EXPLICACIONES MEDIANTE REALIDAD AUMENTADA DE SISTEMAS DE RECOMENDACIÓN SOCIAL EN EL DOMINIO DEL OCIO

Diego Acuña Berger, Daniel Calle Sánchez, Carlos Gómez Cereceda, Zihao Hong

GRADO EN INGENIERÍA DEL SOFTWARE. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin de Grado en Ingeniería del Software

CURSO 2018-2019

Director/es y/o colaborador:

Juan Antonio Recio García
Guillermo Jiménez Díaz

Autorización de difusión

Diego Acuña Berger, Daniel Calle Sánchez, Carlos Gómez Cereceda, Zihao Hong

Fecha

Los abajo firmantes, matriculados en el Grado en Ingeniería del Software de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores el presente Trabajo Fin de Grado: "EXPLICACIONES MEDIANTE REALIDAD AUMENTADA DE SISTEMAS DE RECOMENDACIÓN SOCIAL EN EL DOMINIO DEL OCIO", realizado durante el curso académico 2018-2019 bajo la dirección de Juan Antonio Recio García y Guillermo Jiménez Díaz en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen

Actualmente, en un mundo cada vez más tecnológico, es muy común que surjan aplicaciones diseñadas para satisfacer las necesidades de la sociedad en distintos ámbitos, como pueden ser el de la salud, la educación o el ocio entre otros. Nosotros con nuestro proyecto decidimos centrarnos en el ámbito del ocio y, más concretamente, en el mundo del cine.

Nuestra aplicación surge de la necesidad de aquellas personas que quieren ver una película acompañados ya que, pocas personas hoy en día deciden ir al cine solas y podemos estar de acuerdo en que la mayoría de personas prefieren ver una película en compañía. Para dar respuesta a ésto, nuestra aplicación permite crear planes para ver una cierta película con amigos o personas que tengan un gusto similar. Además, existe la posibilidad de que un usuario no sepa que película quiere ver, por lo que la aplicación tendrá la capacidad de recomendar a los usuarios películas que se ajusten a los gustos que tienen.

Lo más relevante de nuestra aplicación es la facilidad con la que un usuario puede interactuar con las películas y con los usuarios. Con esto nos referimos a que, gracias a la Realidad Aumentada, podremos reconocer con la cámara de nuestro dispositivo carteles de películas e incluso nos permite el reconocimiento facial de usuarios, para así obtener explicaciones sobre lo que estamos reconociendo en tiempo real y así poder interactuar a través de la Realidad Aumentada.

Palabras clave

Lista de palabras clave

- Realidad Aumentada
- Sistemas de Recomendación
- Filtrado Colaborativo
- Ocio
- Película
- Aplicación móvil

Abstract

Nowadays, in a more and more technological world, it is very common that applications designed to satisfy society′s needs in different ambits, such as health, education or leisure appear. With our project we decided to focus on leisure′s ambit and, more concretely, in the cinema′s world.

Our application comes up from the need of those people who want to watch a movie accompanied, since few people nowadays decide to go to the cinema alone and we can agree that the majority of people prefer to watch a movie in company. To answer this, our application allows the creation of plans to go watch a certain movie with friends or even people who have a similar taste. Also, it is possible that an user doesn′t know what movie he wants to see, so that the application will be capable of recommending the users movies that suit their tastes.

What is more important about our application is the ease with which an user can interact with the films and the users. With this we mean that, thanks to augmented reality, we will be able to recognize with our device′s camera movie posters and it even allows facial recognition of users, so we can obtain explications about what we are recognizing in real time and so be able to interact with it through augmented reality.

Keywords

Keywords list

- Augmented Reality
- Recommendation Systems
- Collaborative Filtering
- Leisure
- Film
- Mobile application

Índice general

Agradecimientos

Nos gustaría agradecer a nuestra familia y compañeros/as la confianza depositada en nosotros así como la realimentación recibida por parte de toda persona que se ha involucrado en mayor o menor medida en la realización de nuestro proyecto, ya fuera mediante comentarios constructivos o la redirección a la documentación de tecnologías que usaríamos posteriormente.

Capítulo 1

Introducción

En este capítulo introduciremos la idea de nuestro proyecto, de dónde surge, qué objetivos queremos lograr y cómo nos hemos organizado a la hora de trabajar. Con este proyecto queremos desarrollar una aplicación para solventar una serie de situaciones en las que una persona tiene ganas de ver cierta película, ya sea porque se ha encontrado el cartel de la nueva película de su saga favorita anunciada en la calle y no sabe con quién ir, o porque quizás un amigo le ha sugerido que le acompañe a ver una película de la que no saben nada por el mero hecho de ir al cine. Decidimos ponernos manos a la obra y desarrollar una aplicación móvil que permitiera a sus usuarios actuar en distintas situaciones para que con el simple uso de la cámara de su smartphone, e incluso sin ella, pudieran de forma fácil e interactiva, organizarse para ir al cine o a ver la película a casa de otra persona. Además, podrían apoyarse en la información que la aplicación les proporciona sobre las distintas películas para decidir si les es interesante ir a verla, sobre todo si es para ir al cine ya que el usuario podría ahorrarse el gasto de acabar viendo una película que no le gusta por no poder saber si se ajusta a sus gustos. Además, tener una red de contactos de personas interesadas en ver películas que a ti también te gusten y tener la posibilidad de quedar en un lugar, fecha y hora determinados refuerzan el atractivo de nuestra aplicación en el dominio del ocio. Nuestro proyecto ofrecerá a los usuarios explicaciones sobre el producto que se esté reconociendo mediante la Realidad Aumentada, es decir, al usuario se le recomendarán productos relevantes de la aplicación según sus gustos, también se le mostrará información

sobre los objetos que se reconocen.

1.1. Antecedentes

Para entender de donde surge nuestra idea tenemos que remontarnos al momento en el que se propuso este proyecto. El principal motivo es la mejora de un Trabajo de Fin de Grado anterior al nuestro[34], cuyo enfoque y finalidad era el mismo. Nuestro objetivo principal es ampliar su funcionalidad diferenciándonos, sobre todo, en las tecnologías usadas y en los casos de uso representados.

En nuestra aplicación no solo nos centramos en ir al cine y reconocer películas que estuvieran en la cartelera para que nos recomiende la que más se ajuste a nuestros gustos. Decidimos que el usuario podría crear planes con una película que le interesase ver y que sus amigos u otros usuarios pudieran unirse para ir todos juntos a verla. También añadiremos más interacción sobre las escenas de Realidad Aumentada, permitiéndonos ver la información sobre la película reconocida, ver trailers o guardarla en la lista de favoritos. Además, con la Realidad Aumentada, se podrán realizar reconocimientos faciales sobre los usuarios para añadirlos como amigos, para después recomendarnos en tiempo real que películas podemos ir a ver con éstos.

1.2. Objetivos

Nuestro objetivo principal es crear una aplicación que ayude a los usuarios a realizar planes con sus amigos para ir al cine a ver películas. Además, mostrará qué películas pueden interesar al usuario en base a sus gustos. También se usarán tecnologías de Realidad Aumentada que ayuden al usuario a usar la aplicación. La lista de subobjetivos es la siguiente:

- Estudiar las tecnologías de Realidad Aumentada actuales.
- Analizar sistemas de recomendación, así como su uso dentro del proyecto.
- Definir casos de uso reales que aporten valor a la aplicación.

- Diseñar una aplicación para dispositivos móviles que aplique las anteriores tecnologías.
- Diseñar y aplicar una arquitectura que cumpla las necesidades del proyecto.

1.3. Plan de trabajo

Para comenzar con nuestro proyecto analizaremos el estado del arte, en el Capítulo 2, en el que estudiaremos los conceptos básicos de la Realidad Aumentada y buscaremos las tecnologías actuales a la fecha del proyecto que funcionen en dispositivos móviles. También realizaremos un estudio de las distintas fuentes de información de las que podemos obtener datos y un estudio de técnicas de recomendación.

Después realizaremos las tareas de diseño, descritas en el Capítulo 3. Primero, un análisis de aquellas aplicaciones que se asemejen a nuestra idea. A continuación, nos plantearemos los distintos escenarios en los que sería usada nuestra aplicación. Posteriormente realizaremos una lista de las funcionalidades más importantes de nuestra aplicación y diseñaremos las interfaces de usuario que acompañarán a dichas funcionalidades.

El siguiente paso, descrito en el Capítulo 4, comenzará con el desarrollo de una serie de prototipos con distintas tecnologías que nos permitan decidir cuales utilizar en la implementación de la aplicación final. Después diseñaremos la arquitectura e implementaremos la aplicación, para lo cual aplicaremos algunas prácticas de la metodología SCRUM. El desarrollo se realizará con iteraciones cercanas a las dos semanas. Comenzará con la selección de objetivos más prioritarios y terminará con las reuniones de revisión junto con los directores del Trabajo de Fin de Grado. Tras cada iteración, los miembros del equipo realizaremos una reunión de retrospectiva para identificar en qué hemos fallado durante esta iteración y cómo podemos mejorar para la siguiente.

Finalmente en el Capítulo 5, hablaremos de las conclusiones que hemos obtenido tras realizar el proyecto, seguido del trabajo futuro que se podría realizar sobre nuestra aplicación para conseguir futuras mejoras en el Sección 5.2.

1.4. Introduction

In this chapter we are going to introduce the idea of our project, where it comes from, what objectives we want to achieve and how we have organized ourselves to work together. With this project we want to develop an application to solve a series of situations in which a person wants to watch a certain movie, either because he has encountered the poster of a new movie of it′s favorite saga announced on the street and doesn′t know who to go with, or because maybe a friend of him has suggested on going to see a movie together they don′t know about by the mere fact of going to the cinema. We decided to get to work and develop a mobile application that would allow it′s users to act in different situations so that with the simple use of their smarthpone′s camera, or even without it, they could get organized to go to the cinema or to watch the movie in another person′s house in an easy and interactive way. Also, they could rely on the information that the application provides them about the different movies so they can decide whether to watch it or not, especially if it is for going to the cinema since the user could save himself the expense of ending up watching a movie thath he doesn′t like because he can′t know if it fits his tastes. Furthermore, having a network of contacts of people interested in going to watch films that you like too and having the possibility of meeting on a certain place, date and hour will reinforce the attractive of our application in leisure′s domain.

1.5. Background

To understand where our idea comes from we have to go back to the moment in which this project was proposed. The main reason is the improvement of a Final Degree Project previous to ours[34], which approach and purpose was the same. Our main objective is to extend it′s functionality differentiating us, especially, in the technologies used and in the represented use cases.

In our application we don′t just focus on going to the cinema and recognizing films that were on the billboard so that it can recommend us the movie that fits our tastes better.

We decided that an user could create plans with a film he is interested in watching and that his friends or other users could join so they can all go together. We will also add more interaction to the augmented reality scenes, allowing us to see the information about the film we are recognizing, watch trailers or save it to our favorites list. Furthermore, with augmented reality, it will be possible to realize facial recognitions of users so we can add them as friends, so it can afterwards recommends us films that we can go watch with them in real time.

1.6. Objectives

Our main objective is to create an application that will help the users to carry out plans with their friends to go to the cinema to watch movies. It will also show what films are interesting for the user based on his tastes. We will also use augmented reality technologies that will help the user to use the application. The list of subobjectives is the next one:

- Study the current augmented reality technologies.
- Analize recommendation systems, as well as it′s use inside the project.
- Define real uses cases that will add value to the application.
- Design an application for mobile devices that will apply the technologies described before.
- Design and apply an architecture that meets the needs of the project.

1.7. Workplan

To start with our project we will analize the state of art, in Chapter 2, in which we will study the basic concepts of augmented reality and we will search the current technologies to the project′s date that work in mobile devices. We will also perform a study of the different

information sources from which we can obtain data and we will also perform a study about recommendation techniques.

Afterwards we will perform design tasks, described in Chapter 3. First, an analysis of those applications which are similar to our idea. Then, we will consider the different scenarios in which our application will be used. Later we will do a list of the most important functionalities of our application and design the user interfaces that will accompany these functionalities.

The next step, described in Chapter 4, will start with the development of a series of prototypes with different technologies that will allow us to decide which ones to use in the implementation of our final application. After we will design the architectura and implement the application, for it we will apply some practices of the SCRUM methodology. The development will be carried out with iterations close to the two weeks of duration. It will start with the selection of the most priority objectives and will end with review meetings alongside the Final Degree Project directors. After each iteration, all of the team members will attend a retrospective meeting to identify in what we have failed during this iteration and how we could improve for the next one.

Finally in Chapter 5, we will speak about the conclusions we have obtained from implementing this project, followed by the future work that could be done in our application tu achieve future improvements in Section 5.2.

Capítulo 2

Estado del arte

Para que esta aplicación sea usable e interactiva, en esta sección de la memoria procederemos a realizar un estudio de las posibles tecnologías que nos permitan llevar a cabo nuestros objetivos, tales como la Realidad Aumentada, las distintas fuentes de información y algunas técnicas de recomendación.

2.1. Realidad Aumentada

El gran peso de nuestra aplicación está basado en la tecnología de la Realidad Aumentada que, tras investigar su significado en varias fuentes de información, decidimos elegir la siguiente definición:

> " La Realidad Aumentada nos permite añadir capas de información visual sobre el mundo real que nos rodea, utilizando la tecnología, dispositivos como pueden ser nuestros propios teléfonos móviles. Esto nos ayuda a generar experiencias que aportan un conocimiento relevante sobre nuestro entorno, y además recibimos esa información en tiempo real. Mediante la Realidad Aumentada el mundo virtual se entremezcla con el mundo real, de manera contextualizada, y siempre con el objetivo de comprender mejor todo lo que nos rodea. "[21]

La Realidad Aumentada se aplica a diversas áreas como seguridad, logística, sanidad, ocio, etc... Se puede aplicar a tantas disciplinas como podamos imaginar y no sólo se ha

Figura 2.1: *Telemadrid anuncia el uso de RA en las elecciones de 2019*

Figura 2.2: *Antena 3 usa RA para las elecciones del ayuntamiento de Madrid*

convertido en una herramienta útil, sino que es una tecnología atractiva y además signo de modernidad. Por estas razones actualmente vemos ejemplos como el de las televisiones, que compiten por aplicarla a sus programas y presumir de dar el mejor servicio a sus espectadores como vemos en la Figura 2.1 y Figura 2.2. Con esta situación podemos por tanto estar de acuerdo en que es una tecnología en auge actualmente.

Los smartphones son los dispositivos que más interaccionan con esta tecnología, ya que prácticamente todas las personas poseen uno y están dotados de un amplio abanico de sensores necesarios para su correcto funcionamiento. Por esto elegimos este tipo de dispositivos para nuestro proyecto ya que, esta tecnología, aporta un gran valor añadido a nuestra aplicación, incluso siendo el grueso de la misma.

La Realidad Aumentada en cualquier dispositivo se basa en dónde nos encontramos y hacia donde estamos mirando, es decir, si usamos la cámara del teléfono, según a donde estemos mirando a través de ésta, la Realidad Aumentada superpondrá objetos virtuales o información a nuestra realidad, con un cierto tamaño y una cierta posición basándose en las medidas de lo que vemos a través de nuestra cámara. Por ejemplo, la famosa aplicación de Pokemon Go, que superpone la imagen de un pokemon a la realidad que vemos con la cámara para que podamos atraparlo, haciendo así que su uso sea más interactivo y visual para el usuario, como podemos observar en la Figura 2.3. En la Figura 2.4 y Figura 2.5 podemos ver otros ejemplos del uso de la Realidad Aumentada en dispositivos móviles,

como el reconocimiento de monumentos turísticos o para medir ciertas distancias.

Figura 2.3: *RA en Pokemon Go*[30]

Figura 2.4: *RA para zonas turísticas*[33]

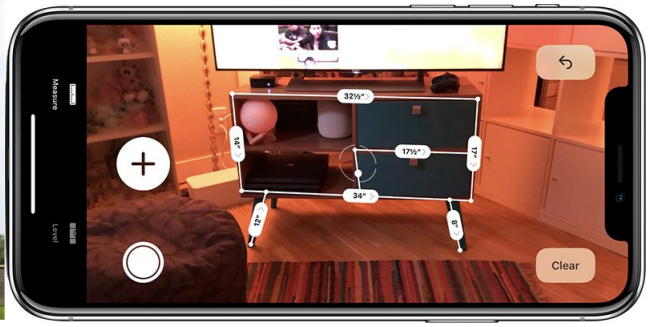

Figura 2.5: *RA para mediciones*[16]

2.1.1. Tecnologías actuales

A continuación, describiremos las distintas tecnologías que hemos investigado para la implementación de la Realidad Aumentada.

- ARCore[9]: esta plataforma creada por Google para desarrollar aplicaciones de Realidad Aumentada con soporte para Android, Android NDK, iOS, Unity y Unreal Engine, aunque las funcionalidades que se ofrecen para iOS y Unity for iOS se limitan a Cloud Anchors. Los anchors son herramientas que permiten que objetos virtuales aparezcan en un lugar captado por la cámara de nuestro dispositivo, estos son compartidos en la nube para que multitud de dispositivos disfruten de la misma experiencia, aquellos que posean iOS podrán usarlos utilizando ARKit.

  ARCore usa tres características a través de la cámara del dispositivo:

  - Motion tracking: permite al dispositivo conocer la posición relativa del mundo.
  - Environmental understanding: permite al dispositivo detectar el tamaño y localización de todos los tipos de superficies.
  - Light estimation: permite al dispositivo estimar las condiciones de luz del entorno actual.

  Puntos positivos:

  - Buena integración con Android.
  - Tiene una curva de aprendizaje media y con su versión 1.5 demuestra una estabilidad interesante respecto a su reciente creación.

  Puntos negativos:

  - Poca documentación para soluciones más complejas.

- No todos los dispositivos son compatibles. Esto depende de que las empresas que desarrollan estos dispositivos cumplan unos requisitos para asegurar que la experiencia con ARCore es la adecuada y de la versión del sistema operativo.

- ARKit[1]: es una librería desarrollada por Apple que podemos utilizar para crear experiencias de Realidad Aumentada persistente y compartirlas entre distintos dispositivos iOS. Esta tecnología se basa en la odometría visual inercial, esta es capaz de reconocer las imágenes que capta la cámara y la forma en la que la luz se refleja en los diferentes elementos que aparecen para obtener un mapa 3D del entorno y, calcular las distancias que hay entre los diferentes objetos desde la posición del dispositivo. Tras iniciar la Realidad Aumentada con ARKit, nuestro dispositivo crea un entorno virtual donde nuestro dispositivo representa la coordenada (0,0,0), con los tres ejes, el horizontal, vertical y el de profundidad, pero también es necesario otro eje, el eje W que representa la rotación del dispositivo. En la Figura 2.6 se puede observar cómo cambia el elemento virtual según como situemos nuestro smartphone.

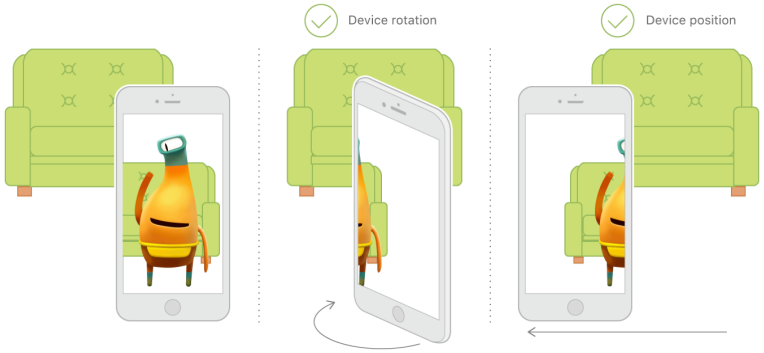

Figura 2.6: *RA en ARKit*[10]

Puntos positivos:

- Detecta imágenes 2D incluso en movimiento y objetos 3D.
- Buena integración con iOS.

Puntos negativos:

- Es necesario ordenadores con macOS para el desarrollo.

- Wikitude[32]: es un kit de desarrollo para Realidad Aumentada con soporte para Android, iOS, Unity, Cordova, Xamarin, Titanium y React Native. Utiliza ARCore o ARKit cuando los dispositivos lo soportan y en caso contrario utiliza tecnología de Wikitude para que el número de dispositivos compatibles sea mayor. Algunas de las tecnologías que soporta Wikitude son la geolocalización, el reconocimiento de imágenes y el reconocimiento en la nube. Para entender cómo funciona este SDK tenemos que describir tres términos básicos fundamentales:

  - Target: conjunto de datos extraídos de una imagen, empleados por un rastreador al reconocerla para realizar alguna acción.
  - Target Collection: es un conjunto de targets empleado por el rastreador para reconocer imágenes en el mundo real.
  - Client Tracker: rastreador que usa la cámara para analizar objetos 2D, analiza el Target Collection y busca similitudes con la imagen reconocida.

  Puntos positivos:

  - Compatibilidad con muchos dispositivos.
  - Dispone de reconocimiento de imágenes en la nube.

  Puntos negativos:

  - Su licencia es de pago, aunque hay versiones limitadas gratuitas, pero limitan su funcionalidad.
  - Documentación obsoleta para algunas plataformas de desarrollo.

- Vuforia[31]: Kit de desarrollo para Realidad Aumentada con soporte para Android, iOS, UWP y Unity. Su integración con Unity permite crear aplicaciones y juegos para que sean usados en Android y iOS simplemente arrastrando y soltando elementos. Usa

un seguimiento basado en marcadores, los marcadores son las imágenes u objetos reconocidos por la aplicación para desencadenar la visualización del contenido virtual sobre su posición en la vista de la cámara. Los Image Targets u objetivos de imagen, son un tipo específico de marcador, son imágenes que se registran manualmente en la aplicación y que pueden ser guardadas en un Cloud, reduciendo así el peso de tener las imágenes a reconocer en el dispositivo. También se puede realizar un seguimiento sin marcado, basado en el GPS o en un giroscopio para insertar modelos de objetos virtuales en la realidad que capta la cámara.

Puntos positivos:

- Se encuentran fácilmente ejemplos y documentación sobre su uso.
- Compatibilidad con muchos dispositivos.
- Dispone de reconocimiento de imágenes en la nube.

Puntos negativos:

- Su licencia es de pago, aunque puede usarse gratis para desarrollar y probar. Con límites en su funcionalidad.

- ViroReact[15]: Plataforma para construir aplicaciones con Realidad Aumentada usando React Native. Es capaz de combinar tecnologías web que se pueden utilizar en distintas plataformas con llamadas a APIs internas de las distintas plataformas móviles. De este modo, podemos generar aplicaciones fácilmente que funcionen en Android utilizando el potencial de ARCore y en iOS utilizando el de ARKit. A pesar de ser software privativo, su licencia no tiene limitaciones en las funcionalidades.

  Puntos positivos:

  - Curva de aprendizaje fácil.
  - Se utiliza el mismo código para generar aplicaciones en iOS y Android.

Puntos negativos:

- Al basarse en React Native, que no tiene versión estable, provoca problemas con las versiones de dependencias.
- Configuraciones tediosas y largas compilaciones.

- Expo AR[7]: Es un conjunto de librerías escritas de forma nativa para cada plataforma, proporcionando acceso a la funcionalidad del sistema del dispositivo (como pueden ser la cámara, notificaciones emergentes, contactos, almacenamiento local y otro tipo de hardware) desde JavaScript. Está desarrollado para suavizar las diferencias entre plataformas lo máximo posible, consiguiendo que los proyectos que se realicen sean muy portables ya que pueden ejecutarse en cualquier entorno nativo que contengo el SDK de Expo. También proporciona componentes de interfaz de usuario para manejar una variedad de casos de uso que casi todas las aplicaciones móviles tienen que cubrir pero no están construidas en el núcleo de React Native, por ejemplo, iconos, vistas borrosas (aquellas que aparecen de fondo para que el usuario se centre más en los elementos que están superpuesto a la imagen), etc.

  Puntos positivos:

  - Curva de aprendizaje fácil.

  Puntos negativos:

  - En el momento del desarrollo de este proyecto solo tiene soporte para iOS con ARKit.
  - Al basarse en React Native que no tiene versión estable provoca problemas con las versiones de dependencias.
  - Configuraciones tediosas y largas compilaciones.

2.2. Fuentes de información

Como nuestra aplicación muestra información sobre las películas que se recomiendan y que los usuarios han guardado, hemos tenido que investigar de qué fuentes de información podríamos obtener estos datos tan relevantes. Con la finalidad de extraer estos datos, guardarlos en contenedores de datos y por último, representarlos ante los usuarios.

- FilmAffinity[8]: es una de las páginas webs más relevantes para obtener información sobre películas y series, además de críticas y valoraciones de profesionales y sugerencias personalizadas. Esta página fue creada con el objetivo de ser un sistema de recomendación, siendo un factor que nos beneficia ya que podemos estar seguros de que sus datos nos servirán para realizar recomendaciones a los usuarios. Por otro lado, la página está en dos idiomas, español e inglés, otro factor a favor ya que nuestra aplicación muestra los datos en inglés. Sin embargo, al ser una página en la que se realizan críticas serias y profesionales, las valoraciones suelen ser bajas.

- MovieLens[18]: es una fuente de información, quizás menos conocida que FilmAffinity y que IMDB pero que ofrece sugerencias personalizadas y la posibilidad de valorar. Se trata de un sistema de recomendación basado en una técnica de recomendación llamada filtrado colaborativo[28], técnica candidata a usar en nuestra aplicación y de la que hablaremos posteriormente. Los datos que nos proporciona están en inglés, valiéndonos perfectamente para nuestra aplicación.

- IMDB[11]: se trata de una base de datos online donde se guarda información sobre películas y series. Es una de las páginas más usadas a nivel mundial en estos aspectos. Actualmente, le pertenece a una de las empresas que más clouds de almacenamiento tiene en el mundo, Amazon, por lo que podemos asegurar su fiabilidad. El idioma de la página también es el inglés.

- Rotten Tomatoes[29]: es una web donde usuarios y críticos profesionales puntúan películas y espectáculos televisivos. La puntuación es representada con tomates frescos si

el 60 % de las críticas son positivas y si es menor de ese porcentaje es representada con un tomate podrido explotando. Existen certificados de una distinción especial si además cumplen más requisitos:

- Una puntuación constante en el Tomatómetro de 75 % o más.
- Al menos cinco reseñas de los mejores críticos.
- Las películas en "wide release" deben tener un mínimo de 80 revisiones.
- Las películas en "limited release" deben tener un mínimo de 40 revisiones.
- Solo las temporadas individuales de un programa de televisión son certificables, y cada uno ha de tener un mínimo de 20 revisiones.

Estos certificados además de que tienen que ser valorados por un grupo de jurados, pueden además ser retirados una vez que esta película o serie no sea capaz de mantener los requisitos.

Finalmente, hemos elegido IMDB como fuente de información para nuestra aplicación, ya que es una de las páginas webs más conocidas mundialmente. Su fiabilidad está asegurada y además estamos más familiarizados con ella ya que es la que usamos normalmente en nuestro día a día.

2.3. Técnicas de recomendación

Comenzaremos esta sección definiendo qué es un sistema de recomendación y para qué se usan. Los sistemas de recomendación (SRs)[27] son herramientas software y técnicas que proporcionan sugerencias de productos útiles para un usuario. Las sugerencias están relacionadas a varios procesos de toma de decisiones como pueden ser productos que comprar, qué música escuchar, o que noticias online podrías leer. Un SR normalmente se centra en un tipo específico de producto (por ejemplo, CDs o noticias) y de acuerdo a su diseño, su interfaz gráfica de usuario, y el núcleo de la técnica de recomendación usada para generar

recomendaciones están todas personalizadas para proveer de sugerencias útiles y efectivas para ese tipo específico de objeto.

Hoy en día, con toda la información que dejamos en internet, los sistemas de recomendación comienzan a ganar popularidad. Muchas empresas mundialmente conocidas nos ofrecen recomendaciones de sus productos en base a un análisis de nuestros datos y comportamientos. Por ejemplo Youtube, que registra los últimos videos que hemos visto y nos recomienda vídeos similares, basándose en el contenido de dichos vídeos. Otro caso es Netflix, que recomienda películas en base a factores como las interacciones de los usuarios en la página web o de usuarios con gustos similares[23]. Es una de las empresas pioneras en el sector de las recomendaciones junto con Amazon. Netflix además tiene numerosas técnicas de recomendación debido a un concurso que hizo en 2009, ofreciendo un millón de dólares a quienes mejor optimizaran sus sistemas de recomendación[22].

Los sistemas de recomendación se pueden dividir según su forma de realizar las recomendaciones:

- Filtrado basado en el contenido[26]: al usuario se le recomendará aquella información que le ha interesado en el pasado, es decir, se le mostrarán elementos similares, independientemente de lo que opinen otros usuarios. Se basa en contenido de los objetos, se analiza el objeto para sacar características relevantes de ello, y a partir de las características extraídas, recomendar otros objetos con características similares.

- Filtrado Demográfico[19]: al contrario que el filtrado basado en el contenido que se basa en los objetos, este se basa en el usuario, se realiza en función de las características que tienen los usuarios, como edad, sexo, profesión, etc. Ya que se asume que usuarios que, por ejemplo, han nacido en la misma década van a tener preferencias similares frente a los de otras décadas. O que a los profesores de matemáticas les gustan más los números que a los profesores de filología.

- Filtrado Colaborativo[28]: esta técnica busca similitudes de preferencias entre usuarios,

y recomienda objetos que le hayan gustado a otros usuarios similares. La determinación de usuarios con preferencias similares se basa en las valoraciones que hayan dado los usuarios a los objetos.

- Filtrado Híbrido[20]: Mezcla las tres técnicas descritas anteriormente, aprovechando los aspectos positivos que ofrece cada una. Incluso se combina con técnicas de inteligencia artificial como la lógica borrosa o la computación evolutiva.

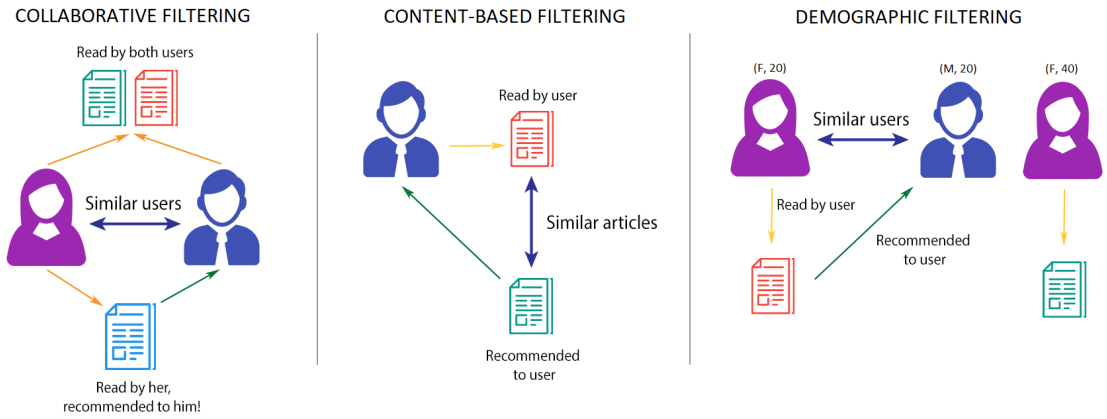

Figura 2.7: *Tipos de sistemas de recomendación*[4]

En nuestra aplicación, los objetos serán las películas que almacenaremos en nuestro contenedor de datos. Nos basaremos en la técnica de filtrado colaborativo, ya que permitimos a los usuarios la posibilidad de interactuar en el sistema de recomendación valorando las películas que les gusten. Otro motivo es que, tanto el filtrado basado en el contenido como el filtrado demográfico, necesitan retener información extra sobre los usuarios, como los objetos, representados en forma de etiquetas, para finalmente llevar a cabo una recomendación.

Para la implementación de los sistemas de recomendación hemos encontrados tres librerías:

1. Mahout: Se trata de una librería que abarca la mayoría de necesidades para la implementación del sistema de recomendación.

2. Lenskit: Es otra librería de Java para la implementación de sistemas de recomendación.

3. LibRec: Se trata de una librería que está subida en un repositorio de GitHub.

Al final nos hemos decantado por Mahout, ya que aporta una documentación más completa. Otro de los motivos es que ofrece clases para realizar la conexión directamente con la base de datos sin tener que leer desde ficheros csv. Por otro lado, en el testeo de Lenskit, cuando intentábamos reproducir el ejemplo que ofrecían, nos encontramos que algunas funciones estaban obsoletas y, por tanto, no nos transmitía mucha confianza. Por último, Mahout destaca por ser más fácil de usar y minimizar nuestro esfuerzo en aprender esta nueva tecnología.

2.4. Conclusiones

En este capítulo hemos podido conocer de primera mano la tecnología que representa el grueso de nuestra aplicación, la Realidad Aumentada. Se han descrito los distintos tipos de librerías que hemos investigado para su implementación, así como las diferentes páginas web de las que podríamos obtener la información necesaria para nutrir a nuestra aplicación y a nuestros usuarios. Finalmente hemos incluido un breve resumen de las técnicas de recomendación que más se ajustan a nuestra aplicación. En los próximos capítulos se tratará más a fondo este tema, cuando se haya descrito con más detalle nuestra aplicación.

Capítulo 3

Diseño de la aplicación

En este capítulo pretendemos explicar cómo hemos diseñado la aplicación. Comenzaremos realizando un análisis de la competencia mediante el cual analizaremos aplicaciones que ofrezcan funcionalidades parecidas a las que nuestra aplicación ofrece. Continuaremos detallando una serie de escenarios reales en los que nuestra aplicación puede ser usada, a partir de estos escenarios extraeremos los requisitos funcionales. Finalmente, detallaremos las interfaces de usuario que hemos implementado.

3.1. Análisis de la competencia

Consideramos que la aplicación que hemos diseñado consta de dos funcionalidades claramente diferenciadas:

1. Realidad aumentada.

2. Recomendación y gestión de películas.

Además, pensamos que existen distintos tipos de competencia dependiendo de cada una de ellas.

- Respecto a la funcionalidad de recomendación y gestión de películas, compite claramente con aplicaciones como IMDB[11] o MovieBase[17]. Estas aplicaciones proporcionan, herramientas para guardar películas y recomendar éstas a los usuarios en función de

sus gustos. Estas funcionalidades son muy parecidas a las que nosotros hemos decidido proporcionar a nuestros usuarios, con la diferencia que nuestra aplicación, además de las funcionalidades anteriormente mencionadas, permite realizar más acciones sobre las películas como crear planes con amigos.

- A diferencia de la funcionalidad de recomendación y gestión en donde hay una gran variedad de aplicaciones que ofrecen servicios parecidos a los nuestros, únicamente hemos encontrado una aplicación (Paramount AR+[2]) que ofrezca servicios parecidos a los nuestros relacionados con Realidad Aumentada. Según las especificaciones de la aplicación permite identificar pósteres y mostrar información sobre la película detectada, servicio muy parecido al que nosotros hemos proporcionado.

Para intentar alejarnos de esta competencia, hemos decidido unir ambas funcionalidades en una aplicación y, además, proporcionar al usuario más funcionalidades como escanear una aplicación con Realidad Aumentada, ver su información y añadirla a un plan con amigos para ver dicha película. Para funcionalidades como ésta y otras en las que unimos la Realidad Aumentada con la recomendación de películas y gestión de planes no hemos encontrado actualmente ninguna aplicación que proporcione estos servicios.

Además, la gestión de amigos con Realidad Aumentada es una funcionalidad que, tras nuestro análisis de la competencia, solamente la posee nuestra aplicación. Se basa en enfocar con la cámara a un usuario de la aplicación y que mediante Realidad Aumentada muestre visualmente la información de los tres planes activos que tiene este usuario, una vez ha sido añadido como amigo, que más podrían gustarme y cuanto se estima que me gustaría ese plan.

3.2. Escenarios de uso

Antes de decidir como implementaríamos nuestra aplicación decidimos que sería importante establecer una serie de escenarios en los que podría ser usada. Tras esto pudimos

ponernos de acuerdo en qué funcionalidades incluiríamos y que éstas se ajustasen a las necesidades de los usuarios. Además, estos escenarios deberían usar la Realidad Aumentada para que les aporte valor, ya que, como hemos expuesto anteriormente en el Capítulo 2, el grueso de nuestra aplicación se centra en esta tecnología. Los escenarios de uso que vamos a describir a continuación son distintas situaciones del mundo real en las que los futuros usuarios de nuestra aplicación se podrían encontrar y como nuestra aplicación ayudaría a afrontar dichos escenarios. De esta manera, analizando las situaciones que se podrían dar en un futuro podemos extraer funcionalidades.

Escenario 1. Reconociendo carteles de películas. Te apetece ir al cine y encuentras el cartel de una película que te interesa ver en una revista. Sacas tu móvil, abres la aplicación y la escaneas. Con la Realidad Aumentada serás capaz de ver la valoración de la película, acceder a su tráiler en Youtube, ver la información referente a la película (título, director, duración, sinopsis) y guardarla como favorita.

Escenario 2. Reconociendo usuarios. Has quedado con un amigo y le convences de que la aplicación es muy útil, se descarga la aplicación y le quieres añadir como amigo para realizar futuros planes. Abres la aplicación y le escaneas. Con la Realidad Aumentada podrás añadirle como amigo.

Escenario 3. Obtener recomendación. Estás planeando con un amigo ver una película, sin embargo, no conseguís poneros de acuerdo en qué película ver. Abres la aplicación y escaneas a tu amigo. Mediante Realidad Aumentada podrás obtener la información de las películas que más os podrían gustar a los dos.

Escenario 4. Quedar con amigos para ver una película. Una vez que has guardado una película porque te interesa ir a verla con alguien, ya sea que quieras verla en el cine o en tu casa, sea fin de semana o no, la aplicación te facilitará quedar con las personas que estén

interesadas en verla, además de que la aplicación supondrá un apoyo para que los usuarios tomen sus decisiones, mostrándoles cuánta afinidad a dicha película tienen.

Escenario 5. Usar la aplicación sin la Realidad Aumentada. Te apetece iniciar un plan para ir a ver una película con unos amigos, sin embargo, en ese momento no tienes la película delante para poderla escanear.La aplicación te permitirá buscar planes, crearlos, borrarlos y unirte a ellos, buscar películas y valorarlas y buscar usuarios y añadirlos sin hacer uso de la Realidad Aumentada.

3.3. Requisitos funcionales

En la siguiente sección describiremos los requisitos funcionales de nuestra aplicación. Con estos requisitos pretendemos mostrar los servicios que nuestra aplicación proporcionará. Para poder entender mejor estos servicios que proporcionaremos, hemos decidido agrupar la aplicación en cuatro grandes grupos: planes, películas, usuarios y recomendaciones.

Planes:

Cuando hablamos de un plan, nos referimos a un elemento creado a partir de una película. Un plan para un usuario significa que ese usuario tiene interés en ver la película con la que se creó dicho plan. Los planes están compuestos por una película, un usuario creador del plan, y los usuarios que se hayan unido.

Funciones:

1. Crear plan: un usuario crea un plan a partir de una película que ha guardado, con una descripción, fecha, hora y lugar determinados.

2. Unirse a un plan: solo podrán unirse los amigos del usuario creador del plan.

3. Salirse de un plan: una vez te has unido al plan puedes salir del mismo.

4. Eliminar un plan: solo puede eliminar el plan la persona que lo ha creado.

5. Información de un plan: la película que se quiere ver, los usuarios que se han unido, el usuario que lo ha creado, la fecha, el lugar, una descripción y la hora.

Películas guardadas:

Las películas guardadas son aquellas que el usuario ha decidido guardar para ver su información más tarde o para posteriormente añadirlas a un plan.

Funciones:

1. Guardar película: cuando un usuario reconoce una película con la cámara, se le permitirá guardar la película como favorita. También podrá guardar la película si la busca desde la aplicación sin necesidad de la Realidad Aumentada.

2. Información de una película: un usuario puede ver la información de una película que haya guardado. Podrá ver una pequeña sinopsis, el género, el director, su valoración y su tráiler en Youtube.

3. Valorar una película: también se le permite a un usuario valorar una película a su gusto.

Usuarios:

Usuario es toda aquella persona que se haya registrado en la aplicación y se haya creado un perfil.

Funciones:

1. Añadir como amigo: cuando un usuario reconoce a otro con la cámara se le permitirá añadirle como amigo. También podrá hacerlo si le busca desde la aplicación sin necesidad de la Realidad Aumentada.

2. Eliminar un amigo: existe la posibilidad de eliminar a un usuario de tu lista de amigos.

3. Información de un usuario: un usuario puede ver la información de otro usuario, como su nombre y su foto de perfil. Si son amigos podrá ver además 3 planes que le interesan

a ambos.

Recomendaciones:

Aquellas películas o planes que son afines a un cierto usuario de la aplicación según sus gustos.

Funciones:

1. Recomendar películas: al usuario de la aplicación se le recomendarán películas que se ajusten a sus gustos para que le sea más fácil tomar una decisión.

2. Recomendar planes: cuando un usuario quiere ver una película con un amigo, se da la opción de que, enfocando al amigo mediante Realidad Aumentada pueda ver los planes que tiene su amigo y cuanto se estima que le podrían gustar. De esta forma el usuario puede hacerse una idea de cómo de afín es a los planes de su amigo y si encaja en alguno.

3.4. Interfaz de usuario

A continuación, describiremos las distintas interfaces de usuario que hemos realizado. En ellas hemos recogido las funcionalidades anteriormente para tratar de atender a los escenarios planteados inicialmente.
Hemos dividido nuestra aplicación en 3 vistas diferentes: la interfaz de Mis planes, la de Recomendaciones y la de Mis películas. Además, poseemos una serie de interfaces de Realidad Aumentada que mostraremos. A continuación hablaremos de las interfaces de usuario más relevantes que hemos diseñado:

3.4.1. Interfaz principal

Tenemos una barra en la parte inferior para movernos por las 3 secciones de nuestra aplicación: mis planes, recomendaciones y mis películas. Cada sección incluye un icono para

apoyar a la representación. Además, encontramos un botón con un símbolo de una cámara en todas las secciones que nos permitirá acceder a la interfaz de Realidad Aumentada.

3.4.2. Interfaz de mis planes

El objetivo de esta vista es el de mostrar los planes públicos que existen, es decir, aquellos en los que estás o hayas creado, indicando con una imagen de fondo la película para la que se creó el plan, el usuario que creó dicho plan y los usuarios que se han unido. Podemos observar como cada elemento representa un plan, con el fondo siendo el cartel de la película, la fecha en la que tendrá lugar el plan en la esquina superior izquierda, justo debajo el título del plan, arriba a la derecha el número del plan (algo simple pero que nos indica claramente lo que representa este elemento) y en la parte inferior aparecen las fotos de los usuarios que se han unido al plan. Todo esto podemos verlo en la Figura 3.1.

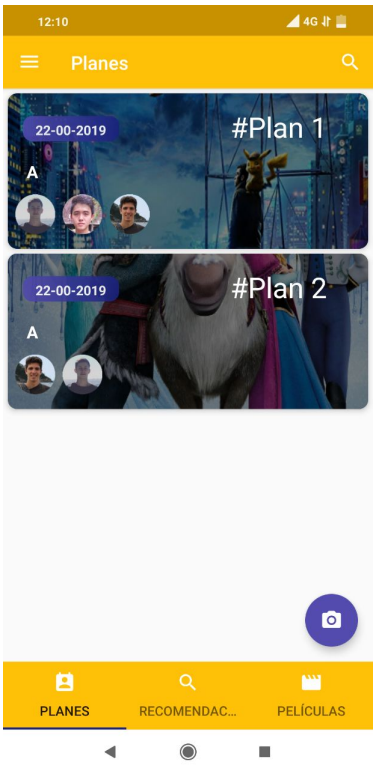

Figura 3.1: *Lista de planes*

3.4.3. Interfaz de información de un plan

Figura 3.2: *Información del plan*

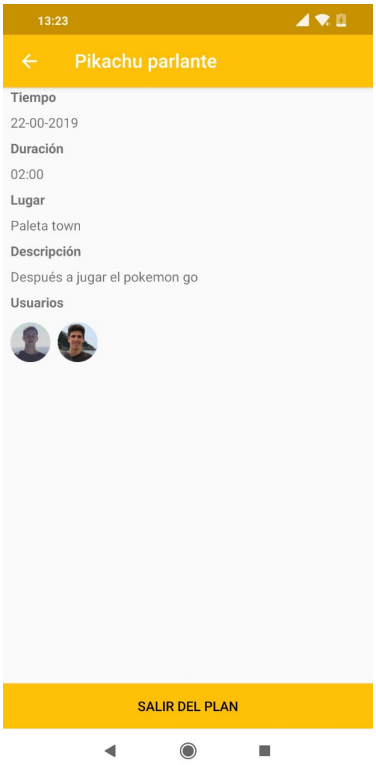

Figura 3.3: *Información del plan*

Esta interfaz aparece cuando pulsamos en un plan, (Figura 3.2). Nos aparecerá en grande la imagen de la película que se quiere ir a ver con dicho plan, junto con información relevante de dicho plan, que podemos ver en la Figura 3.3:

- Fecha y hora del plan.
- Lugar: localización que haya puesto el creador del plan para ver la película.
- Descripción: breves anotaciones características que haya escrito el creador sobre el plan.
- Usuarios unidos: imágenes de los usuarios que se han unido al plan.

La información de cada plan la introduce el usuario mediante un formulario muy simple cuando crea un plan con una película. Además, nos aparece un botón en la parte inferior con el texto UNIRSE AL PLAN o SALIR DEL PLAN, según estemos o no ya dentro del plan. También en la parte inferior derecha de la imagen de la película tenemos un botón que nos redirige a la interfaz de información de dicha película. En la parte superior derecha nos aparecería un icono de una basura, lo que nos permitirá borrar el plan, si somos el creador.

3.4.4. Interfaz Mis películas

Para mostrar las películas guardadas decidimos usar una cuadrícula que muestre solamente los carteles de las películas, es una interfaz simple pero muy visual. Para acceder a la información de cada película simplemente debemos pulsar en uno de los carteles.

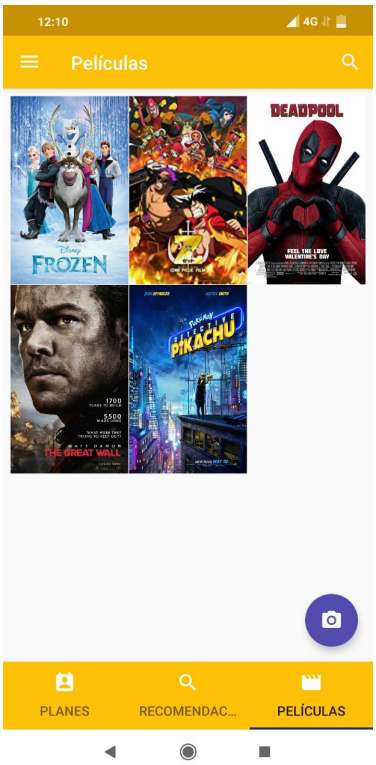

Figura 3.4: *Lista de películas guardadas*

En la Figura 3.4 podemos observar cómo es la interfaz. En este caso el usuario solo ha guardado 2 películas.

3.4.5. Interfaz de información de una película

Tras pulsar en una de las películas que hemos guardado nos aparecerá dicha interfaz. Esta vista nos permite ver en primer plano el cartel de la película y la información correspondiente a la misma, como puede ser la sinopsis de la película, el género y el nombre del director, podemos observarlo en la Figura 3.5 y la Figura 3.6. Además, la interfaz cuenta con un indicador circular en el que se muestra, haciendo uso de colores, la nota de la película. Si pulsamos este, nos permite valorar la película mediante un deslizador. Si hacemos scroll hacia abajo, la imagen de la película se irá ocultando para ofrecernos una mejor visión de la información de la película. En la parte inferior se nos presentan dos botones, uno para crear un plan con la película con el texto: AÑADIR AL PLAN. El otro botón con el símbolo de reproducir un vídeo, nos llevará al tráiler de la película en Youtube. Arriba a la derecha observamos el icono de un corazón, lo que nos permitirá quitar esta película de nuestras favoritas y ya no aparecerá en la lista de películas guardadas.

Figura 3.5: *Información de la película*

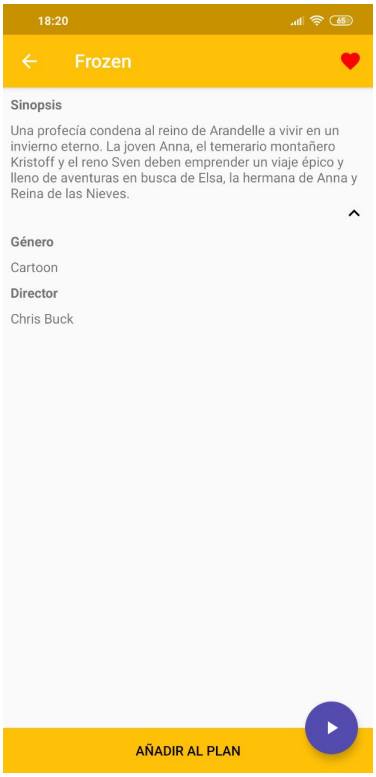

Figura 3.6: *Información de la película*

3.4.6. Interfaz Recomendaciones

Esta interfaz cuenta con tres secciones de recomendaciones (películas recomendadas específicamente para el usuario, películas populares y en estreno), cada una de estas secciones tiene un conjunto de carteles de películas que se le recomiendan al usuario. Cada sección presenta scroll horizontal para poder visualizar todas las películas que se le recomiendan en dicha sección, (ver en las Figuras 3.7 y 3.8). La primera sección son las películas recomendadas al usuario según sus gustos, la segunda representa las películas más populares y las películas que están siendo estrenadas. Al pulsar en cada una de ellas accederemos a la información de cada película (Figura 3.5) pudiendo guardarla como favorita, si no lo hemos hecho ya.

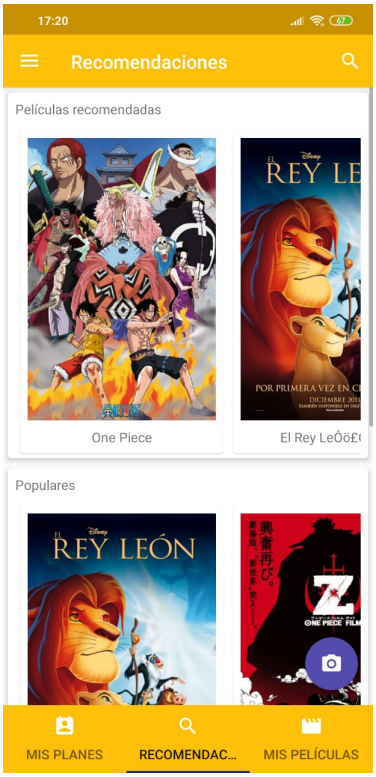

Figura 3.7: *Recomendaciones de películas*

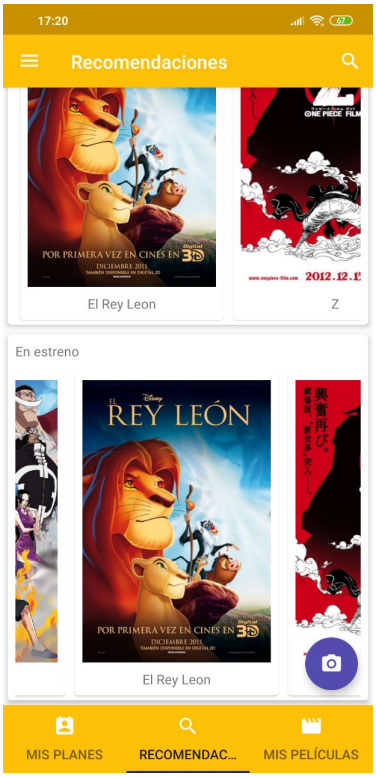

Figura 3.8: *Recomendaciones de películas*

3.4.7. Interfaz Usuario

Al pulsar la foto de un amigo en cualquier lugar de la aplicación, podremos acceder a una vista de ese usuario. En esta interfaz podremos ver la imagen del usuario, el nombre de este, los planes que tiene actualmente activos y el estado de nuestra relación con él. Este estado puede ser:

1. Enviar petición de amistad. No hemos interactuado con este usuario y tenemos la posibilidad de enviarle una petición de amistad.

2. Esperando respuesta. Le hemos enviado en algún momento una petición de amistad y estamos esperando su respuesta.

3. Aceptar o rechazar. El usuario en el que estamos nos ha enviado una petición de amistad y tenemos la posibilidad de aceptarlo o rechazarlo.

4. Eliminar amigo. El usuario que estamos viendo es nuestro amigo y tenemos la opción de dejar de serlo.

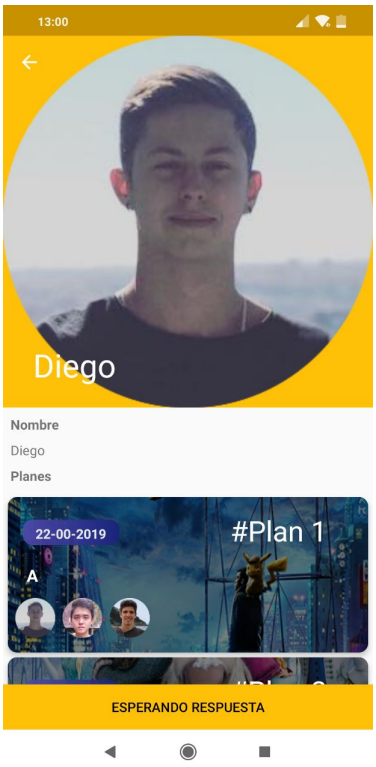

Figura 3.9: *Vista de usuario*

3.4.8. Interfaz Lista de amigos

Hemos decidido incorporar esta vista a nuestra aplicación ya que, es una forma muy intuitiva para que el usuario pueda ver una lista de sus amigos y usuarios a los que ha enviado una petición de amistad.

La vista es una lista en la que se muestra la foto del amigo o usuario al que se le he enviado la petición de amistad y su nombre.

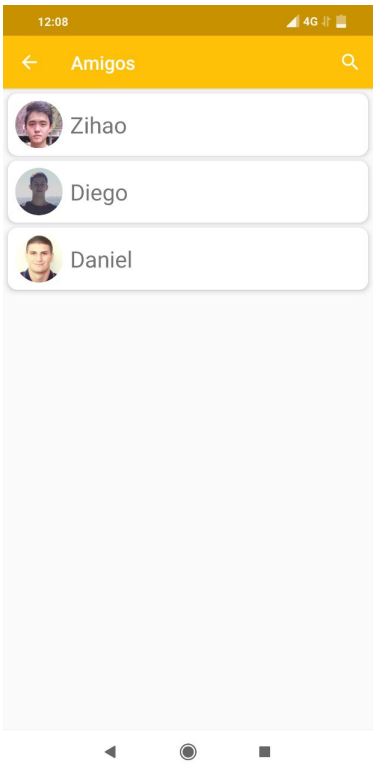

Figura 3.10: *Lista de amigos*

3.4.9. Búsquedas en las vistas

Para facilitar al usuario el uso de la aplicación cuando haya muchos datos en esta, hemos implementado un buscador para cada una de las siguientes vistas:

1. En la vista de Mis planes para permitir al usuario encontrar un plan determinado.
2. En la vista de Recomendaciones para encontrar una película determinada.
3. En la vista de Mis películas para, al igual que en la vista anterior, encontrar una película determinada.

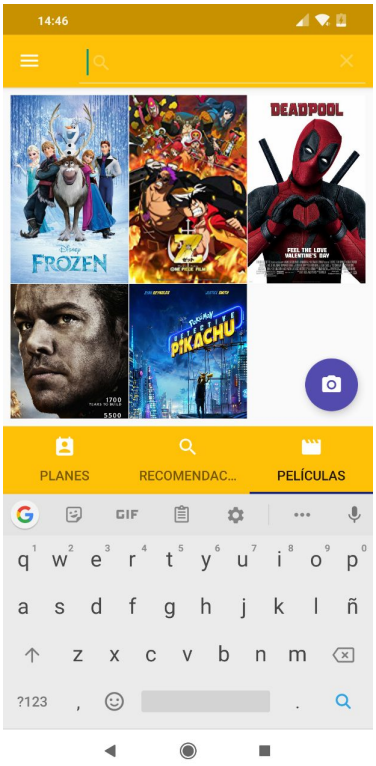

Figura 3.11: *Búsqueda de Mis películas*

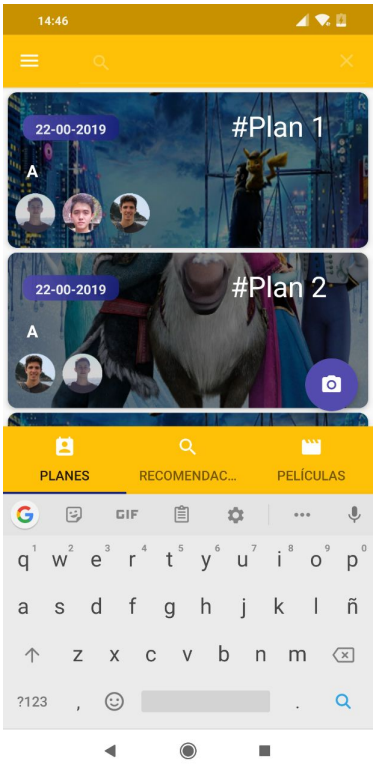

Figura 3.12: *Búsqueda de Mis planes*

3.4.10. Interfaz de Realidad Aumentada principal

Una interfaz muy simple en la que tenemos una línea horizontal que se mueve de arriba hacia abajo y viceversa en la pantalla como si se tratara de un escáner, para que el usuario pueda entender que debe escanear una imagen, como podemos ver en la Figura 3.13.

Figura 3.13: *Escáner en Realidad Aumentada*

3.4.11. Interfaz de Realidad Aumentada tras reconocer un cartel de película

En esta interfaz destacamos los distintos componentes que aparecen al usar la cámara de nuestro dispositivo móvil y reconocer el cartel de una película. Cuando reconocemos un cartel, sobre la imagen de la película nos aparecerá en la parte superior una valoración numérica de la película, la cual la hemos sacado de IMDB. justo en el medio de la imagen encontramos el icono típico de Youtube que, al pulsarlo, nos abrirá nuestra aplicación de Youtube para que el usuario pueda ver el tráiler de esta película. En la esquina inferior derecha encontramos otro icono que al ser pulsado lo que hará será desplegar una serie de botones con distinta funcionalidad, uno de ellos nos redirige a la información de la película en IMDB, que sería el icono representado por una letra i, el otro botón representado por una mano con un pulgar hacia arriba nos permitirá guardar la película en nuestra lista de

películas guardadas, para posteriormente acceder a su información, valorarla o crear un plan con ella. Todo esto lo podemos observar en la Figura 3.14.

Figura 3.14: *Realidad aumentada tras reconocer una película*

3.4.12. Interfaz de Realidad Aumentada al reconocer a un usuario que no es amigo

Se trata de una interfaz sencilla en donde a la persona que está usando la cámara, al enfocar a la imagen de otro usuario, se le proporciona el nombre del usuario al que está enfocando (siempre que el usuario enfocado esté registrado) junto a un botón de añadir amigo, como podemos ver en la Figura 3.15.

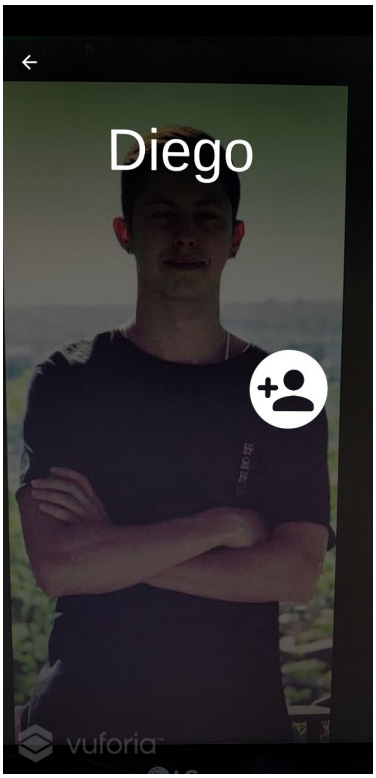

Figura 3.15: *Realidad aumentada tras reconocer a un usuario que no es amigo*

3.4.13. Interfaz de Realidad Aumentada al reconocer a un usuario que sí es amigo

Esta interfaz aparece al enfocar con la cámara a un usuario que es amigo del que enfoca, es una interfaz más compleja que la anteriormente descrita.

Cuando un usuario enfoca a otro que es amigo, se proporciona la opción de incorporarse a uno de los planes del amigo. Para ello, se le muestra una interfaz con los tres planes que, según nuestro algoritmo de recomendación, más pueden interesarle de los que su amigo tiene. En caso de que haya menos de tres planes para mostrar, mostrará esos.

La interfaz proporciona la imagen de las películas de los tres planes. Además, mediante una interfaz dinámica podemos movernos de un plan a otro. De esta forma podemos ver qué usuarios hay en cada plan.

Al usuario al que se le están mostrando los planes de su amigo que el algoritmo le

recomienda, además de los carteles de las películas, se le muestra información de los usuarios que se encuentran en cada uno de los planes.

Esta información la mostramos colocando sobre el cartel las imágenes de los usuarios que están apuntados al plan. Además, alrededor de la imagen de cada usuario se sitúa un indicador que muestra cuanto calcula el algoritmo de recomendación que le va a gustar a cada usuario el plan del que forman parte. De esta manera, el usuario que está pensando en apuntarse al plan cuenta con toda la información posible para decidir si finalmente hacerlo o no.

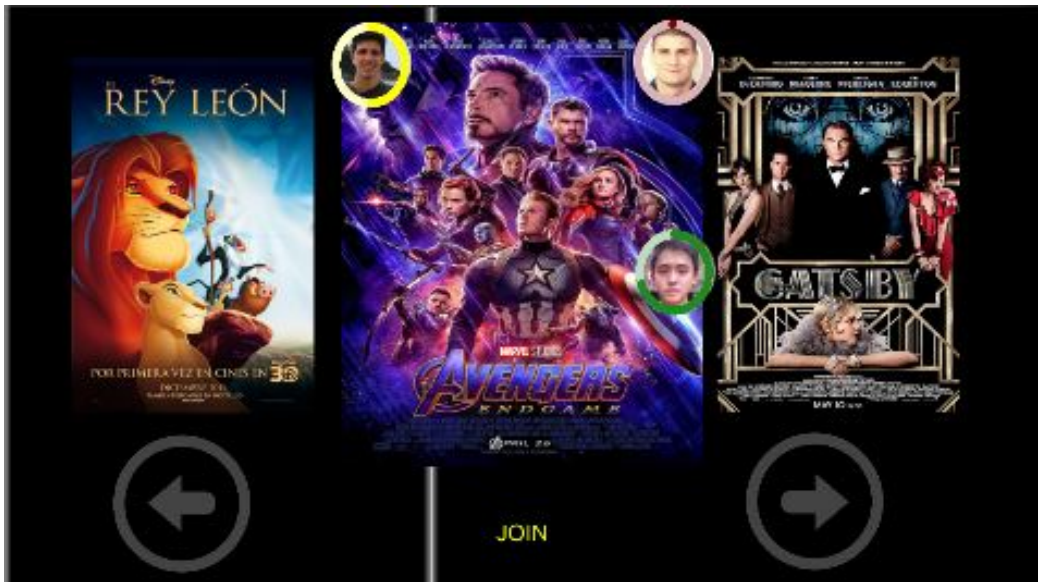

Figura 3.16: *Vista encargada de recomendar planes en RA al enfocar a un amigo*

Como se puede ver en la Figura 3.16, en la interfaz aparecen tres películas, esto se debe a que el recomendador ha encontrado tres planes que al usuario le pueden gustar dentro de los que el amigo al que está apuntando tiene. Únicamente se muestra la información relacionada con los usuarios del plan situado en el centro para no saturar la interfaz. Mediante los botones de izquierda y derecha el usuario puede moverse entre planes para poder ver la información necesaria de cada plan y elegir aquel que más le guste.

La información visual que aparece relacionada con los usuarios representa:

- Arriba a la izquierda, la foto del usuario que está valorando unirse al plan, es decir,

el que ha enfocado a su amigo para ver en cuál de sus planes se puede unir. Además, aparece una representación gráfica de cuanto estima el algoritmo de recomendación que le puede gustar la película alrededor de su foto.

- Arriba a la derecha el amigo que se encuentra ya en el plan y al que se le está apuntando con la cámara. Además de la representación gráfica de cuanto estima el algoritmo de recomendación que le gusta el plan.

- El resto de las posiciones (hasta 6) identifican al resto de usuarios que se hayan unido a ese plan con sus fotografías y la representación gráfica de cuanto se estima que les gusta el plan del que forman parte.

Además, aparece un botón situado debajo del plan central que permite al usuario que lo pulse unirse al plan. Este botón mostrará una interfaz como la de la Figura 3.2 y 3.3 para que el usuario pueda ver todos los detalles del plan y decidir si se quiere unir.

3.5. Navegación entre interfaces

A continuación, explicaremos el diagrama mostrado en la Figura 3.17. Este diagrama muestra la manera en que se puede navegar entre las distintas interfaces de la aplicación.

1. El usuario al entrar en la aplicación accede a la vista de recomendación (Figura 3.17). Esta vista como hemos explicado anteriormente permite navegar haciendo uso de la barra de navegación inferior, con las vistas de Planes (Figura 3.1) y la vista de Mis películas (Figura 3.4).

2. Presionando cualquiera de los planes mostrados en la lista de planes se accede a la vista de cada plan (Figura 3.2).

3. Presionando alguna de las películas que aparecen en la vista de recomendaciones accedemos a la vista de cada película (Figura 3.5).

4. Al presionar alguna de las películas de la vista de Mis películas se accede a la vista de cada película (Figura 3.5).

5. Desde cualquiera de las vistas citadas en el punto 1 se puede acceder a la Realidad Aumentada haciendo uso del botón habilitado para ello.

6. Al pulsar la imagen de un usuario desde cualquier vista en la aplicación, nos redirigirá a la vista de ese usuario. (Figura 3.9).

7. Una vez estamos en la Realidad Aumentada, dependiendo de a qué decidamos enfocar con la cámara accederemos a una vista u otra:

    a) Al enfocar a un usuario que no es amigo se accederá a la vista en Realidad Aumentada habilitada para añadirlo como amigo. (Figura 3.15).

    b) Al enfocar a un usuario que sí es amigo accederemos a la interfaz en la que podremos unirnos a alguno de los planes de nuestro amigo. (Figura 3.16). Si finalmente decidimos unirnos a uno de sus planes, accederemos a la vista fuera de la Realidad Aumentada habilitada para ello (Figura 3.2).

    c) Al enfocar a una película, podremos ver todos los detalles de esta en Realidad Aumentada. (Figura 3.14). Si decidimos ver más información de la película fuera de la Realidad Aumentada, al presionar el botón habilitado para ello, accederemos a la interfaz de Película (Figura 3.5).

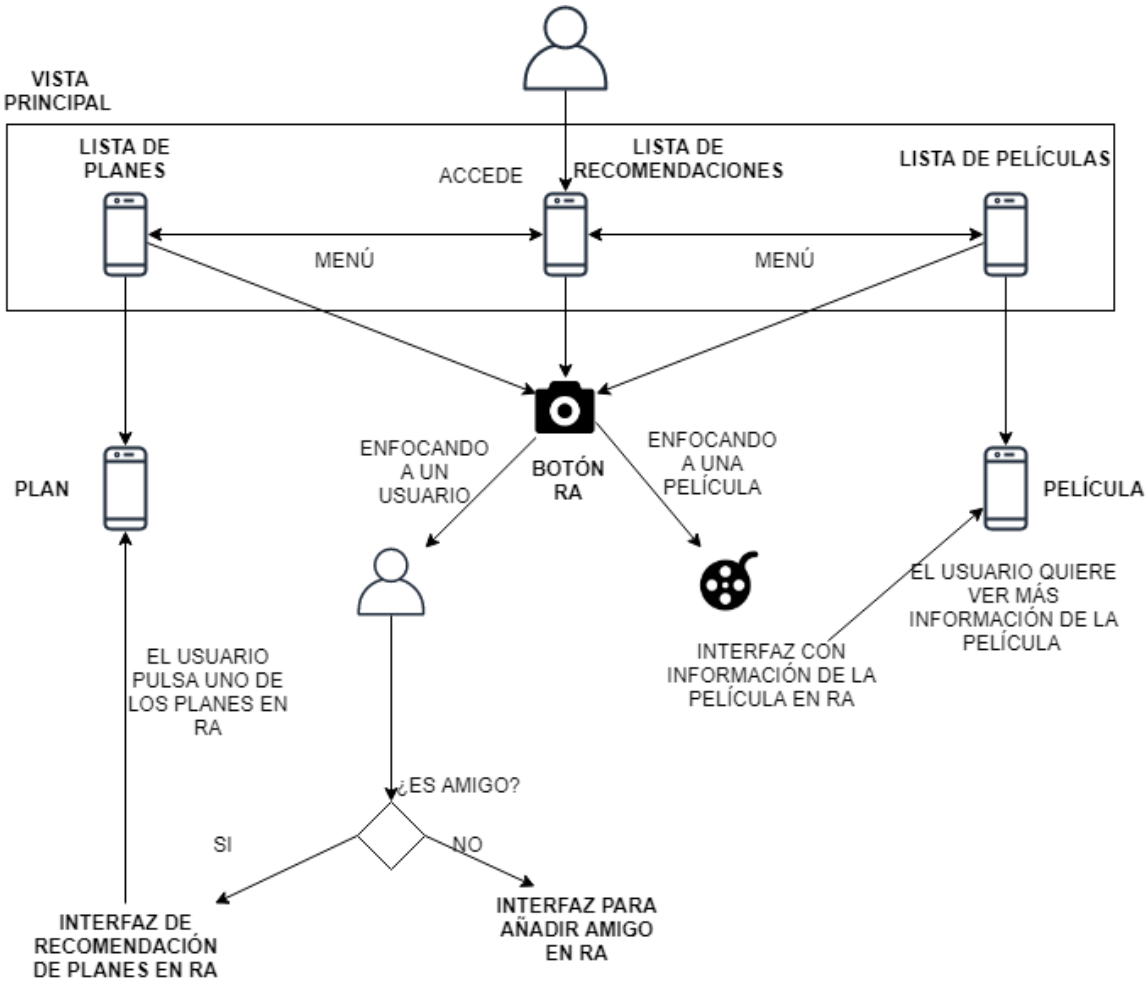

Figura 3.17: *Diagrama que muestra cómo es la navegación entre interfaces en la aplicación*

3.6. Conclusiones

En este capítulo hemos observado los distintos escenarios que hemos planteado para el uso de nuestra aplicación, a partir de ellos hemos sacado una serie de requisitos funcionales que posteriormente han sido mejor especificados tras las descripciones de las interfaces de usuario que hemos diseñado, apoyándonos en imágenes para mostrar visualmente a lo que nos referimos en cada subsección de las interfaces de usuario.

Capítulo 4

Implementación

Nuestra aplicación es una aplicación móvil desarrollada en Android, Unity y Vuforia y con un servidor desarrollado en Java usando Spring. En este capítulo comenzaremos describiendo todos los prototipos de implementación que hemos desarrollado, en los cuáles veremos los pros y contras de las tecnologías que hemos probado y el porqué de nuestra elección de implementar la aplicación con estas tecnologías. Después describiremos cómo hemos implementado nuestra aplicación, desde la arquitectura que hemos seguido, las tecnologías usadas para la parte frontend y la parte backend junto a las dificultades que nos hemos encontrado mientras trabajábamos. Finalmente hablaremos de las herramientas de trabajo utilizadas para facilitarnos el trabajo en equipo. La aplicación FilmAR que se ha construido como solución para este proyecto se encuentra en https://filmar-team.github.io/FilmAR/. Esta aplicación tiene una licencia Apache License 2.0.

4.1. Pruebas de arquitectura

Estas pruebas sirven para conocer si las tecnologías seleccionadas de la Subsección 2.1.1 son viables dentro de nuestro proyecto. Desarrollaremos prototipos con funcionalidades básicas y evaluaremos las partes positivas y negativas de dichas tecnologías. Esto nos permitirá decidir cuales son las mejores opciones para construir el software.

4.1.1. ARCore

Al comenzar la fase de desarrollo de proyectos, pensamos que una de las tecnologías que debíamos investigar y probar debía ser ARCore. Esto se debía a que, la empresa detrás de esta tecnología es Google y esto podría significar que tendríamos más material de consulta y ejemplos en comparación con otras tecnologías de fabricantes con menos recursos. ARCore se encuentra disponible para Java, Unity, Unreal e iOS. Comenzamos realizando el "Quickstart" para Android y posteriormente para Unity. Tras realizar los proyectos propuestos por ARCore, realizamos algún proyecto propio para comprobar si la herramienta se ajustaba a la idea que teníamos para nuestro futuro proyecto. Tras realizar ambos proyectos, concluimos que, aunque ARCore reúne las características necesarias para en un futuro convertirse en una de las tecnologías más importantes en Realidad Aumentada, no íbamos a seleccionarla para nuestro proyecto por las siguientes razones:

- Dispone de mucha documentación para comenzar a usar la herramienta, pero poca para realizar tareas más complejas.
- Resulta muy útil para realizar superposiciones de modelos 3D sobre superficies. Sin embargo, una de las funcionalidades más importantes que nuestra aplicación requería era la interacción con la Realidad Aumentada mediante el uso de botones, imágenes y la carga dinámica de elementos para posicionar en la pantalla de Realidad Aumentada e interaccionar con el usuario. En este sentido ARCore no está, por el momento, tan preparada como otras tecnologías.

4.1.2. Viro Media

El objetivo del prototipo realizado con Viro Media es reconocer imágenes almacenadas en el dispositivo para mostrar texto y objetos virtuales. Además de probar tecnologías de desarrollo móvil web como en este caso React Native para plataformas iOS y Android. Comenzamos construyendo una interfaz sencilla con botones que nos redirigen a la escena de

Realidad Aumentada (ver Figura 4.1 y Figura 4.2). Para esta interfaz utilizamos NativeBase que es una librería que nos permite realizar una aplicación con apariencias de tipo iOS o Android según el dispositivo.

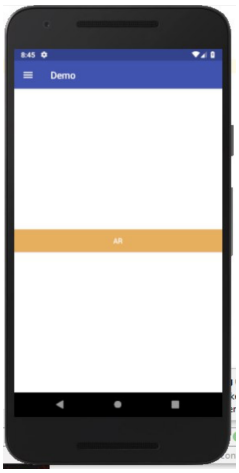

Figura 4.1: *Visualización con NativeBase en Android*

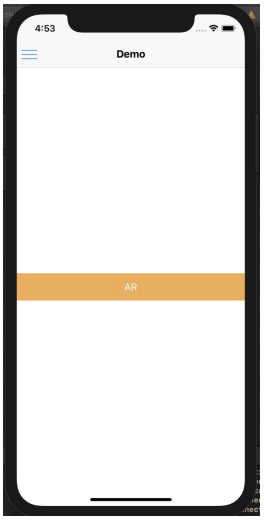

Figura 4.2: *Visualización con NativeBase en iOS*

Para la escena de Realidad Aumentada mostramos texto y al detectar el póster de Pantera Negra, reacciona mostrando una animación de dicho superhéroe saliendo del póster (ver Figura 4.3).

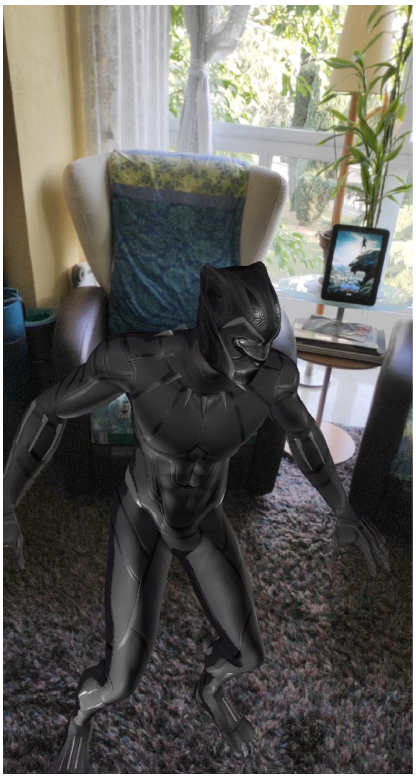

Figura 4.3: *Visualización de RA*

Una de las ventajas que apreciamos fue la facilidad del lenguaje, en este caso Javascript, utilizando la popular librería ReactJS y la buena documentación de Viro Media que hacían que el proceso de codificación fuera agradable.

Uno de los problemas que presentaba este prototipo era que las dependencias de Viro Media entraban en conflicto con las de NativeBase imposibilitándonos la forma de encontrar versiones compatibles. Utilizamos las últimas que, a pesar de lanzar advertencias, funcionaba en el ejemplo realizado.

Otro problema fue la compilación de la aplicación, Viro Media tiene una aplicación para probar lo que desarrollamos conectándose a nuestro ordenador a través de la red. El problema es que algunos recursos, como los iconos que utilizaba NativeBase, no eran descargados, por lo que la mejor forma era probar la versión compilada de iOS y Android.

La forma de compilar la aplicación era un proceso costoso para los ordenadores, lento y

con multitud de problemas según se ampliaban las librerías que se utilizan.

La conclusión que obtuvimos de este prototipo fue que Viro Media y React Native son tecnologías muy prometedoras, pero debido a los problemas surgidos y a que todas sus versiones no eran estables vimos un claro riesgo para el proyecto.

4.1.3. Vuforia + Android

En este prototipo utilizamos la librería nativa de Vuforia para Android para realizar las pruebas de tecnología de reconocimiento de imágenes tanto en local como usando la nube que nos ofrecía Vuforia, para la posterior renderización de objetos y textos.

Las características tecnológicas de este prototipo son las siguientes:

1. La librería de Vuforia para Android está diseñada a muy bajo nivel.
2. Vuforia para dibujar en 3D usa la librería OpenGL (ver Figura 4.5).
3. OpenGL utiliza una serie de espacios donde se van colocando los elementos (ver Figura 4.4):
   a) Local space: Es el espacio local de cada objeto.
   b) World space: Es el mundo donde se encuentran los objetos.
   c) View space: El mundo visto desde la perspectiva de la cámara.
   d) Clip space: Se integra con la pantalla del móvil y, definiendo los límites visibles, se establecen unas coordenadas de rango (-1,-1) - (1,1).

   Las transformaciones de estos espacios se realizan mediante matrices 4x4, en las que la primera fila hace referencia a la coordenada x, la segunda a la coordenada y y la tercera a la coordenada z, mientras que la última columna hace referencia a los desplazamientos de los objetos en esos 3 ejes.

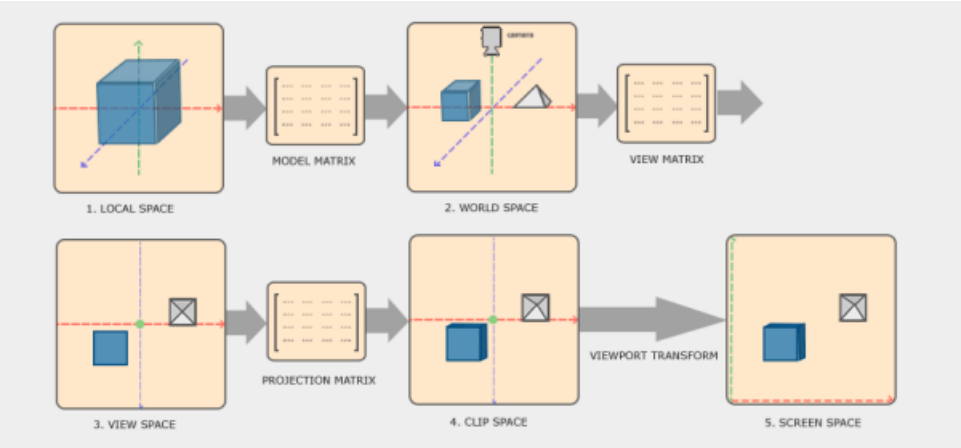

Figura 4.4: *Esquema de los distintos espacios que usa OpenGL*[24]

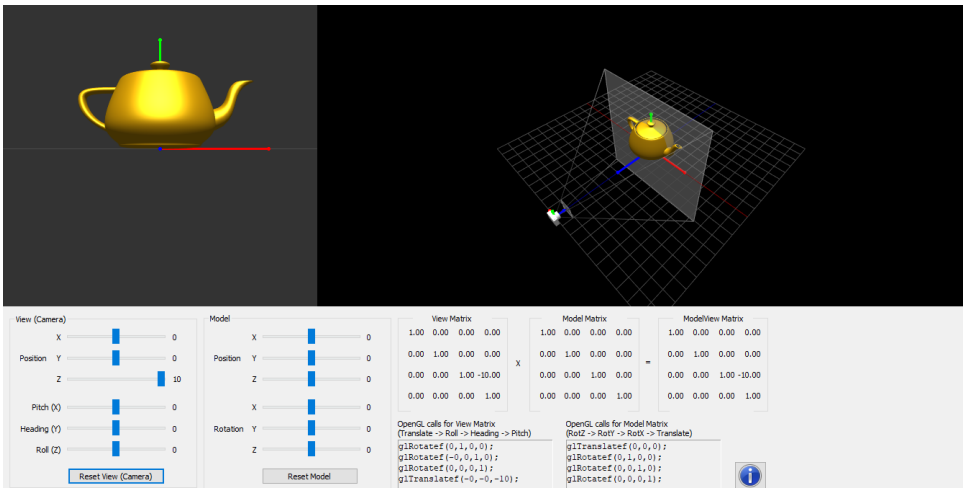

Figura 4.5: *Captura de un ejemplo de un modelo en 3D*

4. En OpenGL es necesario escribir código para que las tarjetas gráficas rendericen el modelo 3D, el lenguaje que se usa es GLSL. Este código de GLSL se escribe en forma de String y se llama a un método que proporciona OpenGL, podemos ver un ejemplo de este código en la Figura 4.6.

```
private String vertexShaderCode =
        // This matrix member variable provides a hook to manipulate
        // the coordinates of the objects that use this vertex shader
        "uniform mat4 uMVPMatrix;" +
        "attribute vec4 vPosition;" +
        "void main() {" +
        // The matrix must be included as a modifier of gl_Position.
        // Note that the uMVPMatrix factor *must be first* in order
        // for the matrix multiplication product to be correct.
        "  gl_Position = uMVPMatrix * vPosition;" +
        "}";
private String fragmentShaderCode =
        "precision mediump float;" +
        //"uniform vec4 vColor;" +
        "void main() {" +
        "  gl_FragColor = vec4(0.2f, 0.7f, 0.9f, 1.0f);" +
        "}";
```

Figura 4.6: *Código en GLSL*

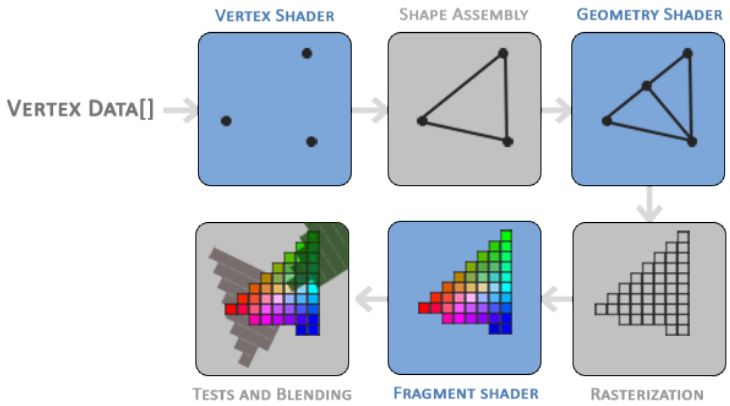

Figura 4.7: *Pipeline de la construcción de un modelo*[25]

Figura 4.8: *Mapa de bits de caracteres usado*

5. Otro aspecto a tener en cuenta es que OpenGL solo nos ofrece lo básico, no nos ofrece métodos para dibujar directamente objetos, sino que hay que seguir un pipeline de procesos para conseguir dibujar algo (ver Figura 4.7). Esto consiste en pasar un array de números (cada tres para definir un punto) a las tarjetas gráficas, establecer triángulos entre los puntos (más arrays de números) definir colores a partir de los puntos (más arrays...), y con el código del shader, ejecutar estos datos.

6. Por último, como sólo ofrece métodos básicos, no hay métodos de escritura de texto, y la forma que encontramos para que funcione fue usar un bitmap con los caracteres (ver Figura 4.8). Este fue el principal motivo de rechazar esta tecnología ya que, dependiendo de la funcionalidad que se desee implementar, es necesario codificar a muy bajo nivel, lo que costaría mucho tiempo y esfuerzo.

4.1.4. Vuforia + Unity

Para realizar este prototipo utilizamos Unity como herramienta básica para realizar la aplicación y Vuforia para dar soporte a la Realidad Aumentada. El prototipo a desarrollar consistió en un modelo 3D de un dragón que aparecía al detectar una imagen que previamente habíamos establecido como "imagen objetivo".

Figura 4.9: *Modelo en 3D que aparecía al detectar la imagen*

Pese a que nadie del equipo había utilizado Unity previamente el resultado fue bastante positivo ya que:

1. Unity resultó ser intuitivo y relativamente fácil en cuanto al aprendizaje de las funcionalidades básicas.

2. Vuforia parecía estar muy probada e incluía de serie muchas funcionalidades.

3. Vuforia tenía la opción de utilizar un Cloud para almacenar las imágenes objetivo.

4.1.5. Vuforia + Unity + Android

Una vez realizado el prototipo en Vuforia con Unity comenzamos a investigar cómo realizar el resto de la aplicación que no requería de Realidad Aumentada. Hasta este momento teníamos claro que Vuforia con Unity era la mejor combinación para realizar la parte de Realidad Aumentada. Sin embargo, tras realizar pruebas desarrollando en Unity interfaces y lógica nos dimos cuenta de que Unity no era igual de intuitivo ni eficaz a la hora de realizar estas tareas que necesitábamos para desarrollar el resto de la aplicación que no necesitaba Realidad Aumentada. Por este motivo intentamos buscar la opción de realizar una aplicación en la que la Realidad Aumentada estuviese diseñada en Unity con Vuforia y el resto de la aplicación en Android (ver Figura 4.10).

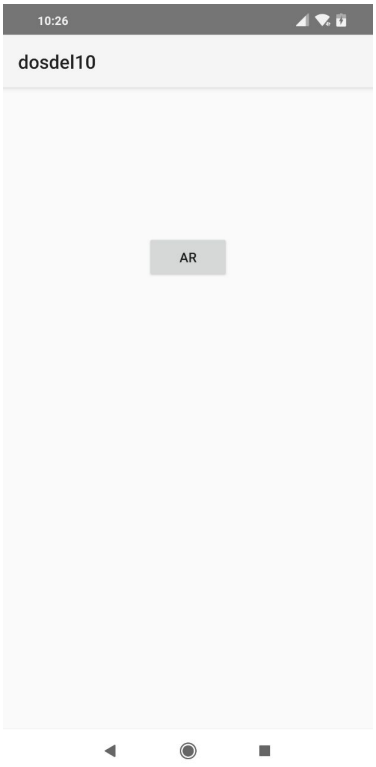

Figura 4.10: *Botón que comunica Android con Unity*

Finalmente conseguimos tener ambos proyectos independientes. La Realidad Aumentada se desarrollaba en Unity con Vuforia y se exportaba a un proyecto Android donde se

encontraba el resto de la aplicación. Esto nos permitía realizar la Realidad Aumentada con la herramienta que tras las primeras tomas de contacto habíamos comprobado que era la mejor (Unity con Vuforia) y del mismo modo realizar el resto de la aplicación con la mejor herramienta para esta parte (Android Studio).

Tras realizar este prototipo, consideramos que estas herramientas podrían ser las que usásemos en la aplicación final puesto que:

1. Como ya habíamos descubierto en el prototipo anterior, Unity era una herramienta muy completa y junto con Vuforia nos proporcionaban todas las herramientas necesarias para cumplir con los casos de uso de Realidad Aumentada que teníamos en mente.

2. Al haber encontrado la forma de combinar Unity y Android no teníamos que renunciar a ninguna de las dos herramientas. Lo que nos permitía explotar las cosas buenas de ambas herramientas.

3. La comunicación entre Unity y Android era relativamente sencilla pese a ser dos proyectos distintos.

4.1.6. Server en Spring

Para la realización de la parte backend de la aplicación explicada a continuación, decidimos incorporar la tecnología de Spring para codificar un servicio web REST en Java y el acceso a datos mediante MySQL. Para el prototipo seguimos un tutorial[14] de 3 partes para crear un servicio web REST que gestionara una simple entidad de contactos de personas, con atributos como el nombre completo del contacto, su número de teléfono o su correo electrónico.

También se investigaron distintas formas de realizar el acceso a la base de datos desde el servidor, nos decantamos por una clase que nos ofrecía implementados los métodos básicos para crear, leer, actualizar y borrar datos. Además de la opción de poder crear nuestros propios métodos.

Para este prototipo decidimos usar MySQL como sistema de gestión de bases de datos relacional, aunque a la hora de probar a levantar nuestro servidor de prueba tuvimos que rehacer este prototipo, además de adaptarlo para que gestionara entidades de películas, y que usara PostgreSQL.

4.1.7. Microservicios

En backend, una de los arquitecturas más usadas hoy en día es aquella basada en microservicios[1][2][5]. Este patrón de arquitectura permite la creación de un servidor gigante formado por pequeños servicios interrelacionados entre sí. Esto hace más fácil el despliegue y apagado de los pequeños servicios ya que no se tiene que detener el proyecto entero cuando se hace una modificación. Se trata de una tecnología compuesta por dos componentes fundamentales:

1. Puerta de enlace (**Gateway**), es el punto de acceso del patrón, donde llegan todas las peticiones HTTP.

2. Registrador de servicios (**Registry**), es el componente que registra todos los servicios. Para organizarlos y distribuirlos.

Al final se consiguió implementar mediante **Eureka** (registrador de servicios) y **Zuul** (puerta de enlace). Sin embargo, se rechazó la implementación en el proyecto pues no era lo bastante grande como para necesitarlo. Para nuestra aplicación nos bastaba con una simple API Rest.

4.2. Arquitectura

En cuanto a la arquitectura usada en nuestra aplicación hemos adjuntado un esquema que podemos observar en la Figura 4.11. Nuestra aplicación móvil estará implementada tanto en Android como en Unity. La aplicación general está en Android, mientras que la parte de Realidad Aumentada está desarrollada en Unity y usa Vuforia. Para el reconocimiento de imágenes con Vuforia usamos un almacenamiento en la nube gratuito. Para la parte del servidor, hemos realizado una API Rest con Spring, usando además una base de datos

relacional en PostgreSQL. La comunicación entre cliente y servidor se realiza mediante peticiones HTTP a nuestro servidor desplegado en Heroku el cual nos devolverá los datos en formato JSON y posteriormente los mapearemos a una clase Java.

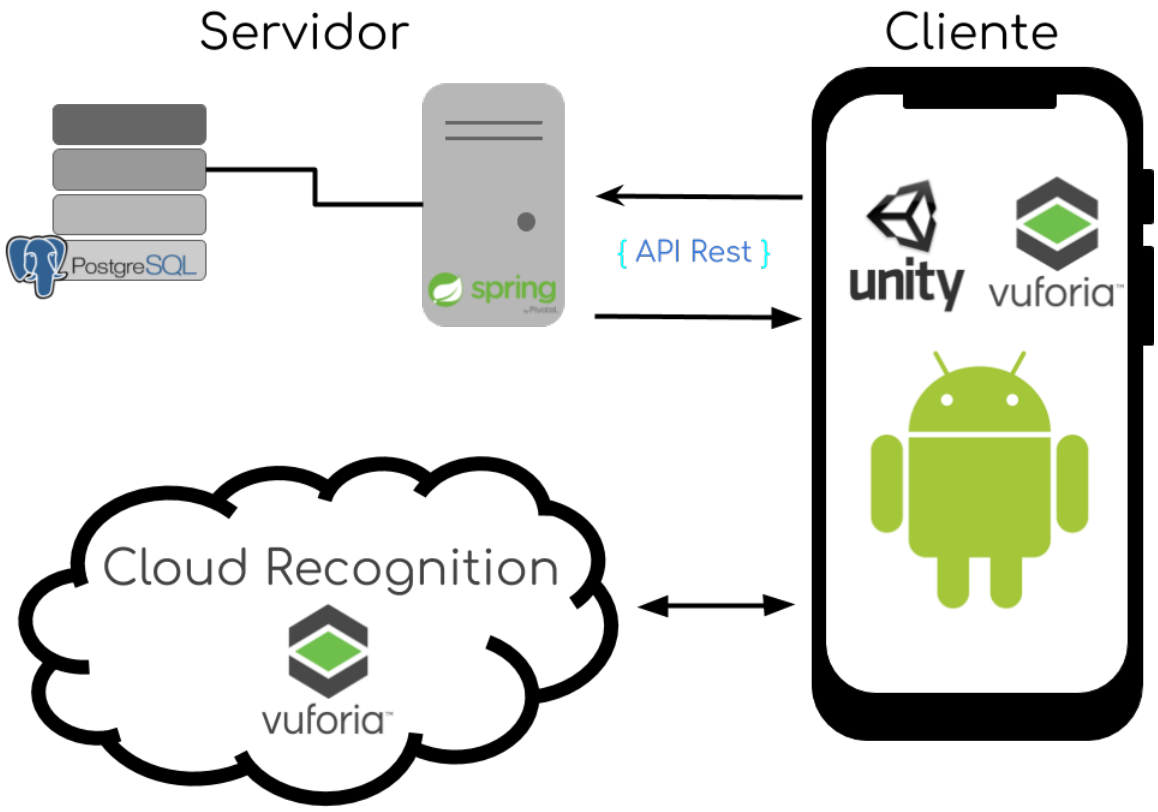

Figura 4.11: *Arquitectura*

4.3. Servidor

Nuestro servidor está programado en Java, usando el framework Spring para la creación de una API Rest que, mediante peticiones HTTP y una base de datos relacional en PostgreSQL, nos proporcionará toda la información necesaria para alimentar de datos a nuestro cliente. También se encuentra en el servidor la implementación del sistema de recomendación.

En la Figura 4.12, podemos ver como se comunican los distintos componentes dentro

del servidor para dar respuesta a las peticiones HTTP que se realizan desde la aplicación del cliente. Las peticiones se gestionan mediante controladores que, a través de los servicios de aplicación, llevan a cabo la lógica de negocio, comprobando si los datos son correctos. Nuestros servicios de aplicación, tras realizar las comprobaciones necesarias, utilizan repositorios para realizar las operaciones contra la base de datos. Por otro lado encontramos el sistema de recomendación que genera predicciones usando los datos de las valoraciones de los usuarios. El sistema de recomendación también usará un servicio de aplicación para realizar las operaciones necesarias y, a través de los repositorios, acceder a las entidades que necesite.

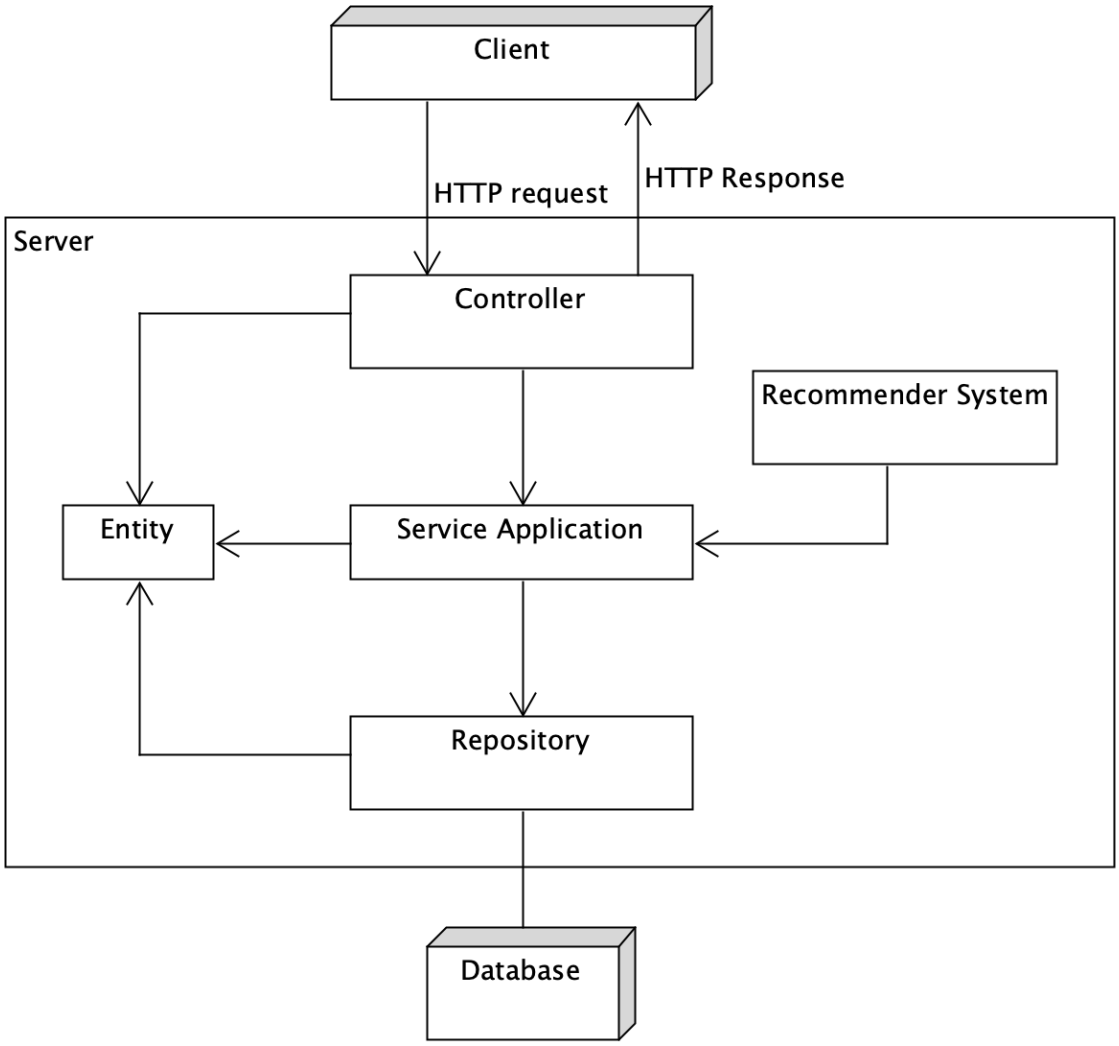

Figura 4.12: *Diagrama de funcionamiento del servidor*

4.3.1. Entidades

En el servidor tendremos todas las entidades que necesitamos (Ver Figura 4.13), éstas son representadas mediante clases Java y anotaciones de JPA, para facilitar el mapeo entre tablas de la base de datos y dichas entidades. Hemos implementado las siguientes entidades:

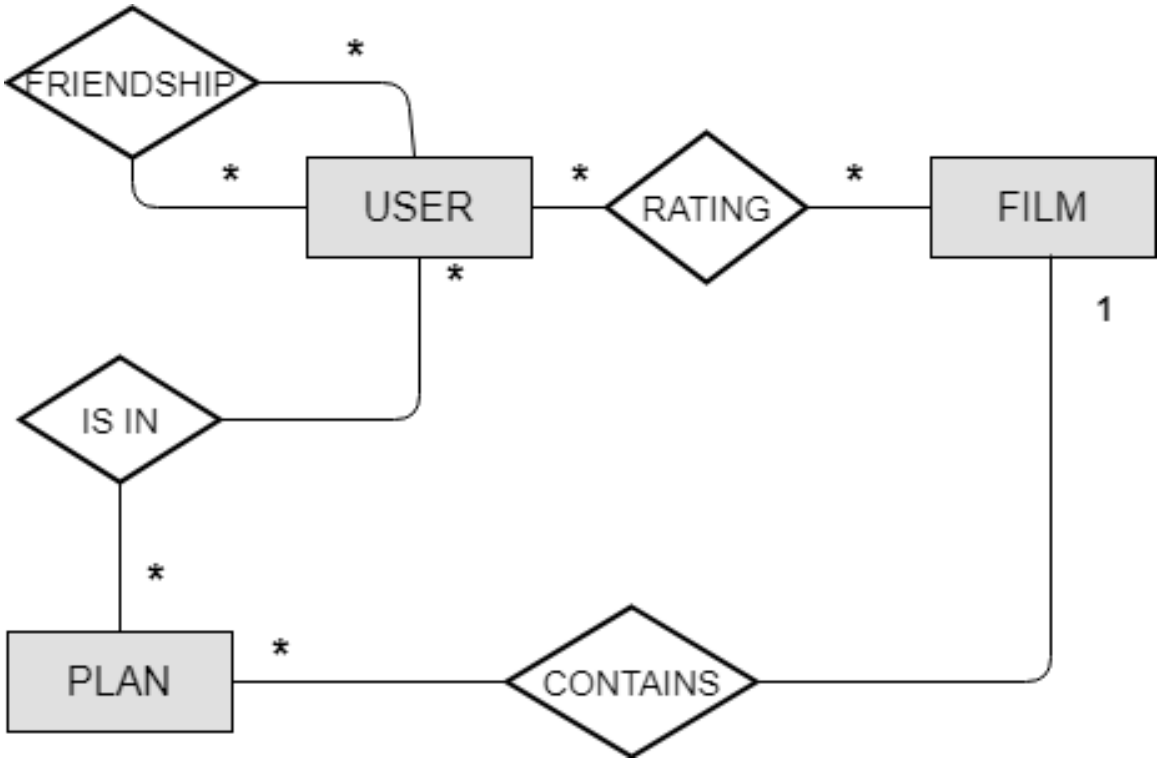

Figura 4.13: *Modelo entidad-relación*

- Film: entidad referente a las películas que guardamos, con su respectivo título, director, duración, valoración, url del tráiler, sinopsis, url de la imagen de la película, género, país y fecha de estreno.
- User: entidad para representar a los usuarios de la aplicación, con su nombre, foto, email y contraseña.
- Friendship: entidad para representar la relación de amistad entre usuarios.
- Plan: entidad para representar los planes. Estos poseerán una película, un usuario creador, fecha, hora, lugar, título, descripción y una lista de usuarios unidos.
- UserFilm: entidad intermedia para representar la relación de usuarios con películas, es decir, las películas que son guardadas por cierto usuario.

4.3.2. Repositorios

Tendremos un repositorio por cada entidad. En Spring las clases de tipo repositorio son aquellas clases que heredan de la clase JPARepository. Son una serie de interfaces que nos proporcionan ya implementados los métodos de tipo CRUD (crear, leer, actualizar y borrar) para la entidad que gestiona dicho repositorio. También se pueden crear métodos personalizados. Los repositorios nos simplifican la forma de acceder a la base de datos ya que nos evita tener que implementar métodos para ello.

4.3.3. Servicios de Aplicación

Tendremos un servicio de aplicación por cada entidad. Son las clases encargadas de la lógica de negocio de nuestra aplicación. Se comprueba que los datos que le llegan a los servicios de aplicación a través de los controladores o del sistema de recomendación sean correctos. Tras comprobar la validez de los datos, el servicio de aplicación llamará a los métodos correspondientes de la clase repositorio que corresponda.

4.3.4. Controladores

Para cada una de estas entidades necesitamos una clase Controller, que actuará de controlador para saber redirigir cada petición HTTP a su correspondiente servicio de aplicación. A la búsqueda por ID tuvimos que añadir la búsqueda por UUID en usuarios y películas.El ID es necesario para utilizarlo en el sistema de recomendación de Mahout yel UUID es necesario para el reconocimiento de imágenes de Vuforia. Las distintas peticiones HTTP que realizamos son las siguientes:

Film		
Obtener todas las películas	GET	/films/
Guardar una película	POST	/films/
Actualizar una película	PUT	/films/
Buscar una película por su nombre	GET	/films/search/{name}
Buscar una película por su id	GET	/films/{id}
Buscar una película por su uuid	GET	/films/uuid/{uuid}
Borrar una película	DELETE	/films/{id}

User		
Obtener todos los usuarios	GET	/users/
Guardar un usuario	POST	/users/
Actualizar un usuario	PUT	/users/
Buscar un usuario por su nombre	GET	/users/search/{name}
Buscar un usuario por su id	GET	/users/{id}
Buscar un usuario por su uuid	GET	/users/uuid/{uuid}
Borrar un usuario	DELETE	/users/{id}
Obtener amistades de un usuario	GET	/users/{id}/friendships
Obtener amigos de un usuario	GET	/users/{id}/friends
Obtener planes de un usuario	GET	/users/{id}/plans
Obtener películas que gustan a un usuario	GET	/users/{id}/films
Login de un usuario	POST	/users/login

UserFilm		
Obtener todas las valoraciones	GET	/user-films/
Guardar valoración	POST	/user-films/
Actualizar valoración	PUT	/user-films/{userId}/{filmId}/rate/{rating}
Buscar una valoración	GET	/user-films/{userId}/{filmId}
Borrar una valoración	GET	/user-films/{id}

Recommender		
Obtener todas las recomendaciones	GET	/recommendations/
Obtener películas random	GET	/recommendations/random
Obtener películas más relevantes	GET	/recommendations/trending
Obtener películas más nuevas	GET	/recommendations/premiere
Recomendaciones para un usuario	GET	/recommendations/{id}
Obtener las recomendaciones de planes	GET	/recommendations/{id}/plans/{friend}
Borrar todas las recomendaciones	DELETE	/recommendations/
Borrar todas las recomendaciones	DELETE	/recommendations/{id}

Images		
Obtener imagen	GET	/{image}/

Friendships		
Obtener todas las amistades	GET	/friendships/
Petición de amistad	POST	/friendships/{requesterId}/{friendId}/request
Aceptación de amistad	POST	/friendships/{requesterId}/{friendId}/accept
Declinar amistad	DELETE	/friendships/{requesterId}/{friendId}/decline
Eliminar amistad	DELETE	/friendships/{requesterId}/{friendId}

Plans		
Obtener todos los planes	GET	/plans/
Guardar un plan	POST	/plans/
Unirse a un plan	PUT	/plans/{id}/join/{userId}
Eliminar un plan	DELETE	/plans/{id}
Obtener planes por id	GET	/plans/{id}
Obtener los usuarios unidos a plan	GET	/plans/{id}/joined-users
Obtener los usuarios de un plan	GET	/plans/{id}/users
Buscar un plan por nombre	GET	/plans/search/{title}

4.3.5. Sistemas de recomendación

Implementamos el sistema de recomendación para obtener películas y planes que pueden ser interesantes para cada usuario, en base a sus gustos. Estas recomendaciones se utilizan en la parte de cliente y se muestran al usuario a través de la interfaz. Los datos de las recomendaciones se guardan en una tabla de la base de datos. Cada periodo de tiempo establecido previamente un procedimiento se ejecuta y actualiza la tabla para aplicar los nuevos datos guardados a las recomendaciones. Este procedimiento usa una funcionalidad de la librería Mahout, para extraer los datos de valoraciones de usuarios a películas. Con estos datos y usando una técnica de filtrado colaborativo genera las recomendaciones. Como generar las recomendaciones es un proceso costoso y lento guardamos los resultados para ser utilizados.

4.4. Cliente

Nuestra aplicación está implementada principalmente en Android, las interfaces principales para el uso de los planes, recomendaciones y acceso a la información de las películas guardadas han sido desarrolladas mediante Android Studio.

Sin embargo,la parte del cliente que otorga el gran valor a nuestra aplicación está desarrollada en Unity usando la librería de Vuforia. Todas las escenas que aparecen al reconocer carteles de películas e imágenes de usuarios fueron implementadas de esta forma.

La parte de la aplicación realizada en Unity, utiliza una serie de clases de la parte desa-

rrollada en Android para realizar las peticiones al servidor y así poder mostrar información específica cuando se reconoce una cierta imagen.

Toda la parte de la aplicación que hace uso de Realidad Aumentada está desarrollada en Unity con Vuforia. Vuforia, como hemos explicado anteriormente, sirve para identificar fotografías que se han fijado previamente como objetivos. Esta técnica nos ha resultado muy útil para el reconocimiento de carteles de películas. Sin embargo, nos encontramos con un problema a la hora de identificar usuarios para todas las mecánicas relacionadas con ellos en Realidad aumentada tales como agregar amigos u obtener recomendaciones. Este problema surge de que Vuforia aún no incluye en su librería el reconocimiento facial. Por este motivo hemos decidido a modo de prototipo, que nuestra aplicación reconozca las imágenes de los usuarios.

Como comentaremos más adelante en Sección 5.2, si en un futuro Vuforia introduce la posibilidad de reconocer mediante reconocimiento facial, actualizaríamos esta mecánica.

Al principio, tanto la parte de Android como la parte de Unity, realizaban peticiones al servidor de forma independiente. Más tarde, para tener un mejor control y no hacer peticiones de dos sitios diferentes, decidimos implementar la obtención de datos únicamente en Android y que Unity usara las clases de Android para acceder al servidor.

En la Sección A.3 podemos encontrar una guía con las instrucciones para construir la aplicación y para integrar la parte de Unity y Android.

Como la mayor parte de la aplicación está implementada en Android Studio, la estructura de ésta se adapta a lo que ofrece el IDE, consistiendo en el flujo de actividades.

Una Activity (Subsección 4.4.1) es cada una de las pantallas o vistas de una aplicación[3], las cuales pueden ser modificadas a través de Adapters (Subsección 4.4.2), para crear listas de objetos o incluso dividir una misma actividad en varias.

Tras dividir una actividad obtenemos Fragments (Subsección 4.4.5), que son considerados secciones de una actividad con un comportamiento determinado[6]. Los Fragments los hemos usado para hacer una actividad principal con 3 secciones por las que se puede navegar, donde

cada sección representa un Fragment.

Las Entities (Subsección 4.4.4) son los modelos de datos correspondientes a las tablas almacenadas en la base de datos. Tendremos una Entity en Android por cada Entidad del Servidor (Subsección 4.3.1), para así mapear los datos de las Entities del Servidor a las Entities de la parte Android.

También tendremos las clases View (Subsección 4.4.7) que permiten configurar comportamientos personalizados para las vistas, como ocultar y mostrar objetos según los gestos que se realicen.

Para acceder al servidor, debemos implementar unas clases para obtener datos, llamadas Request (Subsección 4.4.6).

Como tenemos una parte de cliente implementada en Unity, y anteriormente hemos dicho que Unity utiliza las clases de Android para acceder al servidor, hemos implementado el patrón Command (Subsección 4.4.3). Esto permite que la Activity creada para comunicar Android con Unity use comandos para saber que petición realizar al Servidor.

La arquitectura de la parte de cliente podemos verla en la Figura 4.14.

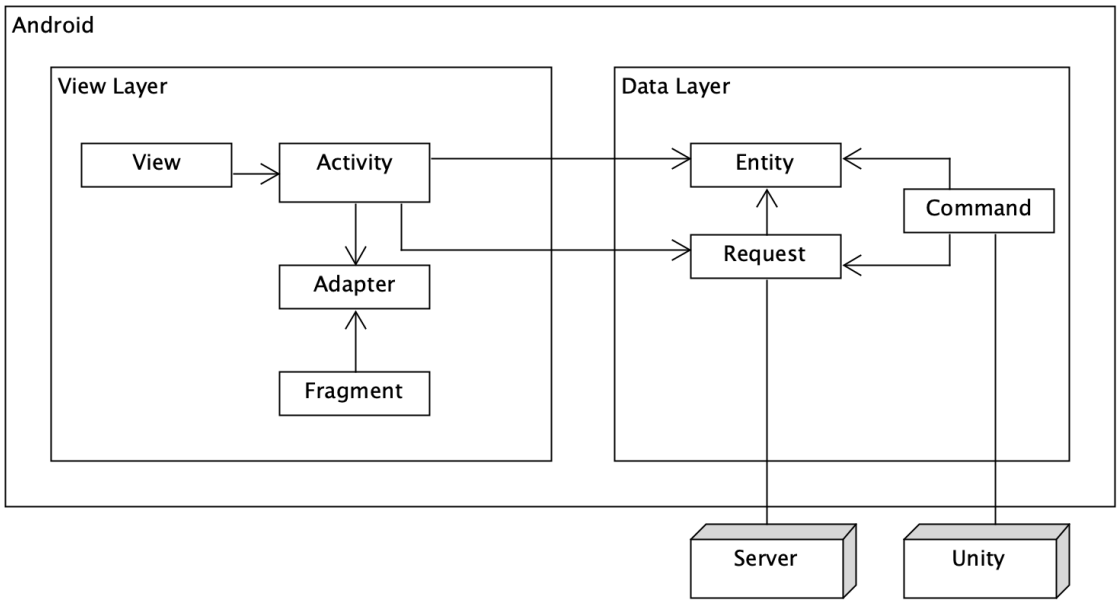

Figura 4.14: *Diagrama de arquitectura de cliente*

4.4.1. Actividades

Son los componentes de la aplicación que representan una pantalla con la que los usuarios pueden interactuar para realizar una determinada acción (ver Figura 4.15).

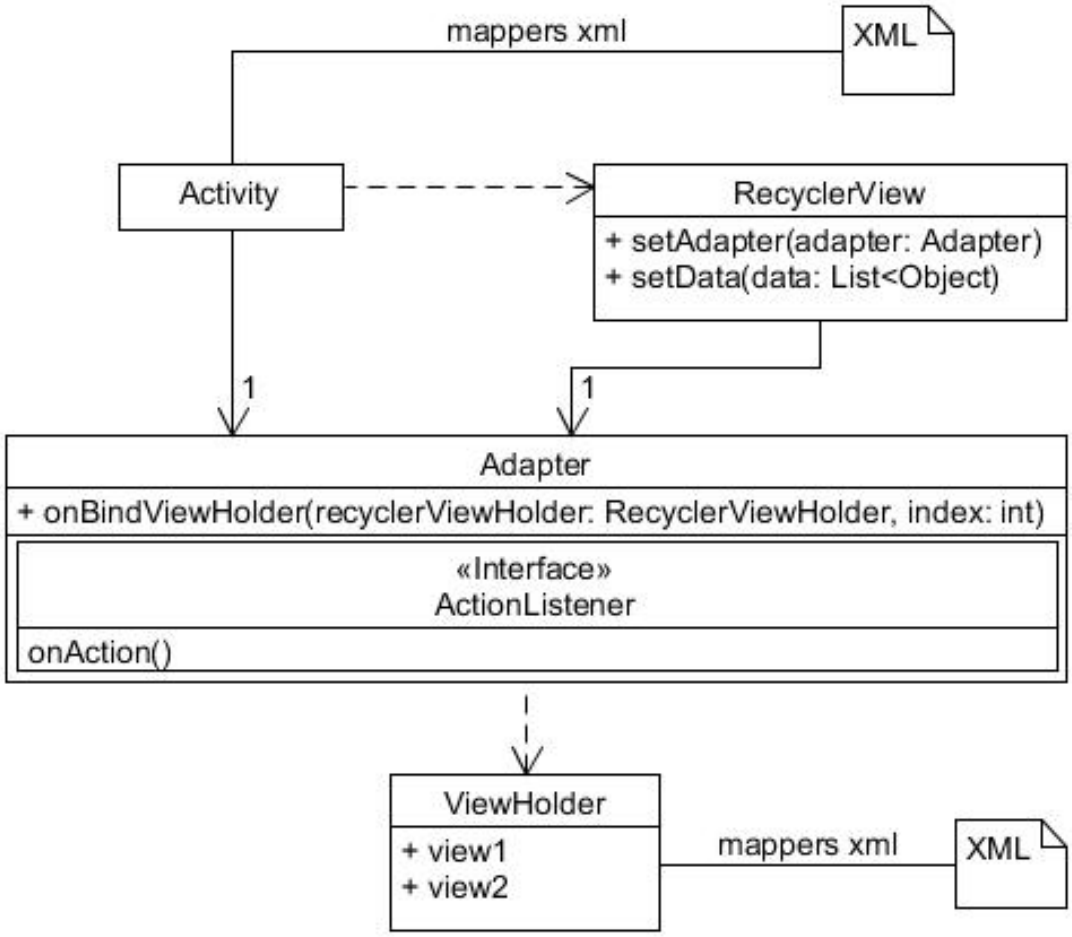

Figura 4.15: *Diagrama de Actividades*

Entre nuestras actividades podemos destacar dos:

- MainActivity: La primera actividad (actividad principal) que se activa al iniciar la aplicación, la cual comprueba el estado de sesión del usuario, si está logueado se queda en la actividad, y si no, se redirige al usuario a otra actividad llamada LoginActivity que mediante un formulario muy simple nos permite iniciar sesión o registrarnos. En

MainActivity se encuentra un paginador (ViewPager) de 3 fragmentos, donde cada fragmento representa las interfaces de planes, recomendaciones y películas guardadas correspondientemente. Podemos observarlo en la Figura 3.1, Figura 3.4 y Figura 3.7 del Capítulo 3.

- UnityPlayerActivity: Se trata de una actividad especial que nos permite comunicar la parte desarrollada en Unity con nuestro proyecto Android. Aquí se encuentran llamadas a funciones desde la parte de Unity, se realizan las órdenes y se devuelven a Unity.

4.4.2. Adaptadores

Son elementos de manipulación de datos que se aplican a vistas y componentes. En el proyecto se les ha dado mucho uso a las vistas de tipo RecyclerView para representar listas de objetos. Para manejar los datos de cada elemento de la lista, se ha utilizado un RecyclerView.Adapter, el cual manipula cada dato correspondiente a un elemento de la lista. También se han creado otros adaptadores, como uno para manejar tres fragmentos dentro de una actividad, como hemos dicho anteriormente al describir la actividad principal (MainActivity).

4.4.3. Comandos

Aquí se encuentran todos los elementos relacionados con el patrón comando (ver Figura 4.16). Ha sido diseñado para realizar la conexión con la parte del proyecto desarrollado en Unity. Al principio las peticiones al servidor se realizaban tanto en Android como en Unity. Esto causaba que tuviéramos implementaciones duplicadas y originaba problemas si había que modificar estos métodos, porque era necesario aplicar los cambios en ambas capas. Decidimos utilizar desde Unity los métodos de Android para resolver el problema. Para evitar que Unity tuviera múltiples dependencias con la capa de Android, decidimos aplicar el patrón comando. Este patrón nos permite abstraer la capa de Android con la capa

de Unity y minimizar las dependencias que eran necesarias.

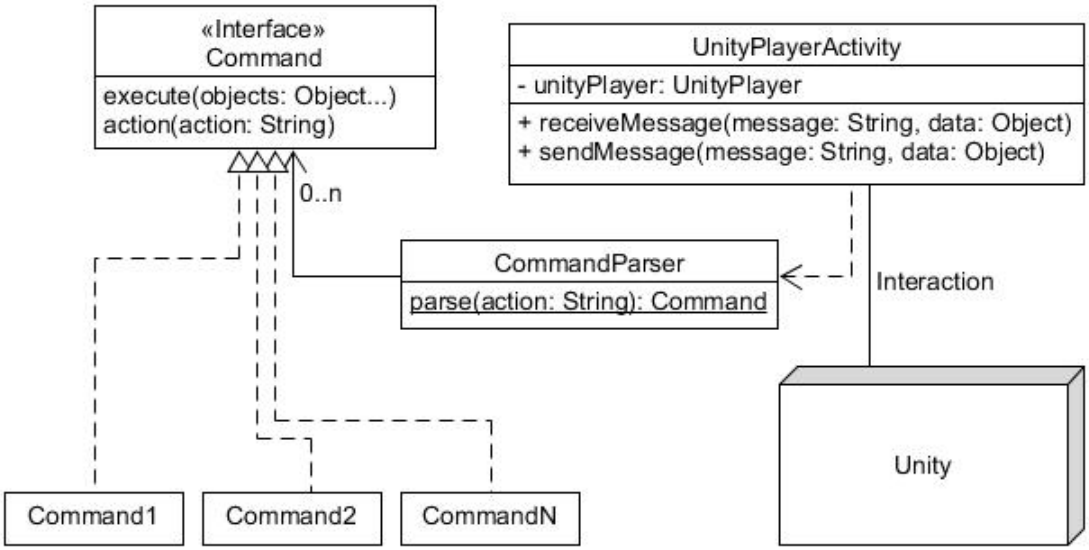

Figura 4.16: *Patrón comando*

4.4.4. Entidades

Todos los modelos de datos que podemos recibir del servidor, los cuales hemos descrito anteriormente en este capítulo, en la Sección 4.3, son representados por entidades en nuestro proyecto de Android, con los mismos atributos que tienen sus entidades correspondientes en el servidor.

4.4.5. Fragmentos

Representan un comportamiento o una parte de la interfaz de usuario en una actividad. Se utilizan para formar una vista compuesta de paginación en una actividad. Aquí es donde tendremos los 3 componentes principales de nuestra aplicación, la actividad referente a mis planes, la actividad de las recomendaciones y la de mis películas.

4.4.6. Petición de servicios

Se trata de un conjunto de clases que solo poseen métodos estáticos para llevar a cabo peticiones HTTP al servidor. Se ha creado una clase genérica para la creación y llamada de peticiones HTTP, se ha diseñado de manera que las respuestas se devuelvan en forma de callback para un uso más sencillo (ver Figura 4.17).

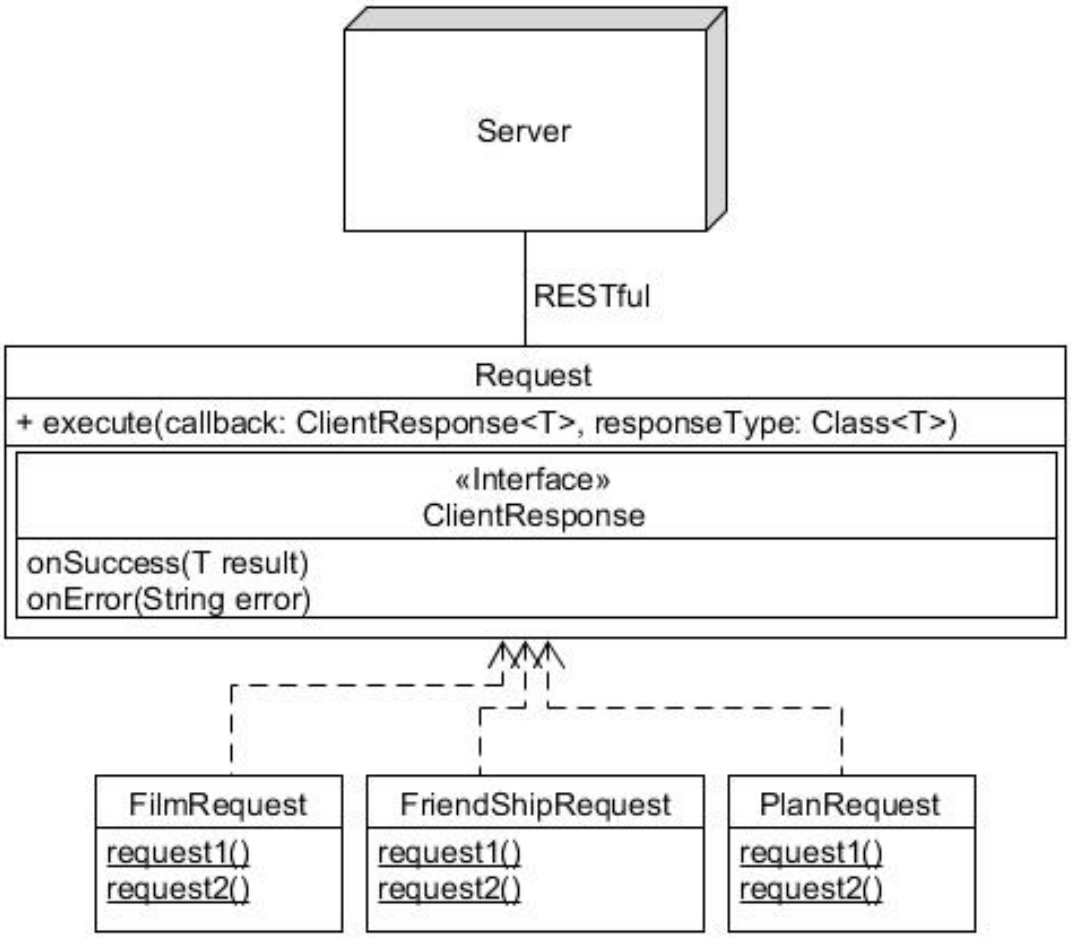

Figura 4.17: *Petición de servicios*

Los servicios que hemos implementado son los siguientes:

- FilmRequest: servicio encargado de realizar las peticiones sobre películas al servidor, como podría ser buscar una película por su ID.

- FriendshipRequest: servicio responsable de obtener los datos relacionados a las relaciones entre amistad, como podría ser aceptar una petición o comprobar si dos usuarios son amigos.

- PlanRequest: servicio cuya finalidad es la de conseguir la información relacionada con los planes, como podría ser la operación de crear un plan y unirse o salirse de uno ya existente.

- RecommendationRequest: este servicio se ocupa de obtener las distintas recomendaciones de películas y planes de un usuario.

- UserFilmRequest: servicio creado para obtener la información referente a ciertos métodos de la relación entre películas y usuarios. Por ejemplo, con este servicio podríamos guardar una película como favorita de un usuario. También es usado para cuando un usuario valora cierta película.

- UserRequest: servicio que nos permite obtener todo lo relacionado con los usuarios, ya sean sus perfiles, los amigos de un usuario, los planes en los que está o las películas que ha guardado.

4.4.7. Vistas

Entre los componentes que ofrece Android existen algunas funcionalidades no soportadas. Por ello hemos creado componentes personalizados para satisfacer estas funcionalidades:

- AnchorVisibilityBehavior: Implementa el comportamiento de los elementos que aparecen y desaparecen en las actividades, por ejemplo, en la Figura 3.2 tenemos un botón para la información de la película pero al hacer el scroll hacia abajo, en la Figura 3.3, vemos que éste desaparece.

- CustomViewPager: el ViewPager es lo que usamos en la MainActivity para colocar los tres fragmentos. Se crea esta clase para desactivar la acción de cambiar entre fragmentos al deslizar con el dedo en la pantalla del smartphone. Esto nos daba problemas

debido a que en el fragmento de las recomendaciones tenemos elementos con scroll horizontal y la acción de deslizar con el dedo en la pantalla interfería.

- ExpandableTextView: Es una clase que extiende de un TextView, se le añade un método para que cuando haya mucho texto, se pueda acortar. Además dispone de un botón para expandir.

4.5. Despliegue

El servidor puede desplegarse como un Docker o en Heroku. Para el desarrollo del proyecto hemos utilizado Heroku por la facilidad y comodidad que ofrece. Usamos una cuenta gratuita que nos permite un alojamiento limitado con las características suficientes para el desarrollo del proyecto. Utilizamos un despliegue automático que al realizar cambios en una rama del repositorio de GitHub, Heroku descarga esos cambios y lo despliega en el servidor. En la Sección A.2 encontramos una guía de como desplegar la aplicación con tu usuario de Heroku. En la Sección A.1 vemos como desplegar la aplicación en un Docker.

4.6. Herramientas de trabajo

Las distintas herramientas de trabajo que hemos decidido usar para que el trabajo realizado fuera más eficiente son las siguientes:

- Telegram: hemos usado esta aplicación de mensajería para poder comunicarnos entre todos los miembros pertenecientes al proyecto, ya fuera para la sugerencia de ideas o que un bot de GitHub nos avisara de los cambios en los distintos repositorios que hemos creado.

- Slack: esta otra herramienta de comunicación nos ha permitido crear canales específicos para cada parte de nuestro proyecto y así poder clasificar las conversaciones que teníamos por temas, por ejemplo, tenemos un canal para hablar solo de la parte

backend, otro para hablar de la parte de Realidad Aumentada, etc. Cosa que Telegram no nos permitía y resultaba lioso encontrar ciertos temas o puntos de nuestras conversaciones.

- Trello: nos ha servido como apoyo para realizar nuestros sprints, Trello nos ofrece la posibilidad de crear un tablero para escribir tareas a modo de post-it y asignarlas a usuarios que estén dentro del tablero, así sabemos qué tipo de tareas ha realizado o está realizando cada miembro en todo momento.

- GitHub: todos nuestros repositorios están en GitHub, lo que nos permite subir cambios y trabajar en paralelo en distintas ramas, agilizando así nuestro trabajo y otorgándonos un control de versiones muy estable.

- Visual Studio Code: utilizamos este editor de código junto con la extensión de LaTeX Workshop para escribir la memoria en LaTeX.

- Android Studio: usamos este IDE para construir la aplicación Android que además nos dota de herramientas de depuración.

- Google Drive: almacenamiento en la nube donde tenemos toda la información relevante para el TFG.

- Heroku: plataforma en la que desplegamos nuestra aplicación de servidor.

- Unity: Utilizamos esta plataforma de desarrollo para construir la parte de la aplicación relacionada con Realidad Aumentada.

- Visual Studio: Utilizamos Visual Studio para codificar los scripts en C# necesarios para el funcionamiento de la parte de Realidad Aumentada en Unity.

- Postman: aplicación para el envío de peticiones HTTP REST, que utilizamos para depurar los servicios de Spring.

- MockFlow: herramienta online para el diseño de mockups, nosotros la utilizamos para el diseño de las vistas de la aplicación Android.

4.7. Conclusiones

A continuación, mostraremos una lista de todas las dificultades encontradas a la hora de implementar nuestra aplicación:

- Diseño de las interfaces para que fueran familiares e intuitivas para el usuario. Tuvimos que rehacer muchas veces los distintos bocetos de las interfaces ya que no eran intuitivos para los usuarios. Gracias a las sugerencias que se nos iban haciendo conseguimos mejorarlas bastante.

- Desconocimiento del entorno de desarrollo y peculiaridades del desarrollo de aplicaciones en Android. Para resolver nuestro desconocimiento de Android realizamos una serie de prototipos y buscamos tutoriales que nos sirvieran como ejemplo para desarrollar cada parte de nuestra aplicación.

- Incompatibilidad de realización de peticiones HTTP en ciertas versiones de Android. Tras probar la aplicación en un smartphone con versión de Android 9.0 descubrimos que los permisos para hacer peticiones de este tipo a un servidor no funcionaban con la configuración inicial de nuestro proyecto. Para solucionarlo cambiamos una serie de líneas de código en nuestro archivo AndroidManifest.xml para solventarlo.

- Problemas a la hora de intentar desplegar el servidor de forma correcta para que fuera accesible siempre. Para que el servidor estuviera siempre levantado se nos ocurrió realizar su despliegue en Heroku para así poder realizar peticiones en cualquier momento desde la app.

- Poca familiaridad con el uso de Unity y el lenguaje C# para realizar la parte de Realidad Aumentada. Para familiarizarnos con el entorno implementamos una serie

de prototipos y buscamos tutoriales que nos sirvieran como guía para diseñar nuestra parte de Realidad Aumentada.

- Caducidad de las licencias gratuitas para el uso del Cloud de Vuforia. La solución a este problema fue muy simple, crearnos cuentas nuevas cada vez que se acabara la licencia para poder seguir accediendo a este servicio.

- Primera vez usando el framework de Spring para la realización de una API Rest. Su solución fue seguir un tutorial paso a paso para crearlo con una sola entidad e ir ampliándolo a todas las entidades que necesitaba nuestra aplicación.

- Comunicación entre la parte cliente en Android y el servidor. Conseguimos realizar servicios en Android que realizaban las peticiones al servidor de forma asíncrona, obteniendo su resultado en una función callback para usar dicha información como correspondiera en cada caso.

- Comunicación entre la parte cliente en Unity con la parte cliente en Android. Para abstraer la lógica desarrollada en Android a la de Unity implementamos clases que usaban el patrón Command para facilitar la comunicación entre estas dos partes de nuestro proyecto.

- Rendimiento en cuanto al reconocimiento de ciertas imágenes, sobre todo de personas, debido a la calidad de las mismas. El Cloud de Vuforia no las reconoce tan fácilmente como las de carteles de películas. Para arreglarlo tuvimos que buscar imágenes de usuarios con una mejor calidad para facilitar a la librería su reconocimiento, finalmente optamos por poner imágenes de los integrantes del proyecto durante el desarrollo de la aplicación.

- Almacenamiento de imágenes en Google Drive y su obtención desde Android y Unity, junto con el retardo que esto produce. Para no tener las imágenes de los usuarios en la aplicación se nos ocurrió guardarlas en Google Drive y obtenerlas con la librería

de Picasso de Android, pero esto era muy lento y decidimos guardarlas en el mismo servidor.

- Incompatibilidades con las nuevas actualizaciones de Unity. Hemos tenido cuidado con no actualizar el IDE de Unity porque nuestro proyecto quedaría obsoleto, ya nos vimos en la situación de que uno de los miembros tenía una versión superior a la de los demás y tuvimos que rehacer lo que habíamos implementado.

Capítulo 5

Conclusiones

El objetivo de nuestro proyecto ha sido la realización de explicaciones mediante Realidad Aumentada sobre carteles de películas y caras de usuarios. La información que mostramos debe ser relevante y entendible para el usuario, tanto para las películas como para los usuarios de la aplicación que se reconozcan con la cámara del dispositivo. Para lograr esto hemos implementado una aplicación móvil completa que nos permitiese realizar dicha función.

Cuando comenzamos a hablar sobre este proyecto, lo que nos motivó fue el reto de aprender nuevas tecnologías desconocidas para nosotros y afrontarlas en un proyecto que soluciona un problema concreto. La solución a este problema es una aplicación móvil que nos ofrece la facilidad de crear y gestionar planes con amigos para ir al cine. Usando un sistema de recomendación somos capaces de tener una previsión sobre lo que nos podría gustar la película en base a otras que hayamos valorado anteriormente. Con el uso de tecnología de RA buscamos que el usuario se sienta más cómodo utilizando la aplicación, siendo propio dispositivo móvil capaz de reconocer los carteles de películas, o las imágenes de usuarios, ofreciéndonos la posibilidad de realizar acciones sobre éstas.

Al principio buscamos aprender los aspectos más básicos que nos permitieran diseñar e implementar la aplicación. Para ello realizamos un estudio del estado del arte en el que desempeñamos las siguientes tareas:

- Buscamos información y explicamos que es la Realidad Aumentada y cuál es su estado actual. Describimos diferentes librerías de Realidad Aumentada con sus aspectos más

destacables.

- Descubrimos y especificamos distintas fuentes de información de las cuales extraer los datos necesarios sobre las películas de la aplicación, que se mostrarán o se utilizarán para el sistema de recomendación.
- Buscamos información sobre los distintos tipos de técnicas de recomendación y definimos las más interesantes.

En el siguiente paso, diseñamos la aplicación y definimos su funcionalidad, realizando las siguientes tareas:

- Realizamos un análisis de competencia en el cual descubrimos aplicaciones similares que nos ayudaron a visualizar el diseño de nuestra aplicación.
- Definimos escenarios donde se podría usar nuestra aplicación.
- En base a los escenarios anteriormente definidos, establecimos los requisitos funcionales que describen la funcionalidad de la aplicación.
- Diseñamos las interfaces de la aplicación para las funcionalidades anteriormente comentadas.

Posteriormente, pasamos a la parte de implementación donde realizamos las siguientes tareas:

- Desarrollamos prototipos con distintas librerías de realidad aumentada en aplicaciones móviles y valoramos la experiencia con ellas.
- Definimos una arquitectura REST para nuestra aplicación y un reconocimiento de imágenes en la nube.
- Implementamos un servidor en Spring desplegado en Heroku que usa como base de datos PostgreSQL.

- Implementamos la aplicación móvil en Android y la parte de RA en Unity con la librería Vuforia.
- Valoramos distintos sistemas de recomendación y finalmente lo implementamos con Mahout.

Finalmente, fuimos capaces de aplicar los conceptos generales que aprendimos en el grado en un entorno nuevo y desafiante, gracias al valor humano de un equipo multidisciplinar en el cual los integrantes complementan sus puntos débiles individuales.

5.1. Conclusions

Our project′s objective is based on carrying out explanations through augmented reality about movie posters and user′s faces. The information that we show must be relevant and understandable for the user, both for the movies and for the users of the application which are recognized by the device′s camera. To achieve this we implemented a complete mobile app that allowed us to perform this functionality.

When we started talking about this project, the fact of learning new technologies and facing them to solve a specific problem was the main issue that encouraged us. Our solution to this problem is a mobile application that offers us the ease of creating and managing movie-based plans with friends. We are able to get predictions about the likeliness of enjoyment for a movie by using a recommendation system, based on a historical film valorations list. Using Augmented Reality, our objective is that a user feels more comfortable using this application, being the mobile device capable of recognizing movie posters, or user images, offering us the possibility of performing actions on these.

At the beginning we seek to learn the most basic aspects that would allow us to design and implement the application. In order to do so we carried out a study of the state of art where we did the following tasks:

- We searched for information and explained what Augmented Reality is and what is

it′s current state. We described different Augmented Reality libraries with their most remarkable aspects.

- We discovered and specificated different information sources from which we could extract the necessary data about the films in our application, which will be showed or used for the recommendation system.
- We searched for information about the different types of recommendation techniques and we defined the most interesting ones.

In the next step, we designed the application and defined it′s functionality, doing the following tasks:

- We made a competition analysis in which we discovered similar applications to ours which helped us visualize our application′s design.
- We defined scenarios where our application could be used.
- Based on those scenarios previously defined, we established the functional requirements which describe the application′s functionality.
- We designed the interfaces of the application for the functionalities previously commented.

Later, we started with the implementation part, where we did the following tasks:

- We developed prototypes with different augmented reality libraries for mobile devices and we valued our experience with them.
- We defined a REST arquitecture for our application and image recognition on the cloud.
- We implemented a server on Spring deployed on Heroku which uses PostgreSQL as a database.

- We implemented the mobile application on Android and the AR part on Unity using Vuforia′s library.
- We valorated different recommendation systems and finally implemented one using Mahout.

Finally, we were capable of applying the general concepts that we learned in our degree in a new and challenging environment, thanks to the human value of a multidisciplinary team in which the members complement each others individual weak points.

5.2. Trabajo futuro

Uno de los mayores problemas del proyecto fue la dificultad de integrar distintas tecnologías entre sí y que estas cumplieran con todos los requisitos que buscábamos. Esto ocasiono que el tiempo de aprendizaje y adaptación del uso de estas tecnologías se disparará, por tanto, a las primeras funcionalidades que teníamos en mente tuvimos que priorizar las que daban el uso básico al problema que planteábamos. A continuación, destacaremos todas esas funcionalidades o mejoras que nos gustaría que tuviera en un futuro la aplicación para que la usabilidad y posibilidades sean más ricas para el usuario:

1. Una de las situaciones que se pueden encontrar nuestros usuarios es que quieren ir al cine a ver una película con amigos, pero no les importe que película si no pasar un buen rato con buena compañía. En este caso la aplicación podría crear planes y una vez se unan todos los amigos sea capaz de recomendar la película que mejor se ajusta a sus preferencias.

2. Otra situación similar a la anterior es que queramos que nos recomiende la película que mejor se adapta a los gustos del grupo, pero acotando a las películas que elijamos previamente.

3. En el momento del desarrollo de este proyecto el reconocimiento facial no está integrado en las librerías de RA que hemos investigado y la complejidad de integrar tecnologías

de reconocimiento facial, así como su aprendizaje no era viable para el tiempo de este proyecto. Con lo que se decidió realizar el reconocimiento del usuario con imágenes asemejando esta funcionalidad. Creemos interesante el uso de reconocimiento facial para funcionalidades de usuario, aunque pueda ser polémica es algo nuevo que aporta valor a la aplicación.

4. La escena de RA esta implementada en Unity por lo que podemos exportarla a otras plataformas como iOS, consiguiendo con esto que la aplicación se abra a más usuarios y estos consigan un mayor crecimiento y enriquecimiento de la aplicación.

5. Facilitar la compra de entradas desde la aplicación para que los usuarios puedan realizar la mayoría de las acciones, necesarias para ir al cine, redireccionando a la página web de venta de los cines a los que se quieren acudir para ver la película.

6. Publicar la aplicación a una plataforma de distribución digital de aplicaciones como Google Play, para que los usuarios tengan acceso a instalarla en sus dispositivos.

7. Automatizar con web scraping la obtención de datos de las fuentes de información de películas para que la aplicación siempre tenga la información actual.

8. Utilizar distintos idiomas para el contenido de las películas en la base de datos del servidor usando una columna que indique el idioma de la información.

9. Una de las cosas más importantes que falta por realizar es una evaluación con usuarios que nos proporcionen retroalimentación de la aplicación de la cual obtendremos una visión más amplia sobre si satisface la necesidad al problema que tratamos en el proyecto o si hay que realizar cambios.

Capítulo 6

Contribución al proyecto

En este capítulo especificaremos las aportaciones de cada integrante del grupo, las cuales se ordenarán por las fases del proyecto.

6.1. Diego Acuña Berger

6.1.1. Estado del arte

- Investigación de las distintas fuentes de información para la obtención de los datos referentes para las películas, valorando cuál sería la que mejor información aportaba para el proyecto.

6.1.2. Diseño de la aplicación

- Diseños de vistas de la aplicación con MockFlow. Es decir, el diseño de los primeros bocetos de las interfaces que terminaron por ser lo que hemos diseñado en el Capítulo 3.
- Diseño de las interfaces de usuario de los planes y películas guardadas. Tras varias pruebas y diseños fallidos diseñé la forma de mostrar los planes y películas basándome en aplicaciones del ámbito del cine o de las series, como puede ser TV Time[13]
- Diseño de la vista que aparece tras guardar una película mediante la Realidad Aumentada, con la que se confirma que película se ha guardado.

6.1.3. Implementación

- Estudio y creación del prototipo de servidor en Spring. Búsqueda de tutoriales y la codificación de un pequeño servidor en local usando el framework de Spring, tras el prototipo contribuí a su ampliación y modificación para que se ajustase correctamente a lo que necesitaba nuestra aplicación.

- Obtención de imágenes mediante Picasso a la aplicación.

- Funcionalidad para guardar películas mediante Realidad Aumentada, esto implica la comunicación con Android y Unity para saber la relación que tiene la película que se ha reconocido desde el cloud, con la película que se guarda en el servidor y otorgar feedback al usuario de su correcto funcionamiento.

- Disposición y diseño de botones que aparecen tras reconocer una película con Realidad Aumentada. Los botones que mostrábamos al principio no tenían mucho sentido con sus iconos y ocupaban mucho espacio, opté por incluir un botón inicial que al pulsarse se desplegase y mostrase más botones por encima del mismo, en los cuales puedes acceder a la información de la película o guardarla.

- Implementación de funciones del servidor tales como unirse a un plan, salirse de un plan y buscar los planes de un usuario concreto. Cuando me dedicaba a implementar ciertas vistas de la aplicación de Android, como mostrar los planes o las películas guardadas, me vi en la necesidad de incluir funciones en el servidor para obtener los datos que necesitaba, las cuales no nos habíamos planteado ninguno hasta el momento en el que fueron necesarias.

6.1.4. Memoria

- Redacción del resumen tanto en español como en inglés.

- Redacción de la introducción a nuestro proyecto así como de los antecedentes.

- Redacción de los escenarios en los que se usaría nuestra aplicación.
- Redacción de los requisitos funcionales de la aplicación.
- Redacción de las distintas interfaces de usuario de la aplicación.
- Redacción del prototipo del servidor que implementé con Spring.
- Redacción de cómo está implementado nuestro servidor, es decir, qué entidades poseemos, que servicios de aplicación, controladores y repositorios.
- Revisión de calidad de la memoria. Me he ocupado de revisar las posibles faltas ortográficas o fallos de estilo al usar LaTeX.
- Redacción de las dificultades de trabajo así como de las soluciones que les dimos durante el desarrollo del proyecto.

6.2. Daniel Calle Sánchez

6.2.1. Estado del arte

- Investigación de las tecnologías de Realidad Aumentada ViroReact y Expo AR. Con el fin de decidir el entorno en el que finalmente se implementará la aplicación.

6.2.2. Diseño de la aplicación

- Colaboración en la identificación de requisitos funcionales de la aplicación.
- Creación de los segundos bocetos para las interfaces en MockFlow, definiendo un recorrido en el uso de la aplicación.

6.2.3. Implementación

- Creación de un prototipo con ViroReact y React Native siguiendo los ejemplos de la documentación oficial de Viro Media[15], incluye un menú desde el que se accede a dos

escenas de RA, una que muestra texto y otra que muestra al personaje de una película saliendo del poster de la película cuando el dispositivo lo reconoce. También se usa la librería llamada NativeBase que muestra distinta apariencia en las interfaces si es iOS o Android. Se solucionó problemas de dependencias al usar las distintas librerías.

- Creación de un prototipo con ARCore usando Kotlin como lenguaje en la plataforma de Android, siguiendo los ejemplos proporcionados por la documentación de Google ARCore[9].
- Gestión de los repositorios utilizados para el proyecto en GitHub.
- Diseño de la arquitectura que ha permitido la comunicación entre servidor y cliente.
- Configuración de los recursos y la aplicación en Heroku para el despliegue automático.
- Implementación de los primeros servicios para planes y amigos que contenían todas las operaciones CRUD en Spring.
- Cambios en los servicios de usuarios y películas para añadir nuevas operaciones y datos.
- Implementación de controles a los datos de entrada en los servicios del Spring.
- Uso de patrón Transfer para los datos de entrada y salida en Spring.
- Uso de los patrones Factoría y Singleton en Spring.
- Cambios en los métodos HTTP de los servicios de Spring para usar los correctos que describen los estándares de API REST.
- Implementación del reconocimiento de usuarios con Vuforia en Unity para que al detectar que la imagen es la de un usuario mostrar los paneles correspondientes y si es una película mostrar los anteriores ya existentes.

- Cambios en la lista de planes en la interfaz de Android para mostrar fecha, imágenes de usuarios.
- Guiar las reuniones de retrospectiva utilizando algunas dinámicas como la técnica del barco velero para recolectar datos del equipo y generar ideas. Después de la anterior, aplicamos la lista abierta de acciones para decidir qué hacer con el fin de mejorar en la siguiente iteración.
- Búsqueda y configuración de herramientas que mejoren el rendimiento de trabajo en equipo como Slack, Trello.

6.2.4. Memoria

- Definir estructura de la memoria.
- Redacción de Capítulo 5 Conclusiones.
- Redacción de Sección 5.2 Trabajo futuro.
- Referencias a la bibliografía.
- Revisión de la memoria.

6.3. Carlos Gómez Cereceda

6.3.1. Estado del arte

- Investigación de Vuforia en Unity y su exportación a Android, con el fin de decidir el entorno en el que finalmente se implementaría la aplicación.
- Estudio del funcionamiento del Cloud de Vuforia, con el objetivo de decidir su viabilidad en la incorporación al proyecto y si con la licencia gratuita ofrecida por Vuforia era suficiente.

- Investigación de ARCore con el objetivo de decidir si las funcionalidades ofrecidas por esta tecnología eran las que necesitábamos.

6.3.2. Diseño de la aplicación

- Diseño de las interfaces mostradas en Realidad Aumentada cuando se reconoce a un usuario o a una película.

- Primeras interfaces desarrolladas con Realidad Aumentada en las que se mostraban vídeos. En un primer momento, pensamos que sería buena idea mostrar los tráileres de las películas en Realidad Aumentada, aunque finalmente se desechó la idea porque saturaba la interfaz.

- Participación en la identificación de escenarios y creando los primeros bocetos.

6.3.3. Implementación

- Estudio y creación de una aplicación desarrollada enteramente en Unity con Vuforia. Puesto que ninguno de los integrantes del equipo habíamos tenido ninguna experiencia con Unity ni con Vuforia, mediante tutoriales y documentación desarrollé un prototipo básico en el que se detectaba una imagen objetivo y se situaba encima de ésta un modelo 3D.

- Una vez desarrollado el prototipo anterior, investigué como de factible era realizar el resto de la aplicación en Unity. Realicé alguna interfaz y funcionalidades básicas, finalmente decidimos que no era viable ya que Unity no proporciona demasiadas facilidades ya que no es para lo que está diseñado. Por lo investigué cual era la forma de exportar el proyecto de Unity a uno en Android, creando un prototipo simple en Android que al pulsar un botón llevase a Unity.

- Finalmente, al prototipo anterior le incluí el Cloud de Vuforia para no sobrecargar la aplicación con imágenes.

- Estudio y creación de un prototipo de ARCore en Android. Con la guía oficial de ARCore realicé un prototipo de ARCore, sin embargo, la comparación Vuforia era muy grande, éste último era mucho más sencillo de aprender e intuitivo por lo que finalmente no se realizó el proyecto con ARCore.

- Una vez teníamos el proyecto en Android con la parte de Realidad Aumentada en Unity con Vuforia investigué como enviar información de una parte a otra, ya que en cierto modo son partes distintas de la aplicación.

- Desarrollé las primeras versiones de la parte de Realidad Aumentada en las que se mostraban vídeos con los tráileres de las películas y texto con la sinopsis.

- Comunicación en Vuforia con el servidor, aunque finalmente decidimos no usar este método y que todas las llamadas al servidor se realizasen desde Android.

- Diseñé e implementé la mecánica que permite distinguir si el objeto al que se está enfocando es una película o un usuario sin necesidad de tener dos tipos de Realidad Aumentada. Al igual que la mecánica anterior, una vez que se sabe que es un usuario, desarrollé la mecánica para saber si es un amigo o no.

- Creación con ayuda de los ejemplos de Vuforia del escáner que proporciona al usuario el feedback de que está escaneando.

- Creación de la vista en Unity que se muestra al identificar una película mediante Realidad Aumentada, además de su funcionalidad y comunicación con Android. Ejemplo: mostrar tráiler, puntuación, información...

- Creación de la vista en Unity que se muestra al identificar a un usuario no identificado como amigo. Así como la funcionalidad en Realidad Aumentada y Android para agregar a dicho usuario como amigo.

- Creación de la vista en Unity que se muestra al identificar a un amigo en donde se muestran los tres planes recomendados y la información visual de cuánto se prevé que les guste ese plan a los usuarios dentro del plan.
- Implementación en Android de la mayoría de las funciones que reciben información de Unity, realizan alguna petición al servidor y devuelven dicha información a Unity.
- Creación en Android de métodos de comunicación con el servidor para acciones como añadir amigos, borrar plan, obtener amistad, obtener planes...
- Creación en el servidor de la primera versión de usuarios y su relación con películas.

6.3.4. Memoria

- Revisión de la memoria.
- Aportaciones en distintas partes como herramientas usadas, contribuciones, dificultades encontradas.
- Redacción de parte de las interfaces de usuario que se muestran en Unity y en Android.
- Redacción de la guía para generar un APK.

6.4. Zihao Hong

6.4.1. Estado del arte

- Estudio de la implementación de un prototipo que usara Vuforia y Android, además de las dependencias que necesitaba. Investigación del pipeline que usan para renderizar objetos en el espacio, la librería de dibujo en 3D que se usa internamente OpenGL, incluso aprendizaje de lenguajes de bajo nivel (GLSL) que se usa en las tarjetas gráficas. Para finalmente decidir su viabilidad para incorporar al proyecto

- Investigación sobre los métodos de recomendación, mediante búsquedas de recursos en la web, prueba de ejemplos dados, realización de simulaciones contra la base de datos visualizando los resultados y establecimiento del método final del sistema.
- Estudio del funcionamiento del Cloud que ofrece Vuforia para el reconocimiento de imágenes. Simulaciones de ejemplos contra el Cloud y pruebas de servicios REST que ofrecen para en un futuro la automatización de subidas de imágenes al Cloud.
- Investigación de la seguridad que necesita tener el servidor para restringir accesos.

6.4.2. Diseño de la aplicación

- Diseño de los primeros bocetos para los casos de usos y escenarios, proponiendo situaciones que se podían dar en las que usar la aplicación y sus respectivas soluciones.
- Diseño de las interfaces de usuario de los amigos y en cómo se hacen peticiones de amistad, cómo responden a estas peticiones y el borrado lógico o físico de las amistades.
- Diseño de las interfaces de sesión de un usuario en la aplicación.
- Diseño inicial del esquema de relación de la base de datos.

6.4.3. Implementación

- Implementación de distintos ejemplos de proyectos Unity sobre Realidad Aumentada con Vuforia, verificando los distintos requisitos que necesitábamos para el proyecto.
- Implementación de las primeras interfaces del proyecto de Unity con Vuforia generando dos paneles al lado de la imagen del póster de la película para representar la información de la misma.
- Creación de prototipo con Vuforia en Android, utilizando los métodos que ofrecían para reconocer imágenes, e implementación con OpenGL para dibujar figuras sobre objetos mediante un lenguaje a bajo nivel (GLSL).

- Intento de escribir texto en el proyecto de Vuforia, el cual se consiguió mediante uso de mapa de bits, pero se rechaza finalmente pues se trata de una implementación muy complicada.

- Creación de prototipo de la arquitectura de Microservicios, para posterior incorporación en el proyecto. Utilizando un gateway llamado Zuul y un cloud de registros llamado Eureka, de los cuales los demás servicios se registran y zuul los organiza.

- Evaluación del uso del Docker ofrecido por la universidad, creación de scripts para imágenes de Docker y su posterior despliegue como contenedor en local, preparación de las distintas herramientas necesarias para el despliegue del servidor en un entorno Linux, preparación de bases de datos dentro del contenedor, creación de distintos perfiles para el despliegue en el proyecto del servidor y final implementación de scripts usados para activar el servicio del servidor en Docker.

- Implementación inicial del servidor para su conexión con la base de datos junto con sus entidades más fundamentales.

- Implementación de rutas en el servidor, como consecuencia también sus servicios de aplicación, repositorios, etc.

- Creación de la base de la de arquitectura del proyecto de Android para establecer el flujo de actividades inicial en el proyecto de Android.

- Implementación de un Viewpager para integrar tres interfaces en una actividad para que el usuario pueda moverse entre ellas. Se establece como la principal interfaz al entrar en la aplicación.

- Implementación de actividades de sesión y sus correspondientes controles. Se ha conseguido que la sesión del usuario siga activa incluso cuando se cierra la aplicación, solo se termina cuando el usuario decide cerrar su sesión.

- Implementación de un panel de menú a la izquierda de la aplicación para darle más funcionalidad a la aplicación.
- Establecimiento de la clase RecyclerView para mostrar listas de objetos en la aplicación, ya que es la más usada en Android, la clase ListView está obsoleta.
- Creación de algunas vistas propias con características especiales para satisfacer las necesidades que tiene la aplicación para mostrar algunas cosas.
- Establecimiento de la conexión del proyecto de Android con el proyecto de Unity mediante una actividad llamada UnityPlayerActivity, la cual recibe, ejecuta y retorna datos desde Unity. Para ello se ha usado el patrón de diseño de comando para abstraer dependencias entre Android y Unity.
- Diseño de una clase genérica para hacer peticiones contra el servidor y así facilitar su uso. Las respuestas se devuelven en forma de callback ya que estas son asíncronas.
- Establecimiento de algunos efectos cuando se haga una transición de actividades dentro de la aplicación.
- Implementación en el servidor del sistema de recomendación usando la librería de Mahout, que ofrece un método para la conexión con la base de datos PostgreSQL sin tener que leer los datos de un fichero csv.
- Revisor de código en todos los proyectos, principalmente en Android y en el servidor para la generalización del estilo de código y correcciones en cuanto se identifiquen errores.
- Establecimiento del uso de patrones de diseño, como comandos, singleton, constructor, etc. Los cuales se encuentran en la implementación de Android.

6.4.4. Memoria

- Explicación de técnicas de recomendación en el Sección 2.3.
- Explicación del prototipo de Vuforia y Android en el apartado Subsección 4.1.3.
- Redacción de la parte cliente de la aplicación en el apartado Sección 4.4.
- Explicación de sistemas de recomendación en el apartado Subsección 4.3.5.
- Redacción de la guía de despliegue del servidor en Docker en el Sección A.1.

Bibliografía

[1] Apple. ARKit 2. https://developer.apple.com/arkit/, 2019. Último acceso Mayo de 2019.

[2] Paramount AR+. Paramount AR+. https://play.google.com/store/apps/details?id=com.paramount.ar.paramountar&hl=es_419, 2019. Último acceso Mayo de 2019.

[3] Andrea Ardións. ¿qué es un activity en android studio? https://androidstudiofaqs.com/conceptos/que-es-un-activity-en-android, 2015. Último acceso Mayo de 2019.

[4] Pio Calderon. An overview of recommendation systems. http://datameetsmedia.com/an-overview-of-recommendation-systems/, 2017. Último acceso Mayo de 2019.

[5] Paul Chapman. Microservicios. https://spring.io/blog/2015/07/14/microservices-with-spring, 2018. Último acceso Mayo de 2019.

[6] Android Developer. Fragmentos. https://developer.android.com/guide/components/fragments?hl=es-419, 2019. Último acceso Mayo de 2019.

[7] Expo. Expo augmented reality. https://docs.expo.io/versions/latest/sdk/AR/, 2019. Último acceso Mayo de 2019.

[8] FilmAffinity. FilmAffinity. https://www.filmaffinity.com/es/main.html, 2019. Último acceso Mayo de 2019.

[9] Google. ARCore overview. https://developers.google.com/ar/discover/, 2019. Último acceso Mayo de 2019.

[10] Marius Horga. Using arkit with metal. http://metalkit.org/2017/07/29/using-arkit-with-metal.html, 2017. Último acceso Mayo de 2019.

[11] IMDB. IMDB. https://www.imdb.com/, 2019. Último acceso Mayo de 2019.

[12] Kong. Microservicios. https://microservices.io/patterns/microservices.html, 2018. Último acceso Mayo de 2019.

[13] Toze Labs. Tv time la guia de series nº 1. https://play.google.com/store/apps/details?id=com.tozelabs.tvshowtime&hl=es, 2019. Último acceso Mayo de 2019.

[14] main_bug. Como crear un microservicio o servicio web rest con spring boot (parte 1). http://sinbugs.com/como-crear-un-microservicio-o-servicio-web-rest-con-spring-boot-1/, 2017. Último acceso Mayo de 2019.

[15] Viro Media. Viro React. https://viromedia.com/viroreact, 2019. Último acceso Mayo de 2019.

[16] melty. Photo de memoji-jeux-en-realite-virtuelle-siri-5-nouveautes-a-tester-d-urgence-sur-ios-12. https://www.melty.fr/memoji-jeux-en-realite-virtuelle-siri-5-nouveautes-a-tester-d-urgence-sur-ios-12-ga
html, 2018. Último acceso Mayo de 2019.

[17] Moviebase. Moviebase. https://play.google.com/store/apps/details?id=com.moviebase&hl=es, 2019. Último acceso Mayo de 2019.

[18] MovieLens. MovieLens. https://movielens.org/, 2019. Último acceso Mayo de 2019.

[19] Ricardo Moya. Filtrado demográfico. https://jarroba.com/que-son-los-sistemas-de-recomendacion/, 2013. Último acceso Mayo de 2019.

[20] Ricardo Moya. Filtrado híbrido. https://jarroba.com/que-son-los-sistemas-de-recomendacion/, 2013. Último acceso Mayo de 2019.

[21] Neosentec. ¿qué es la realidad aumentada? https://www.neosentec.com/realidad-aumentada/, 2019. Último acceso Mayo de 2019.

[22] Netflix. Netflix prize. https://www.netflixprize.com/, 2009. Último acceso Mayo de 2019.

[23] Netflix. Netflix recommendation factors. https://help.netflix.com/en/node/100639, 2019. Último acceso Mayo de 2019.

[24] Learn OpenGL. Coordinate systems. https://learnopengl.com/Getting-started/Coordinate-Systems, 2019. Último acceso Mayo de 2019.

[25] Learn OpenGL. Hello triangle. https://learnopengl.com/Getting-started/Hello-Triangle, 2019. Último acceso Mayo de 2019.

[26] Denis Parra. Filtrado basado en contenido. https://recommendersys.wordpress.com/2017/11/05/filtrado-basado-en-contenido/, 2017. Último acceso Mayo de 2019.

[27] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer US, 2011.

[28] Juan Saavedra. Sistemas de recomendación, parte 2: filtrado colaborativo. https://medium.com/@eng.saavedra/sistemas-de-recomendaci%C3%B3n-parte-2-b8a5dc9dc730, 2017. Último acceso Mayo de 2019.

[29] Rotten Tomatoes. Rotten Tomatoes. https://www.rottentomatoes.com, 2019. Último acceso Mayo de 2019.

[30] Vibhav Vibsz. Pokemon go ar contest winners announced. https://otakukart.com/animeblog/2017/11/19/pokemon-ar-contest-winners-announced/, 2017. Último acceso Mayo de 2019.

[31] Vuforia. Vuforia overview. https://library.vuforia.com/getting-started/overview.html, 2019. Último acceso Mayo de 2019.

[32] Wikitude. Wikitude augmented reality sdk. https://www.wikitude.com/products/wikitude-sdk/, 2019. Último acceso Mayo de 2019.

[33] Kirsten Winkler. How social learning through implied networks will change the travel experience. https://bigthink.com/disrupt-education/how-social-learning-through-implied-networks-will-change-the-travel-experience, 2011. Último acceso Mayo de 2019.

[34] Cristina Delgado Rodríguez, Ignacio Rocillo Landa, Jorge Muñoz Rodríguez, Jorge Rueda Garzón y Sergio Fuentes Urabayen. Recomendación dinámica de productos a grupos con realidad aumentada. *Universidad Complutense de Madrid*, 2015-2016.

Apéndice A

Guía de instalación para el servidor

A.1. Docker

Esta es una guía para el despliegue del servidor en Docker.

El repositorio del proyecto del servidor se encuentra en:

https://github.com/DanielCalle/TFG-Server

Para ello primero tenemos que tener docker instalado, el cual se puede instalar por el siguiente enlace:

https://www.docker.com/

Tras descargarse el repositorio e iniciado docker, situarse en la carpeta raíz del proyecto mediante el comando **cd**.

Listing A.1: *Despliegue de la instancia de docker*

```
# Ir a la raiz del proyecto
cd <path-del-proyecto>
# Despliegue
docker-compose up
```

Con la sentecia **docker-compose up** se despliega la instancia de docker con las configuraciones que están en el archivo docker-compose.yml. Esperar hasta que en el terminal de comandos aparezcan las líneas de la Figura A.1

```
Creating tfg-server_filmar_1 ... done
Attaching to tfg-server_filmar_1
```

Figura A.1: *Docker desplegado*

Ahora que se ha levantado la instancia de docker y sus respectivos puertos mapeados. Abrir un nuevo terminal de comandos y conectarse al contenedor mediante ssh con la contraseña **tfg-ucm**

Listing A.2: *Conexión ssh*

```
# El puerto 22022 es el que se habia mapeado
ssh root@localhost -p 22022
```

Una vez conectado al contenedor, hay que ir a la ruta en la que se encuentra el proyecto: `/usr/src/app`

Listing A.3: *Ir a la ruta del proyecto*

```
# Ir a ruta
cd /usr/src/app
```

Antes de levantar el servicio, hay que preparar postgresql. Hay que ejecutar el script de configuración llamado **database.sh**. Para ello hace falta primero darle permiso de ejecución al script.

Listing A.4: *Configuración postgresql*

```
# Dar permiso de ejecucion al script
chmod +x database.sh
# Ejecutar el script
./database.sh
```

Una vez configurado postgres, se procede al despliegue del servicio. Con **service.sh**.

Listing A.5: *Despliegue*

```
# Dar permiso de ejecucion al script
chmod +x service.sh
# Para iniciar el servicio
./service.sh start
# Para reiniciar el servicio
./service.sh restart
# Para parar el servicio
./service.sh stop
```

Puede que haya problemas con el formato de los scripts pues los scripts se escribieron en Windows, para ello hace falta descargarse **vim**, y cambiarle el formato a los scripts.

Listing A.6: *Ayudas*

```
# Descarga de vim
apt-get install -y vim
# Cambiar formato
vim script.sh
:set fileformat=unix
:wq
```

Una vez terminada la ejecución, ya está levantado el servicio en el puerto 8080, se puede probar con el siguiente enlace en cualquier navegador `localhost:8080/films`.

Si se quiere ver la estructura de la base de datos se encuentra en el siguiente fichero: `path-al-proyecto/src/main/webapp/WEB-INF/sql/filmar_low.sql`.

Cuando queramos terminar el proyecto, habríamos que desconectar la conexión de ssh y apagar la instancia de docker.

Listing A.7: *Salir de ssh y docker*

```
# Para desconectar la conexion con el ssh
exit
# Para apagar la instancia de docker
docker-compose down
```

A.2. Heroku

Para comenzar debemos crear una cuenta en Heroku. Una vez la tenemos creamos una nueva aplicación a la cual daremos un nombre como vemos en la Figura A.2 y este nombre a su vez será parte de la URL pública.

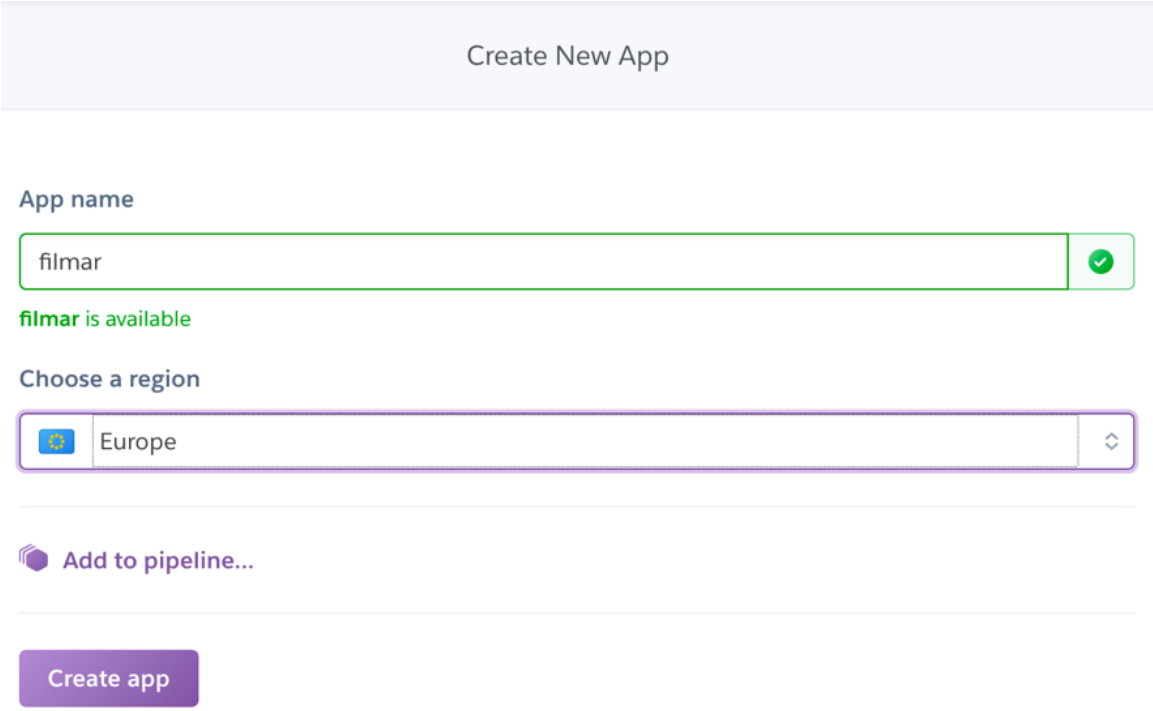

Figura A.2: *Crear nueva aplicación en Heroku*

Cuando creemos la aplicación accederemos a la configuración en la que veremos opciones como en la figura Figura A.3.

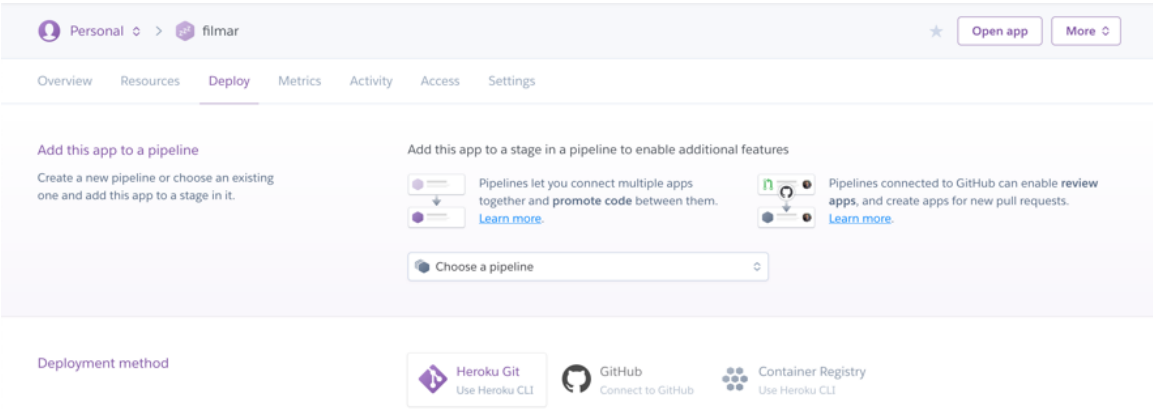

Figura A.3: *Configuración de la aplicación*

Entramos en la pestaña de recursos y buscamos Postgres como vemos en la Figura A.4.

La seleccionamos y agregamos la versión gratuita.

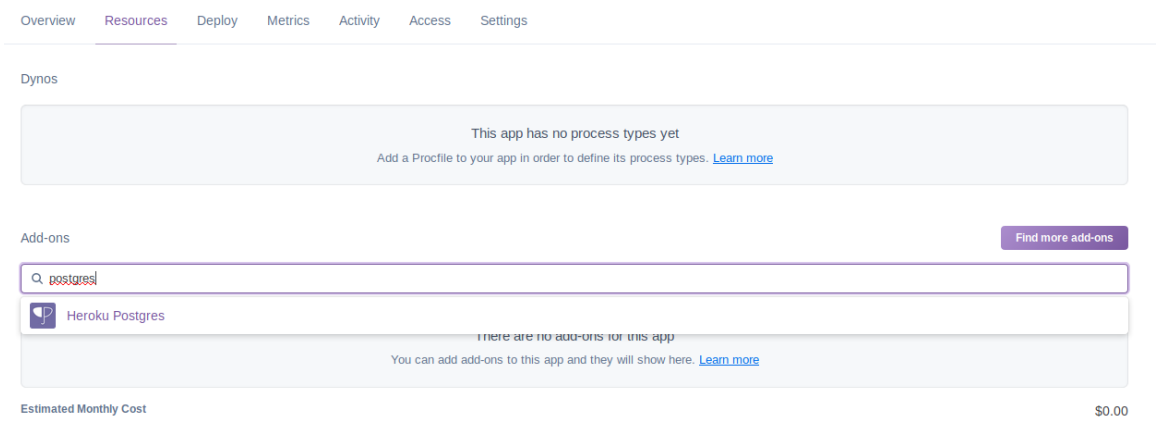

Figura A.4: *Añadir nuevos recursos*

Si accedemos al recurso de la base de datos que acabamos de crear y Entramos en la configuración podemos ver las credenciales que necesitamos añadir a nuestro servidor para que funcione correctamente como vemos en la Figura A.5.

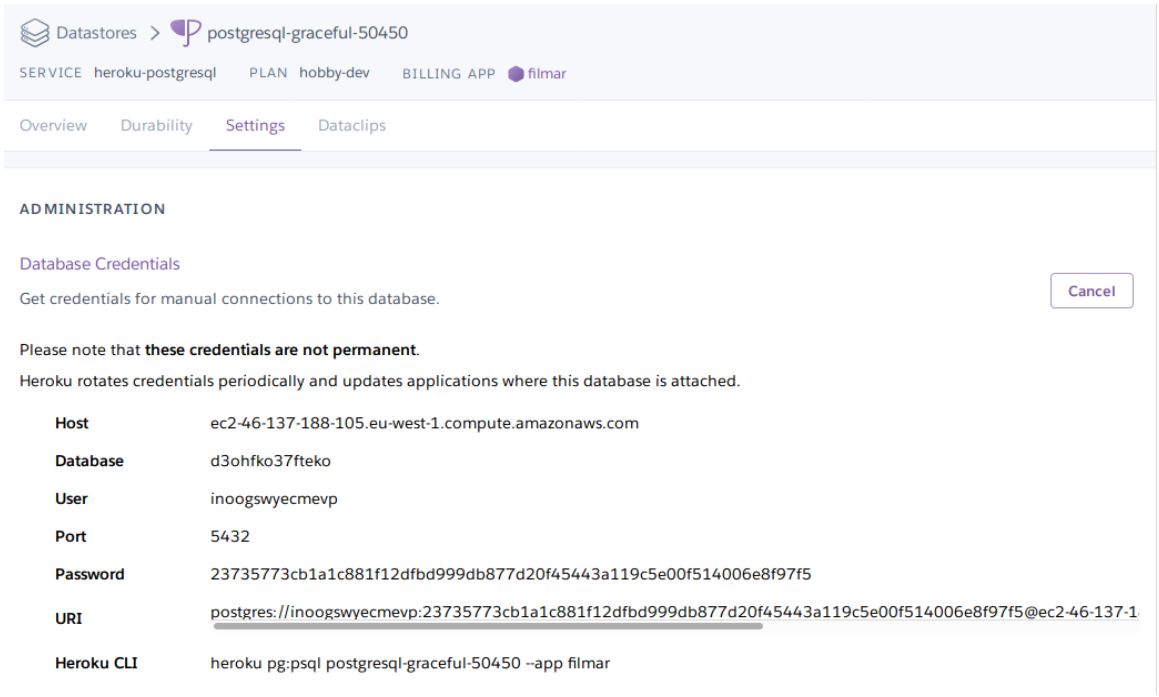

Figura A.5: *Ver credenciales*

Si vamos a la pestaña de despliegue de la aplicación, como vemos en la Figura A.6 nos ofrece diversos métodos de despliegue:

- Heroku Git
- GitHub
- Container Registry

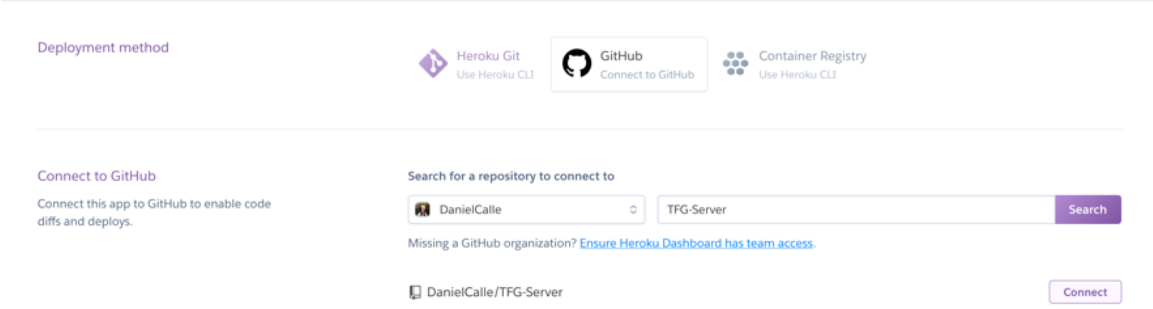

Figura A.6: *Métodos de despliegue*

Elegimos la opción de GitHub que nos proporciona un despliegue fácil y rápido. Tenemos que vincular la cuenta de nuestro GitHub (Importante ser el dueño del repositorio).

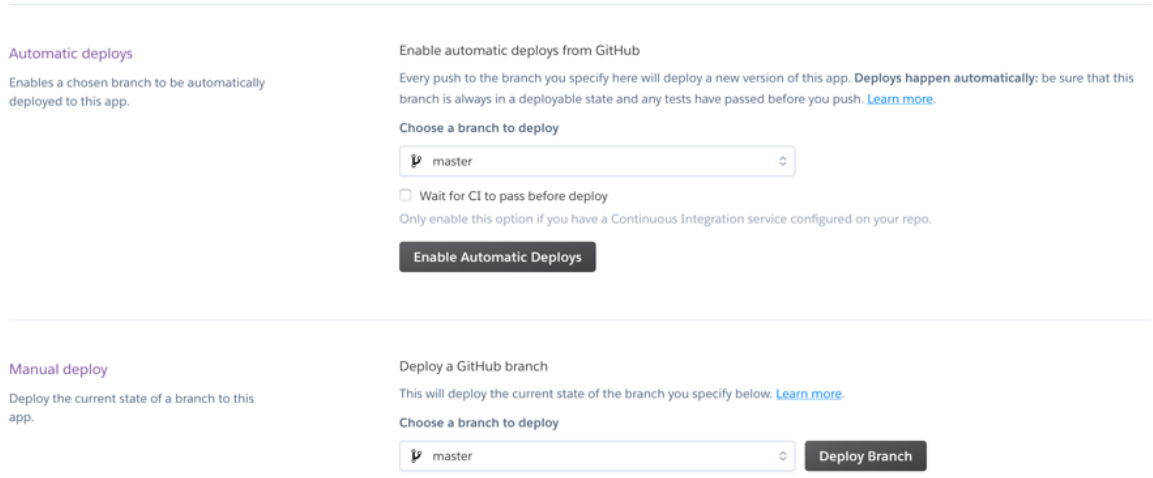

Figura A.7: *Opciones de despliegue utilizando Github*

Una vez vinculada la cuenta de GitHub tenemos dos métodos de despliegue como vemos en la Figura A.7. Un método automático para desplegar los cambios de una rama del repositorio que se accionará cada vez que subamos cambios y otro método manual que se desplegará sólo cuando presionemos el botón. Cuando elijamos una de las opciones, la aplicación estará lista para usarse.

A.3. Guía para generar un apk

Como hemos comentado anteriormente, nuestra aplicación consta de dos partes: La parte de Realidad Aumentada y la parte de Android. Por ello, a la hora de generar una apk completa de la aplicación habrá que generar estas partes por separado.

A.3.1. Generar la parte de la aplicación en Unity.

Para generar la aplicación en Unity:

1. Abrimos en Unity el proyecto. El nombre de este es TFG-Vuforia.

2. A continuación debemos seleccionar la escena en la que se encuentra la aplicación en Unity. Esta escena se encuentra en el directorio principal Assets >Scenes y su nombre es CloudRecognition.

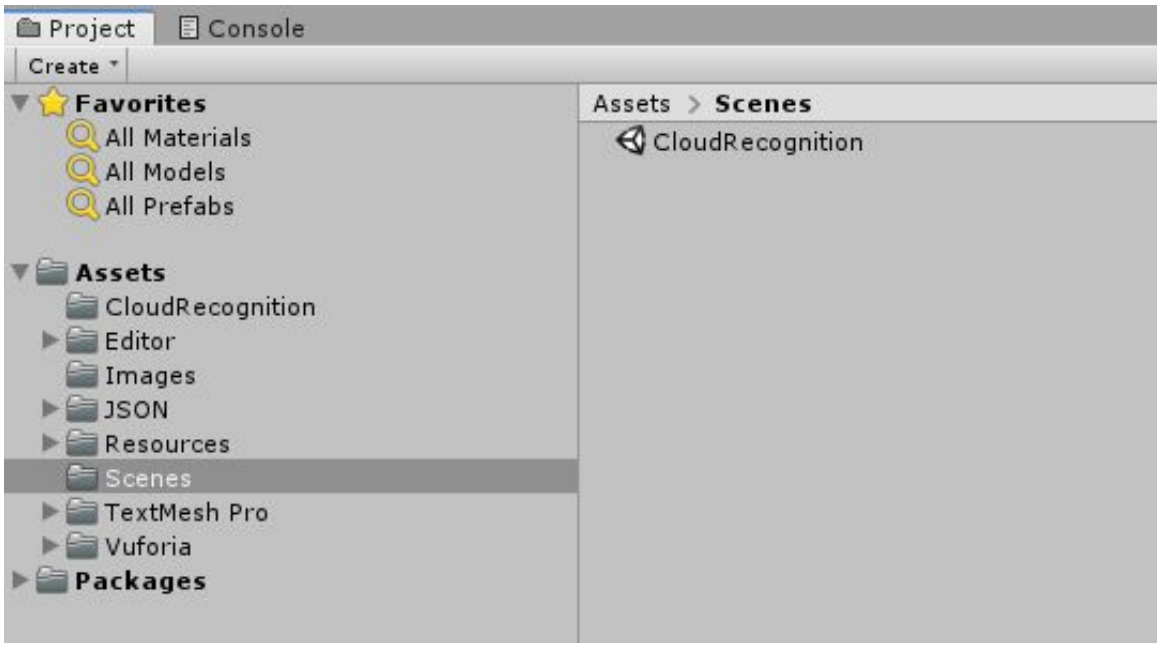

Figura A.8: *Ubicación de la escena principal*

3. Una vez localizada la escena principal, debemos seleccionarla. Para ello la arrastraremos al panel de trabajo (situado arriba a la izquierda).

Figura A.9: *Panel principal*

4. El siguiente paso es exportar este proyecto en Unity para que pueda usarlo la parte de Android. Para ello, pulsaremos File >Build Settings. Aparecerá una vista como

la de la Figura A.10, es muy importante antes de exportar la aplicación a Android tener seleccionado Android (a la izquierda). Para seleccionarlo se marca y se presiona el botón de Switch Platform.

Una vez que nos hemos asegurado de tener seleccionada la plataforma Android ya podemos exportar el proyecto. Para ello seleccionamos la opción que dice Export proyect con el Build system de Gradle, como se indica en la Figura A.10.

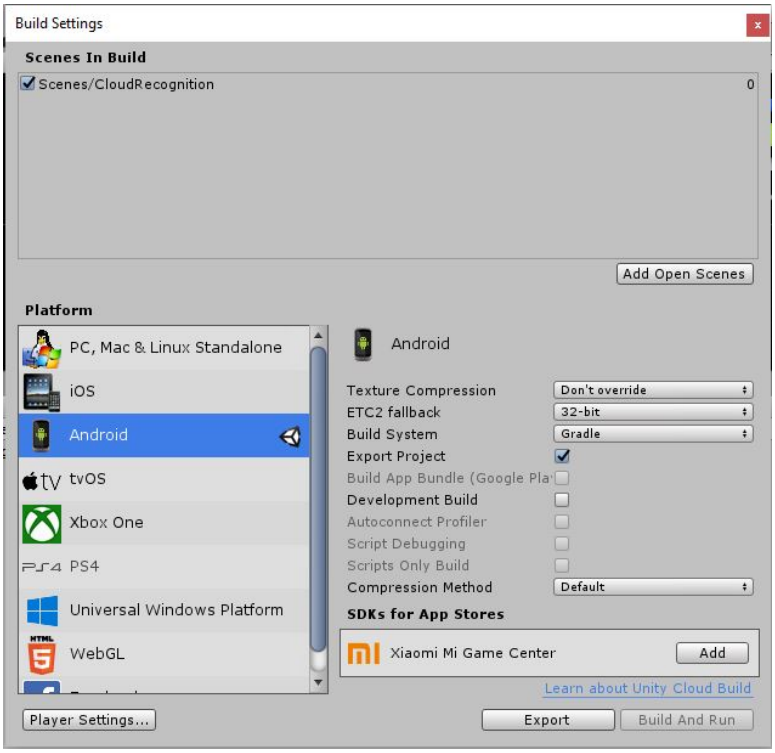

Figura A.10: *Build Settings*

A.3.2. Cómo unir la parte de Unity y Android

Como solución de la sección anterior, se habrá generado en la carpeta especificada un proyecto en Android llamado FilmAR.

Si fuésemos a crear un proyecto en Android desde cero, partiríamos desde este proyecto generado por Unity.

Si lo que queremos es actualizar nuevos cambios en Unity en la aplicación de Android

seguiremos los siguientes pasos:

1. Accederemos a los directorios FilmAR >src >main de las dos aplicaciones (la generada por Unity con los nuevos cambios y la aplicación en Android general de la aplicación).

2. Eliminaremos la carpeta Assets de la aplicación general y la cambiaremos por la nueva carpeta Assets generada por Unity.

De esta manera, tendremos la aplicación en Android actualizada a los nuevos cambios de la parte de Realidad Aumentada.

A.3.3. Generar un APK

Para generar un APK, simplemente tendremos que ir en Android Studio a la pestaña Build y hacer click en generar APK. De esta forma podremos descargar la aplicación.