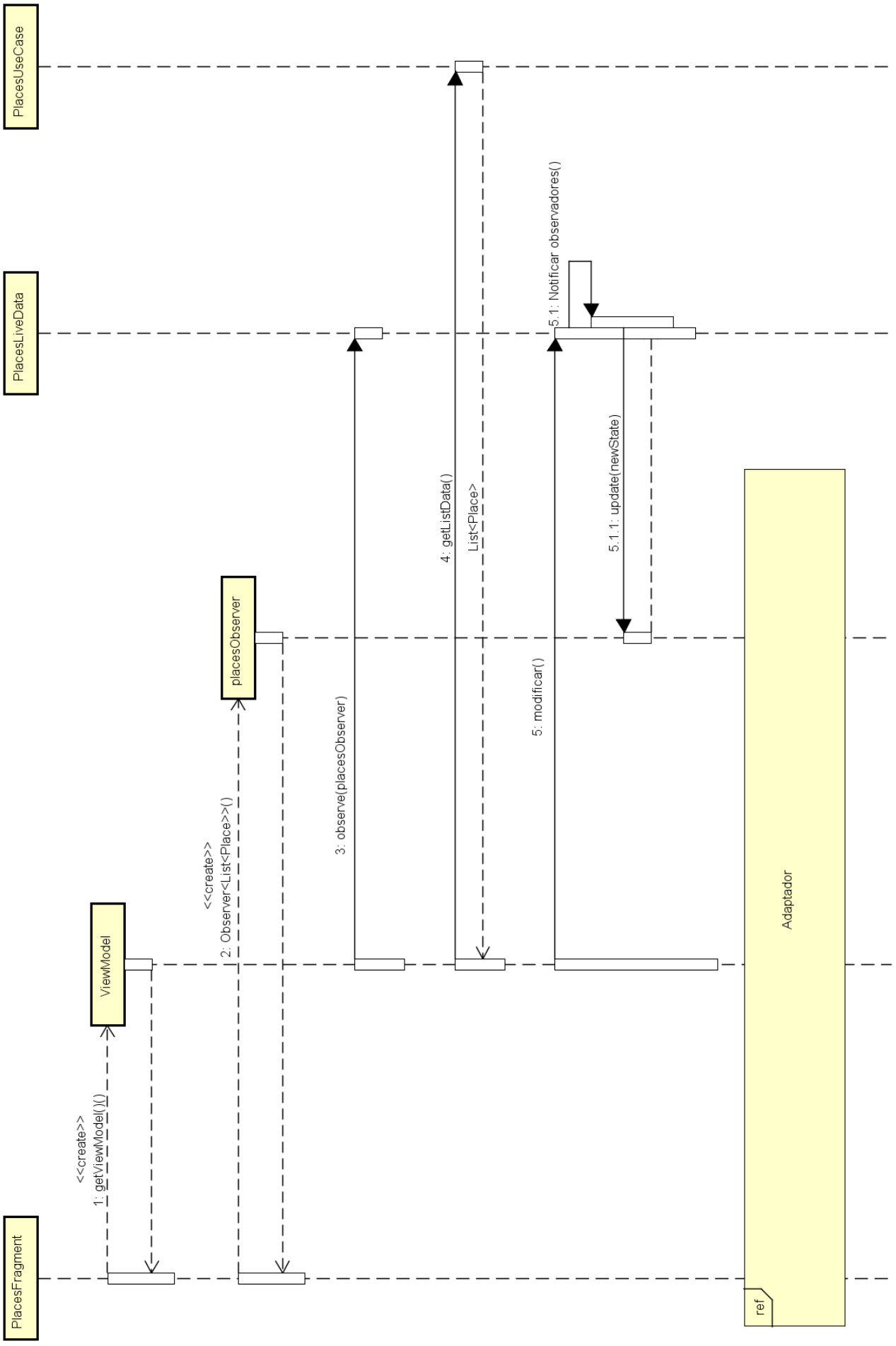


Figura 6.7: Diagrama de secuencia del patrón observador.

6.5. Patrón adaptador

Este patrón se usa para implementar la vista de la lista de lugares que se explicará con más detalle en el capítulo de implementación. Para esta vista necesitamos un adaptador que convierta nuestros objetos de tipo Place a contenedores que aparecerán en la vista de la clase ViewHolder.

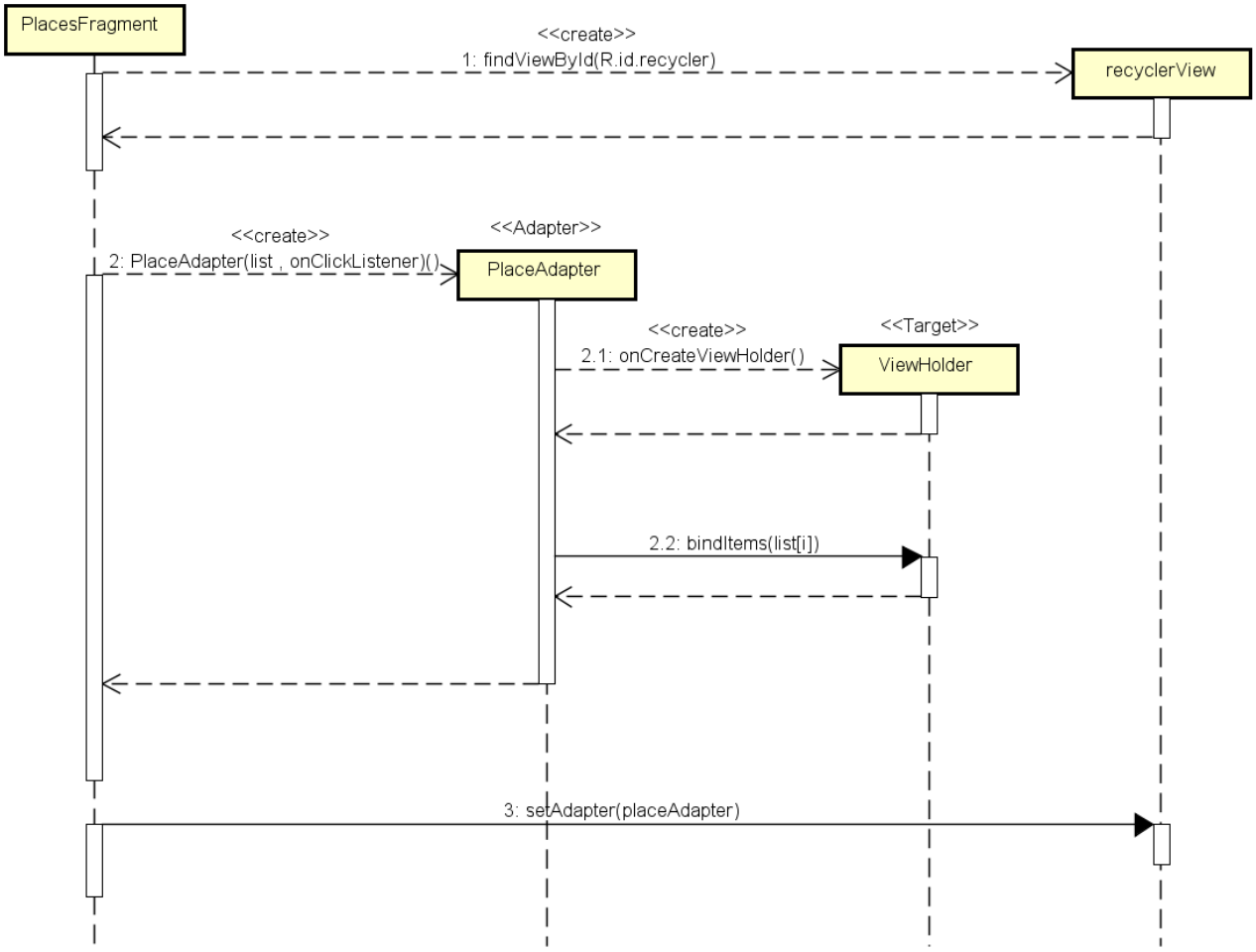


Figura 6.8: Diagrama de secuencia del patrón adaptador.

6.6. Gestión de los datos

Esta aplicación precisa obtener la lista de lugares que se pueden visitar antes de mostrar la primera vista que aparece cuando la aplicación se inicia. Como los lugares que se incluirán en la aplicación son limitados, ya que no existen fotografías antiguas de muchos lugares de la ciudad, se crea una lista con objetos Place que contienen toda la información que usa el sistema. Esta lista es creada por la clase PlacesDataSet. La lista de emplazamientos es la siguiente:

- Iglesia de Santa María la Antigua.
- Iglesia de Nuestra Señora de las Angustias.

- Teatro de la comedia.
- Monumento al conde Pedro Ansúrez.
- Plaza de Fuente Dorada.
- Plaza Mayor de Valladolid.
- Sala de cine Roxy.
- Teatro Lope de Vega.
- Iglesia de San Lorenzo.
- Casa consistorial de Valladolid.
- Catedral de Valladolid.
- Calle de Cánovas del Castillo.
- Universidad de Valladolid.
- Teatro Calderón.
- Fábrica de Harinas La Perla.
- Biblioteca de Castilla y León.
- Iglesia de San Agustín.
- Mercado del Val.
- Museo Nacional de Escultura.
- Iglesia de San Pablo.
- Iglesia de Santiago Apóstol.

Capítulo 7

Implementación

Este capítulo se divide en dos secciones: la implementación de la aplicación Android nativa y la implementación de la librería Unity de realidad aumentada.

7.1. Implementación de la aplicación Android

7.1.1. MainActivity

Para esta parte del proyecto se ha utilizado una actividad principal (MainActivity) que controla el resto de fragmentos mediante el componente Navigation. El ciclo de vida de esta actividad se mantiene independientemente de lo que le pase a los fragmentos, y sólo contiene código para configurar la toolbar (barra superior de la aplicación) y lanzar el componente de navegación.

```
13  class MainActivity : AppCompatActivity() {
14      private val toolbar : MaterialToolbar by lazy { findViewById<MaterialToolbar>(R.id.toolbar) }
15      private lateinit var navController: NavController
16      private lateinit var appBarConfiguration: AppBarConfiguration
17
18      override fun onCreate(savedInstanceState: Bundle?) {
19          setTheme(R.style.AppTheme_NoActionBar)
20          super.onCreate(savedInstanceState)
21          setContentView(R.layout.activity_main)
22          setSupportActionBar(toolbar)
23          navController = findNavController(R.id.nav_host_fragment)
24          appBarConfiguration = AppBarConfiguration(navController.graph)
25          this.setupActionBarWithNavController(navController, appBarConfiguration)
26      }
27
28      override fun onSupportNavigateUp(): Boolean {
29          return navController.navigateUp(appBarConfiguration) || return super.onSupportNavigateUp()
30      }
31
32  }
33
```

Figura 7.1: Código de MainActivity.

La función onSupportNavigateUp se sobre-escribe para hacer que la barra superior mues-

tre una flecha para navegar hacia atrás cuando se avanza en el grafo de navegación.

Figura 7.2: Toolbar configurada para navegación.

El componente de navegación se encarga de iniciar el ciclo de vida del fragmento marcado como inicial en el grafo. En este caso el fragmento inicial es PlacesFragment (vista de lugares).

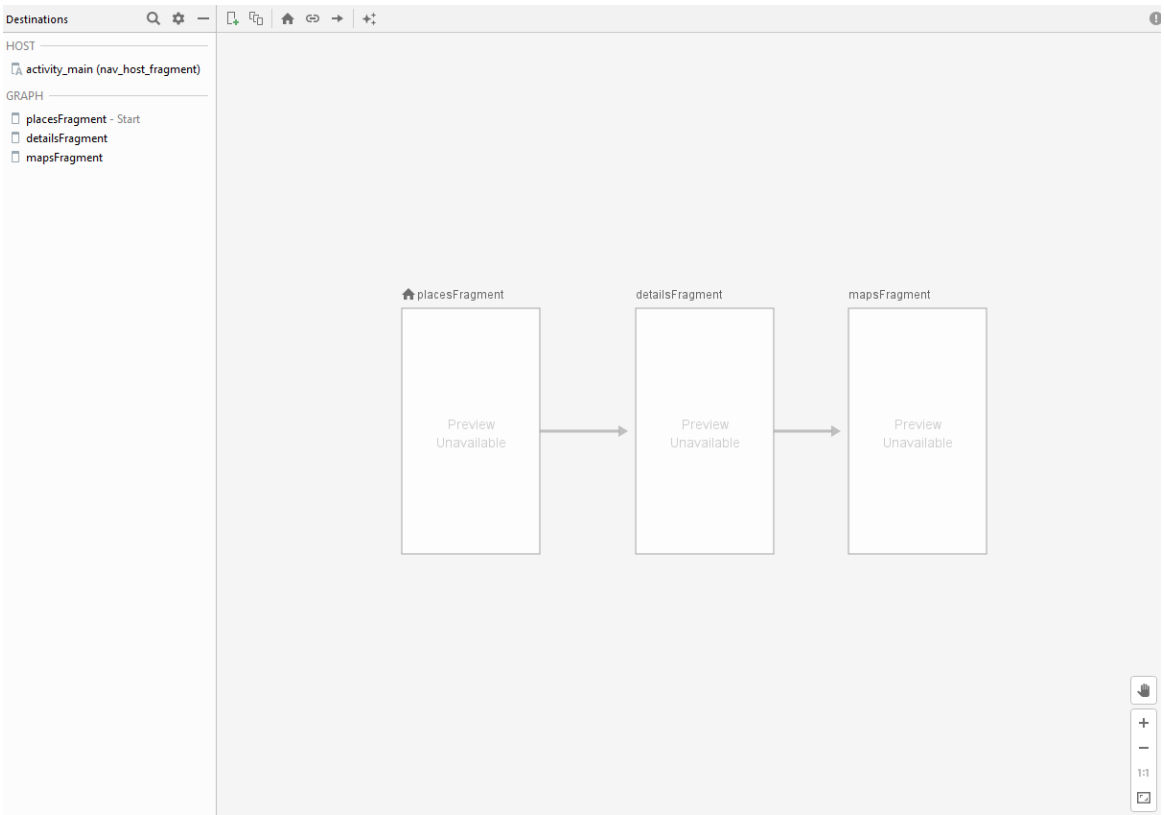


Figura 7.3: Grafo de navegación usado por el componente Navigation.

7.1.2. Fragmento de lugares

Para esta vista se quiere implementar una lista de elementos, cada uno con un título y una fotografía. La mejor forma de implementar una lista es mediante RecyclerView, ya que es una versión más flexible y avanzada de ListView, que es la otra forma de implementar este tipo de vistas. RecyclerView tiene un rendimiento muy superior y soluciona una serie de problemas que presentaba ListView (sólo permite orientación vertical, bajo rendimiento ya que requería muchas operaciones de findById, etc).

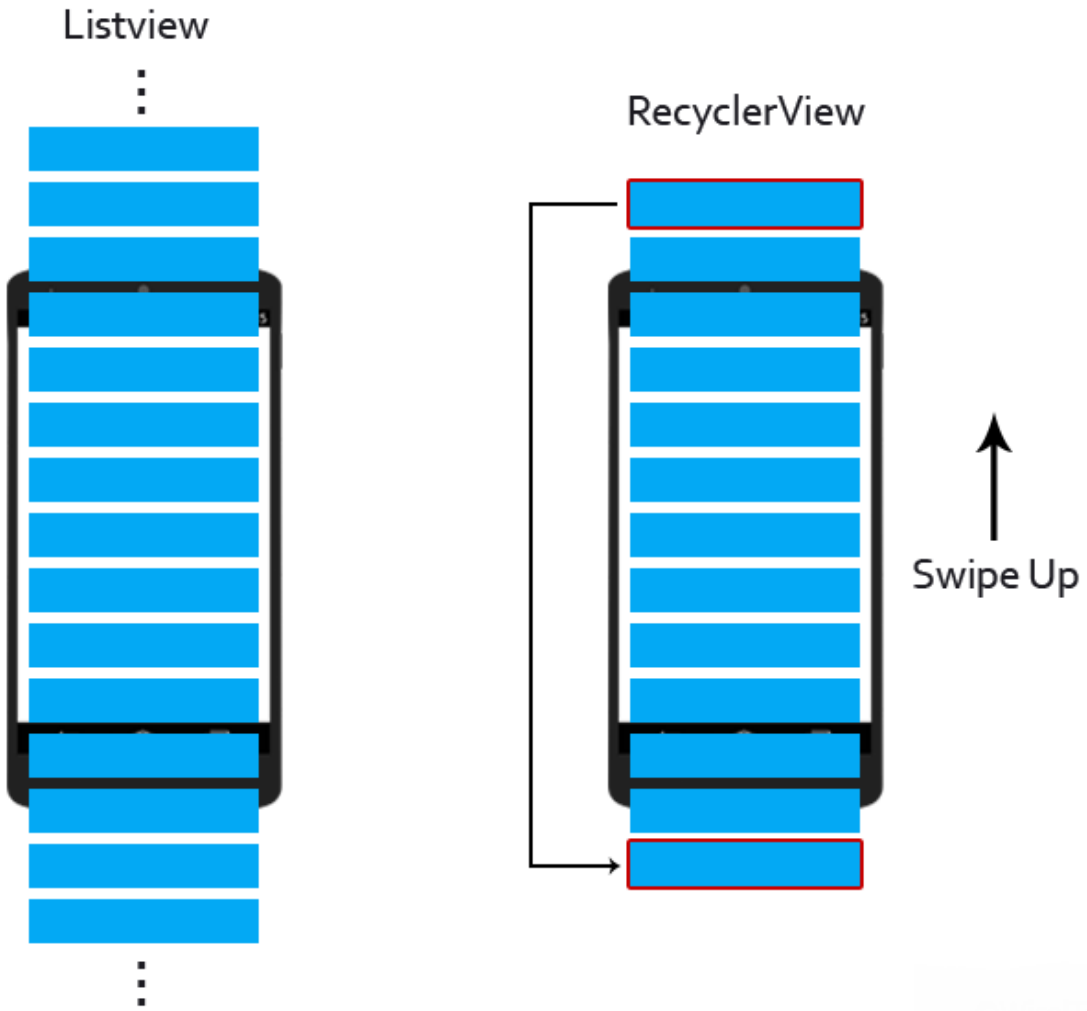


Figura 7.4: Diferencia entre el funcionamiento de ListView y RecyclerView [16].

RecyclerView utiliza varios componentes para poder funcionar, ya que por sí mismo sólo crea los contenedores necesarios para almacenar cada objeto de la lista y los gestiona. Cuando el usuario se desplaza a través de la lista, el objeto RecyclerView toma las vistas que no están en pantalla y vuelve a vincularlas con los datos que se desplazan en la misma. Para explicar estos distintos componentes se debe explicar primero cómo se implementa una vista común en Android.

Una vista consta de dos partes: un diseño o layout que define los elementos que con-

tendrá la vista a implementar, y un fragmento o actividad asociado a ese layout que se encarga de la lógica de dicha vista. Los componentes que se necesitan para implementar este tipo de vista son:

- **ViewHolder:** Es una clase que extiende RecyclerView.ViewHolder y que se crea en el interior de la clase Adapter (que se explica más adelante). Las vistas que se incluyen en la lista son instancias de la clase ViewHolder.
- **Adapter:** Es una clase que extiende RecyclerView.Adapter y se encarga de administrar los objetos contenedores de vistas. Esta clase tiene los datos necesarios que se quieren mostrar esta lista, y los enlaza a cada objeto. En este caso se tiene una lista de lugares y la clase PlaceAdapter se encarga de asignar el nombre y la imagen de cada elemento de la lista a cada elemento de la vista que forma la lista de RecyclerView. Para vincular esta imagen se ha utilizado la librería Glide [17], que facilita el uso de imágenes en Android para cargarlas sobre ImageView en el layout.

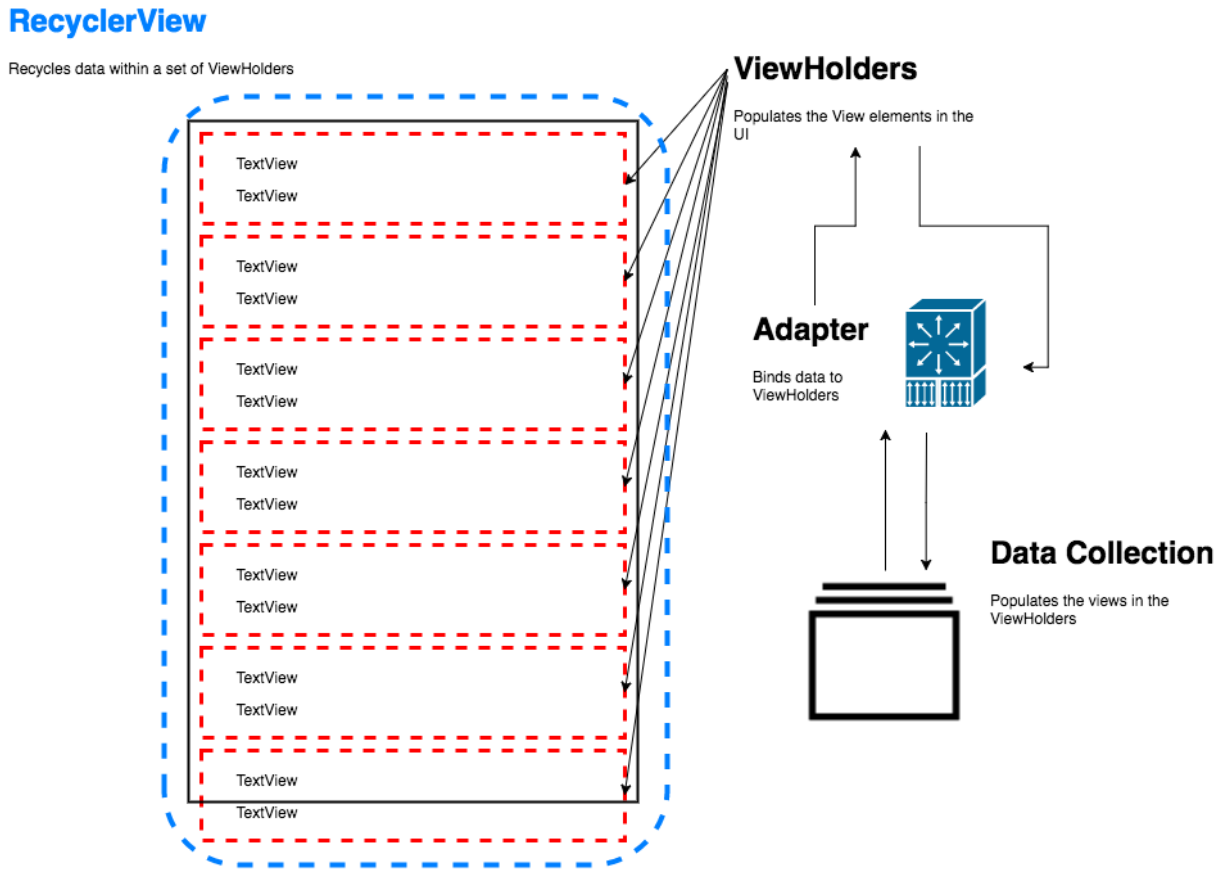


Figura 7.5: Diagrama del funcionamiento de la clase Adapter [18].

- **Layout del fragment:** El layout de la vista de lugares (PlacesFragment) contendrá únicamente el widget de RecyclerView.
- **Layout del objeto contenedor:** Este es el diseño de cada objeto que se encontrará en la lista. Se ha implementado utilizando un CardView, y en su interior un layout que tiene

un ImageView (contenedor de imagen) y un TextView (contenedor de texto). La clase adaptadora se encarga de colocar el título y la imagen de los objetos Place de la lista en estos contenedores.

- **El propio fragment:** PlacesFragment se encarga de, una vez su ciclo de vida comienza, obtener la lista de lugares, crear una instancia de PlacesAdapter proporcionándole esta lista y otra de RecyclerView proporcionándole el adapter.

De esta forma se implementa esta vista que mostrará todos los sitios que se pueden visitar. Pero para implementar la arquitectura MVVM, no se puede crear simplemente una lista de lugares en el propio fragment, ya que el objetivo es que la vista conozca la menor cantidad de datos posibles. En este caso la lista es siempre la misma, pero si fuera necesario obtenerla de una base de datos o de una API externa se podría implementar de forma muy similar. La obtención de los datos se produce en una sola clase separada del fragment al que queremos que lleguen, y una clase ViewModel (PlacesViewModel) se encarga de hacer que estos datos lleguen al fragment que observa esta lista. La secuencia de eventos es la siguiente:

1. PlacesFragment crea un observador que la clase ViewModel utiliza para observar el objeto lista de tipo MutableLiveData.
2. PlacesFragment pide al ViewModel que le proporcione la lista de lugares.
3. ViewModel pide a la clase PlacesUseCase que le proporcione esta lista.
4. PlacesUseCase a su vez le pide a la clase PlacesDataSet que genere esta lista y se la devuelva.
5. Cuando ViewModel tiene la lista creada, cambia el objeto MutableLiveData que contiene una lista de lugares.
6. Al producirse este cambio, el observador notifica a PlacesFragment que se encarga de actualizar la vista con la nueva lista.

De esta forma queda implementada toda la funcionalidad de la primera vista de la aplicación.

7.1.3. Fragmento de detalles

Para esta vista se quieren implementar dos elementos que formen los detalles de cada lugar que podemos visitar. Estos elementos serán una galería que muestre varias fotografías y que el usuario puede deslizar para cambiar de imagen, y un cuadro de texto que proporcione al usuario información acerca del lugar. También contendrá un botón que al ser pulsado muestre la ruta hacia ese lugar y otro botón que abrirá la cámara para interactuar con la realidad aumentada.

Los dos primeros elementos mencionados cambiarán en función del elemento sobre el que el usuario ha pulsado en la vista de lugares. Para pasar estos datos de un fragmento a otro, se utilizan los argumentos del componente Navigation. Al crear la acción que se utilizará para cambiar de vista, le damos los argumentos que queremos comunicar. Estos se fijan en el grafo de navegación.

Una vez comienza el ciclo de vida del fragmento de detalles, se obtiene la instancia del ViewModel DetailsViewModel. Este objeto se va a encargar de proporcionar a DetailsFragment una lista de SlideModel. SlideModel es un objeto que pertenece a la librería que se usará para implementar la galería de imágenes, llamada ImageSlideshow [19]. Para crear cada uno de estos objetos, se utiliza una imagen y un título como parámetro. Usando el identificador del lugar como argumento, DetailsViewModel pide la lista a DetailsUseCase, que a su vez se la pide a DetailsDataSet que cree la lista en función del identificador que obtiene. Una vez se ha creado, el objeto observado notifica al fragmento para que actualice la vista. Funciona de la misma forma que el fragmento anterior pero con el argumento para crear la lista de forma dinámica. Para actualizar la vista obtenemos del layout el elemento ImageSlider (también perteneciente a la librería ImageSlideShow) y le proporcionamos la lista que se observa. El resultado es el siguiente:

Figura 7.6: Ejemplo de vista de la galería de imágenes.

Como se aprecia en la imagen, la librería implementa un contador de la imagen que se muestra en ese momento, y hay otras dos imágenes que el sistema mostrará si el usuario desliza la imagen. Debajo de esta galería se mostrará un cuadro de texto con la descripción del lugar. Esta descripción se pasa por parámetro cuando se cambia de fragmento y se coloca en el TextView.

Por último en la parte inferior de esta vista se muestran dos FloatingActionButton. Estos elementos son botones circulares que actúan de la misma forma que un botón pero ocupan un espacio reducido, y no se dispone de mucho espacio en esta vista. Uno de ellos abrirá la vista de navegación en el mapa, y el otro abrirá la cámara para la realidad aumentada. En la siguiente figura podemos ver un ejemplo del resultado.

Figura 7.7: Ejemplo de vista de información de un lugar y botones de navegación.

En esta vista, si cambiamos la orientación del dispositivo a horizontal, se producen varios problemas con la galería de imágenes y el resto de elementos. Mientras que en las otras vistas el resultado es adecuado, en esta vista se produce el siguiente efecto:

Figura 7.8: Ejemplo de vista de detalles con orientación horizontal.

Las imágenes de la galería apenas se ven, y para solucionar este problema se emplea un layout de landscape (vista horizontal) que se activa cuando el dispositivo cambia a esta orientación. En este layout se colocan los mismos elementos pero la galería se coloca en la mitad izquierda de la vista, mientras que el texto y los botones se colocan en la mitad derecha. el resultado es el siguiente:

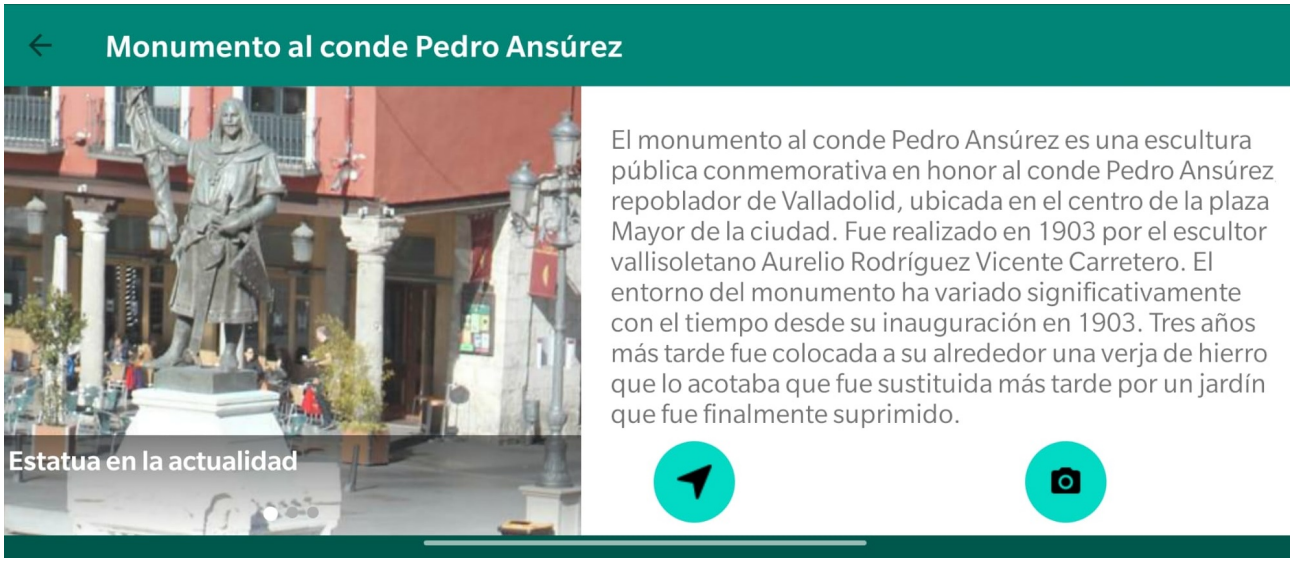


Figura 7.9: Ejemplo de vista de detalles con orientación horizontal con layout de landscape.

Así el usuario puede ver las imágenes completas y el espacio de la vista queda bien distribuido para los elementos de la misma.

7.1.4. Fragmento de mapa

Para esta vista se quiere implementar un mapa que muestre la ubicación actual del usuario y la ruta más cercana dibujada sobre este mapa. Además se implementará un botón que pueda abrir la cámara igual que en el fragmento anterior.

Por defecto Android Studio permite implementar un fragmento de mapa con un marcador de ejemplo. Se ha usado esta implementación por defecto añadiendo la ubicación actual al mapa. Para dibujar la ruta, se obtiene la localización del dispositivo (pidiendo antes permisos al usuario y en caso de no estar activada, pedir que se active) usando LocationManager y también conocemos la ubicación por latitud y longitud del lugar a visitar. Con estos dos datos, se forma una url y después se hace una petición a la API de Google Maps Directions. Esta API nos devuelve un objeto JSON con distintos campos de los que sólo se utilizan algunos para esta funcionalidad. Para transformar este objeto JSON a información que pueda tratarse, se emplea una clase DTO. Esta clase tiene los mismos campos que el JSON y una vez se transforma, obtenemos una lista de objetos LatLng (ubicaciones con latitud y longitud). Esta lista se transforma a una línea de tipo Polyline mediante una función. Una vez tenemos este objeto listo, se añade al mapa. Podemos configurar el color, grosor, etc de esta línea y se muestra una vez que el mapa hace zoom en nuestra ubicación. El resultado es el siguiente:

Figura 7.10: Ejemplo de vista de ruta dibujada sobre el mapa.

Para poder acceder a la ubicación del dispositivo y mostrar la ruta, necesitamos tanto permisos de ubicación, servicio de localización activado y conexión a internet. Todas estas comprobaciones se llevan a cabo al inicio del ciclo de vida del fragmento. Primero se piden permisos de ubicación mediante el siguiente mensaje:

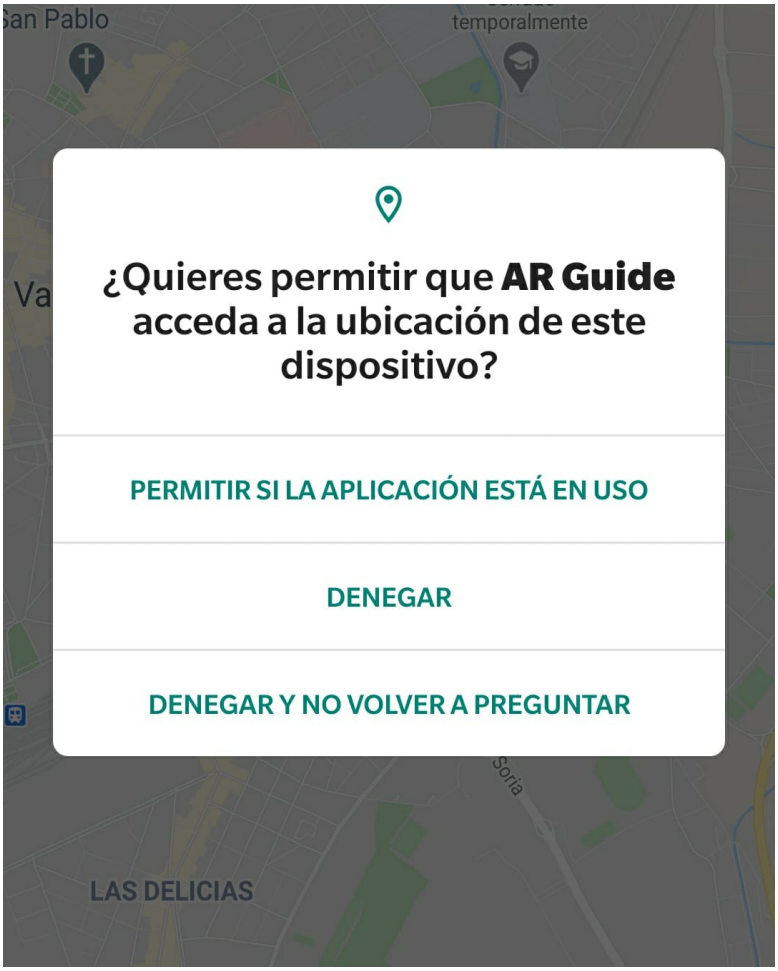


Figura 7.11: El sistema pide los permisos de ubicación al usuario.

Si el usuario rechaza esta petición, se le mostrará el siguiente mensaje con un botón que puede pulsar para dar estos permisos en caso de haberlos rechazado por error:

Figura 7.12: El sistema muestra mensaje de aviso al usuario.

Una vez se han otorgado permisos de ubicación, si la localización del dispositivo no está activada, el sistema mostrará el siguiente mensaje al usuario:

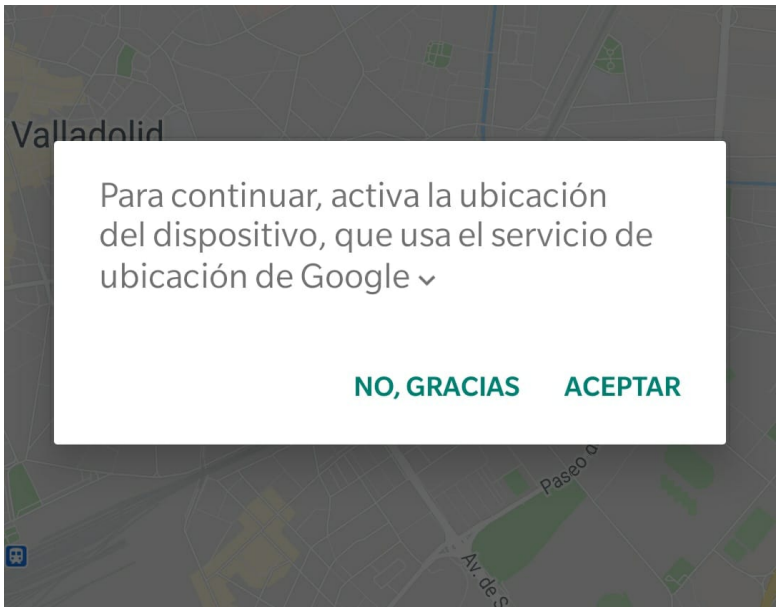


Figura 7.13: El sistema pide al usuario activar la ubicación.

Si el usuario rechaza esta petición, también se le mostrará un aviso (con un mensaje diferente) y con un botón que si es pulsado por el usuario mostrará de nuevo la petición de activación de la localización.

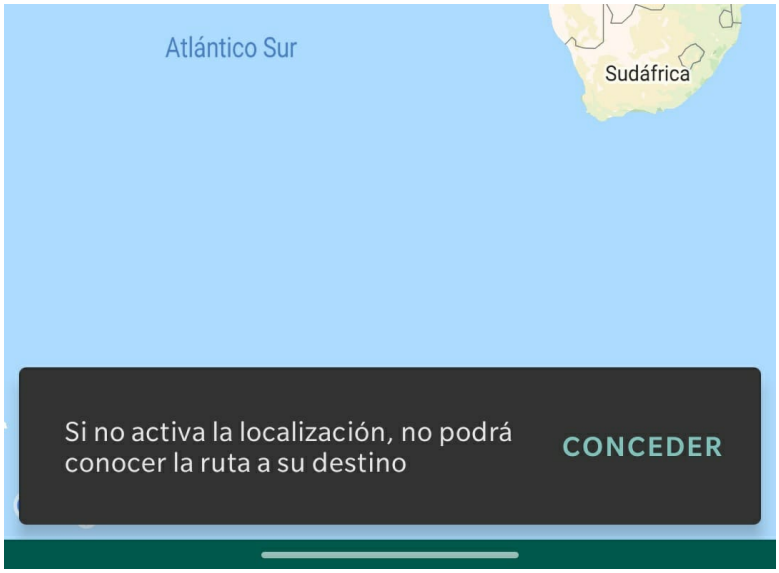


Figura 7.14: El sistema muestra un mensaje de aviso para activar la ubicación al usuario.

Si el dispositivo no tiene conexión a internet, se mostrará el siguiente mensaje al usuario para que la active:

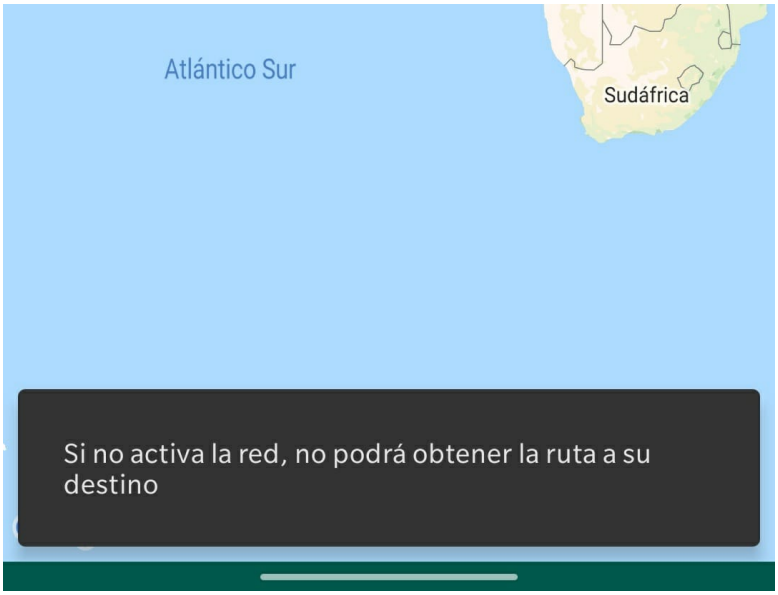


Figura 7.15: El sistema muestra un mensaje de aviso para notificar que el dispositivo no tiene conexión.

7.2. Implementación de la librería de realidad aumentada

Se ha empleado Unity 2019 junto al plugin de Vuforia para la implementación de esta parte del proyecto. Para poder utilizar este plugin se debe marcar la casilla de instalación del plugin cuando se instala la versión de Unity. El objetivo es crear una librería capaz de reconocer una imagen que se le proporciona y, una vez la reconoce, colocar un modelo 3D (con una imagen antigua) sobre este marcador, de forma que el modelo se superponga y cubra el mundo real. Hay que tener en cuenta que las imágenes antiguas que podemos usar son limitadas. Esto quiere decir que el marcador a reconocer debe ser identificado desde la misma perspectiva (o muy similar) desde la que fue tomada la imagen antigua. En las siguientes imágenes se muestra un ejemplo del marcador que debemos utilizar para cierta imagen antigua del Teatro Calderón de Valladolid:

(a) Imagen antigua del teatro.

(b) Imagen actual del teatro.

Figura 7.16: Comparación de modelo 3D con marcador.

Lo primero que se necesita una vez el plugin de Vuforia está instalado en Unity, es una base de datos de marcadores que se crea a través de la web de desarrollador de Vuforia. En este portal se puede obtener una licencia gratuita para esta simulación. Esta licencia contiene una clave que debe ser introducida en la configuración de Vuforia en Unity. Una vez se dispone de una licencia, se puede elaborar una base de datos de marcadores en la web.

Para crear la base de datos, sólo se necesita proporcionarle un nombre y ya se pueden agregar marcadores. Para añadir uno se usa el siguiente formulario:

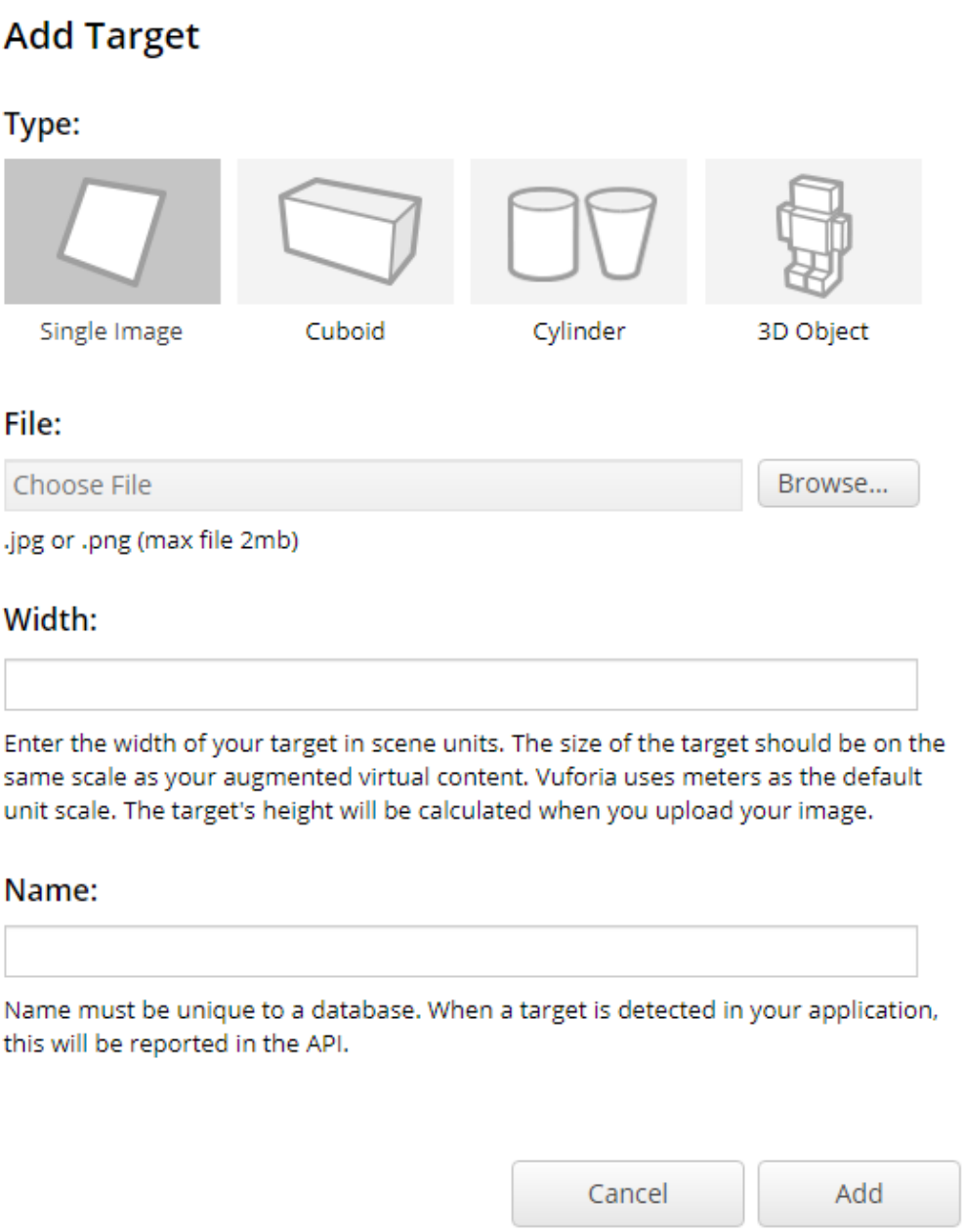
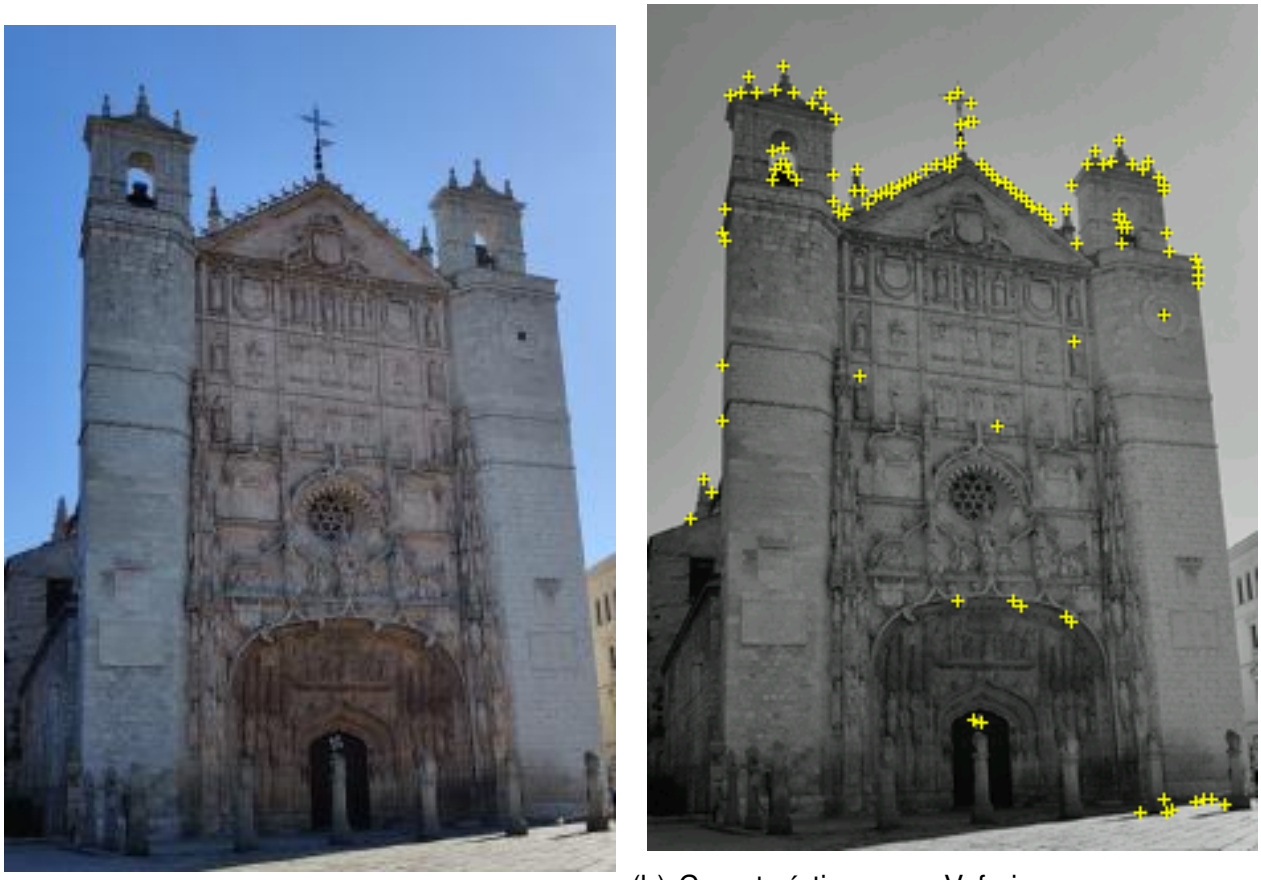


Figura 7.17: Formulario para añadir un marcador a la base de datos de Vuforia.

Para este proyecto se han utilizado marcadores de tipo ”Single image”. Son los más adecuados para reconocer imágenes planas, y hay que tener en cuenta que crear un modelo 3D de los edificios a reconocer es una tarea demasiado costosa. Además, el reconocimiento de imagen funciona de forma adecuada. También hay que proporcionarle la imagen que se quiere establecer como marcador a identificar, un nombre, y el ancho estimado de la imagen en ”scene units” (metros por defecto), para mejorar el rendimiento del reconocimiento.

Una vez se completa este formulario, aparecerá en la lista de marcadores de la base de datos y se le asignará un índice de las características reconocibles de la imagen que se le ha proporcionado. Este índice puede variar de 0 (no reconocible) a 5 (muy fácilmente

reconocible). Por ejemplo, si se trata de una imagen con colores muy similares y pocas formas, tendrá un índice de reconocimiento menor que una imagen con colores más variados o patrones reconocibles. En la siguiente figura se muestra un ejemplo de una imagen con un bajo índice de reconocimiento (índice 3):

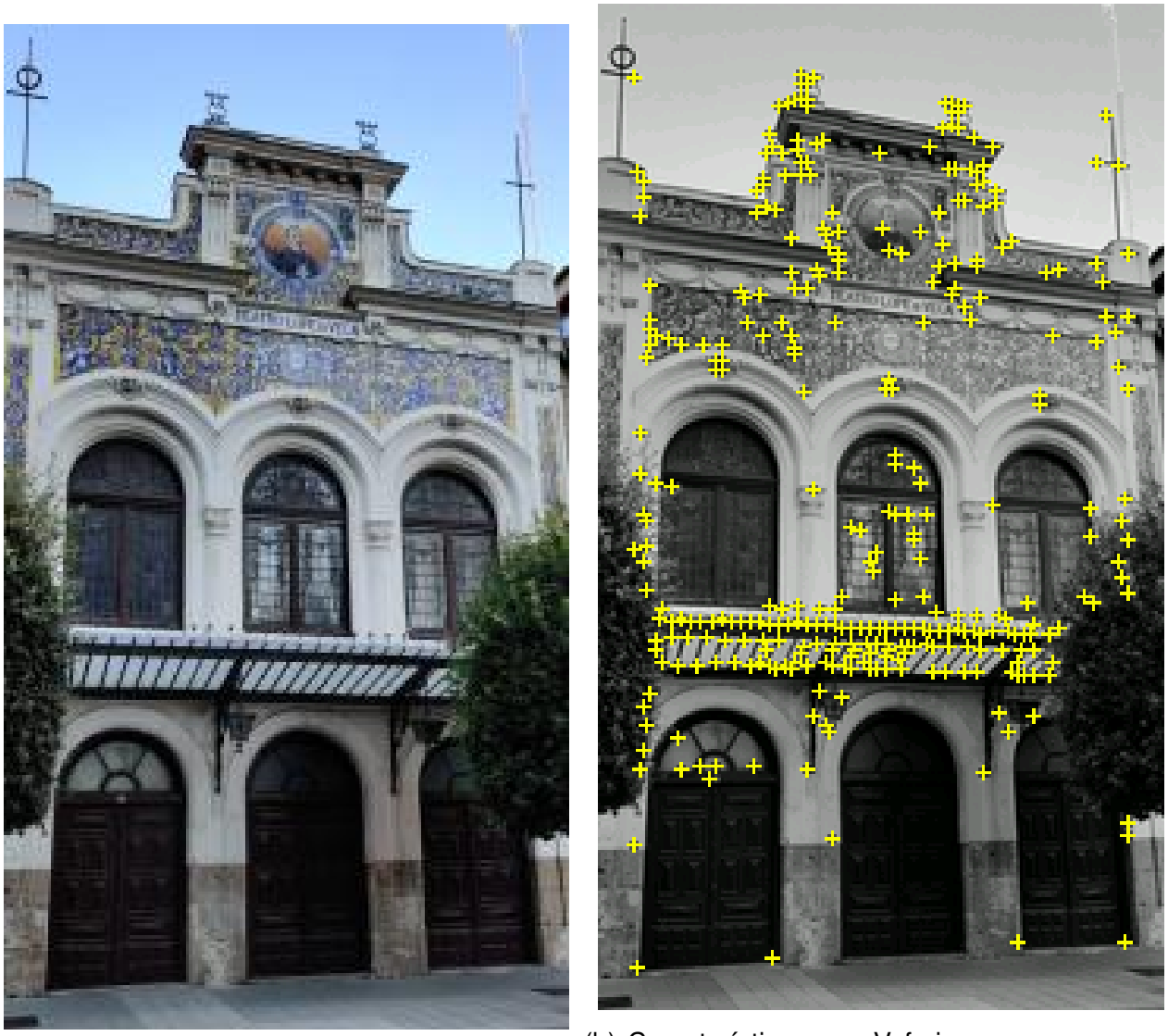


(a) Marcador de la Iglesia de San Pablo.

(b) Características que Vuforia marca para reconocer la imagen.

Figura 7.18: Marcador con bajo índice de reconocimiento junto a sus características.

Como se puede apreciar en la figura, Vuforia asigna características reconocibles casi únicamente a la parte superior de la iglesia, donde se produce un contraste de color y de forma más notable. Además, la fotografía está tomada cuando la fachada de la iglesia apenas está iluminada, lo que hace que el color y las formas de la misma sean menos distinguibles entre sí. A continuación se muestra una figura con un índice de 5:



(a) Marcador del Teatro Lope de Vega.

(b) Características que Vuforia marca para reconocer la imagen.

Figura 7.19: Marcador con alto índice de reconocimiento junto a sus características.

En este segundo marcador, la cantidad de características reconocibles por Vuforia es mucho mayor que en el caso anterior, por lo que es fácil pensar que será más fácil reconocerlo. Tiene más colores, más formas reconocibles y en más lugares diferentes. A pesar de esto, el marcador de la Iglesia de San Pablo es muy fácilmente reconocible ya que las características marcadas por Vuforia son muy sencillas de detectar cuando es enfocado por una cámara: no hay vegetación, la forma siempre coincide y todos los puntos están muy próximos. Sin embargo, en el caso del segundo marcador, la imagen presenta vegetación, cuya forma varía constantemente debido a su movimiento, no hay muchas características en la zona externa de la fachada y la vegetación y la zona central llena de marcadores están en un plano distinto al de la fachada, más cerca del dispositivo (presenta profundidad). Esto hace a este marcador más difícil de reconocer cuando se le enfoca con la cámara.

Este índice no refleja la capacidad de reconocimiento que tendrá el sistema respecto a ese marcador, simplemente indica la cantidad de características que Vuforia usará para in-

tentar reconocerlo. Por lo general, es más sencillo identificar marcadores que no presenten profundidad o sean de un tamaño muy grande. Por ejemplo, resulta más fácil reconocer la fachada de una iglesia muy grande que reconocer una estatua. Además, también resulta más sencillo reconocer objetos cuanto más iluminados estén. Cuando un marcador es difícil de reconocer para el sistema, se puede solventar incluyendo varios marcadores del mismo lugar desde perspectivas ligeramente distintas. De esta forma el reconocimiento no fallará si el objetivo es solamente reconocible desde una perspectiva exacta.

Una vez se hayan añadido todos los marcadores a la base de datos, se puede descargar, tanto en formato para importar a Unity o para otros IDE como Android Studio. En este caso se usa Unity. Cuando se importa esta base de datos a Unity, se puede encontrar en la pestaña Databases en la configuración de Vuforia. A partir de este momento ya se puede empezar a añadir modelos 3D que aparecerán cuando el marcador se reconozca, y mediante Unity se pueden colocar estos modelos 3D justo en el lugar que sea necesario en cada caso.

El primer paso a llevar a cabo es la creación de la cámara principal que contendrá todos los objetivos de imagen y su respectivo modelo 3D. Para ello en la escena principal se crea el objeto ”AR Camera” que proporciona el plugin de Vuforia por defecto. Este objeto es como cualquier otro objeto de Unity, se le pueden agregar scripts personalizados o desactivar su comportamiento por defecto. Se le asigna la etiqueta de MainCamera de Unity para evitar conflictos.

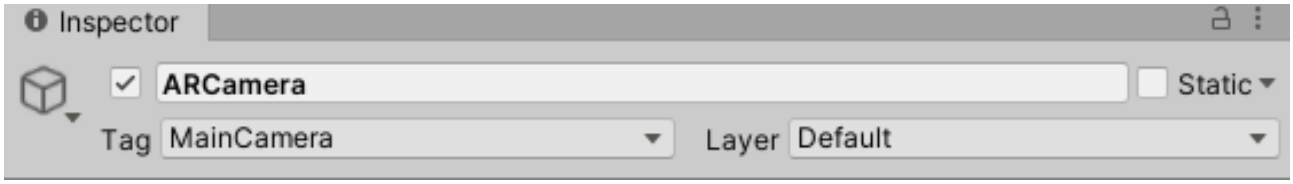


Figura 7.20: Etiqueta del objeto AR Camera.

Al añadir un objeto de Vuforia al proyecto, se generarán una serie de archivos y la estructura del proyecto será la siguiente:

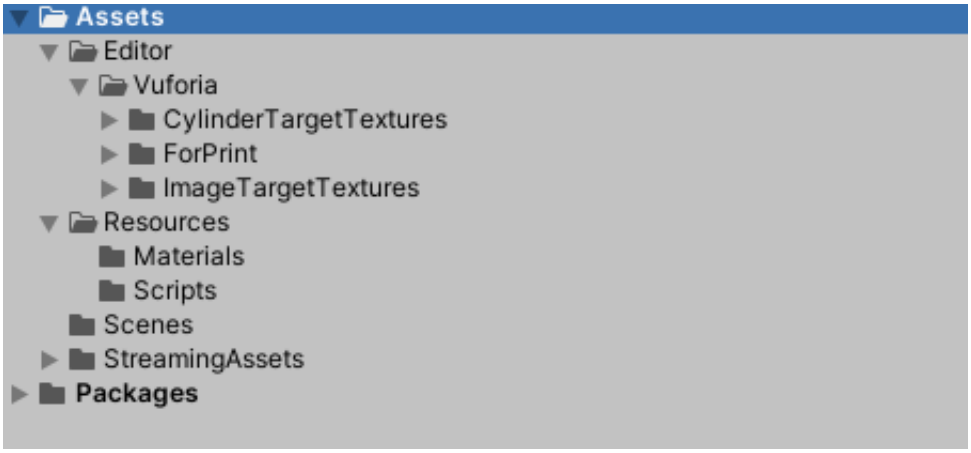


Figura 7.21: Estructura del proyecto Unity.

El siguiente paso es añadir los objetivos de imagen a la escena (contenidos dentro de la cámara en la jerarquía). Para esto se crea un nuevo objeto Image de Vuforia. Este objeto permite seleccionar un marcador de la base de datos que se ha añadido anteriormente. Una vez se ha elegido qué marcador va a ser, se añade un objeto de Unity 3D de tipo Quad (un plano) al interior del objeto Image Target en la jerarquía de Unity. Esto implica que al ser reconocido ese marcador, se mostrará el objeto 3D Quad en la pantalla. Pero este objeto está vacío, y se necesita que muestre una imagen antigua del mismo lugar que muestra el marcador. Para esto, se añade una imagen al fichero Resources, y se crea un Material al que se le asigna como textura la imagen que se acaba de añadir. En caso de que la imagen esté recortada, el tipo del objeto material debe ser de Sprite 2D, que soporta imágenes png con zonas transparentes y se adapta. Una vez el material con la textura correspondiente está listo, se añade al Quad que se ha creado previamente. Por último, hay que ajustar la posición, tamaño, escala y rotación de este Quad para que la imagen antigua coincida en la medida de lo posible con el objetivo de imagen. El resultado es el siguiente:

Figura 7.22: Ejemplo de un objetivo de imagen con su modelo 3D en la escena.

Cuando se reconoce el marcador del ejemplo anterior, el modelo 3D se muestra de esta forma:

Figura 7.23: Ejemplo de reconocimiento y superposición de realidad aumentada en la calle.

La jerarquía una vez todos los marcadores con sus modelos 3D han sido añadidos es la siguiente:

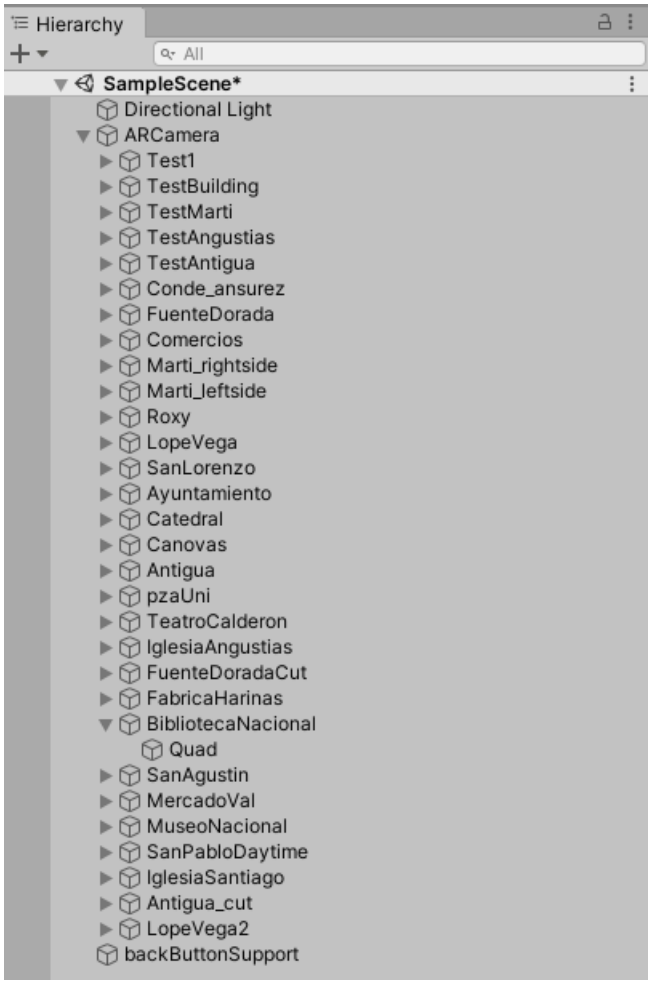


Figura 7.24: Ejemplo de reconocimiento y superposición de realidad aumentada en la calle.

Como se puede ver en la figura anterior, el último objeto de Unity llamado backButtonSupport es un objeto del juego básico de Unity por defecto que lleva asociado un script. Este script se encarga de que, una vez se detecte una pulsación del botón atrás de Android, se cierre este proceso de la librería, ya que de otra forma no es posible volver de la cámara a la aplicación Android nativa.

Por otra parte, en la configuración de Vuforia, se pueden ajustar ciertos parámetros. Se puede activar AR Core si el dispositivo que usa la aplicación es compatible con esta tecnología, pero tras llevar a cabo varias pruebas se ha comprobado que la experiencia de usuario con AR Core activado es de peor calidad, ya que el modelo 3D que se muestra se mantiene en pantalla aunque ya no se detecte el marcador, y además el reconocimiento empeora debido a que se pierde el enfoque predeterminado del motor de Vuforia, que optimiza enormemente la capacidad de reconocimiento. Para mejorar este reconocimiento aún más, se ha configurado el modo ”Quality” en lugar de ”Speed”. Esto permite una identificación mucho más eficaz ya que el modelo 3D se mueve constantemente si el reconocimiento no es del todo adecuado.

7.3. Integración de la librería Unity en la aplicación Android

Para poder utilizar la aplicación de realidad aumentada que se ha implementado en el apartado anterior, debemos exportarla como una librería. Para ello debemos configurar algunos parámetros en Player Settings de Unity, como el nombre de paquete, versión mínima de Android y arquitecturas que pueden ejecutar la aplicación. La versión mínima de Android será 4.4 y las arquitecturas serán las dos que Unity permite seleccionar, tanto ARM64 como ARMv7. Tras esto se exporta la aplicación y, en el directorio raíz del archivo que se obtiene, se encuentra un fichero denominado unityLibrary. Este debe moverse a la raíz del proyecto de la aplicación nativa Android.

Una vez la librería ha sido añadida, hay que modificar el archivo settings.gradle del proyecto, y añadir la ruta de la librería para incluirla. También se debe añadir como directorio de librerías el directorio que contiene la aplicación Unity de la siguiente forma:

```
allprojects {
    repositories {
        google()
        jcenter()
        flatDir {
            dirs "${project(':unityLibrary').projectDir}/libs"
        }
    }
}
```

Figura 7.25: Directorio de librería local añadido.

Una vez sea reconocida, se añade la librería como dependencia en el archivo build.gradle (como el resto de librerías empleadas). Para abrir la cámara mediante esta librería, se crea un Intent que la lance. Como se ha explicado en el apartado anterior, se creó un script para poder volver al presionar el botón atrás desde la cámara. Pero al cerrar la actividad de Unity, también destruye la actividad desde la que fue llamada. Por este motivo, se crea una actividad intermedia distinta a MainActivity, llamada IntermediateActivity. Esta actividad, al crearse, lanza la librería Unity. De esta forma, cuando se vuelve de la aplicación de realidad aumentada, se destruye la actividad intermedia, y no MainActivity. Cuando es eliminada, regresa a la última actividad cuyo ciclo de vida sigue activo, que es MainActivity y esta actividad muestra el último fragment desde el que se abrió la cámara. Tras este proceso, el árbol de directorios es el siguiente:

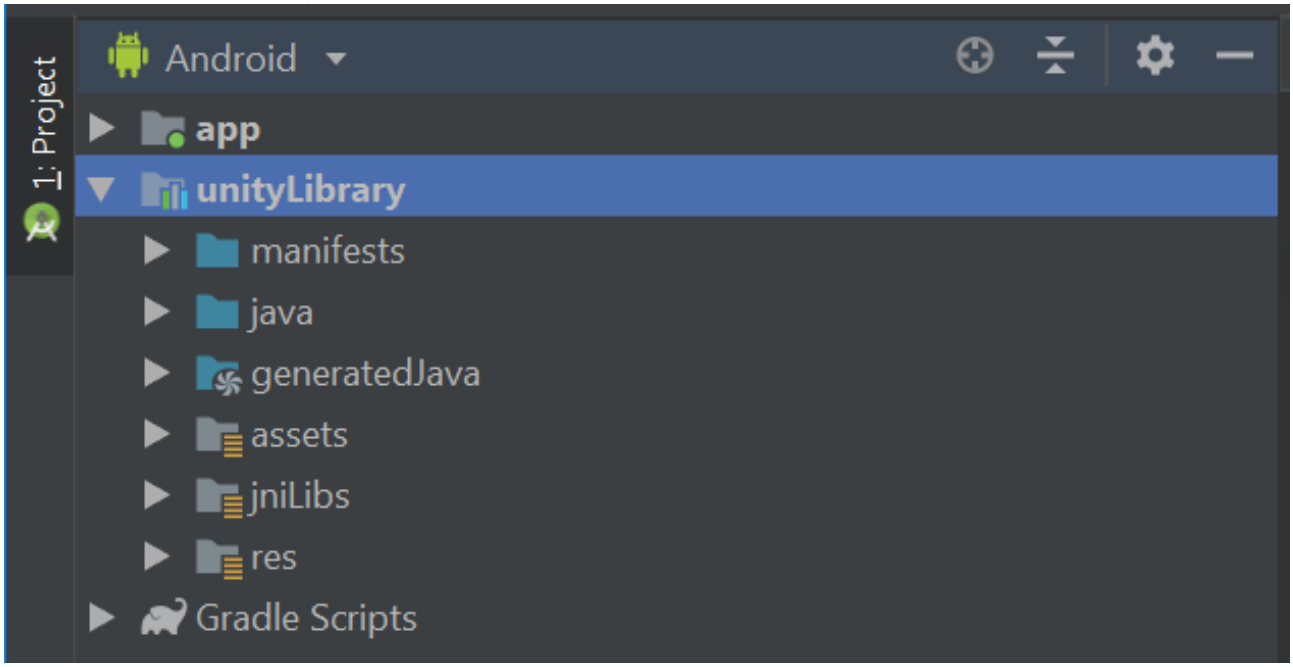


Figura 7.26: Árbol de directorios con librería añadida.

Capítulo 8

Pruebas

En este capítulo se mostrarán las distintas pruebas que se han llevado a cabo para comprobar el correcto funcionamiento del sistema.

8.1. Pruebas de caja blanca

Estas pruebas se han ido llevando a cabo durante la implementación del proyecto, ya que van ligadas al código fuente y a través de ellas se pueden solucionar errores en etapas tempranas de desarrollo para que no afecten a implementaciones posteriores.

8.2. Pruebas de caja negra

Estas pruebas incluyen todas aquellas realizadas posteriormente a la implementación del proyecto, sin tener en cuenta el código fuente y cómo ha sido implementado el mismo. Son guiadas por los casos de uso y requisitos del sistema, y, en el caso de las pruebas de la aplicación de realidad aumentada, se busca comprobar cómo se comporta el sistema en el mundo real y comprender en qué situaciones y para qué marcadores es más sencillo para el sistema reconocer los lugares. Las primeras pruebas realizadas van ligadas a los casos de uso:

Identificador	P01
Nombre	Ver lista de lugares
Prueba	El usuario verá una lista de lugares que puede visitar cuando inicie la aplicación.
Resultado	Tras ver la pantalla de carga, la lista de lugares se muestra correctamente.
Valoración	Correcta.

Cuadro 8.1: P01 - Ver lista de lugares

Identificador	P02
Nombre	Ver detalles de un lugar
Prueba	El usuario verá los detalles de un lugar al pulsar en cualquier lugar de la lista de lugares.
Resultado	La aplicación muestra una nueva vista con los detalles del lugar que se ha pulsado.
Valoración	Correcta.

Cuadro 8.2: P02 - Ver detalles de un lugar

Identificador	P03
Nombre	Ver galería de imágenes
Prueba	Se deslizará la imagen de la lista de lugares hacia la izquierda cuando no haya más imágenes disponibles a su izquierda y lo mismo a la derecha.
Resultado	La galería no presenta ningún cambio.
Valoración	Correcta.

Cuadro 8.3: P03 - Ver galería de imágenes

Identificador	P04
Nombre	Ver ruta en el mapa sin servicios
Prueba	Se intentará ver la ruta en el mapa cuando no hay permisos de ubicación, conexión a internet y la ubicación está desactivada para comprobar las peticiones.
Resultado	Se muestra un mensaje de aviso para pedir conexión a internet.
Valoración	Correcta.

Cuadro 8.4: P04 - Ver ruta en el mapa sin servicios

Identificador	P05
Nombre	Ver ruta en el mapa sin servicios de ubicación
Prueba	Se intentará ver la ruta en el mapa cuando no hay permisos de ubicación y la ubicación está desactivada para comprobar las peticiones.
Resultado	Se muestra un mensaje de aviso para pedir permisos de ubicación al usuario.
Valoración	Correcta.

Cuadro 8.5: P05 - Ver ruta en el mapa sin servicios de ubicación

Identificador	P06
Nombre	Ver ruta en el mapa sin ubicación activada
Prueba	Se intentará ver la ruta en el mapa cuando la ubicación está desactivada para comprobar las peticiones.
Resultado	Se muestra un mensaje de aviso para pedir al usuario que active el servicio de localización.
Valoración	Correcta.

Cuadro 8.6: P06 - Ver ruta en el mapa sin ubicación activada

Identificador	P07
Nombre	Rechazo de petición de permisos
Prueba	Se rechazará la petición de permisos de ubicación.
Resultado	Se muestra un mensaje de aviso temporal para pedir al usuario que proporcione permisos a la ubicación.
Valoración	Correcta.

Cuadro 8.7: P07 - Rechazo de petición de permisos

Identificador	P08
Nombre	Rechazo de activación de servicio de localización
Prueba	Se rechazará la petición activación del servicio de localización.
Resultado	Se muestra un mensaje de aviso temporal para pedir al usuario que active el servicio de localización.
Valoración	Correcta.

Cuadro 8.8: P08 - Rechazo de activación de servicio de localización

Identificador	P09
Nombre	Rechazo de activación de servicio de localización
Prueba	Se rechazará la petición activación del servicio de localización.
Resultado	Se muestra un mensaje de aviso temporal para pedir al usuario que active el servicio de localización.
Valoración	Correcta.

Cuadro 8.9: P09 - Rechazo de activación de servicio de localización

Identificador	P10
Nombre	Pérdida de servicio de ubicación mientras el mapa está activo
Prueba	Se desactivará el servicio de localización mientras el mapa con la ruta se muestra al usuario.
Resultado	El mapa muestra la última localización conocida del usuario, que deja de actualizarse.
Valoración	Correcta.

Cuadro 8.10: P10 - Pérdida de servicio de ubicación mientras el mapa está activo

Identificador	P11
Nombre	Abrir cámara de realidad aumentada
Prueba	Se pulsará el botón de cámara tanto en la vista de detalles como en la vista de mapa.
Resultado	Tras la splash screen de Unity, se muestra la cámara del dispositivo.
Valoración	Correcta.

Cuadro 8.11: P11 - Abrir cámara de realidad aumentada

Identificador	P12
Nombre	Cerrar cámara de realidad aumentada
Prueba	Se pulsará el botón ir hacia atrás cuando el sistema esté mostrando la cámara del dispositivo.
Resultado	Se cierra la cámara y tras una pequeña demora se vuelve a la vista desde la que se abrió la cámara.
Valoración	Correcta.

Cuadro 8.12: P12 - Cerrar cámara de realidad aumentada

8.3. Pruebas de la realidad aumentada

Para verificar el funcionamiento de la aplicación de realidad aumentada, se han visitado todos los lugares añadidos a la misma a lo largo de varios días para observar cómo se reconocen los marcadores. Los marcadores más difíciles de reconocer se han ajustado progresivamente hasta alcanzar el resultado esperado.

Se ha observado que los marcadores que el sistema no puede reconocer con facilidad provocan que el modelo 3D que aparece se mueva mucho, ya que intenta detectarlo de forma continua y cada vez que lo reconoce el modelo 3D se ajusta a la nueva posición. A medida que el reconocimiento mejora, el movimiento de los modelos 3D disminuye, y en lugares que se detectan de forma sencilla el modelo 3D permanece inmóvil. Para mejorar la

facilidad del sistema de reconocer un marcador, se emplean varias técnicas. Es importante no añadir imágenes de marcador con características que pueden cambiar mientras es enfocado: Partes de la calle, carreteras, vegetación... También es importante añadir más de un marcador para aquellos lugares más difíciles de detectar.

Tras este proceso de pruebas y adaptación de la implementación, prácticamente todos los marcadores se reconocen fácilmente y el modelo 3D no se mueve, lo cual proporciona una experiencia de usuario de más calidad.

Capítulo 9

Conclusiones y trabajo futuro

Una vez el desarrollo del proyecto ha finalizado, se puede decir que se han cumplido los objetivos y requisitos impuestos al inicio del mismo: crear una aplicación de guía turística que soporte realidad aumentada para ver imágenes antiguas sobre la ciudad de la actualidad.

Este desarrollo ha sido muy interesante y enriquecedor ya que me ha permitido aprender mucho más acerca de aplicaciones móviles Android y sus patrones de arquitectura, así como un lenguaje de programación tan potente y moderno como es Kotlin. También he aumentado mis conocimientos acerca de los distintos SDK de realidad aumentada, especialmente Vuforia y Unity.

Por último, he conseguido gestionar y planificar un proyecto de una escala mucho mayor a los que he llevado a cabo en las asignaturas del grado, utilizando apropiadamente un plan de proyecto y un repositorio Git, así como unos requisitos y objetivos a cumplir bien definidos. Como trabajo futuro para ampliar este proyecto, pueden llevarse a cabo las siguientes implementaciones:

- Añadir más ciudades que se puedan visitar con sus distintos atractivos turísticos y patrimonio histórico haría que el número de usuarios que pudieran hacer uso del sistema fuera mucho más elevado.
- Distintos idiomas para la aplicación también harían mucho más fácil su uso para todo tipo de usuarios (y más teniendo en cuenta que se trata de una aplicación de uso turístico).
- Funcionalidad para cambiar de modelo 3D en tiempo de ejecución para que, en caso de haber más de una imagen antigua para la misma perspectiva, el modelo 3D que se muestra cuando se reconoce el marcador pueda ser intercambiado usando la interfaz.

Bibliografía

[1] M. Akçay. *Advantages and challenges associated with augmented reality for education: A systematic review of the literature*. http://www.cs.ucf.edu/courses/cap6121/spr17/readings/ARLit.pdf. (Visitado el 28/08/2020). Nov. de 2016.

[2] Carlos Arce. *Realidad Aumentada*. http://jeuazarru.com/wp-content/uploads/2014/10/RA2013.pdf. (Visitado el 01/09/2020). Oct. de 2014.

[3] S. O'Dea. *Global mobile OS market share — Statista*. https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. (Visitado el 28/08/2020). Feb. de 2020.

[4] Google. *Introducción a Android Studio*. https://developer.android.com/studio/intro. (Accessed on 29/08/2020). Mayo de 2020.

[5] Google. *Kotlin*. https://kotlinlang.org/. (Visitado el 08/29/2020).

[6] Vijay Singh. *Kotlin vs Java: Important Differences That You Must Know*. https://hackr.io/blog/kotlin-vs-java. (Visitado el 29/08/2020). Ago. de 2020.

[7] Google. *Google Maps Platform*. https://cloud.google.com/maps-platform. (Visitado el 29/08/2020).

[8] Unity. *Plataforma de Unity*. https://unity.com/es/products/unity-platform. (Visitado el 29/08/2020).

[9] PTC. *Vuforia: Market-Leading Enterprise AR*. https://www.ptc.com/en/products/vuforia. (Visitado el 29/08/2020).

[10] Google. *Augmented Images for Android NDK*. https://developers.google.com/ar/develop/c/augmented-images. (Visitado el 29/08/2020).

[11] Pablo Domínguez. *En qué consiste el modelo en cascada - Gestiona tu proyecto de desarrollo - OpenClassrooms*. https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada. (Visitado el 29/08/2020). Jun. de 2020.

[12] Wikipedia. *Desarrollo en cascada*. http://es.wikipedia.org/w/index.php?title=Desarrollo%20en%20cascada&oldid=128455675. (Visitado el 29/08/2020). 2020.

[13] Google. *Guía de arquitectura de apps — Desarrolladores de Android*. https://developer.android.com/jetpack/docs/guide#recommended-app-arch. (Visitado el 03/09/2020).

[14] Inducesmile. *Why MVVM and How to execute MVVM combined with Data Binding and LiveData in Four Simple Steps*. https://inducesmile.com/android/why-mvvm-and-how-to-execute-mvvm-combined-with-data-binding-and-livedata-in-four-simple-steps/. (Visitado el 04/09/2020).

[15] Google. *Descripción general de ViewModel — Desarrolladores de Android*. https://developer.android.com/topic/libraries/architecture/viewmodel. (Visitado 04/09/2020).

[16] Kishore C.S. *ListView vs RecyclerView*. https://medium.com/@kish.imss/listview-vs-recyclerview-2965d50b363. (Visitado el 05/09/2020). Sep. de 2018.

[17] Sam Judd. *bumptech/glide: An image loading and caching library for Android focused on smooth scrolling*. https://github.com/bumptech/glide. (Visitado el 05/09/2020).

[18] Amanda Hinchman. *Working with RecyclerView in Android & Kotlin — by mvndy — Medium*. https://medium.com/@hinchman_amanda/working-with-recyclerview-in-android-kotlin-84a62aef94ec. (Visitado el 06/09/2020). Nov. de 2018.

[19] denzcoskun. *ImageSlideshow: Android image slider.* https://github.com/denzcoskun/ImageSlideshow. (visitado el 09/09/2020).

Apéndice A

Manual de instalación

Para poder instalar la aplicación se debe abrir el archivo apk en un dispositivo cuya versión de Android sea 5.0 o superior y cuyo procesador sea ARM64 o ARMv7. Se deben seguir los siguientes pasos:

- Activar la opción de instalación desde orígenes desconocidos en los ajustes del dispostivo (ya que no se encuentra disponible en Google Play).
- Tener al menos 126 MB libres de espacio en la memoria del dispositivo.
- Descargar el archivo con extensión apk.
- Abrir el archivo y seguir los pasos del gestor de instalación que muestra el sistema.

Apéndice B

Manual de usuario

Una vez la aplicación haya sido instalada e iniciada, se mostrará la siguiente pantalla:

Figura B.1: Primera vista de la aplicación.

El usuario puede deslizar hacia arriba o abajo para ver los distintos lugares de la lista. Para mostrar los detalles de uno de ellos, se puede pulsar en su fotografía y aparecerá la siguiente pantalla:

Figura B.2: Vista de detalles de un lugar.

En esta vista se pueden observar varias fotografías (tanto antiguas como actuales) del lugar pulsado anteriormente. En la parte inferior se encuentran dos botones, uno para mostrar la ruta hacia ese lugar en el mapa y otro para abrir la aplicación de realidad aumentada si el usuario ya se encuentra en esa localización. Si el usuario pulsa en el botón de navegación se mostrará la siguiente vista:

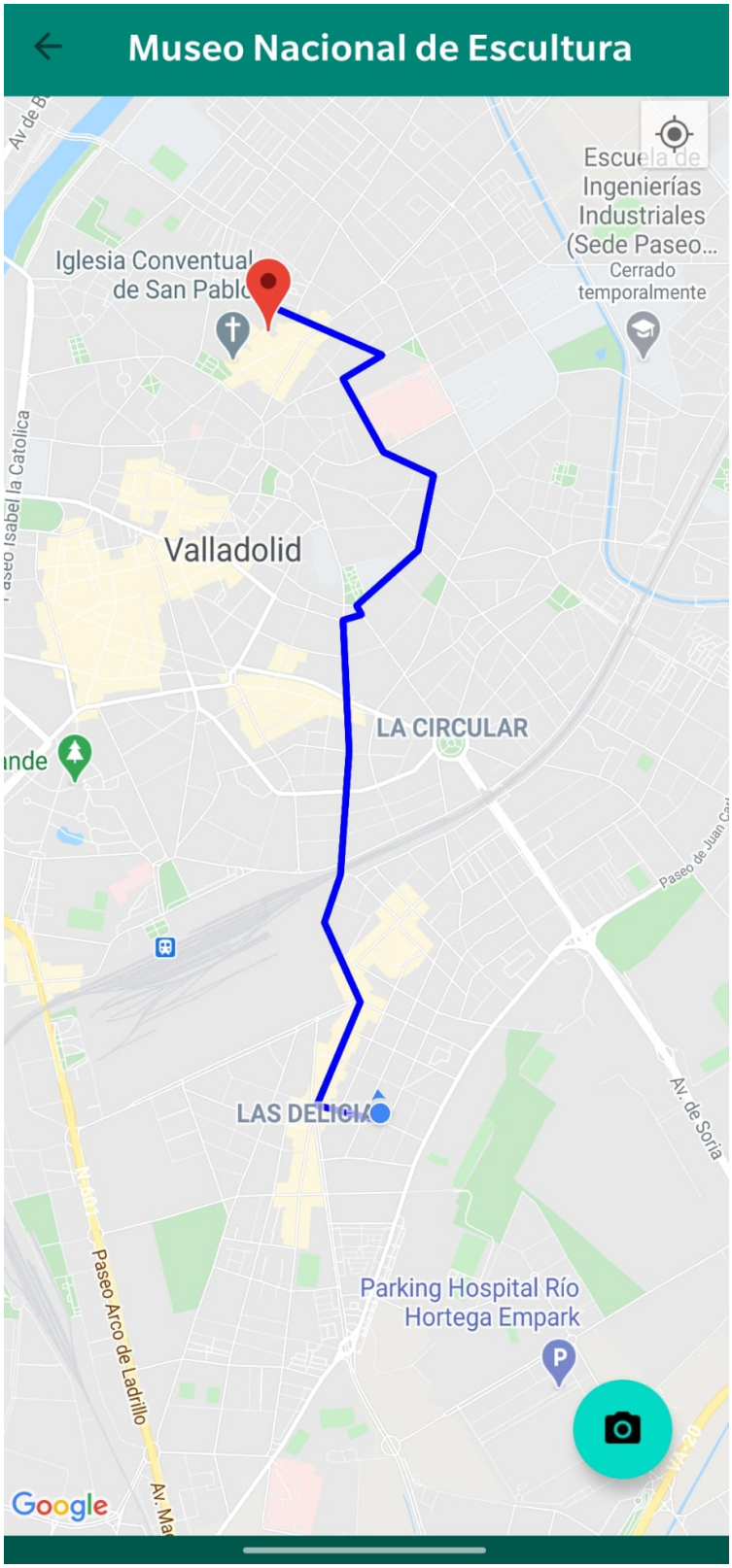


Figura B.3: Vista de ruta en el mapa.

Una vez el usuario haya llegado a la localización marcada en el mapa, deberá iniciar la aplicación de realidad aumentada utilizando el botón que se muestra en la parte inferior derecha de la pantalla. Se le mostrará una pantalla de carga con el logotipo de Unity:

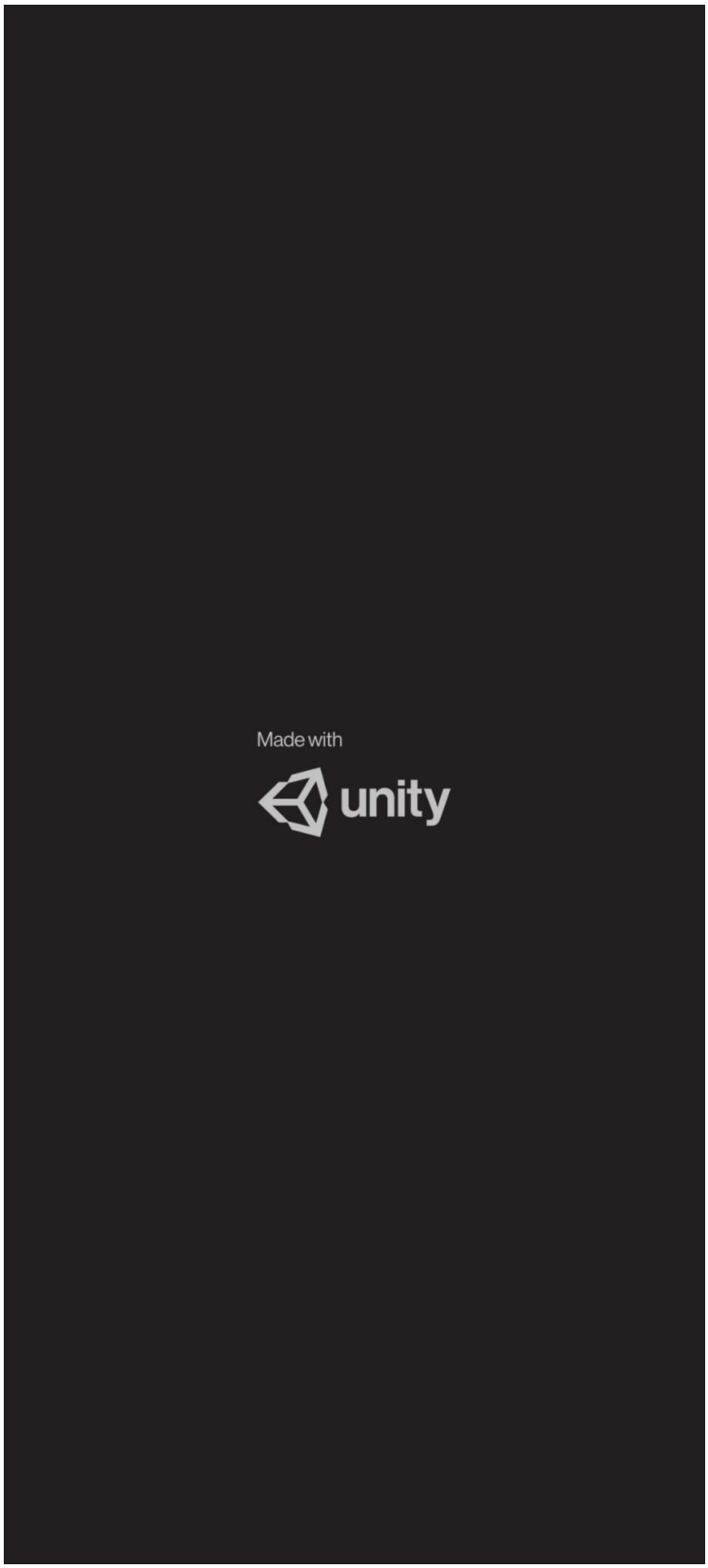


Figura B.4: Pantalla de carga de Unity.

Una vez esta pantalla desaparezca, el sistema mostrará la cámara del dispositivo, con la cual se debe enfocar el lugar desde la ubicación marcada por el sistema. Tan pronto como el marcador sea reconocido, se superpondrá la imagen antigua de ese lugar sobre la vista actual:

Figura B.5: Modelo 3D sobre imagen actual.

Apéndice C

Entregables

Los entregables del proyecto Se encuentran en el siguiente enlace:
https://drive.google.com/file/d/1VSVn8RKSvoN1UMH07MvHyRNwnFoLCowC/view?usp=sharing
Al descargar y extrar el archivo se pueden encontrar los siguientes elementos:

- Código fuente del proyecto Unity: Se encuentra en la carpeta Vuforia. Para poder abrir este proyecto se debe instalar Unity Hub, añadir la versión de Unity 2019 (con soporte para Android, tanto NDK como SDK) y añadir esta carpeta a la lista de proyectos de Unity. Por último se le asigna la versión de Unity con la que se quiere abrir (la descargada anteriormente).
- Código fuente de la aplicación nativa Android: Se encuentra en la carpeta ARGuide. Para poder ver este proyecto se debe instalar Android Studio y abrir un proyecto ya existente, seleccionando esta carpeta.
- Archivo de instalación de aplicación Unity: Es ARLib.apk. Este archivo puede ejecutarse en un dispositivo Android para instalar la aplicación de realidad aumentada.
- Archivo de instalación de aplicación completa: ARGuide.apk. Tras instalar el sistema mediante este archivo se dispondrá de toda la funcionalidad del sistema.

En el siguiente enlace se pueden descargar los diagramas creados para este proyecto:
https://drive.google.com/file/d/1Y-tLmF0oUIZILArhQw8ZwgEM5JUClFwV/view?usp=sharing