Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

CineMapp: Una red social de cine para dispositivos móviles

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Marcos Terol Lloret

Tutor: David de Andrés Martínez

2018-2019

Resumen

El TFG consiste en la creación de una red social de cine donde los usuarios pueden comentar y valorar las películas. Además de esto tendrían acceso a la información de los últimos estrenos y un mapa donde localizar los cines más próximos. Se estudiará la viabilidad de permitir a los usuarios registrados la adquisición de entradas de cine desde la propia aplicación. Para su desarrollo se usarán herramientas externas que alojarán/obtendrán la información relativa a los usuarios, cartelera y películas, y un cliente para el consumo de los servicios desde un dispositivo móvil. El desarrollo se realizará en Android.

Palabras clave: aplicación; cine; película; cartelera; red;

Abstract

The TFG consists in the creation of a social network of cinema where users can comment and value the films. In addition to this they would have access to the information of the latest releases and a map where to find the nearest cinemas. The feasibility of allowing registered users to purchase movie tickets from the application itself will be studied. For its development, the rolled-up tools that will host / obtain information related to users, billboards and films, and a client for the consumption of services from a mobile device will be used. The development will take place on Android.

Keywords: app; cinema; movie; hoarding; network;

Tabla de contenidos

1. Introducción

Con la elaboración del TFG se pretende cubrir la necesidad de una aplicación móvil para buscar los cines cercanos y viendo su cartelera poder comprar las entradas directamente desde la aplicación.

Las aplicaciones han sufrido un cambio radical en cuanto a cantidad y uso. Desde la llegada de las primeras aplicaciones de escritorio, su evolución a aplicaciones web ya por 1987 cuando internet no era accesible al público general, Larry Wall creo el primer lenguaje para desarrollo web el “Perl” [1] pero no fue hasta 1995 cuando Rasmus Lerdorf creó el conocidísimo PHP [2]. Es aquí cuando el desarrollo de aplicaciones web despegó.

El desarrollo web derivó en el desarrollo de aplicaciones móviles, las primeras mundialmente conocidas datan de la década de los noventa como la aplicación del Tetris o el famoso Snake de Nokia. Más tarde en el 2000 se registró el protocolo WAP [3] (Wireless Application Protocol), un estándar abierto para aplicaciones que usan comunicaciones inalámbricas. Esto permitió una distribución mucho más rápida y global de ellas.

Aunque no fue hasta 2008 con el lanzamiento del App Store [4] y la primera publicación del SDK de Android y su respectivo Market cuando hubo un auge muy considerable en su uso.

El concepto que teníamos en usabilidad y utilidad de las primeras aplicaciones móviles hasta el día de hoy ha cambiado radicalmente. Los Smartphones son auténticos ordenadores capaces de realizar la mayoría de las tareas a nivel profesional, doméstico o de ocio. Con el aumento de capacidades de estos terminales ha aumentado también las acciones que pueden realizar las aplicaciones.

En estos momentos ningún empresario pensaría en dirigir una empresa sin usar un smartphone, en su móvil puede tener toda la información económica de la empresa, puede contactar con todos sus proveedores y clientes, es capaz de hacer un seguimiento simultaneo de su producción. Al igual que un estudiante puede leer su temario, contactar con profesores y otros alumnos, seguir su evaluación, recibir notificaciones. Antes debía acercarse a su facultad a leer en un tablón enorme para ver su nota, ahora lo hace con un vistazo rápido desde su smartphone.

Las posibilidades de las aplicaciones en un smartphone son infinitas y solo limitadas por la imaginación del desarrollador.

Este aumento de capacidades ha ido de la mano en el crecimiento de tecnologías para realizar las aplicaciones, mientras que en su inicio se podían contar con los dedos de la mano los lenguajes posibles ahora son muchísimos y creciendo cada año.

Por ejemplo, para crear nuestra aplicación tuvimos varias alternativas. Estaba la opción de crear una aplicación híbrida mediante algún sistema de diseño web, como puede ser JavaScript junto a HTML y PHP, después adaptándolo con alguna herramienta como

Apache, pero hemos preferido centrarnos en un lenguaje conocido como es Java para poder profundizar en él. Android ahora ofrece la opción de desarrollo en Kotlin y aunque parece interesante ya que promete más productividad haciendo las mismas funciones con menor líneas de código nuestro objetivo no era aprender un lenguaje nuevo durante la realización del TFG.

Actualmente en España existen aplicaciones de búsqueda de información de películas generalmente alguna empresa internacional o directamente se puede consultar a Google que tiene una respuesta rápida para estos casos, es decir, no es necesario entrar en ningún enlace, Google responde primero y detrás ya aparecen los enlaces.

Por supuesto, también existen aplicaciones para comprar entradas, pero esto está limitado a las propias compañías de cines que sacan sus propias aplicaciones. Lo que se presente es unificar las apps y poder ver la cartelera de los cines cercanos y sobre todo poder comprar las entradas desde una única aplicación. Estas aplicaciones y su uso tanto en España como Europa lo veremos más especificado en otra sección del proyecto.

1.1 Motivación

La principal motivación del proyecto es el interés en aprender el desarrollo de aplicaciones móviles, más concretamente en Android, ya que es el sistema operativo móvil más usado en la actualidad y con mayor crecimiento que el resto de sistemas como puede ser iOS.

La elección del tema de la aplicación es la necesidad propia de usar una aplicación capaz de unificar la compra de entradas y mi interés personal por el mundo del cine.

1.2 Objetivos

El objetivo de crear este proyecto es la idea de realizar algo que pudiera ser accesible para todo el mundo. Con esto surgió hace años la idea de realizar una aplicación móvil. Se hizo diferentes pruebas, aplicaciones muy básicas y esto es el inicio a la realización de una aplicación más profesional.

Esto se unió a la pasión por el cine y surgió la idea de este proyecto.

Como objetivos técnicos tenemos:

- La realización de una aplicación para Android por la estima que tenemos a este sistema operativo.
- El uso de Java por la necesidad de profundizar en este lenguaje.
- Conectar con una Api de cine.
- Almacenar información, algo imprescindible en este tipo de aplicaciones. Además, es interesante analizar las diferentes formas en las que se puede realizar.

1.3 Estructura

El proyecto se va a estructurar en 6 apartados. El primero es la introducción anteriormente descrita, el segundo es el Estado del Arte donde describiremos las aplicaciones que hay actualmente en el mercado español viendo las características de cada una de ellas y si nos pueden interesar o no en nuestro propio proyecto, además se verá a partir de las alternativas una propuesta conjunta. En tercer lugar, se desarrollará la propia estructura de la aplicación. El cuarto es la implementación, de describirán las tecnologías usadas y las decisiones más importantes en cuanto a desarrollo y elección de herramientas. El quinto será el resultado de la aplicación, es decir imágenes reales de lo que se ha desarrollado. Por último, el sexto será la conclusión del proyecto junto lo aprendido y su relación con los estudios cursados.

2. Estado del Arte

Podemos encontrar multitud de aplicaciones en el Play Store con el objetivo de mostrar los últimos estrenos, seleccionar el cine que queramos y comprar las entradas deseadas. También se pueden encontrar aplicaciones más orientadas a una red social con el tema principal del cine. Lo extraño es encontrar una aplicación que combine ambas.

2.1 Aplicaciones similares

Por un lado, aplicaciones para comprar entradas tenemos que destacar las Apps propias de cada empresa como pueden ser 'Yelmo Cines App[1]' o 'Cinesa: Cartelera de películas'[2]. Con estas Apps el inconveniente principal es que solo sirven para sus propios cines, no permiten la colaboración con ninguna empresa externa.

Por otro lado, las aplicaciones que funcionan como red social destacamos 'IMDb Cine & TV[3]' y 'SensaCine – Cine y Series'[4]. Tienen gran información sobre películas, actores, directores y todo el entorno del séptimo arte, pero no tienen la posibilidad de adquirir entradas a través de ellas.

Empecemos por desarrollar las relacionadas con adquisición de entradas.

Yelmo Cines App

Podemos encontrar principalmente tres secciones.

La primera nos muestra la sección 'Películas' que nos muestra la cartelera. Cuando seleccionamos en cualquiera de las películas nos enseña un tráiler y en dos subapartados los horarios de las salas cercanas a nuestra ubicación y un pequeño resumen de esta.

La segunda es 'Productos' donde nos muestra todos los productos alimenticios que podemos precomprar antes de nuestra llegada al local para consumir durante nuestro pase. Esta característica está en implementación y en la mayoría de sus cines no está disponible.

[1] https://play.google.com/store/apps/details?id=es.yelmocines.geopush&hl=es
[2] https://play.google.com/store/apps/details?id=com.codiwans.cinesa&hl=es
[3] https://play.google.com/store/apps/details?id=com.imdb.mobile&hl=es
[4] https://play.google.com/store/apps/details?id=com.sensacine.android&hl=es

La tercera es ‘Cines’ donde mediante localización GPS nos muestra los cines que tengamos más cercanos. Podemos seleccionar uno para que nos redirija a Google Maps o para ver su cartelera.

Después hay otras dos secciones que sirven para la asistencia técnica por si surge cualquier problema con la compra y el apartado de perfil donde podemos ver compras pasadas y nuestras tarjetas asociadas.

Cinesa: Cartelera de películas

Podemos decir que Cinesa básicamente usa la misma estructura que Yelmo, pero con un aspecto más cuidado. Sustituye el apartado de ‘Productos’ por uno de promociones y tiene más información tanto de las películas como de los cines, aunque nada realmente destacable.

Ahora hablemos de las relacionadas con las redes sociales.

Primero debemos destacar que estas redes son redes sociales verticales, es decir, las que están centradas en un tema específico y por tanto sus usuarios tienen al menos un gusto común, en este caso el cine.

IMDb Cine & TV

Perteneciente a Amazon, IMDb International Movie Data base como su propio nombre indica es más bien una Base de datos global de todo el contenido de series y cine, es una herramienta muy poderosa que no solo posee información de películas sino una gran cantidad de usuarios que puntúan y critican toda obra cinematográfica. Además, contiene novedades, rumores, cifras de taquilla y producción entre otras cosas.

El usuario puede crear listar de contenido por ver o visto, compartirlo, ver los últimos estrenos, la cartelera, buscar casi cualquier obra, ver avances, seleccionar cines favoritos y más.

SensaCine – Cine y Series

Con muchos menos usuarios que IMDb se ha hecho con un hueco importante en este sector, al menos a nivel europeo. Aunque dispone de menos material prácticamente tienen las mismas características. Su ventaja es que al no pretender albergar tanto contenido se muestra de una forma más clara y simple.

A continuación, comprobamos a través de una tabla MoSCoW las características que nos interesa de cada aplicación.

Tabla 1: Tabla MoSCoW comparativa.

	Must	**Should**	**Could**	**Won't**
Yelmo Cines App	-Lista de los últimos estrenos. -Mapa donde nos muestra los cines más cercanos. -Adquisición de entradas desde la App.	-Página de perfil del usuario. -Ver las sesiones disponibles de los cines cercanos.	-Página de asistencia técnica. -Tráiler de películas.	-Productos alimenticios que tiene el cine.
Cinesa: Cartelera de películas	-Lista de los últimos estrenos. -Mapa donde nos muestra los cines más cercanos. -Adquisición de entradas desde la App.	-Página de perfil del usuario. -Valoración de películas. -Ver las sesiones disponibles de los cines cercanos.	-Tráiler de películas.	-Información detallada sobre los diferentes tipos de salas.
IMDb Cine & TV	-Lista de los últimos estrenos. -Información de las películas. -Poder comentar películas.	-Valoración de películas.	-Información de los actores y demás miembros que participan. -Tráiler de películas. -Crear listas sobre contenido que ya he visto o deseo ver.	-Información sobre series o TV. -Últimas noticias. -Información detallada de costes de producción. -Información sobre premios y eventos relacionados. -Rumores o cotilleos sobre famosos.
SensaCine – Cine y Series	-Mapa donde nos muestra los cines más cercanos. -Lista de los últimos estrenos. -Comentarios de películas.	-Valoración de películas. -Ver las sesiones disponibles de los cines cercanos.	-Tráiler de películas.	-Información sobre Bandas Sonoras originales. -Información sobre series.

2.2 Propuesta

A partir de las características anteriormente descritas y viendo la tabla 1 de comparación podemos tener una idea clara de que queremos que tenga nuestra aplicación. En primer lugar, queremos una lista de las películas que estén actualmente en cartelera. De estas películas pretendemos que se muestre una información más detallada de ellas.

Como queremos un aspecto de red social, la idea es que los usuarios puedan mostrar su parecer de estas películas, por tanto, deben tener la capacidad de poder comentar las películas y valorarlas.

Además, la idea es que puedan ver desde cualquier parte del mundo donde están situados los cines más cercanos. Se añadirá un mapa para que el usuario a partir de su ubicación vea que cines tiene más próximos.

Por último, se pretende que el usuario sea capaz de ver la cartelera de un cine en partículas, su horario y que pueda comprar las entradas desde la misma aplicación.

3. Especificación

A continuación, se describirán las diferentes capas de las que se compone el proyecto empezando por la capa de negocio, justo después de describirá la capa de presentación para relacionarla con la capa anterior. Por último, la capa de persistencia.

3.1 Capa de Negocio

En la capa de Negocio describiremos los diferentes casos de uso que tiene la aplicación.

3.1.1 Visionar últimos estrenos

Tabla 2 : Caso de uso 1

Caso de Uso 1	Visionar últimos estrenos.
Objetivo	Ver una lista de las últimas películas estrenadas.
Descripción	El usuario podrá ver fácilmente una lista detallada de los últimos estrenos, con la portada, el nombre y una breve descripción.
Precondición	Ninguna.
Secuencia Normal	1.Entrar en la aplicación. Es la página por defecto de la app.
Postcondición	Ninguna.

3.1.2 Visionar detalle película

Tabla 3: Caso de uso 2

Caso de Uso 2	Visionar detalle película.
Objetivo	Ver el detalle de la película seleccionada.
Descripción	El usuario podrá ver la descripción detallada de la película
Precondición	Estar en la página principal.
Secuencia Normal	1. Seleccionar una película de la lista
Postcondición	Ninguna.

3.1.3 Agregar comentario

Tabla 4 : Caso de uso 3

Caso de Uso 3	Agregar comentario
Objetivo	Poner un comentario.
Descripción	El usuario podrá comentar públicamente la película que desee para así compartir su opinión con el resto de los usuarios.
Precondición	Estar en la página principal. Estar previamente logueado.
Secuencia Normal	1. Seleccionar una película de la lista. 2. Seleccionar el TextBox. 3. Escribir comentario. 4. Pulsar el button Save.
Postcondición	Ninguna.

3.1.4 Iniciar sesión/Registrase

Tabla 5: Caso de uso 4

Caso de Uso 4	Iniciar Sesión/Registrarse
Objetivo	Iniciar sesión/Registrarse
Descripción	El usuario podrá registrarse o iniciar sesión en Facebook, Google o correo electrónico.
Precondición	No haber iniciado sesión con anterioridad.
Secuencia Normal	1. Entrar en el menú lateral izquierdo. 2. Seleccionar la sección Cuenta. 3. Pulsar el tipo de cuenta con el que queramos registrarnos. 4. Según el tipo hacer el proceso propio de cada uno.
Postcondición	El usuario queda registrado en nuestra base de datos y/o el usuario ha iniciado sesión y puede realizar más acciones.

3.1.5 Ver cines cercanos

Tabla 6: Caso de uso 5

Caso de Uso 5	Ver cines cercanos
Objetivo	Ver cines cercanos
Descripción	El usuario podrá ver los cines próximos a su ubicación.
Precondición	Aceptar el acceso a tu ubicación
Secuencia Normal	1. Entrar en el menú lateral izquierdo. 2. Seleccionar la sección Maps. 3. Pulsar el botón de localización que aparece arriba a la derecha
Postcondición	Ninguna.

3.1.6 Ver cartelera cine

Tabla 7: Caso de uso 6

Caso de Uso 6	Ver cartelera cine
Objetivo	Ver cartelera cine
Descripción	El usuario podrá ver las películas disponibles en el cine seleccionado.
Precondición	Aceptar el acceso a tu ubicación.
Secuencia Normal	1. Entrar en el menú lateral izquierdo. 2. Seleccionar la sección Maps. 3. Pulsar el botón de localización que aparece arriba a la derecha 4. Seleccionar el marcador del cine. 5. Pulsar en la ventana de información que aparece al pulsar el marcador.
Postcondición	Ninguna.

3.1.7 Comprar entrada

Tabla 8: Caso de uso 7

Caso de Uso 7	Comprar entrada
Objetivo	Comprar entrada
Descripción	El usuario podrá comprar la entrada de la película y cine seleccionado en el horario elegido.
Precondición	Aceptar el acceso a tu ubicación. Haber iniciado sesión.
Secuencia Normal	1. Entrar en el menú lateral izquierdo. 2. Seleccionar la sección Maps. 3. Pulsar el botón de localización que aparece arriba a la derecha 4. Seleccionar el marcador del cine. 5. Pulsar en la ventana de información que aparece al pulsar el marcador. 6. Pulsar en el botón con la hora a la que queramos ir.
Postcondición	Ninguna.

3.2 Capa de Presentación

En la capa de presentación veremos los mock-up correspondientes a cada uno de los casos de uso descritos anteriormente. Se unificarán aquellos que resulten repetitivos.

3.2.1 Visionar últimos estrenos, 3.2.2 Visionar detalle película y 3.2.3 Agregar comentario

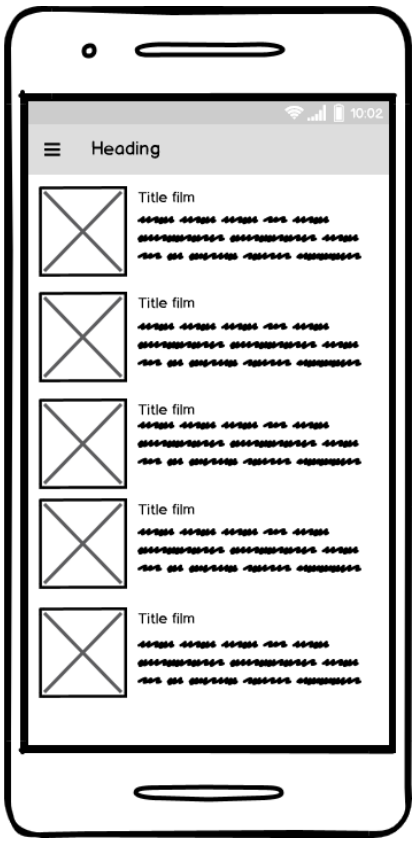


Figura 1: Mock-up pantalla inicial

En este mock-up vemos la ventana principal de la aplicación. Vemos como tiene visible el menú con sus líneas indicadores y el nombre de la pestaña en la cabecera. Aparecen una listview de películas con su imagen correspondiente, el título y una breve descripción. La vista es desplazable, por tanto, podemos deslizar hacia arriba para ver el resto de información disponible.

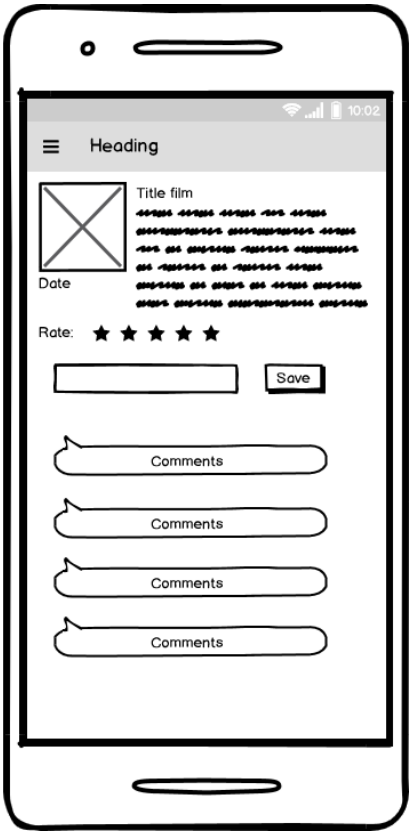


Figura 2: Mock-up detalle película

En la figura 2 vemos que como se ha descrito en el caso de uso, al pulsar en una película del listview nos aparece su información detallada. A parece la imagen, título y descripción de antes, a esto se le añade la fecha de estreno y su valoración en formato de 5 estrellas.

Desde esta ventana observamos que también es posible añadir un comentario, escribiremos en el TextBox, pulsamos el Button “Save” y se añadirá a la lista de comentarios con nuestro nombre e imagen de perfil. (esto no aparece en el mock-up por simplificarlo y quede más claro).

3.2.4 Iniciar sesión/Registrase

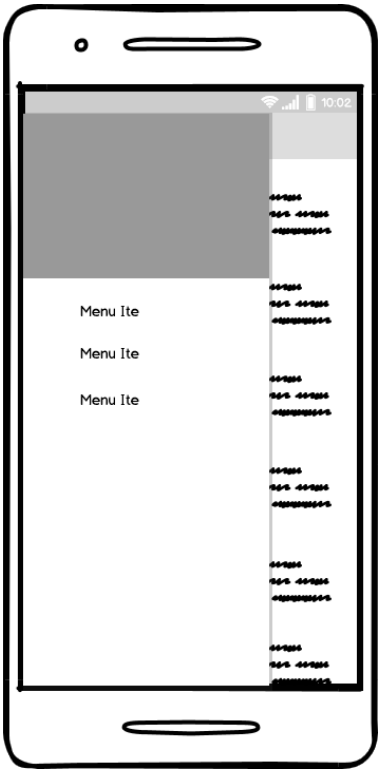
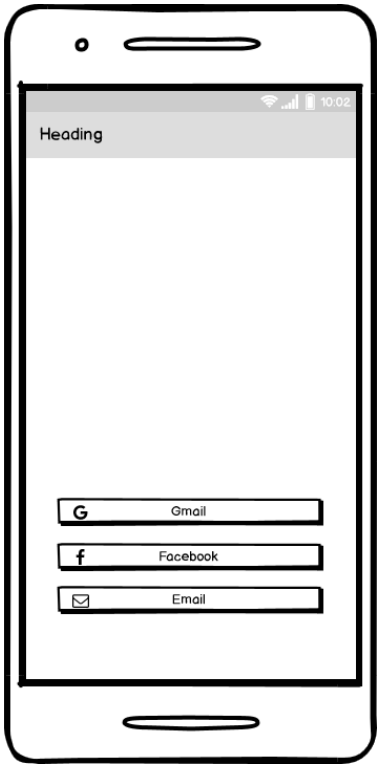


Figura 3: Mock-up menú

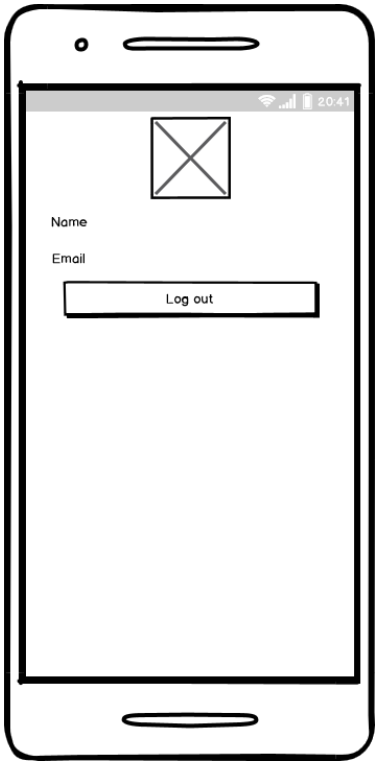
En la figura 3 vemos una idea de cómo será el menú lateral, en la parte superior izquierda estará el icono de la aplicación, su nombre y el nombre de la compañía.

En los elementos puestos como Menú Ite serán “Popular”, “Map” y “Account” para poder acceder a las películas, al mapa de cines cercanos y a la información de nuestra cuenta.



En esta pestaña (Figura 4) vemos el menú de inicio de sesión/ registro. Como se puede observar y se ha descrito con anterioridad, habrá tres opciones Gmail, Facebook y Email. Al pulsar el sistema verá si está registrado ya en nuestra base de datos, en caso de ser así simplemente iniciará sesión y en caso contrario habrá un proceso de registro que acabará igualmente con un inicio de sesión.

Figura 4: Mock-up inicio sesión



En la figura 5 vemos el resultado de inicio de sesión por parte del usuario. Aparecerá el nombre del usuario, su email y la imagen asociada a su cuenta.

Además, habrá un Button de “Log out” por si el usuario prefiere no estar logueado o desea usar otra cuenta.

Figura 5: Mock-up perfil

3.2.5 Ver cines cercanos y 3.2.6 Ver cartelera cine

Figura 6: Mock-up mapa

Al seleccionar el menú lateral, descrito en la figura3, la opción “Map” nos aparecerá un mapa con un botón en la esquina superior derecha. Al pulsarlo nos llevará a nuestra ubicación marcada con un punto junto a los marcadores de los cines más cercanos.

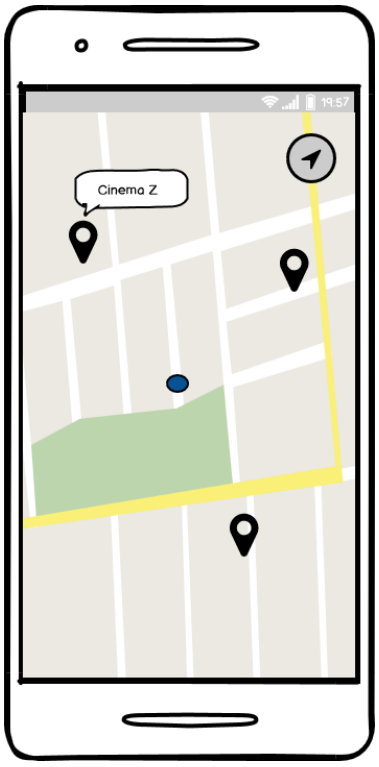
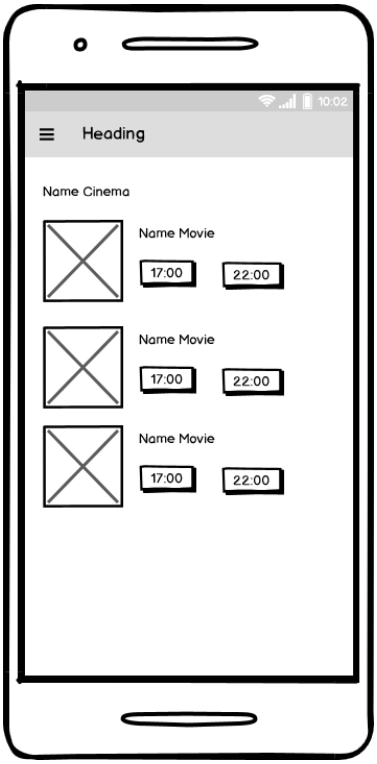


Figura 7: Mock up información marcador

La figura 7 es igual a la 6 con la diferencia de que se muestra la información del marcador. Esta información aparece cuando el usuario pulsa el marcador.



En la figura 8 vemos la ventana que aparece cuando el usuario pulsa encima de la información del marcador de cine. Aparece el nombre del cine y debajo la información de las películas que tiene disponible en cartelera. El nombre, la portada y el horario disponible. El horario está puesto en Buttons para pulsar y poder comprar la entrada en el horario seleccionado.

Figura 8: Mock-up cartelera

3.3 Capa de Persistencia

En este apartado se mostrará un esquema entidad-relación para mostrar de una forma clara como se administran los datos. En nuestro proyecto básicamente se almacenarán dos tipos de datos. El primero de ellos será la información detallada sobre películas y el segundo los datos sobre los cines cercanos.

En este apartado se mostrará solo el esquema y una pequeña explicación sobre ella. Especificaremos en el apartado 4.4 los atributos de cada clase y como hemos manejado y obtenido la información.

3.3.1 Información películas

Vamos a observar el diagrama entidad-relación de toda la información que se guarda sobre las películas.

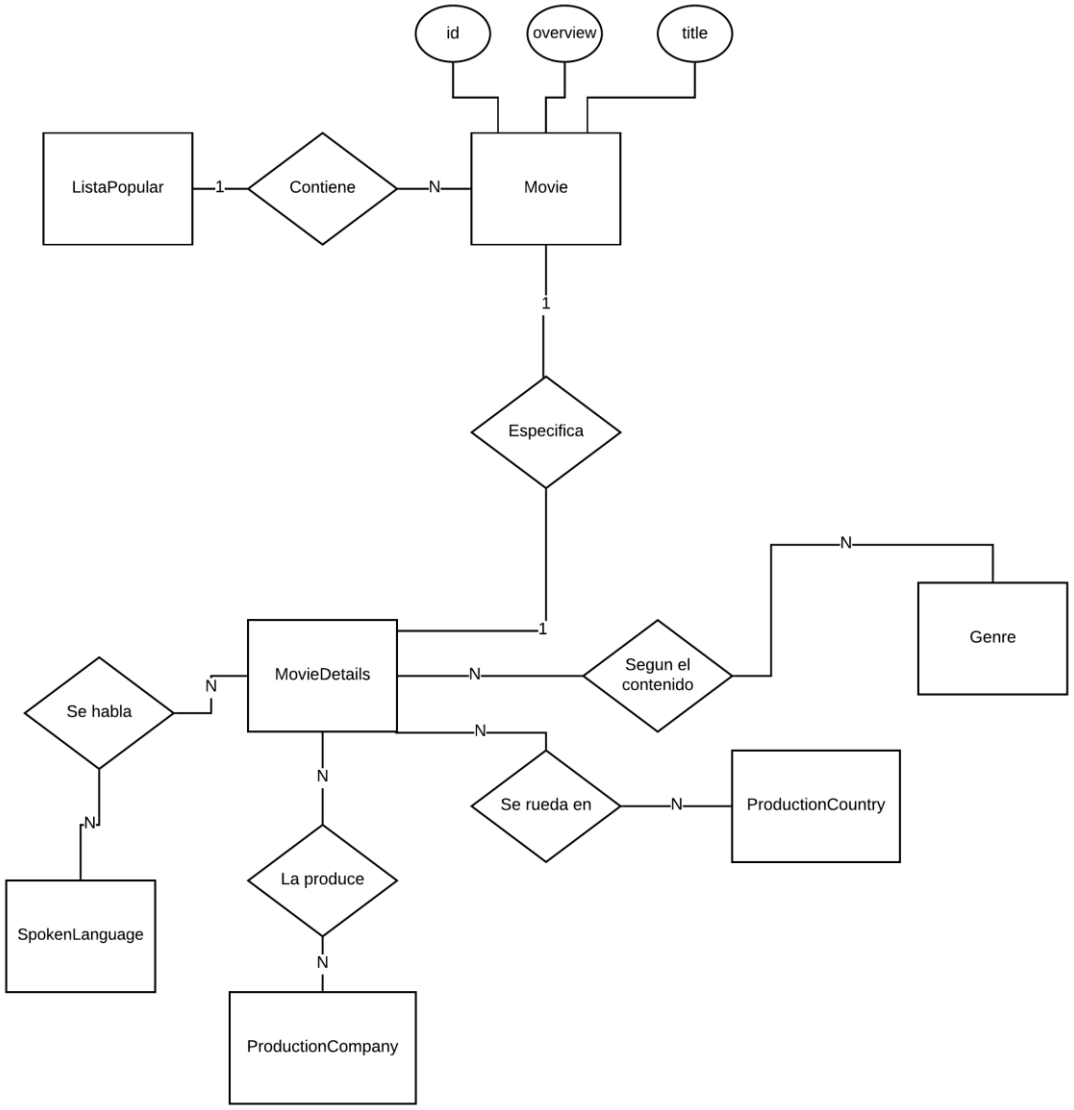


Figura 9: Diagrama entidad-relación información películas

Vemos que la relación empieza desde ListaPopular donde hay un List de Movie, la clase Movie contiene la información básica sobre cada película y por tanto no es necesario, si no queremos detallar, el resto de las clases. Si queremos especificar debemos partir de la clase MovieDetails relacionada con Movie mediante el ID de la película.

MovieDetails deriva en otras clases cuando la información tiene más de un valor. Como vemos en cualquiera de ellas puede tener varios valores, puede haberse rodado en

varias lenguas, puede tener diferentes géneros, ha podido ser producido por varias productoras o puede haberse rodado en varios países. No vamos a entrar en detalle de las clases derivadas de MovieDetails porque, aunque estén implementadas al ser esto una versión inicial de la aplicación tan solo se ha usado los atributos de la clase Movie. Como se ha dicho anteriormente estos atributos se detallarán en el apartado 4.4.

3.3.2 Información Places

Vemos como almacenamos en los diferentes tipos de Objetos la información de los sitios obtenidos a partir de Google Places.

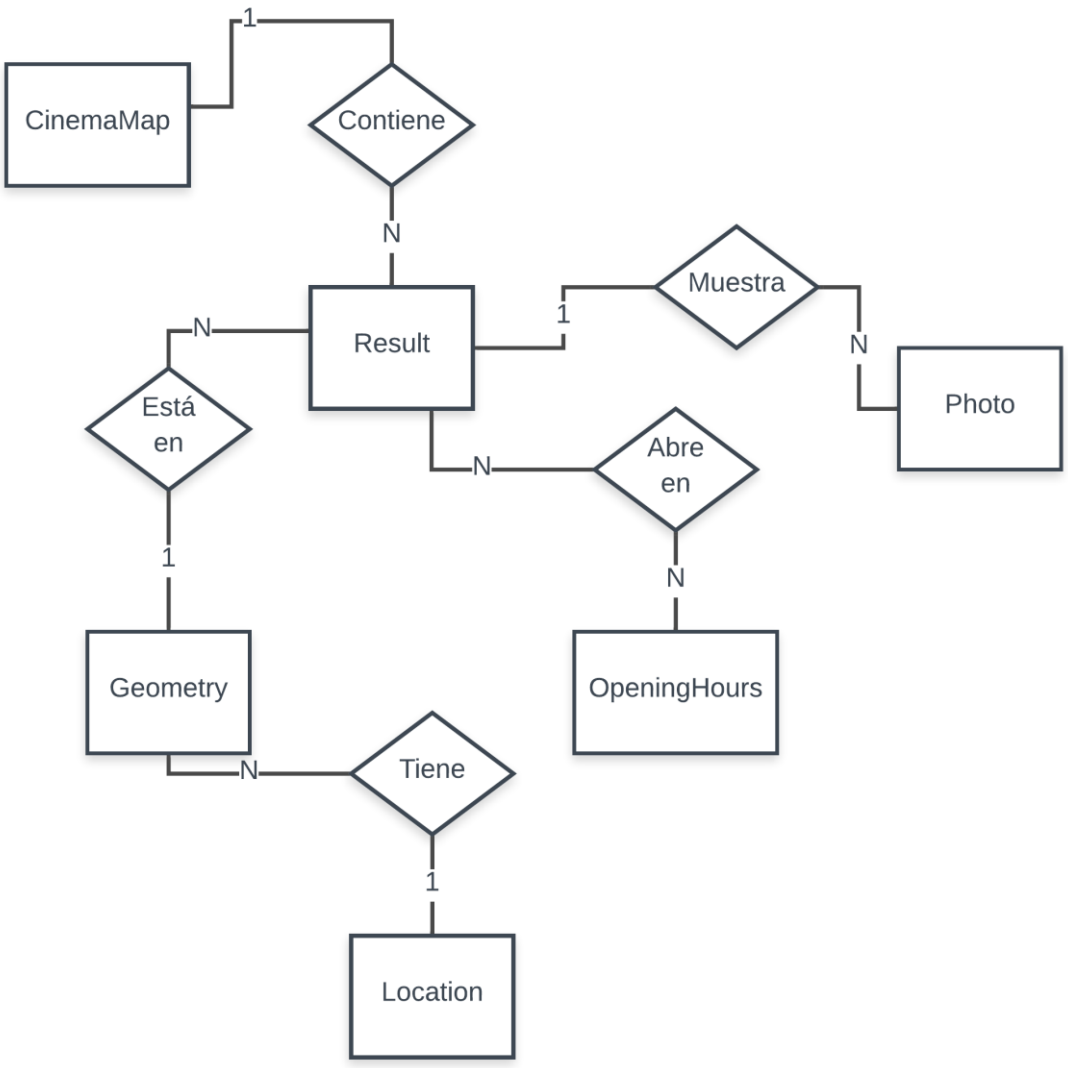


Figura 10: Diagrama entidad-relación información mapa

CinemaMap obtiene una list de Result, la clase Result contiene toda la información que nos puede proporcionar Google Places sobre cierta ubicación. Como podemos ver en el diagrama la clase Result tiene como atributos a parte de los que se mencionarán en el apartado 4.4 Objetos de las clases Geometry, OpeningHours y Photo. A su vez

Geometry contendrá un único atributo, un objeto de la clase Location, Location tiene dos atributos Longitud y Latitud, necesarios para ubicar el sitio que deseemos.

4. Implementación

En este apartado desarrollaremos las tecnologías, herramientas y decisiones de implementación utilizadas en el proyecto para su correcto desarrollo. Por último, se explicará la arquitectura usada para la comunicación con las APIs.

4.1 Tecnologías

4.1.1 Firebase Authentication

Esta tecnología nos permite de forma sencilla guardar los usuarios de nuestra aplicación en la nube de forma segura. Nos proporciona servicios de backend, SDK y bibliotecas de IU para autenticar a los usuarios desde la propia app.

Permite autenticarse desde diferentes tipos de cuenta como pueden ser Facebook, Twitter, Google, email, teléfono o Microsoft entre otros.

4.1.2 Firebase Database

Nos permite sincronizar y almacenar datos en tiempo real. Funciona como una base de datos NoSQL.

4.1.3 Java

Java es uno de los lenguajes Orientado a Objetos más usado del mundo. Una de sus características más importantes es que al compilarse en bytecode puede ser ejecutado en cualquier máquina virtual Java (JVM) sin importar la arquitectura del dispositivo donde se ejecute.

4.1.4 XML

XML (Lenguaje de Etiquetado Extensible) es uno de los formatos más utilizados para el intercambio de información. Es un lenguaje etiquetado que se usa para describir información. En Android es usado para el diseño de Layouts porque es un lenguaje ligero y hace que nuestros diseños no pesen demasiado.

4.1.5 JSON

JSON (JavaScript Object Notation) es un formato de intercambio de información ligero. Se caracteriza por su simple estructura de pares relacionando cada nombre con un valor, es decir un objeto.

4.1.6 The Movie Database (TMDb) API

The Movie Database como su propio nombre indica es una base de datos de películas y series. La fuerza de esta plataforma es que ha sido construida por su propia comunidad y todos los datos que tiene proceden de ella. A partir de estos datos han construido una API accesible con la que cualquier desarrollador pueda manejar toda la información que esta comunidad posee, por ejemplo, información de películas, actores, o últimos estrenos.

4.1.7 Maps SDK for Android

Gracias a Maps SDK se pueden añadir mapas basados en Google Maps en nuestra aplicación. La API automáticamente enlaza a los servidores de Google Maps, esto nos permite el acceso a toda la información de Google Maps y a que el usuario pueda interactuar con el mapa de la aplicación.

4.1.8 Places API

Places API es un servicio que nos devuelve información de establecimientos, localizaciones geográficas o puntos de interés a través de peticiones HTTP.

4.2 Herramientas

4.2.1 Android Studio

Ilustración 1: Android Studio

Android Studio es una plataforma de desarrollo creada por Google para el desarrollo exclusivo de aplicaciones en Android tanto en Java como en Kotlin, tiene incorporado un emulador de los diferentes sistemas operativos que han salido puestos en el dispositivo que nosotros creamos más adecuado. A parte de desarrollo para smartphones se puede desarrollar para cualquier plataforma que use Android como pueden ser Smart TV, relojes o coches.

Existen otras alternativas como Eclipse que en un pasado eran más recomendables, pero Google ha puesto mucho interés en mejorar su plataforma.

4.2.2 Google Cloud Platform

Ilustración 2: Logo Google Cloud Platform

Google Cloud Platform [5]es la plataforma que unifica todos los productos, servicios y herramientas de Google para el desarrollo y testing de aplicaciones.

4.2.3 Firebase

Ilustración 3: Logo Firebase

Firebase[6] es una plataforma que une diferentes herramientas en las que podemos delegar parte el back-end, monitorizar su uso, monetizar los anuncios en caso de que los pongamos y escalar fácilmente nuestra aplicación en caso de un crecimiento de nuestros usuarios.

4.3 Decisiones Implementación

En este apartado se explicará el porqué de algunas de las herramientas y tecnologías explicadas anteriormente, así como las alternativas disponibles. Además de explicar varios dilemas en el desarrollo.

4.3.1 Herramientas y Tecnología

No se puede tratar de explicar sin relacionar la tecnología usada con su herramienta.

Android y Java

La primera decisión lógica después del tema de la aplicación era decidir a qué tipo de sistema queríamos dirigir la aplicación. Había muchas alternativas, la más tentadora era hacer una aplicación híbrida con algún sistema como Angular o Ionic que permitiera la flexibilidad a través de programas como Apache de crearla para Android, iOS o navegador. Aun teniendo experiencia en ellas se decidió rechazarla por la falta de interés en un futuro de profundizar en estas tecnologías. Solo quedaban dos opciones una aplicación nativa en Android o en iOS. Esta decisión fue mucho más simple por el conocimiento previo en Java y desarrollo en Android así que rápidamente se descartó iOS.

[5] https://cloud.google.com/
[6] https://firebase.google.com/?hl=es-419

Como se ha dicho anteriormente por la experiencia en Java se descartó usar Kotlin, aunque es un lenguaje que un futuro se debería al menos tener muy en cuenta.

Ahora que ya estaba claro para que Sistema Operativo desarrollar y en que lenguaje quedaba decidir el entorno de desarrollo. Las alternativas principales para desarrollar en Android nativo son Eclipse y Android Studio. Mientras que Eclipse fue de las primeras en tener un sistema fiable siendo incluso el IDE oficial de desarrollo en Android se ha quedado estancado sin tener un progreso sustancial desde hace años. A su vez Google ha puesto un empeño increíble en seguir mejorando su propio sistema que se ha convertido en un perfecto aliado en el desarrollo de Android, dando soporte nativo a muchas librerías propias, mejorando continuamente el emulador, varias plantillas que ayudan a una mayor rapidez y estandarización, consola de desarrollador entre otras características.

Firebase

Cuando fue decidido todo lo anteriormente descrito se pensó en un sistema backend que permitiera almacenar diferente tipo de información. La alternativa más presente fue el desarrollo de un Servidor en Node, incluso de desarrolló la fase inicial de él, pero esto daba muchos problemas de escalabilidad, de desarrollo y de infraestructura así que se optó por una alternativa adecuada a las características que buscábamos para nuestra aplicación.

Se eligió Firebase porque posee dos tecnologías que son perfectamente adecuadas para nuestros requisitos. La primera Firebase Auth que nos ayuda a despreocuparnos por almacenar y proteger la información de nuestros usuarios y segundo Por Firebase Database un sistema de base de datos NoSQL a tiempo real que nos permite acceder a la información en la nube desde cualquier parte.

Además, Firebase está íntegramente relacionada con Google por tanto es una gran alternativa si se está desarrollando en Android.

Google Cloud Platform

Una de las características que queríamos que tuviera nuestro proyecto fue sin duda que el usuario pudiera acceder a un mapa que le indicara los cines más próximos. La herramienta más fuerte para ello es Google Maps. Para poder usar la información de Google Maps hay que usar Maps SDK for Android y esto solo es accesible con una clave que proporciona Google Cloud Platform. Anteriormente Maps SDK era independiente y de acceso gratuito, al haberse integrado con Google Cloud se ha convertido de pago, pero al menos dejan un año gratuito para la fase desarrollo.

Cuando ya se tuvo acceso a Maps se vio que no estaba tan completo como creíamos así que para completar la información necesaria solo había una opción Places API, que también está integrado en Google Cloud.

The Movie Database (TMDb) API

Con la idea de obtener información de todas las películas se buscó diferentes APIs que nos sirvieran. Hay varias alternativas como The Open Movie Database (OMDb) de acceso gratuito y otras de pago. Se descartó las de pago por motivos obvios, siendo esto un trabajo únicamente académico.

Se decidió por TMDb por su acceso gratuito, sencillez en las consultas, elevada información y su amplia documentación.

4.3.2 Desarrollo

Hay varias decisiones importantes en cuanto a desarrollo. Desde la creación del propio proyecto hasta decisiones que se van tomando a medida que se va encontrando el problema.

Versión Android

Desde su creación se han ido renovando periódicamente las versiones de Android. Una decisión importante es para que versión de Android se quiere dirigir nuestra aplicación.

Cada versión tiene ventajas e inconvenientes. La mayor ventaja es poder usar las últimas novedades/características, pero si se usa una versión demasiado moderna hace que muchos dispositivos no puedan usarla.

Uno de los grandes problemas de Android [5], que por ejemplo su principal competidor-iOS- no sufre,[6] es la fragmentación que existe entre sus diferentes versiones. Aunque es cierto que no es justo compararlo con Apple porque es una única marca con pocos dispositivos sigue siendo un problema muy grave para los desarrolladores y el propio Google.

Teniendo en cuenta la siguiente tabla que Android Studio nos proporciona con los datos oficiales de fragmentación se decidió que la versión ideal era la 6.0 porque pueden acceder la mayoría de los dispositivos y no sacrificamos en exceso las últimas características.

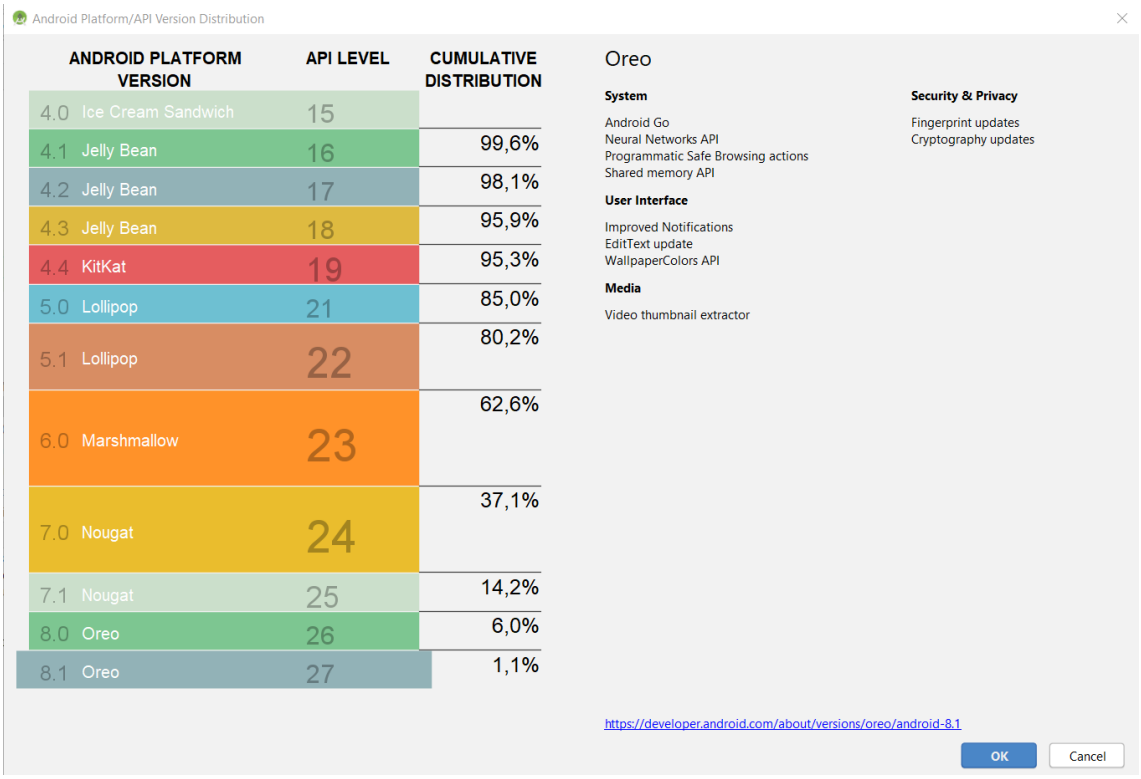


Ilustración 4: Tabla explicativa fragmentación android

Aunque se decidió por dar disponibilidad por la versión 6.0 o lo que es lo mismo el SDK 23, la aplicación esta compilada en SDK 28, es decir, la última versión disponible Android 9.0. Viendo la tabla observamos que a nivel comercial carece en un principio de sentido poner los esfuerzos en últimas versiones, pero esto está hecho a nivel académico así que esto nos permite que Android Studio nos indique que clases o métodos están obsoletos y cuáles son los nuevos.

```
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.cinepolio.marcos.cinemapp"
        minSdkVersion 23
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
```

Ilustración 5: Fragmento código de Gradle

AndroidX

Android Jetpack también llamado AndroidX [7] es la nueva estructura de paquetes diseñados para Android. Se decidió el cambio porque tiene nominaciones más simples, acceso a nuevas librerías y por su retrocompatibilidad.

JSON

Debatimos entre la clase JsonReader y Gson. Ambas tienen el mismo propósito que es convertir información JSON en objetos Java y viceversa. Se optó por Gson porque es mucho más completa y simple que JsonReader que tiene el inconveniente que hay que implementar de forma manual la lectura de los datos mientras que Gson lo realiza de forma más automática.

4.3.3 Características

En este apartado no se pretende explicar las características que tiene la aplicación si no las que no puede tener por diferentes motivos legales.

Se ha tratado de proporcionar a cada cine su cartelera, el problema es que esa información no es pública y a no ser que se haga mediante técnicas de scraping no se puede obtener. Es aquí el primer inconveniente de por qué el resto de aplicaciones en el mercado no disponen de un sistema de ventas si el cine no es de su propia empresa.

El segundo inconveniente y razón principal es que no es posible la compra de entradas si no se llega a un acuerdo comercial con estas compañías cosa que en España no ha ocurrido con ninguna empresa.

Por tanto, habiendo buscado alternativas sin la colaboración de las empresas de cines es imposible implementar estas características. Lo que se ve en la aplicación es un sistema que emula hacerlo porque técnicamente es posible pero no legalmente. Se ha planteado la opción de redirigir a la web de la empresa correspondiente, pero este no es el objetivo de este proyecto y por tanto se ha descartado.

4.4 Arquitectura

Comentaremos el proceso de comunicación entre la aplicación y las APIs tanto de TMDb, FireBase y Google Cloud.

[7] https://developer.android.com/jetpack?hl=es-419

4.4.1 TMDb

Se hace una petición HTTP a la API con la URL correspondiente [7]. Esto se hace a través de Gson que transforma de forma interna el JSON en un objeto Java, en este caso el Objeto ListaPopular, una clase que hemos creado con anterioridad.

```
    CineMappSingleton.getInstance(getContext()).addToRequestQueue(
            new GsonRequest<ListaPopular>(
                    url: URL_BASE+URL_JSON,
                    ListaPopular.class,
                    headers: null,
                    new Response.Listener<ListaPopular>(){
                        @Override
                        public void onResponse(ListaPopular response) {
                            items = response;
                            notifyDataSetChanged();
                        }
                    },
                    (error) → {
                            Log.d(TAG, msg: "Error Volley:"+ error.getMessage());
                    }
            )

    );
}
```

Ilustración 6: Fragmento código PostAdapter.java

De esta imagen habría que comentar lo que hace CineMappSingleton.

Se ha desarrollado esta clase para usar el patrón Singleton. Este patrón consiste en la necesidad en que solo haya una instancia de una determinada clase. Solo se creará la instancia si no existe y guardará su referencia para devolverla.

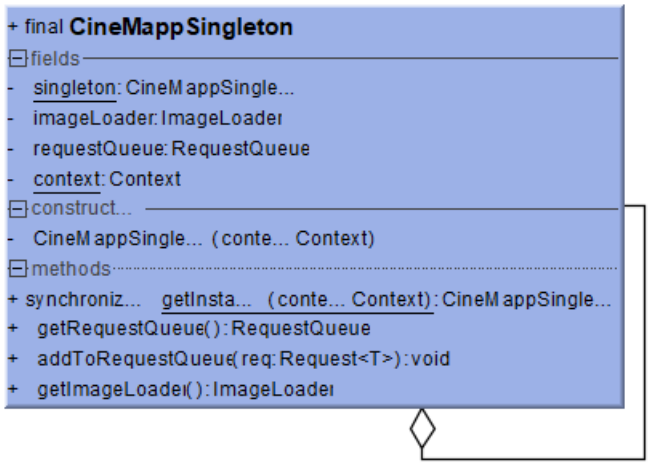


Ilustración 7: Diagrama CineMappSingleton

También comentar que en este caso la URL base es esta: https://api.themoviedb.org/3/movie/popular?api_key=<<api_key>>&language=en-US&page=1

Esto nos responderá un JSON con esta estructura.

Responses application/json

200 401 404 Schema Example collapse all

object		
page	integer	optional
results	array[object] {Movie List Result Object}	optional
poster_path	string or null	optional
adult	boolean	optional
overview	string	optional
release_date	string	optional
genre_ids	array[integer]	optional
id	integer	optional
original_title	string	optional
original_language	string	optional
title	string	optional
backdrop_path	string or null	optional
popularity	number	optional
vote_count	integer	optional
video	boolean	optional
vote_average	number	optional
total_results	integer	optional
total_pages	integer	optional

collapse

Ilustración 8: Tabla explicativa esquema JSON. Origen https://developers.themoviedb.org/3/movies/get-popular-movies

Como podemos comprobar la clase ListaPopular tendrá los siguientes atributos.

Tabla 9: Atributos ListaPopular

Integer	page
List<Movie>	results
Integer	totalResults
Integer	totalPages

De Aquí solo nos interesa comentar results que es un List de la clase Movie que tendrá los siguientes atributos:

Tabla 10: Atributos Movie

String	posterPath;
Boolean	adult;
String	overview;
String	releaseDate;
List<Integer>	genreIds
Integer	id;
String	originalTitle;
String	originalLanguage;
String	title;
String	backdropPath;
Double	popularity;
Boolean	video;
Double	voteAverage;

Como vemos en la tabla 10 la clase Movie ya posee los atributos necesarios para poder usar la información que necesitemos.

4.4.2 FireBase

FireBase funciona de una forma diferente. Hay que integrar su biblioteca a nuestro proyecto y así poder usar todas sus clases.

```
implementation 'com.google.firebase:firebase-auth:16.0.4'
implementation 'com.firebaseui:firebase-ui-auth:4.1.0'
implementation 'com.google.firebase:firebase-database:16.0.3'
```

Ilustración 9: Fragmento código AndroidManifest.xml

Como vemos se han añadido 3 bibliotecas diferentes. Firebase-auth [8] para poder guardar y recibir los datos de los usuarios. Firebase-ui-auth para poder integrar la UI de Firebase y así integrar el inicio de sesión propio de cada plataforma como por ejemplo Google y Facebook y por último Firebase-database [9] para poder acceder a las clases de la Base de Datos NoSQL.

Empezando por Auth se usan las librerías:

- com.google.firebase.auth.FirebaseAuth;
- com.google.firebase.auth.FirebaseUser;

para obtener o guardar los usuarios.

Y las librerías:

- com.firebase.ui.auth.AuthUI;
- com.firebase.ui.auth.IdpResponse;

para la interfaz.

```
auth=FirebaseAuth.getInstance();
if(auth.getCurrentUser()!=null){
    Log.d( tag: "AUTH",auth.getCurrentUser().getEmail());
}else {
    startActivityForResult(AuthUI.getInstance() AuthUI
                    .createSignInIntentBuilder() AuthUI.SignInIntentBuilder
                    .setAvailableProviders(Arrays.asList(
                            new AuthUI.IdpConfig.GoogleBuilder().build(),
                            new AuthUI.IdpConfig.EmailBuilder().build(),
                            new AuthUI.IdpConfig.FacebookBuilder().build())) SignInIntentBuilder
                    .build(),
            RC_SIGN_IN);


}
```

```
mAuthListener = new FirebaseAuth.AuthStateListener() {

    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        updateUi();
    }
};
```

Ilustración 10: Fragmento código LoginActivity.java

Como vemos obtiene la instancia de tipo FirebaseAuth, comprueba si ya hay un usuario logueado. Si es así obtiene la información del que esté autenticado mediante getCurrentUser(), si no es así emplea la clase AuthUI para crear los constructores de los diferentes tipos de inicio de sesión, como podemos ver hay 3. Google, Email y Facebook. Hay más opciones, pero se ha considerado que estos eran los tres mecanismos más simples para el usuario.

En el caso de Database se han usado:

- com.google.firebase.database.DataSnapshot;
- com.google.firebase.database.DatabaseError;
- com.google.firebase.database.DatabaseReference;
- com.google.firebase.database.FirebaseDatabase;
- com.google.firebase.database.ValueEventListener;

```
public void saveComment(View v){

    // newComment=(EditText)v.findViewById(R.id.new_comment);
    // saveComment=(Button)findViewById(R.id.button_save_comment);
     FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

     if(user !=null) {
        String comentario = newComment.getText().toString();
        String id = Integer.toString(ide);
         String url="";
        if(user.getPhotoUrl()!=null){
             url =user.getPhotoUrl().toString();
        }


        Map<String,Object> datoComentario = new HashMap<>();
        datoComentario.put( k: "userId",user.getUid());
        datoComentario.put( k: "comentario",comentario);
        datoComentario.put( k: "userName",user.getDisplayName());
        datoComentario.put( k: "userImg", url);

        mRootReference.child("film").child(id).push().setValue(datoComentario);
         Toast toast1 =
                 Toast.makeText(getActivity().getApplicationContext(),
                         text: "Comentario Guardado", Toast.LENGTH_SHORT);

         toast1.show();
         newComment.getText().clear();

    }else {
        Toast toast1 =
                Toast.makeText(getActivity().getApplicationContext(),
                        text: "Primero loggeate", Toast.LENGTH_LONG);

        toast1.show();
    }
}
```

Ilustración 11: Fragmento código MovieDetailActivity.java

En la ilustración 11 podemos ver el método creado para guardar los comentarios y como actúa con la base de datos. Lo primero que hace es comprobar que el usuario este logueado [10], si no le indica que lo haga.

Después obtiene la información necesaria para poder almacenar un comentario, que es el comentario en sí, el “ide” que es el número de identificación de la película, la foto del usuario, el número de identificación del usuario y su nombre. Todo esto se almacena en un Map<String,Object> y se sube a la base de datos mediante mRootReference.child(X).push.setValue(Y);

mRootReference es la referencia a nuestra base, child se usa para anidar los datos donde deseemos, push() para subirlo y setValue para obtener el valor de lo que queramos almacenar.

Por otra parte, para obtener los datos se ha procedido de la siguiente forma.

```
mRootReference.child("film").child(id).addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        for( final DataSnapshot snapshot:dataSnapshot.getChildren()){
            Log.e( tag: "Datos:", msg: ""+snapshot.getValue());

            mRootReference.child("film").child(id).child(snapshot.getKey()).addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    Comentario user=snapshot.getValue(Comentario.class);
                    String usuario=user.getUserName();
                    String comentario =user.getComentario();
                    String usuarioId=user.getUserId();
                    String usuarioImg=user.getUserImg();
                    Log.e( tag: "Idusuario", msg: ""+usuario);
                    Log.e( tag: "Comentario", msg: ""+comentario);
                    Comentario newUser = new Comentario(usuario, comentario,usuarioId,usuarioImg);
                    adapter.add(newUser);

                }
```

Ilustración 12: Fragmento código MovieDetailActivity.java

De la misma forma en la que se guardan vemos que para obtener la información debemos partir de la referencia a nuestra base de datos, después entrar a través de child que obtiene la referencia a una ruta relativa especifica en nuestra base de datos.[11]

Obtenemos los datos de la ruta que le hayamos especificado y los guardamos en forma del objeto Comentario, este es añadido a nuestro adaptador.

El adaptador lo único que hace es administrar junto a su respectivo layout como aparecerá la información extraída del objeto Comentario.

4.4.3 Google Cloud

Para poder acceder a Google Cloud lo primero que hay que hacer es darse de alta y contratar un plan con ellos. Esta parte la omitiremos porque carece de valor para este tipo de proyecto.

Los dos servicios que utilizamos funcionan de forma distinta. El primero Maps SDK trabaja con las clases de Maps y Places trabaja con consultas http y respuestas JSON.

Maps SDK
Después del registro obtendremos una clave de acceso que habrá que añadir a nuestros manifest.xml [12]

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

Ilustración 13: Fragmento código AndroidManifest.xml

Las clases que hemos usado han sido:

- com.google.android.gms.maps.GoogleMap;

- com.google.android.gms.maps.OnMapReadyCallback;
- com.google.android.gms.maps.SupportMapFragment;
- com.google.android.gms.maps.model.LatLng;
- com.google.android.gms.maps.model.Marker;
- com.google.android.gms.maps.model.MarkerOptions;

Han sido usadas principalmente para obtener el mapa, obtener la localización del dispositivo y añadir marcadores.[13]

Veremos ejemplos más concretos en Places API mediante la ilustración 14.

Places API

```
CineMappSingleton.getInstance(this.getApplicationContext()).addToRequestQueue(new  GsonRequest<CinemaMap>(
        url: "https://maps.googleapis.com/maps/api/place/nearbysearch/json?type=movie_theater&location="+location+"&radius=10000&key="+getString(R.string.google_maps_places_key)
,
        CinemaMap.class,
        headers: null,
        (response) → {
                cines = response;
                List<Result> a = cines.getResults();
                for(int i=0;i<a.size();i++){

                    double la=a.get(i).getGeometry().getLocation().getLat();
                    double ln=a.get(i).getGeometry().getLocation().getLng();
                    String name=a.get(i).getName();
                    mMap.addMarker(new MarkerOptions()
                            .position(new LatLng(la, ln))
                            .title(name));

                }
```

Ilustración 14: Fragmento código MapsActivity.java

Como podemos observar al igual que TMDb hemos enviado una petición a la ruta que nos interesaba, en este caso de cines cercanos a nuestra localización.[14]

Un ejemplo de ruta es

https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters

En nuestro caso los parámetros añadidos han sido el tipo de dato que queremos que nos sea devuelto JSON, el tipo de lugar que deseamos, cines, la localización donde estamos para que muestre los cercanos, el radio desde nuestra localización y la clave que hemos obtenido en el panel de Google Cloud.[15]

Nos devuelve un JSON, Gson lo convierte en un objeto de la clase CineMap con los siguientes atributos.

Tabla 11: Atributos CineMap

List<Object>	htmlAttributions
List<Result>	results
String	status;

De aquí nos interesa results que es una lista de Result. Esta clase tiene los siguientes atributos.

Tabla 12: Atributos Result

Geometry	geometry;
String	icon;
String	id;
String	name;
OpeningHours	openingHours;
List<Photo>	photos
String	placeId;
String	reference;
List<String>	types
String	vicinity;

Como podemos ver en la ilustración 14 usamos los datos de Results para añadir marcadores mediante la clase MarkerOptions usando los valores de geometry y name. Geometry tiene un único atributo de la clase Localization.

Como podemos ver viendo este caso concreto para poder usar Places hemos tenido antes que obtener de Maps las clases Localization y MarkerOptions.

5. Resultados

En este apartado se mostrará el trabajo desarrollado hasta la fecha, poniendo imágenes reales de la aplicación.

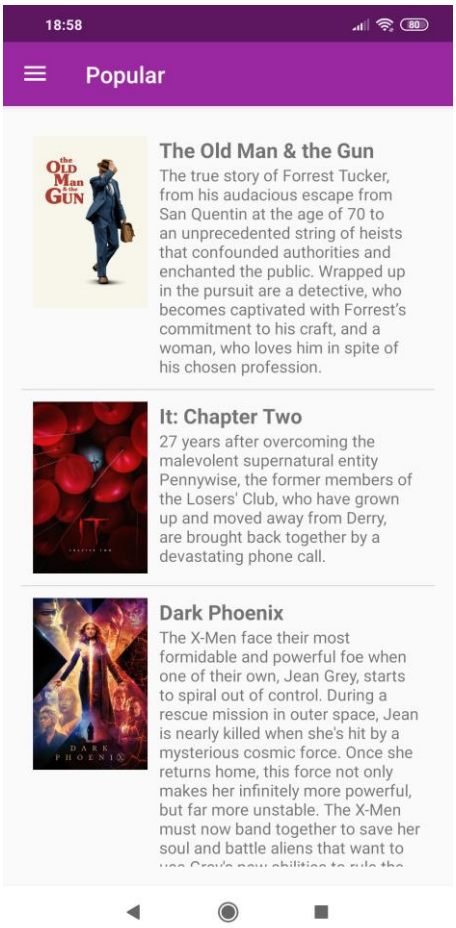


Ilustración 8: Pantalla inicial

Esta es la pantalla de inicio, vemos las últimas novedades.

Se puede observar que la información está en inglés, desde la API de TMDb podemos elegir el idioma en que queramos la información, el motivo de que esté en inglés es que este TFG está en movilidad y se mostró la fase inicial al profesor de la universidad destino del Erasmus

Cuando se selecciona una película de la lista nos aparecerá la siguiente pantalla.

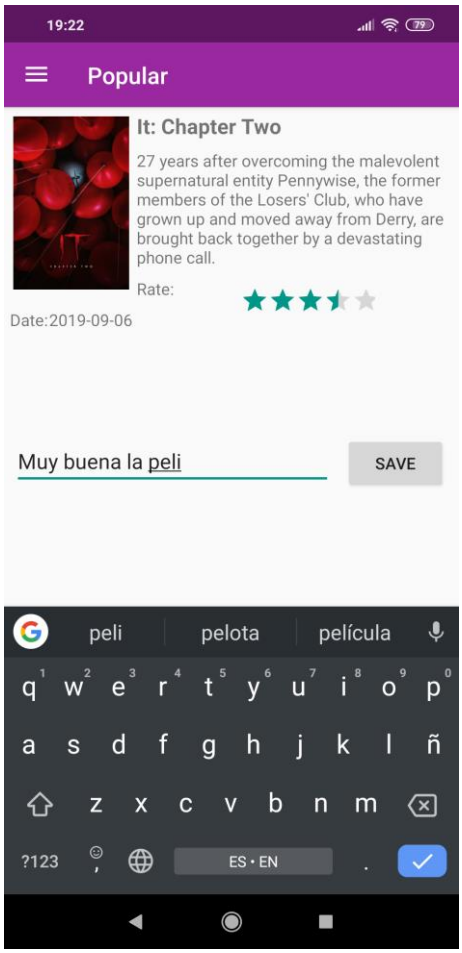


Ilustración9: Pantalla película detalle

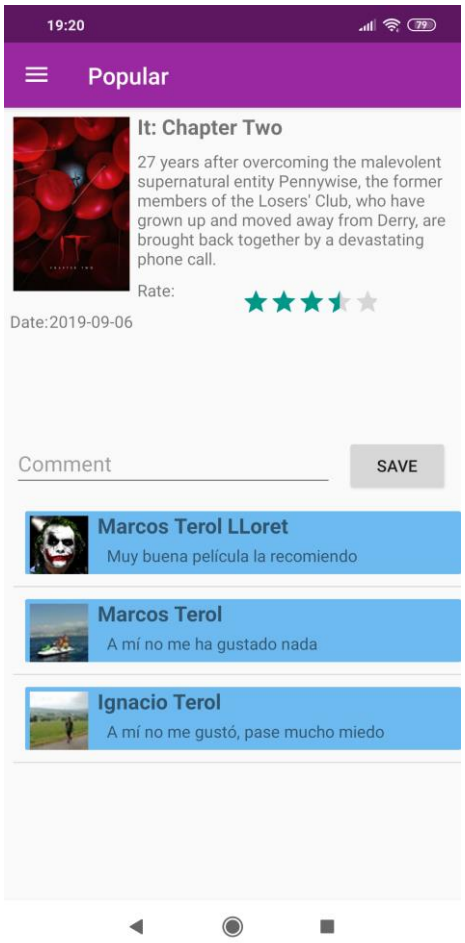


Ilustración 10: Pantalla película detalle con comentarios

En estas dos ilustraciones observamos en la ilustración 9 como aparece cuando no hay comentarios y estamos escribiendo nosotros uno. En la segunda (ilustración 10) cuando ya varios usuarios han comentado sobre la película. Se puede apreciar que en los detalles de la película aparece la imagen de la película, una breve descripción, la fecha de estreno y la valoración que tiene por la comunidad TMDb.

Habiendo visto la pantalla principal y sus detalles pasamos al menú de la aplicación.

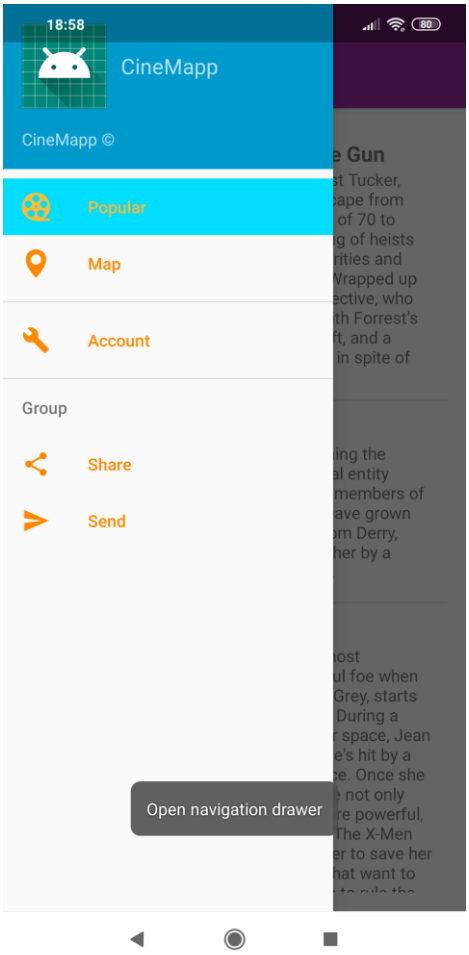


Ilustración 11: Pantalla menú

Podemos observar el menú lateral, se puede acceder de dos formas, pulsando en el icono del menú o desplazando el dedo de la parte izquierda a la derecha. Vemos que tiene la imagen de la app (la app tiene la imagen por defecto), el nombre de la app CineMapp, el nombre de la compañía, en este caso inventada pero manteniendo el nombre.

Bajo donde están los diferentes elementos vemos Popular, es donde vemos el listado de películas. Map donde vemos el mapa que mostraremos más adelante y Account donde mostramos la configuración de la cuenta, también se detallará más adelante.

El último sector es Share y Send, están ideados en el caso de Share para poder compartir la aplicación si nos ha gustado a familiares y amigos y Send para enviar mensajes directamente al desarrollador por si hubiera cualquier problema, pero no están implementados, aunque se pretende hacer en un futuro.

También se puede comprobar que aparece un Toast al abrir el menú, esto se puso para comprobar que funcionaba correctamente. Ahora es prescindible, pero cabe recordar que esta aplicación no está en fase de mercado.

Viendo el menú vamos a continuar por la siguiente sección, el Map.

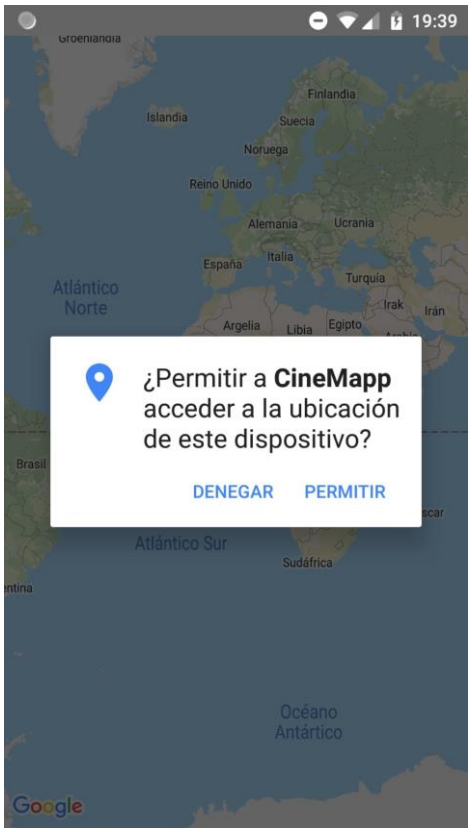


Ilustración 12: Pantalla permisos localización

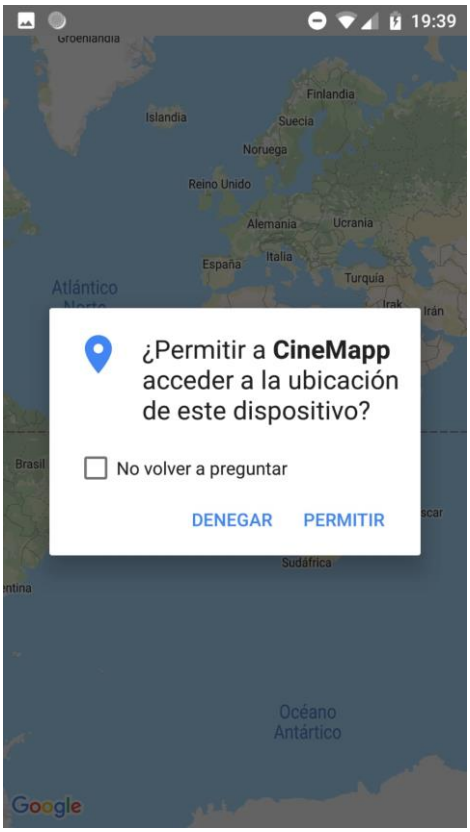


Ilustración 13: Pantalla permisos localización

Como podemos observar si es la primera vez que instalamos la aplicación y usamos la sección de Map nos exigirá concederle permisos para acceder a la ubicación. Implementar este tipo de autorización es exigido por Google desde Android 6.0. [16]

Al ser rechazado insistirá con la pregunta.

Ilustración 14: Pantalla mapa inicial

Cuando ya le hemos dado permisos, nos mostrará nuestra ubicación. Podemos acercarnos manualmente o pulsando en el botón superior derecho.

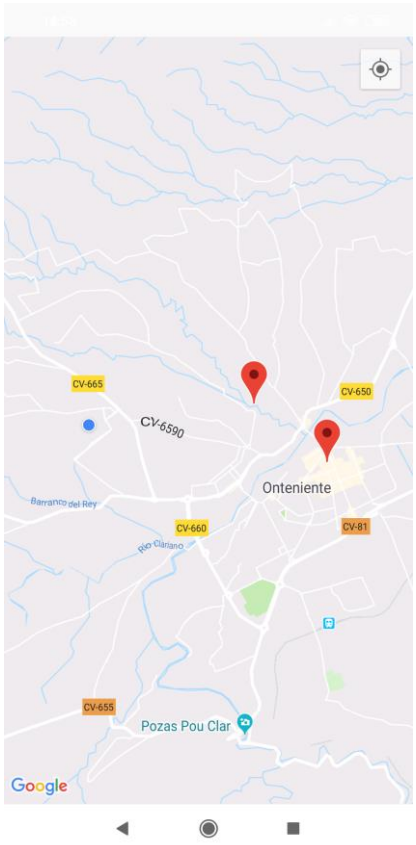


Ilustración 15: Pantalla mapa con marcadores

Podemos observar los 2 marcadores que nos proporcionan nuestra ubicación y en este caso concreto la de dos cines cercanos.

Ilustración 16: Pantalla mapa marcador con información

Cuando seleccionamos un marcador comprobamos que en la esquina inferior derecha aparecen dos botones, el primero es para que nos guie al cine desde nuestra ubicación y el segundo es para abrir esta ubicación directamente desde Google Maps

A parte muestra el nombre del cine justo encima del marcador, como hemos seleccionado ese nos aparece “Cineapolis”.

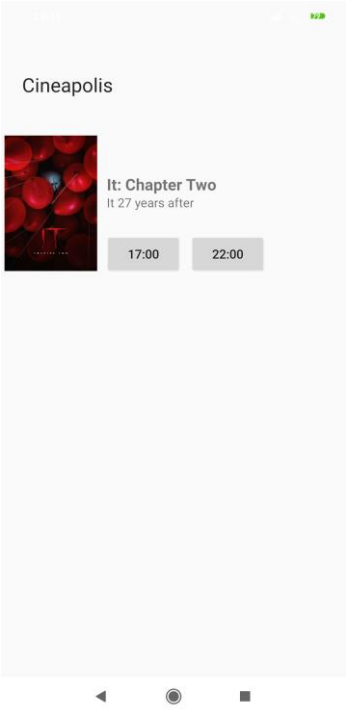


Ilustración 17: Pantalla cartelera

Al pulsar en el marcador nos muestra la cartelera de ese cine y los horarios. Esto es solo un pequeño ejemplo de lo que se pretendía mostrar, aunque como se ha explicado en el proyecto esto ha sido imposible porque las compañías no ofrecen la cartelera. Esta página no tiene funcionalidad, solo es un layout con Texto e imagen por defecto.

Visto este segmento, nos queda el último apartado del menú, la cuenta.

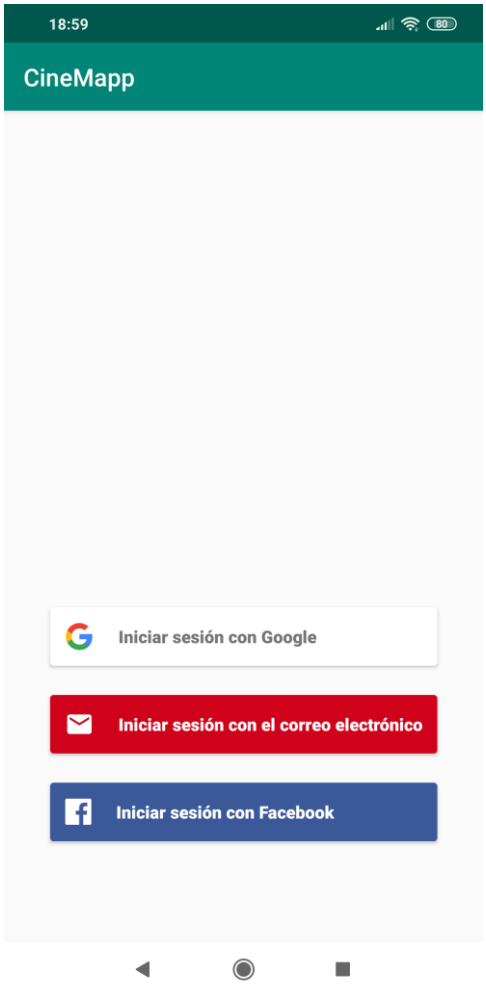


Ilustración 18: Pantalla inicio sesión

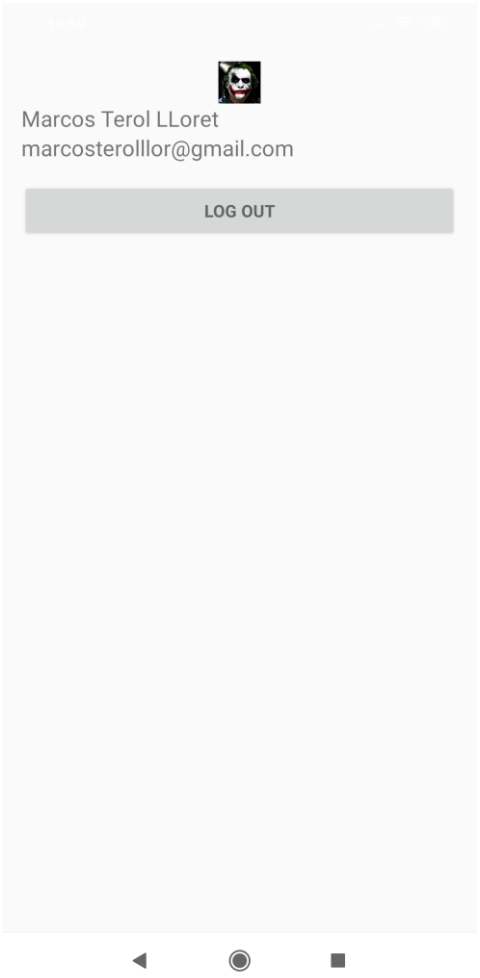


Ilustración 19: Pantalla detalles usuario

Estas dos ilustraciones nos muestran en el primer caso si no estamos logueados y en segundo si ya estamos logueados.

Observamos que en la primera ilustración nos aparecen las 3 opciones que veremos más adelante, inicio con Google, con Facebook y con email.

En la segunda ilustración aparece la imagen de la cuenta, e nombre y su email. Si la cuenta no tiene imagen aparecerá vacío ese segmento.

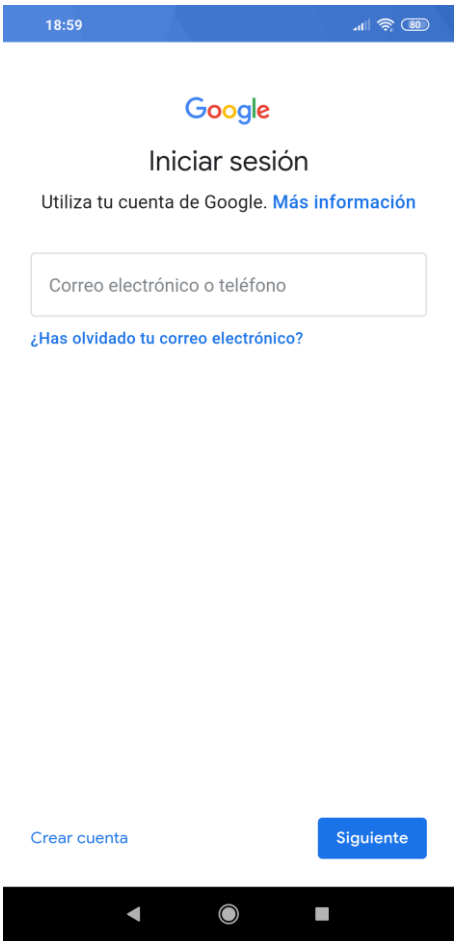


Ilustración 20: Pantalla inicio sesión Google

Ilustración 21: Pantalla inicio sesión Google

Cuando se inicia sesión por Google hay dos opciones, la primera si nuestro dispositivo no tiene ninguna cuenta asociada de Google nos aparecerá el proceso normal de inicio de sesión, el cual no se va a describir porque no compete. En el segundo nos aparece directamente las cuentas usadas en nuestro dispositivo.

Si es la primera vez que iniciamos sesión nos solicitará permisos especiales para obtener los datos asociados a la cuenta. Tan solo email, nombre y foto de perfil.

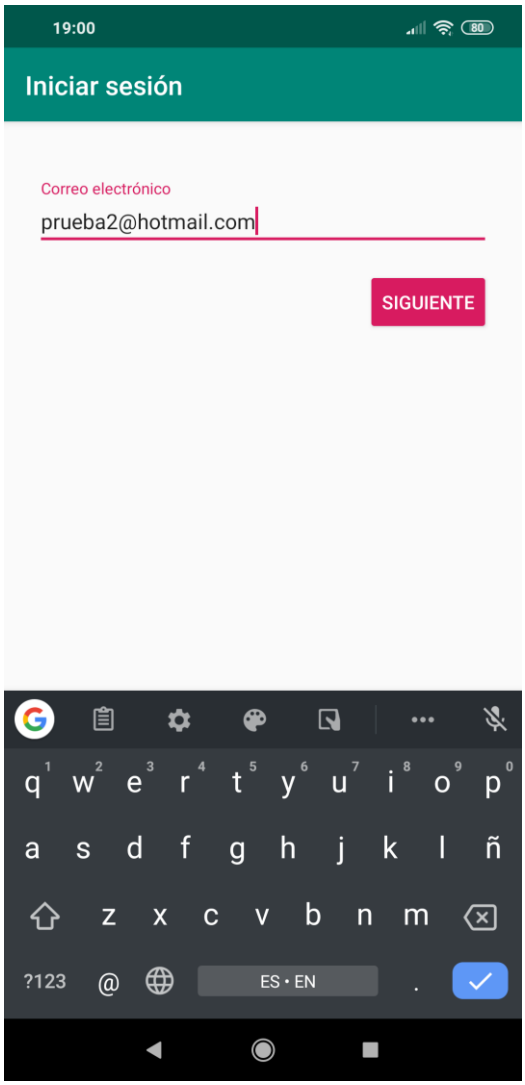


Ilustración 22: Pantalla inicio sesión email

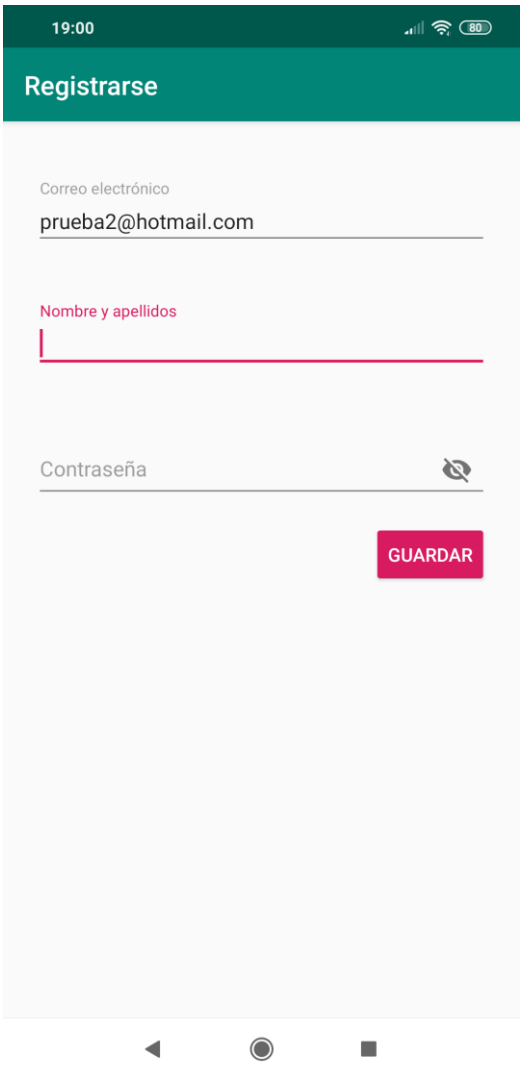


Ilustración 23 Pantalla registro email

En el caso del email nos aparecerá la casilla para ponerlo, si nos hemos registrado anteriormente nos pedirá solo la contraseña, en caso contrario como aparece la segunda imagen nos pedirá más datos para completar el registro.

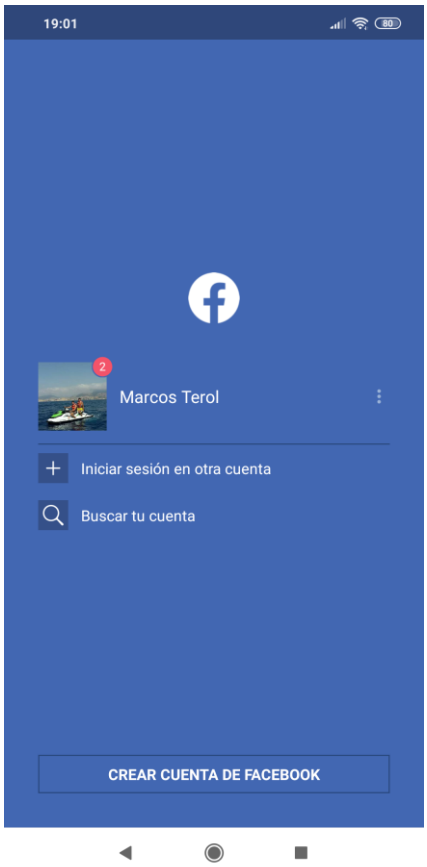


Ilustración 24: Pantalla inicio sesión Facebook

Por último, en el caso de Facebook nos iniciará sesión automáticamente si ya tenemos iniciada la sesión en la propia aplicación de Facebook de nuestro dispositivo, en caso contrario nos pedirá que iniciemos sesión con una cuenta ya guardada o una nueva.

Además, al igual que con el inicio de sesión de Google, si es la primera vez que entramos nos solicitará permisos especiales para que la app tenga acceso a los datos personales de la cuenta.

6. Conclusión

Concluido el proyecto cabe destacar los objetivos cumplidos, los que no se han podido alcanzar, el trabajo futuro y la relación con los estudios cursados.

Los dos objetivos principales, a parte de desarrollar una aplicación en Android nativo, que queríamos llevar a cabo se han cumplido. Por una parte, que el usuario pudiera interactuar, es decir, que pudiera opinar sobre las películas más populares del momento, por otra parte queríamos que el usuario pudiera ver de un vistazo los cines más próximos a su ubicación.

Como se ha ido comentando durante el proyecto se tenía la intención de que el usuario pudiera comprar directamente las entradas desde la aplicación, pero esto ha sido una quimera. Las grandes corporaciones no posibilitan el trabajo de pequeños desarrolladores en temas comerciales y de licencias.

Al ser esto solo una primera versión queda mucho trabajo por delante. Es un proyecto que nos gustaría invertir más tiempo y continuar añadiendo nuevas funcionalidades. Hay varias características que nos gustaría implementar en un futuro, además como hemos visto en el Estudio del Arte algunas de las aplicaciones de la competencia las tienen, no nos podemos quedar atrás. Queremos añadir más información al detalle de las películas. La primera idea es añadir el nombre y foto de los actores y demás personal que trabaje en la obra. También nos gustaría integrar los tráilers dentro de la aplicación.

Como se ha ido viendo a lo largo de proyecto todo lo realizado está relacionado con los estudios cursados. El desarrollo ha sido en Java, lenguaje predominante de enseñanza en nuestro grado integrando patrones y métodos de desarrollo estudiados en las diferentes asignaturas de software. También como en las asignaturas más relacionadas con proyectos y empresa se ha analizado el mercado, se ha construido una idea y a partir de ahí se ha creado. Tampoco debemos olvidar las asignaturas relacionadas con el diseño de interfaces para que sean más atractivas y accesibles al usuario. Por último, hay que destacar que, aunque esto se ha impartido en diferentes asignaturas el proyecto ha sido posible a la unión de los conocimientos adquiridos a lo largo de los diferentes cursos.

7. Referencias

[1] Historia de Perl.

https://www.britannica.com/technology/Perl#ref1072046

[2] Historia de PHP.

https://www.ecured.cu/PHP

[3] Artículo Wireless Application Protocol 2.0 por Chris Bennett.

http://www.informit.com/articles/article.aspx?p=23999

[4] Infographic: The Evolution of The App Stores

https://thetool.io/2017/evolution-app-stores-infographic

[5] Distribución Android

https://developer.android.com/about/dashboards?hl=en

[6] Distribución iOS

https://developer.apple.com/support/app-store/

[7] Documentación API TMDb

https://developers.themoviedb.org/3/

[8] Documentación Firebase Auth

https://firebase.google.com/docs/reference/android/com/google/firebase/auth/FirebaseAuth

[9] Documentación Firebase DataBase

https://firebase.google.com/docs/reference/android/com/google/firebase/database/FirebaseDatabase

[10] Documentación Firebase User

https://firebase.google.com/docs/reference/android/com/google/firebase/auth/FirebaseUser

[11] Documentación a las referencias en la base de datos de Firebase.

https://firebase.google.com/docs/reference/android/com/google/firebase/database/DatabaseReference

[12] Documentación obtener key de Google Cloud.

https://developers.google.com/maps/documentation/android-sdk/get-api-key

[13] Documentación librería Google Maps

https://developers.google.com/maps/documentation/android-sdk/

[14] Documentación librería Google Places

https://developers.google.com/places/android-sdk/

[15] Documentación petición Places Search

https://developers.google.com/places/web-service/search

[16] Permisos Android

https://developer.android.com/guide/topics/permissions/overview?hl=es-419