UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

TheMovieHouse: App móvil para gestión de series y películas

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Avila Solera, Alejandro

Tutor/a: Andrés Martínez, David de

CURSO ACADÉMICO: 2021/2022

Resumen

La evolución de la tecnología en las últimas décadas ha sido inmensa. Los dispositivos móviles se han convertido hoy en día en unos de los dispositivos más utilizados por las personas, y es por ello por lo que tenemos que aprovechar esa ventaja para proporcionarles aplicaciones útiles.

Muchas veces, nos hemos encontrado en el aprieto de querer ver una película o serie y no saber cuál ver. Y hoy en día, al tener la tecnología al alcance de nuestros dedos, acudimos a los móviles a buscar que película/serie poder ver. Por esta razón, este proyecto propone como objetivo la creación de una aplicación móvil cuya función principal sea ayudar al usuario a encontrar series y películas de todo tipo. El usuario tendrá la posibilidad de registrarse en la aplicación para disponer de ciertas funcionalidades, como podría ser guardarse como favorita una película. Toda la información se va a obtener a través de la API que nos proporciona The Movie Database.

Palabras clave: Android, App, Móvil, Películas, Series, Gestor de películas.

Abstract

The evolution of technology in the last decades has been immense. Mobile devices have become one of the most used devices by people today, and that is why we have to take advantage of this to provide them with useful applications.

Many times, we have found ourselves in the predicament of wanting to watch a film or series and not knowing which one to watch. And nowadays, with technology at our fingertips, we turn to our mobiles to find out which film/series to watch. For this reason, this project proposes the creation of a mobile application whose main function is to help the user to find series and films of all kinds. The user will have the possibility to register in the application in order to have certain functionalities, such as saving a film as a favourite. All the information will be obtained through the API provided by The Movie Database.

Keywords: Android, App, Mobile, Films, Series, Film manager.

Tabla de contenidos

Lista de figuras

Lista de tablas

1. Introducción

En este apartado del documento se presenta una concisa explicación sobre el presente trabajo, describiendo la motivación, los objetivos, la metodología utilizada y la estructura del documento.

1.1. Motivación

A lo largo de los años la evolución de la tecnología ha ido mejorando la experiencia del espectador. El cine ha evolucionado de la mano de la innovación de la tecnología. Hace años, si alguien quería ver una película solo tenía dos opciones: ir al cine o verla en la televisión. Hoy en día, existe un abanico extenso de posibilidades para poder disfrutar de una película.

El desarrollo de la tecnología, concretamente de Internet, ha ofrecido unos nuevos aparatos que tienen un tamaño inferior y una potencia comparable a la de los ordenadores, los dispositivos móviles. Estos dispositivos permitieron poner la tecnología al alcance de un número mayor personas, y por ello, en la actualidad más gente recurre a estos dispositivos para conectarse a internet antes que hacer un desembolso importante en un ordenador.

En la actualidad muchas personas están interesadas en el mundo del cine, en el mundo de las películas y de las series. Cada poco tiempo salen a producción cientos de películas, nuevos episodios y nuevas temporadas de las series que más nos gustan.

Muchas veces nos pasa que queremos recordar el nombre de una película o serie que ya hemos visto, y no llegamos a recordarlo. Esto se debe a la gran consumición de otras películas o series, o simplemente por no haberla visto desde hace un tiempo.

También nos pasa que queremos estar al tanto de las novedades, como podría ser estar al tanto de los nuevos lanzamientos de películas en cines o series en televisión. A raíz de estos pequeños problemas y teniendo en cuenta hay un mayor uso de los dispositivos móviles para acceder a internet, surge la idea de este proyecto.

1.2. Objetivos

El objetivo principal del trabajo que se ha desarrollado es diseñar una aplicación Android que facilite al usuario la gestión y administración de películas y series, así como brindarles la posibilidad de buscar y documentarse acerca de dichas películas y series dentro del catálogo.

Lo que se pretende conseguir es reunir en esta aplicación a todo tipo de personas vinculadas con el mundo del cine.

Una vez presentado el objetivo principal del trabajo desarrollado, se presentan los objetivos específicos de la aplicación:

- Buscar películas/series.
- Filtrar películas.
- Crear/eliminar listas de películas/series.
- Añadir película/serie a favoritos.

- Añadir película/serie a la lista de seguimiento
- Ver información detallada acerca de las películas/series.

Por otro lado, con esta aplicación se quiere apoyar dos de los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas, concretamente el número 10 y 13, conocidos por el nombre de “Reducción de las desigualdades” y “Acción por el clima”. En el Anexo 1: ODS se explica con más detalle la relación con los ODS.

1.3. Metodología

Como punto de partida, se tiene que explicar que son las metodologías ágiles. Son aquellas que dividen el trabajo en partes, que permiten adaptarse sobre la marcha y resuelven necesidades en poco tiempo.

Con respecto a las ventajas de las metodologías ágiles, tenemos una mejora de la flexibilidad del proyecto y una reducción de costes (se van identificando los errores a lo largo del desarrollo).

Se ha decido utilizar una metodología ágil porque permite dar una gran flexibilidad al proyecto y gestionarlo eficazmente.

La metodología ágil que se ha seguido es la metodología Kanban [1]. Esta metodología se trata de un método visual de gestión de proyectos que permite a los implicados en el mismo visualizar la carga y los flujos de trabajo. Concretamente se trata de un tablero donde se muestra todo el trabajo organizado por columnas. Hay que recalcar que Kanban es una técnica de organización del trabajo, no de planificación.

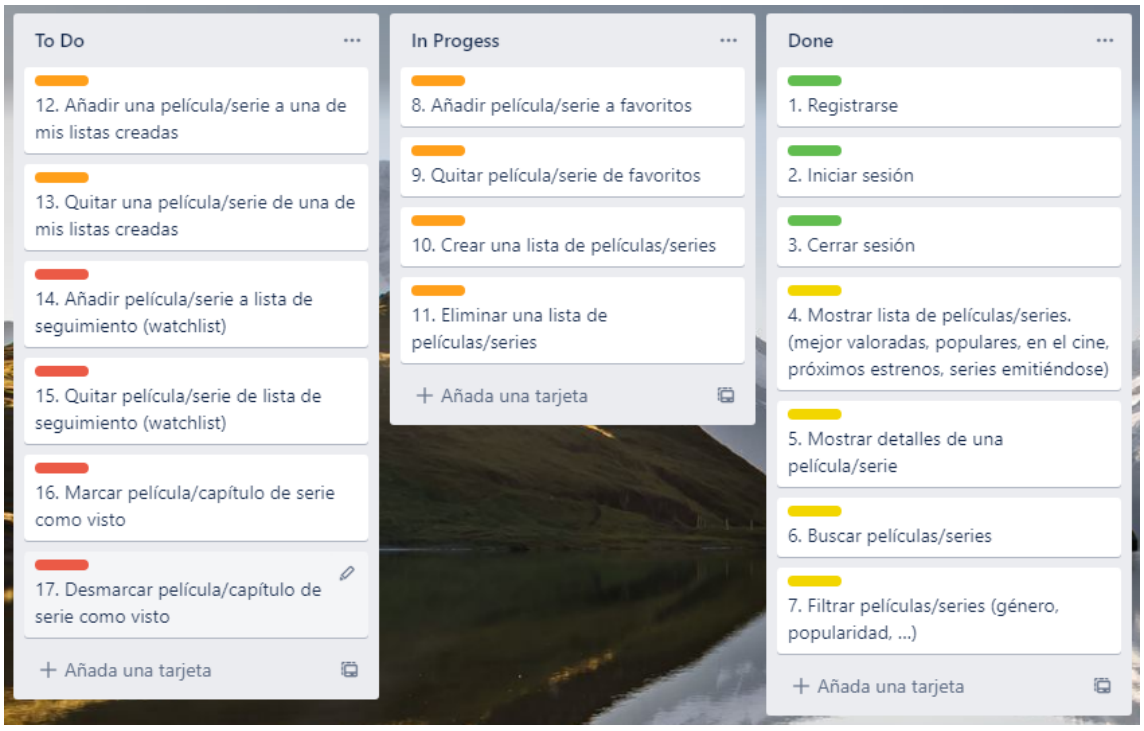

Figura 1: Metodología Kanban usando Trello

Se ha seguido esta metodología usando la herramienta Trello[1]. En la Figura 1 se puede apreciar esa organización del trabajo. Se ha decidido utilizar tres columnas para organizar el trabajo. La primera corresponde al trabajo pendiente o por hacer, la segunda al trabajo en progreso, y la última al trabajo terminado.

1.4. Estructura

A continuación, se va a presentar un pequeño índice genérico de la información que se va a poder encontrar en los siguientes puntos del documento:

1. **Estado del arte**: se analizan y documentan otras aplicaciones del mercado del mismo ámbito que la aplicación planteada.

2. **Análisis del problema**: se analizan de forma más profunda las aplicaciones comentadas en el punto anterior, entrando en detalle acerca de las funcionalidades.

3. **Especificación de requisitos**: se especifican los requisitos funcionales y no funcionales del proyecto.

4. **Diseño de la solución**: se describe la arquitectura que tiene el proyecto, el diagrama de clases y los bocetos de todas las pantallas de la aplicación.

5. **Tecnologías utilizadas**: se mencionan y describen las tecnologías y herramientas utilizadas en el proyecto.

6. **Implementación de la solución**: se describen lo primeros pasos que se han realizado, las bibliotecas utilizadas, la estructura del proyecto, y se comentan algunas partes del proyecto que se consideran relevantes.

7. **Pruebas**: se muestran todas las pruebas realizadas con sus resultados.

8. **Resultado final**: se expone, mediante capturas de pantalla, todas las interfaces de la aplicación desarrollada.

9. **Conclusiones**: se describen las conclusiones, relación con los estudios cursados y futuros trabajos.

10. **Referencias**: se incluyen los datos de las fuentes de información que se han usado como referencia para el desarrollo de este proyecto.

[1] https://trello.com/

2. Estado del Arte

En este apartado del documento se estudian y analizan algunas aplicaciones existentes en el mercado que tienen una funcionalidad similar, o muy parecida a la de la aplicación del proyecto. Las tres aplicaciones para analizar son TV Time, IMDb y Sofa Time.

2.1. TV Time

La aplicación TV Time[2] se encuentra tanto en iOS como Android al momento de redactar este documento. Fue lanzada el año 2011 de forma gratuita. Es compatible con iOS 13.0 y versiones posteriores, y con Android 5.0 y versiones posteriores.

TV Time es una aplicación ideal para aquellas personas que sean amantes de las series y películas. La idea principal es permitirnos un seguimiento de nuestras series y películas favoritas, así como crear una agenda o lista de pendientes. Una aplicación básica que incorpora las funciones útiles para aquellas personas que tienen más de una serie o película pendiente.

2.1.1. Funcionalidades principales

La funcionalidad principal de TV Time es la búsqueda de películas/series dentro de su catálogo, como muestra la Figura 2. Entre otras funcionalidades se destacan las siguientes:

- Añadir película/serie a favoritos.
- Mostrar el seguimiento de tus series/películas.
- Ordenar por tendencia o lo más añadido.
- Filtrar películas (género, año, …).
- Seguir a usuarios.
- Crear una lista de películas/series.
- Mostrar detalles de una película serie.

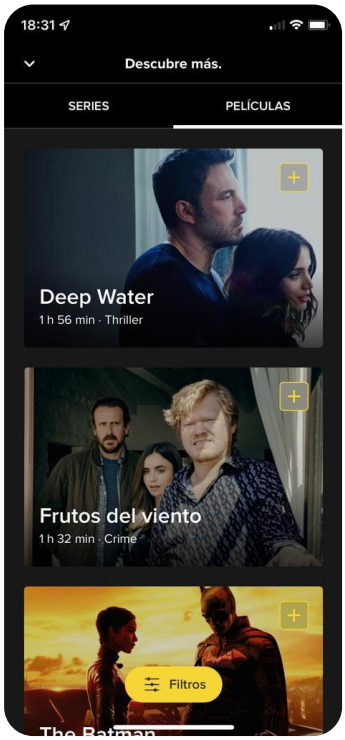

Figura 2: Captura de pantalla del buscador de TV Time

[2] https://www.tvtime.com/

2.2. IMDb

La aplicación móvil IMDb[3] la tenemos disponible tanto en la App Store como en la Play Store al momento de redactar este documento. Fue lanzada el año 1990, y a partir del año 1998 fue adquirida por Amazon. Es compatible con iOS 14.1 y versiones posteriores, y con Android 8.0 y versiones posteriores.

IMDb es la base de datos de películas y TV más grande y popular de todo Internet. Permite mostrar miles de películas, series, programas de televisión, actores y mucha más información centrada en el entretenimiento. La aplicación es totalmente gratuita, pero también dispone de una suscripción de pago llamada IMDbPro, destinada a los profesionales de la industria del entretenimiento.

2.2.1. Funcionalidades principales

Una de las funcionalidades principales de IMDb es la búsqueda de películas/series dentro de su catálogo, como muestra en la Figura 3. Se destacan las siguientes funcionalidades de la aplicación:

- Mostrar lista de películas y series populares.
- Mostrar lista de películas que se encuentran actualmente en el cine.
- Visualizar vídeos de entrevistas, de avances populares, de noticias.
- Mostrar la calificación de cada una de las películas del catálogo.
- Buscar una película dentro del catálogo.
- Visualizar vídeos de entrevistas, de avances populares, de noticias.
- Crear una lista de películas/series.
- Mostrar detalles de una película/serie.

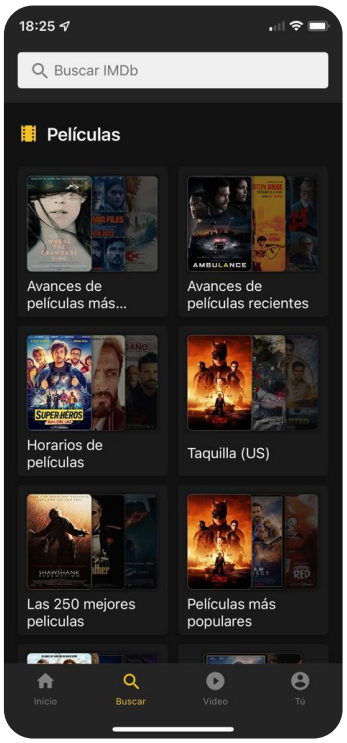

Figura 3: Captura de pantalla del buscador de IMDb

[3] https://www.imdb.com/

2.3. Sofa Time

La aplicación SofaTime[4] se encuentra tanto en iOS como Android al momento de redactar este documento. Fue lanzada el año 2020 de forma gratuita. Es compatible con iOS 13.0 y versiones posteriores, y con Android 5.0 y versiones posteriores.

Se trata de una aplicación que te ayuda a gestionar series y películas. La aplicación es totalmente gratuita, y además incluye una versión de pago en la que habilita muchas más funcionalidades.

2.3.1. Funcionalidades principales

Una de las funcionalidades principales de Sofa Time es el calendario con los próximos estrenos, como muestra la . Entre las funcionalidades que dispone se destacan las siguientes:

- Calendario con los próximos estrenos en el cine.
- Crear una lista de películas/series.
- Buscar una película dentro del catálogo.
- Mostrar la calificación de cada una de las películas del catálogo.
- Mostrar detalles de una película/serie.
- Marcar una película/serie como vista.
- Filtrar películas (género, año, …).

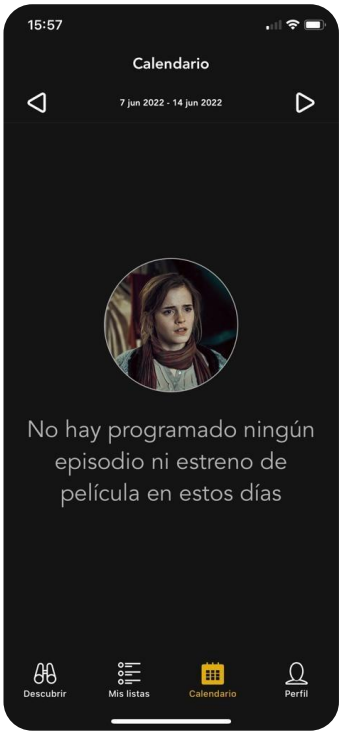

Figura 4: Captura de pantalla del calendario de Sofa Time

[4] https://www.sofatime.app/

3. Análisis del problema

Como punto de partida, se ha realizado una tabla en la que se recogen una gran variedad de funcionalidades comunes y no comunes de todas las aplicaciones analizadas. Algunas de las funcionalidades solo las podemos encontrar en alguna de las aplicaciones, y no en las tres.

Funcionalidad / Aplicación	**TV Time**	**IMDb**	**Sofa Time**
Registrarse	✓	✓	✓
Iniciar sesión	✓	✓	✓
Cerrar sesión	✓	✓	✓
Mostrar lista de películas populares.	✓	✓	✓
Mostrar lista de películas mejor valoradas.		✓	✓
Mostrar próximos estrenos en el cine.		✓	
Mostrar lista de películas que se encuentran actualmente en el cine.		✓	
Buscar una película dentro del catálogo.	✓	✓	✓
Mostrar películas/series similares a una película/serie.	✓	✓	
Añadir película/serie a favoritos.	✓		
Quitar película/serie de favoritos.	✓		
Mostrar detalles de una película.	✓	✓	✓
Proporcionar información de los actores	✓	✓	✓
Filtrar películas/series (género, año, …)	✓	✓	✓

Ordenar por (popularidad ascendente o descendente, valoración, …)	✓	✓	✓
Añadir película/serie a lista de seguimiento (watchlist)		✓	
Quitar película/serie de lista de seguimiento (watchlist)		✓	
Crear una lista de películas/series	✓	✓	✓
Eliminar una lista de películas/series	✓	✓	✓
Añadir una película/serie a una de mis listas creadas	✓	✓	✓
Quitar una película/serie de una de mis listas creadas	✓	✓	✓
Marcar película/capítulo de serie como visto	✓		✓
Desmarcar película/capítulo de serie como visto	✓		✓
Buscar películas o series por actor		✓	✓
Visualizar vídeos de entrevistas, de avances populares, de noticias.		✓	

La aplicación TV Time dispone de bastantes funcionalidades interesantes, entre ellas la búsqueda y el filtrado de películas/series. No obstante, la aplicación llega a ser un poco confusa al tener las interfaces muy cargadas de información.

La aplicación IMDb es de las más completas, dispone de muchas funcionalidades útiles con un gran catálogo de películas y series, pero no tiene una cosa que llama bastante la atención. No proporciona una API pública para obtener datos y trabajar con ellos. Eso no nos facilita la posibilidad de extender la propia funcionalidad de IMDb, o la obtención de información que disponen ellos.

Por último, la aplicación Sofa Time tiene muchas funcionalidades, pero la gran mayoría sólo están disponibles en la versión premium, que es de pago. Entre ellas, funcionalidades cómo mostrar un calendario de los próximos estrenos en el cine, listas ilimitadas, más opciones de ordenación de las que tiene, soporte VIP, sin publicidad y otras muchas más.

3.1. Tabla MoSCow

Las tres aplicaciones analizadas presentan unas funcionalidades muy interesantes. A continuación, se muestra una tabla MoSCow donde se determina la priorización de funcionalidades de la aplicación propuesta dentro del proyecto.

Debe tener (MUST)	- Registrarse. - Iniciar sesión. - Cerrar sesión. - Mostrar lista de películas/series (mejor valoradas, populares, en el cine, …). - Mostrar detalles de una película/serie. - Buscar películas/series. - Añadir película/serie a favoritos. - Quitar película/serie de favoritos.
Debería tener (SHOULD)	- Crear una lista de películas/series. - Eliminar una lista de películas/series - Añadir una película/serie a una de mis listas creadas. - Quitar una película/serie de una de mis listas creadas. - Añadir película/serie a la lista de seguimiento (watchlist). - Quitar película/serie de la lista de seguimiento (watchlist).
Podría tener (COULD)	- Notificaciones de películas/series para ver. - Ajustes - Marcar película/capítulo de serie como visto. - Desmarcar película/capítulo de serie como visto.
No tendrá (WON'T)	- Listado de películas recomendadas para una película - Proporcionar información de los actores. - Visualizar vídeos de entrevistas, de avances populares y de noticias.

Como se puede observar en la tabla MoSCow, hemos definido todas las funcionalidades de mayor a menor prioridad. Se ha definido de esta manera para determinar la importancia de las funcionalidades dentro del proyecto.

Una de las funcionalidades clave de la aplicación es mostrar el listado de películas/series atendiendo a varios criterios, como podrían ser las que están en el cine o las más populares. Cabe destacar que la aplicación no dispone de un listado de películas recomendadas para cierta película, no proporciona información sobre el reparto de las películas series, ni la visualización de vídeos de entrevistas, de avances populares y de noticias.

Algunas mejoras que se podrían incluir a la aplicación es la adición de algunas de las funcionalidades que se encuentran en la tabla MoSCow y que no están incorporadas en la aplicación.

Como conclusión, lo que se pretende con la aplicación es de disponer de una gran variedad de funcionalidades que le sean útiles al usuario, ya sea añadir películas/series a sus favoritas como crear sus listas personalizadas.

4. Especificación de requisitos

La especificación de requisitos de software (**ERS**) es una definición/descripción completa de todo el comportamiento del sistema que se va a desarrollar, incluyendo los requisitos funcionales, especificados a través de casos de uso, y los requisitos no funcionales.

Los requisitos funcionales describen todas las interacciones que tendrán los usuarios con el producto desarrollado, mientras que los requisitos no funcionales son condiciones que imponen restricciones en el diseño y la implementación.

4.1. Requisitos funcionales

A continuación, se muestra el diagrama de casos de uso y un conjunto de tablas donde se detallan cada uno de los casos de uso.

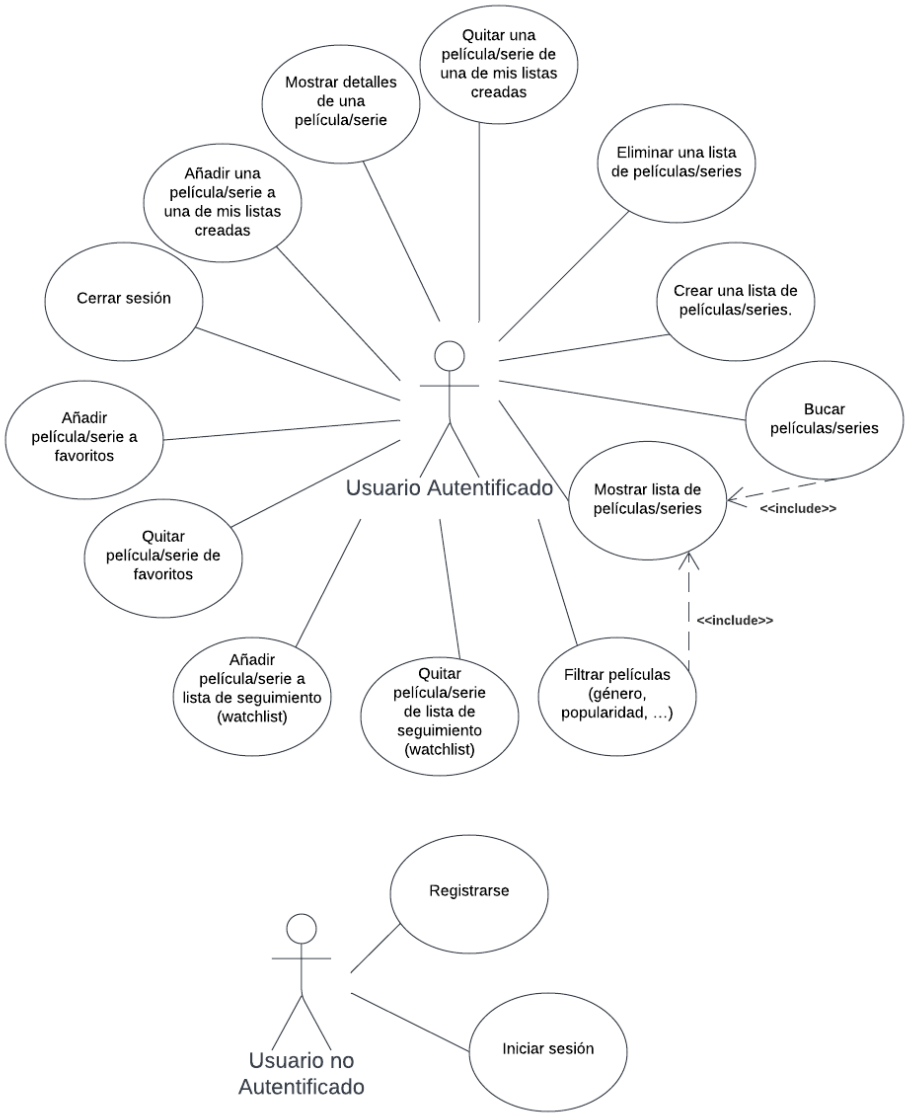

Figura 5: Diagrama de Casos de Uso

Tabla 1: UC-01 Buscar películas/series

Id y Nombre	UC-01 Buscar películas/series.
Descripción	El usuario podrá buscar películas/series en las diferentes listas de estas.
Precondición	Ninguna.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona sobre el menú "Buscar". 5. El usuario introduce el texto que desee. 6. El sistema muestra una lista con los resultados de la búsqueda.
Alternativas /Errores	Ninguno.
Postcondición	Las películas/series deseadas han sido encontradas. No hay películas/series dada esa búsqueda.
Notas	Ninguna.

Tabla 2: UC-02 Mostrar lista de películas/series

Id y Nombre	UC-02 Mostrar lista de películas/series.
Descripción	El usuario podrá mostrar la lista de películas/series.
Precondición	Ninguna.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona en un menú de listas de películas/series (mejor valoradas, más populares, favoritos, de seguimiento, ...). 5. El sistema muestra una lista de películas/series.
Alternativas /Errores	Ninguno.
Postcondición	Las películas/series se muestran en la lista.
Notas	Al iniciar la aplicación e iniciar sesión se le redirige al usuario al listado completo de películas.

Tabla 3: UC-03 Añadir película/serie a favoritos

Id y Nombre	UC-03 Añadir película/serie a favoritos.
Descripción	El usuario podrá añadir una película/serie a la lista de favoritos.
Precondición	La película/serie no debe haber sido añadida a la lista de favoritos anteriormente.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona en un menú de listas de películas/series. 5. El usuario presiona sobre el elemento en forma de corazón de una película/serie que no haya sido marcado anteriormente. 6. El sistema guarda con éxito la película/serie en la lista de favoritos.
Alternativas /Errores	3.1. Si el usuario se encuentra en los detalles de la película, presiona al elemento en forma de corazón que se encuentra sin marcar.
Postcondición	La película/serie se ha añadido a favoritos.
Notas	Ninguna.

Tabla 4: UC-04 Quitar película/serie de favoritos

Id y Nombre	UC-04 Quitar película/serie de favoritos.
Descripción	El usuario podrá quitar una película/serie a la lista de favoritos.
Precondición	La película/serie debe haber sido añadida a la lista de favoritos anteriormente.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona en un menú de listas de películas/series. 5. El usuario presiona sobre el elemento en forma de corazón de una película/serie que haya sido marcado anteriormente. 6. El sistema elimina con éxito la película/serie de la lista de favoritos.
Alternativas /Errores	3.1. Si el usuario se encuentra en los detalles de la película, presiona al elemento en forma de corazón que se encuentra marcado.
Postcondición	La película/serie se ha quitado de favoritos.
Notas	Ninguna.

Tabla 5: UC-05 Añadir película/serie a lista de seguimiento (watchlist)

Id y Nombre	UC-05 Añadir película/serie a lista de seguimiento (watchlist).
Descripción	El usuario podrá añadir una película/serie a la lista de seguimiento.
Precondición	La película/serie no debe haber sido añadida a la lista de seguimiento anteriormente.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona en un menú de listas de películas/series. 5. El usuario presiona sobre el elemento en forma de marcador de una película/serie que no haya sido marcado anteriormente. 6. El sistema guarda con éxito la película/serie en la lista de seguimiento.
Alternativas /Errores	2.1. Si el usuario se encuentra en los detalles de la película, presiona el elemento en forma de marcador que se encuentra sin marcar. 2.2. Si el usuario se encuentra dentro de la "Lista de favoritos", presiona el elemento en forma de marcador que se encuentra sin marcar.
Postcondición	La película/serie se ha añadido a la lista de seguimiento.
Notas	Ninguna.

Tabla 6: UC-06 Quitar película/serie de lista de seguimiento (watchlist)

Id y Nombre	UC-06 Quitar película/serie de lista de seguimiento (watchlist).
Descripción	El usuario podrá quitar una película/serie de la lista de seguimiento.
Precondición	La película/serie debe haber sido añadida a la lista de seguimiento anteriormente.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona en un menú de listas de películas/series. 5. El usuario presiona sobre el elemento en forma de marcador de una película/serie que haya sido marcado anteriormente. 6. El sistema elimina con éxito la película/serie de la lista de seguimiento.
Alternativas /Errores	2.3. Si el usuario se encuentra en los detalles de la película, presiona el elemento en forma de marcador que se encuentra marcado. 2.4. Si el usuario se encuentra dentro de la "Lista de seguimiento", presiona el elemento en forma de marcador que se encuentra marcado.
Postcondición	La película/serie se ha eliminado de la lista de seguimiento.
Notas	Ninguna.

Tabla 7: UC-07 Filtrar películas/series (género, popularidad, ...)

Id y Nombre	UC-07 Filtrar películas/series (género, popularidad, …).
Descripción	El usuario podrá filtrar el listado de películas/series a su gusto.
Precondición	Debe estar en la ventana de búsqueda y presionar el icono de filtrado.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona sobre el menú "Buscar". 5. El usuario presiona el elemento de filtrado de series o películas. 6. El usuario selecciona las opciones a filtrar. 7. El sistema muestra una lista de películas/series con el filtro aplicado.
Alternativas /Errores	Ninguno.
Postcondición	Se muestra la lista filtrada.
Notas	Ninguna.

Tabla 8: UC-08 Crear una lista de películas/series

Id y Nombre	UC-08 Crear una lista de películas/series.
Descripción	El usuario podrá crear una lista para guardar las películas/series que desee.
Precondición	Ninguna.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona sobre el menú “Mis listas”. 5. El usuario presiona el botón de crear listas. 6. El usuario introduce el nombre y la descripción de la lista. 7. El usuario presiona el botón de crear. 8. El sistema muestra el listado de todas las listas, incluyendo la nueva creada.
Alternativas /Errores	Ninguno.
Postcondición	La lista se ha creado con éxito.
Notas	Ninguna

Tabla 9: UC-09 Eliminar una lista de películas/series

Id y Nombre	UC-09 Eliminar una lista de películas/series.
Descripción	El usuario podrá eliminar una lista de las listas existentes.
Precondición	Ninguna.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona sobre el menú “Mis listas”. 5. El usuario presiona sobre el icono en forma de papelera de una lista. 6. El sistema elimina con éxito la lista del listado de mis listas.
Alternativas /Errores	Ninguno.
Postcondición	La lista se ha eliminado con éxito. Todas las películas/series que se encontraban dentro de la lista son eliminadas de la misma.
Notas	Ninguna.

Tabla 10: UC-10 Mostrar detalles de una película/serie

Id y Nombre	UC-10 Mostrar detalles de una película/serie.
Descripción	El usuario podrá visualizar los detalles de una película/serie.
Precondición	Ninguna.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona en un menú de listas de películas/series (mejor valoradas, más populares, …). 5. El usuario presiona sobre una película/serie. 6. El sistema muestra los detalles de esa película/serie.
Alternativas /Errores	Ninguno.
Postcondición	Se muestran los detalles de la película/serie.
Notas	También se puede acceder a los detalles de una película/serie cuando buscamos las mismas mediante el buscador.

Tabla 11: UC-11 Iniciar sesión

Id y Nombre	UC-11 Iniciar sesión.
Descripción	El usuario podrá iniciar sesión en la aplicación para poder acceder a las demás funcionalidades.
Precondición	El usuario debe tener una cuenta para iniciar sesión. Si no es el caso tendrá que registrarse.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario introduce su nombre de usuario y contraseña. 3. El usuario presiona sobre el botón "LOGIN". 4. El sistema verifica los datos y muestra el listado completo de películas.
Alternativas /Errores	4.1. Si los datos introducidos no son válidos aparece un mensaje avisando de ello.
Postcondición	El usuario ha iniciado sesión con éxito.
Notas	Ninguna.

Tabla 12: UC-12 Cerrar sesión

Id y Nombre	UC-12 Cerrar sesión.
Descripción	El usuario podría cerrar sesión en la aplicación.
Precondición	El usuario debe haber iniciado sesión con anterioridad.
Secuencia principal	1. El usuario abre el desplegable lateral de la izquierda. 2. El usuario presiona sobre el menú "Cerrar sesión". 3. El sistema cierra sesión y muestra la vista de iniciar sesión.
Alternativas /Errores	Ninguno.
Postcondición	El usuario ha cerrado sesión con éxito.
Notas	Ninguna.

Tabla 13: UC-13 Añadir una película/serie a una de mis listas creadas

Id y Nombre	UC-13 Añadir una película/serie a una de mis listas creadas.
Descripción	El usuario podrá añadir cualquier película/serie a una de sus listas creadas.
Precondición	La película/serie no debe haber sido añadida a la lista con anterioridad.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona en un menú de listas de películas/series (mejor valoradas, más populares, …). 5. El usuario presiona sobre una película/serie. 6. El usuario presiona sobre el elemento en forma del signo más. 7. El usuario selecciona la lista donde desea guardar la película/serie. 8. El usuario presiona sobre el botón “AGREGAR”. 9. El sistema agrega la película/serie a la lista seleccionada.
Alternativas /Errores	Ninguno.
Postcondición	Se ha añadido una película/serie a una de las listas creadas.
Notas	Ninguna.

Tabla 14: UC-14 Quitar una película/serie de una de mis listas creadas

Id y Nombre	UC-14 Quitar una película/serie de una de mis listas creadas.
Descripción	El usuario podrá quitar cualquier película/serie de una de sus listas creadas.
Precondición	La película/serie debe haber sido añadida a la lista con anterioridad.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario inicia sesión. 3. El usuario abre el desplegable lateral de la izquierda. 4. El usuario presiona sobre el menú "Mis listas". 5. El usuario presiona una lista. 6. El usuario presiona sobre el icono en forma de papelera de una película/serie. 7. El sistema quita con éxito la película/serie de la lista.
Alternativas /Errores	Ninguno.
Postcondición	Se ha quitado una película/serie de una de las listas creadas.
Notas	Ninguna.

Tabla 15: UC-15 Registrarse

Id y Nombre	UC-15 Registrarse.
Descripción	El usuario podrá registrarse para poder acceder a la aplicación.
Precondición	El usuario debe de tener una cuenta de correo para poder registrarse.
Secuencia principal	1. El usuario inicia la aplicación. 2. El usuario presiona sobre el botón "Regístrate". 3. El sistema redirige al usuario a la página de registro de The Movie Database .
Alternativas /Errores	Ninguno.
Postcondición	Se ha registrado un usuario nuevo.
Notas	El registro se ha realizado de esta manera ya que el propio servidor proporciona el inicio de sesión a través de una petición POST, pero no el registro.

4.2. Requisitos no funcionales

Disponibilidad	- El dispositivo móvil debe disponer de una conexión a internet para el acceso a la aplicación.
Tiempo de respuesta	- Tiempo de carga de los datos del servidor más bajo posible.
Arquitectura	- El sistema debe ser fácilmente escalable, con el objetivo de hacer crecer la aplicación al incorporar nuevas funcionalidades en un futuro.
Interfaz del sistema	- La aplicación debe tener una interfaz intuitiva y agradable para el usuario.
Autenticidad	- La aplicación debe contar con un mecanismo de autenticación de usuarios. Solo los usuarios autorizados deben tener acceso a la aplicación.
Confidencialidad	- La aplicación debe garantizar que la información estará protegida y va a estar solamente disponible para aquellos usuarios autorizados.
Seguridad	- La aplicación no debe utilizar los datos de los usuarios con otro fin que no sea para verificar su acceso a la misma.

5. Diseño de la solución

En esta sección se describe las decisiones que se han tomado para llevar a cabo la realización de la solución. Se va a presentar el patrón de diseño MVVM, el diagrama de clases que representa la distribución de los datos, y finalmente los bocetos de la aplicación iniciales de los cuales se partieron.

5.1. Arquitectura MVVM

Se ha decidido seguir el patrón de diseño MVVM ya que nos ayuda a estructurar la aplicación en capas que permiten aislar la interfaz de la lógica de negocio y de la persistencia. Mediante la aplicación de dicho patrón conseguimos una arquitectura organizada, donde obtenemos un código más limpio y una mayor claridad y comprensión del proyecto frente a otros desarrolladores. Además, nos facilita las pruebas, escalabilidad y mantenimiento de los proyectos.

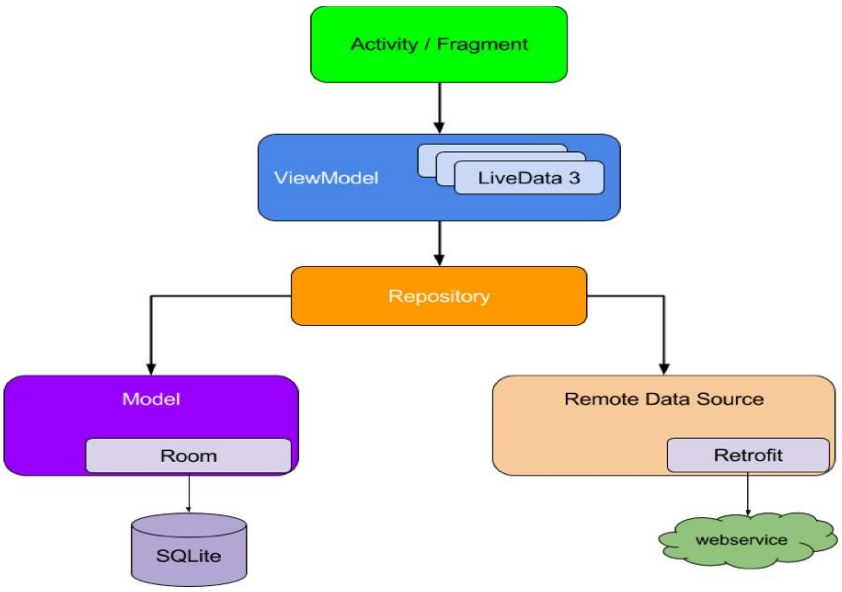

Figura 6: Arquitectura MVVM [2]

En la Figura 6 se muestra gráficamente como se implementa el patrón MVVM de una aplicación Android. Así mismo, Android nos proporciona una colección de bibliotecas que nos ofrecen cierta facilidad a la hora de desarrollar nuestras aplicaciones.

La arquitectura MVVM se compone de tres capas:

- **View** (capa de presentación o de interfaz de usuario): se encuentran las clases que se centran en mostrar los datos de la aplicación en pantalla.

- **ViewModel** (capa Vista-Modelo): se encuentran las clases que se encargan del almacenamiento y administración de los datos relacionados con la vista.

- **Model** (capa de datos): se encuentran las clases que interaccionan con todas las fuentes de datos, ya sean provenientes de bases de datos, APIs, etc.

5.1.1. Capa de interfaz de usuario (View)

En esta capa se define cómo la información y las diferentes funcionalidades se mostrarán gráficamente. Las diversas clases que se encuentran aquí podrían ser las actividades (Activity) o los fragmentos (Fragment).

Una Activity [3] en Android se relaciona a una de las pantallas de nuestra aplicación. Se encargan de crear una ventana en la que posteriormente se coloca la interfaz de usuario. Normalmente se presentan al usuario como ventanas a pantalla completa, pero también se pueden representar de otras formas: como ventanas flotantes, multiventana, …

Un Fragment es un componente que funciona dentro del ámbito de una Activity. Se encargan de ampliar parte de la lógica utilizada para la navegación entre actividades. Siempre va a estar alojado en una Activity, y su ciclo de vida se ve afectado por el ciclo de vida de la actividad anfitriona. Se puede agregar o quitar de una actividad siempre y cuando la actividad se esté ejecutando. Además, se pueden definir varios fragmentos dentro de una misma Activity.

Esta capa tiene la referencia hacía la capa Vista-Modelo.

5.1.2. Capa Vista-Modelo (ViewModel)

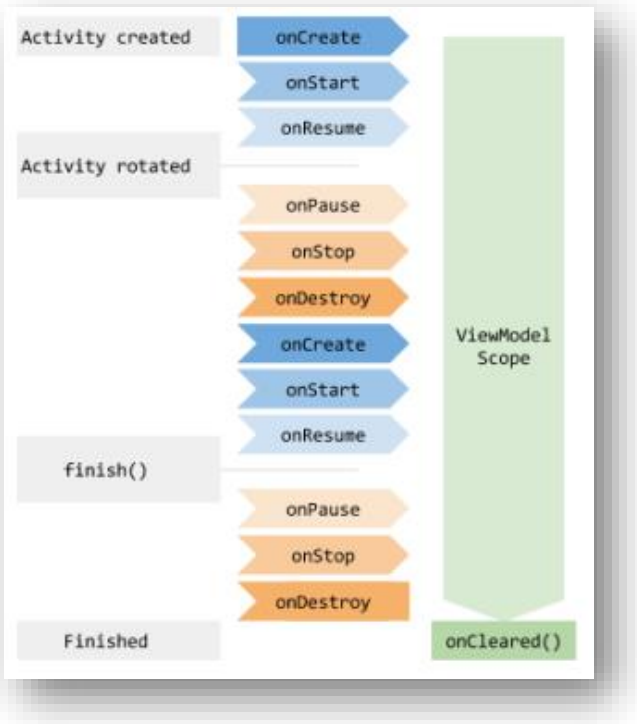

Figura 7: Ciclo de vida de un ViewModel en relación con el ciclo de vida de una Actividad [23]

Este capa se encarga de almacenar y gestionar los datos relacionados con la interfaz de usuario. Las clases que extiendan de ViewModel tienen la posibilidad de hacer que los datos sobrevivan a cambios de configuración. Si se vuelve a crear una actividad, ésta recibe la misma instancia de ViewModel.

En la Figura 7 se puede observar lo que se ha mencionado anteriormente.

Se entiende que el ViewModel existe desde la primera vez que se solicita hasta que finaliza la actividad y se destruye.

Dentro de los ViewModel se encuentran los objetos LiveData y MutableLiveData. LiveData es una clase que hace de contenedor de datos y es observable. Se pueden subscribir a estos objetos y obtener los datos que se vayan añadiendo o actualizando. Un detalle importante es que solo actualiza observadores (o subscriptores) de componentes de aplicaciones que tienen un estado de ciclo de vida activo, como actividades o fragmentos.

De esta manera, lo que ocurre es que se notifica automáticamente a la vista de que los datos han cambiado para que pueda refrescar la interfaz. No se determina, a mano, cuándo se hacen esos cambios.

MutableLiveData es una clase que permite editar el valor almacenado en un objeto *LiveData*. La forma habitual es usar los MutableLiveData en los ViewModel, y solo exponer objetos LiveData inmutables a los observadores.

Esta capa tiene la referencia hacía la capa Modelo.

5.1.3. Capa de datos (Model)

En esta capa se encuentran las clases que trabajan con las diferentes fuentes de datos. Entre estas clases se hacen uso de las bibliotecas Retrofit, Room y Paging. Por ello, en esta capa se encuentran todas las peticiones a la API, y los métodos para acceder a la base de datos local.

Está formada por repositorios que pueden contener muchas fuentes de datos. Estos repositorios nos permiten desacoplar el acceso a la base datos y la API del servidor. Además, los mismos son responsables de exponer datos al resto de la aplicación, y de resolver conflictos entre varias fuentes de datos.

5.2. Diagrama de clases

A continuación, se representa el diagrama de clases UML [4]. Este tipo de diagrama describe una vista estática del modelo de datos. Muestra las clases y los atributos de los que se compone la aplicación, y las relaciones entre las distintas clases.

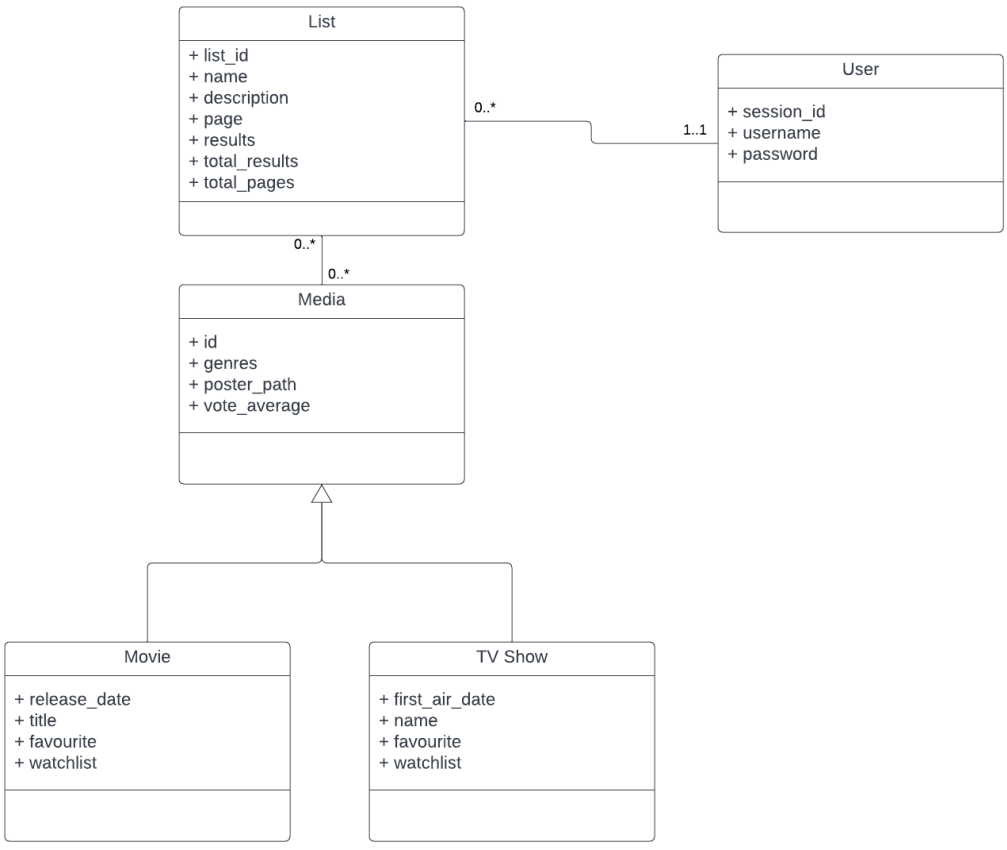

Figura 8: Diagrama de clases UML

La clase User hace referencia a los usuarios de la aplicación. Cada usuario dispone de un nombre de usuario y una contraseña para acceder a la aplicación. Además, tiene un session_id que se utiliza para realizar ciertas peticiones al servidor. La clase List hace referencia a una lista de la aplicación, y dispone de un id, un nombre, una descripción, la página de la lista, un conjunto de películas y/o series en el atributo results, el número de elementos que tiene la lista y el número de páginas. Y la clase Media hace referencia a una película o serie. Las películas y series comparten los atributos id, genres, poster_path y vote_average, por eso se ubican en Media.

Como se puede observar, existe una relación entre la clase User y la clase List, un usuario de la aplicación puede tener cero o muchas listas. Existe otra relación entre la clase List y la clase Media. Esto es debido a que las listas pueden tener de cero a muchas Media, y una Media puede encontrarse en cero o más listas.

Por último, se puede observar que existe una herencia. Esa herencia lo que representa es que los elementos de las listas pueden ser películas o series.

5.3. Bocetos

En este apartado se representan los bocetos realizados antes del desarrollo de la aplicación. Los bocetos están relacionados con los requisitos funcionales que se han mencionado anteriormente.

5.3.1. Inicio de sesión

La Figura 9 representa el inicio de sesión de la aplicación. El usuario debe iniciar sesión para poder acceder a todas las funcionalidades de la aplicación. En el caso de que no tenga una cuenta tiene la posibilidad de registrarse. Hay que indicar que no se ha realizado un boceto del registro de usuario, ya que el registro se realiza a través de la página que nos proporciona la API, The Movie Database.

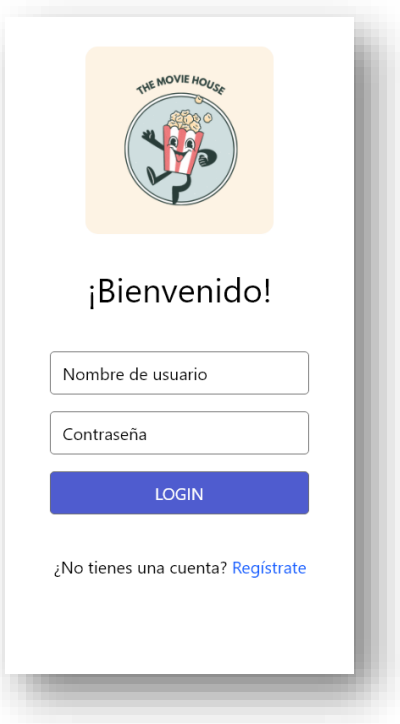

Figura 9: Boceto del inicio de sesión

5.3.2. Listado de películas/series

La Figura 10 representa el listado de películas/series. Cada uno de los elementos que corresponden a las películas/series tienen la imagen de portada, el título, la fecha de estreno, una estrella con la puntuación, y los botones de añadir/quitar el elemento de la lista de favoritos y de la lista de seguimiento (watchlist).

En la parte superior se encuentran unas pestañas que permiten la navegación a otro listado. Se puede navegar hacia otra pestaña clicando en la pestaña que se quiera, o deslizando hacia un lado el listado.

El listado de películas completo es el listado principal, es decir, una vez se inicie sesión en la aplicación se dirigirá al usuario al listado mencionado.

Figura 10: Boceto del listado de películas/series

5.3.3. Listado de películas/series favoritas y en seguimiento

La Figura 12 representa el listado de películas/series favoritas y la Figura 11 representa la lista de seguimiento (watchlist). Ambos listados son similares, aunque en uno de ellos todos los elementos están marcados como favoritos, y en el otro están todos marcados como que están en la watchlist. También se encuentran dos pestañas para navegar al listado de las películas o al de las series.

Hay que señalar que al quitar una película/serie de alguna de las dos listas desaparece de la misma, ya que en ese momento ya no forma parte de ella.

Figura 12: Boceto del listado de películas/series favoritas

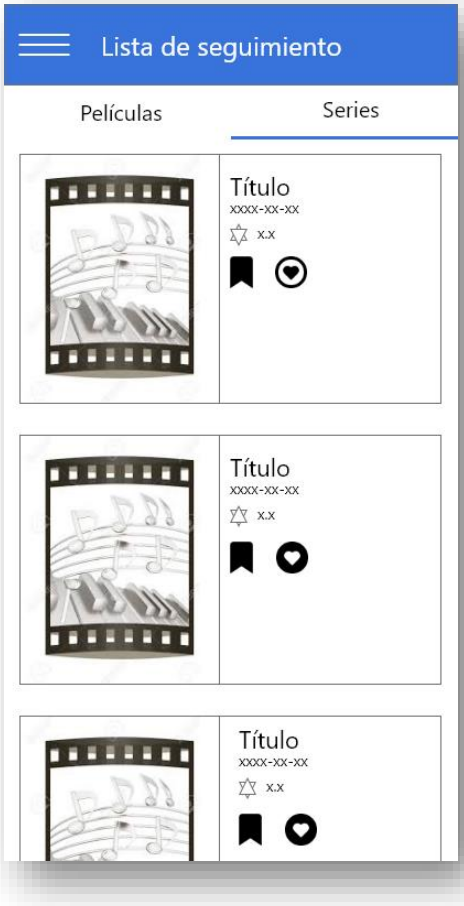

Figura 11: Boceto del listado de películas/series en seguimiento

5.3.4. Navegación principal

La Figura 13 representa la navegación principal de la aplicación. Se trata de un desplegable lateral con el logo, el nombre, y los diferentes menús que permiten navegar de una funcionalidad a otra de la aplicación.

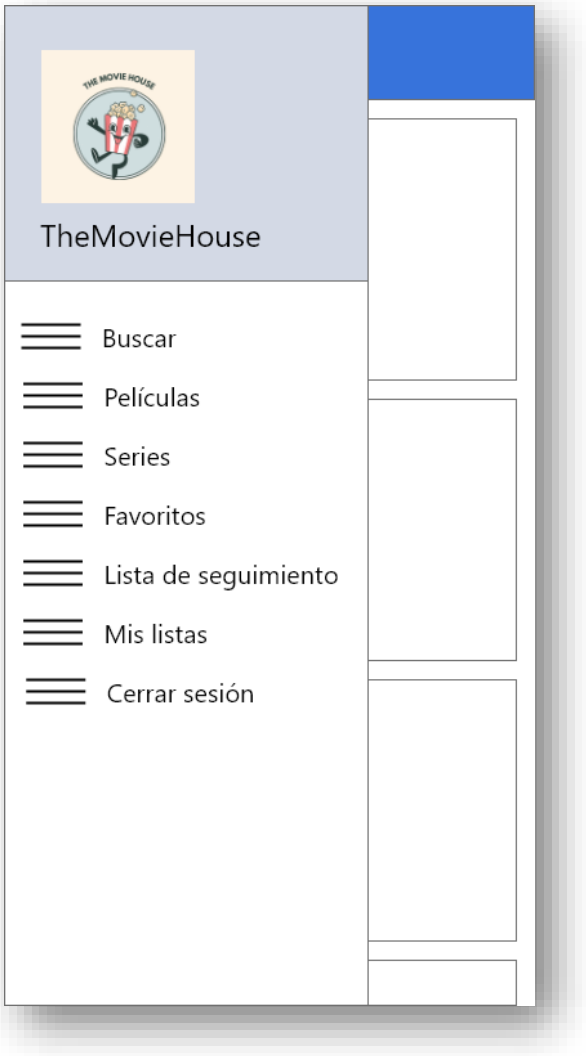

Figura 13: Boceto de la navegación principal

5.3.5. Detalles de la película/serie

La Figura 15 representa los detalles de una película/serie que aparece en cualquiera de los listados. En la parte superior aparece el título correspondiente con la opción de volver atrás. A continuación, se muestra la imagen de la portada de la película/serie, con su nombre y puntuación, la fecha de estreno, el botón para agregarla a una de las listas existentes, y los botones de añadir/quitar el elemento de la lista de favoritos y de la lista de seguimiento (watchlist).

En el caso de tener la película/serie una descripción muy larga o el usuario un dispositivo con una resolución pequeña, se puede desplazar la vista de forma vertical para poder visualizar todo el contenido correctamente.

Si se presiona sobre el botón en forma de signo de más se muestra el diálogo de la Figura 14. Este diálogo contiene un desplegable donde se elige la lista que se desea, y el botón para agregar esa película/serie a la lista seleccionada.

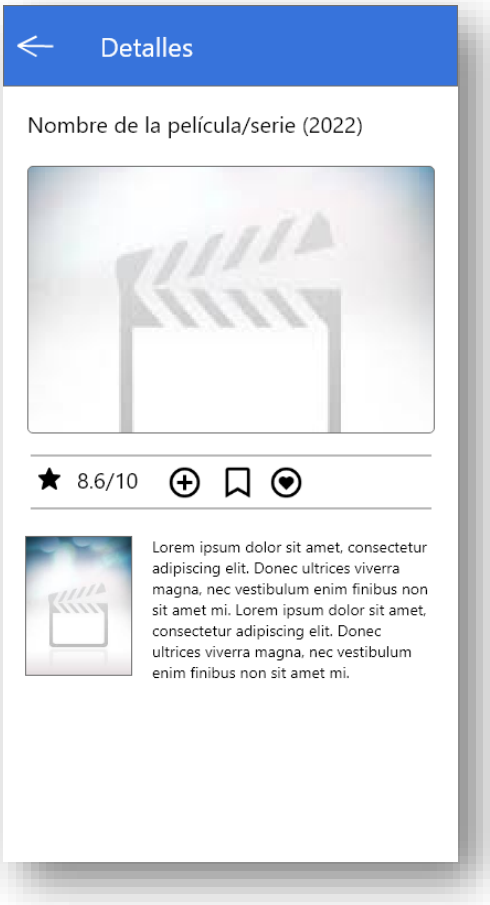

Figura 15: Boceto de los detalles de la película/serie

Figura 14: Boceto de añadir película/serie a lista existente

5.3.6. Buscador de películas/series

La Figura 16 representa el buscador de películas/series. En la parte superior se aprecian el título de la vista y la opción de volver atrás. Aquí también se dispone de dos pestañas para separar la búsqueda de películas y la de series.

Debajo de las dos pestañas se puede visualizar el buscador y el botón que se utiliza para filtrar las películas.

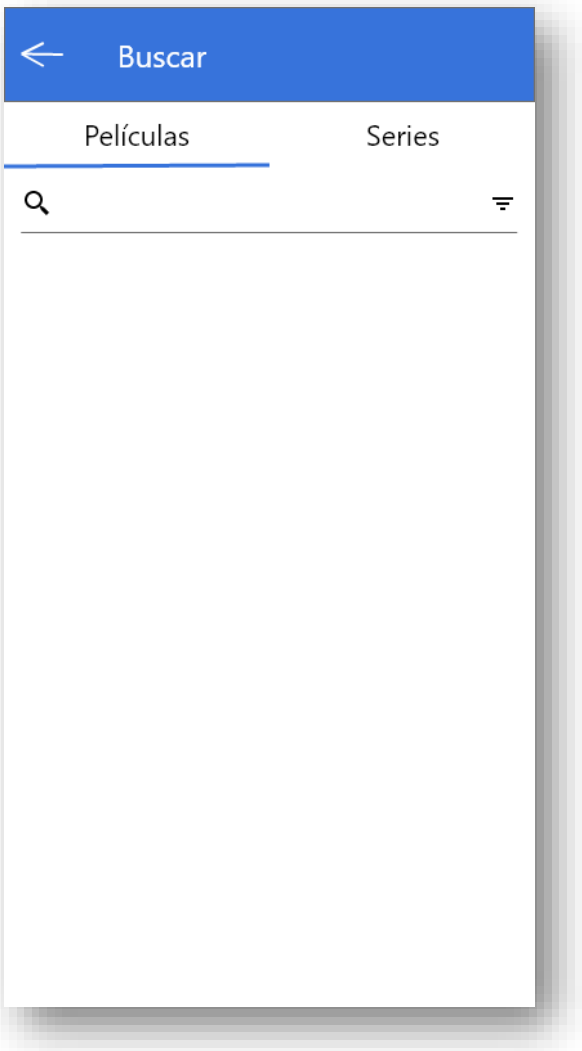

Figura 16: Boceto del buscador de películas/series

5.3.7. Filtro de películas/series

La Figura 17 representa el filtro de películas/series. Se muestra un desplegable con las diferentes formas de ordenar la lista. También se muestra una variedad de géneros disponibles para realizar el filtrado, pudiendo llegar a filtrar por varios géneros a la vez.

Un detalle a señalar es que tanto la búsqueda como el filtrado se realizan por separado para las películas y para las series. Debido a ello, se pueden realizar las búsquedas y filtrados más acotados.

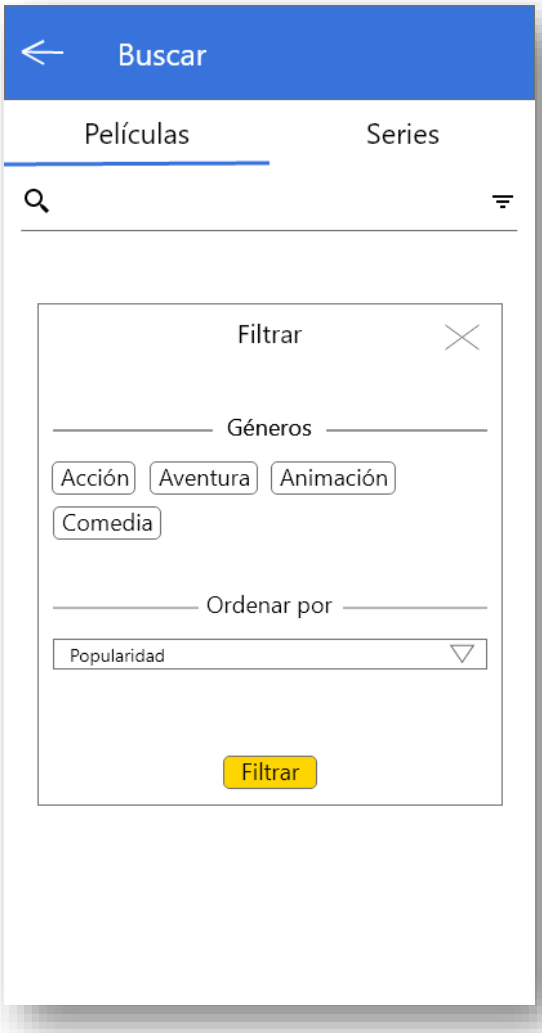

Figura 17: Boceto del filtro de películas

5.3.8. Listado de mis listas

La Figura 19 representa el listado de todas las listas que se cree el usuario. En cada una de las listas se encuentran las películas y series que el usuario ha agregado con anterioridad. Cada una de las listas dispone del nombre, una descripción, y de un botón para borrar la misma.

En esta misma vista el usuario tiene la posibilidad de crear una nueva lista clicando en el botón de la parte superior. Se muestra la Figura 18 con los campos de nombre y descripción de la lista a crear, y el respectivo botón para crearla.

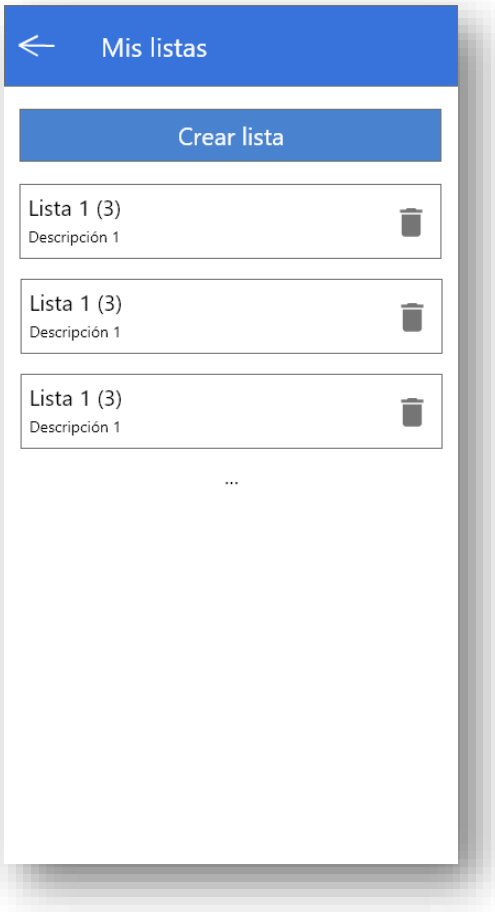

Figura 19: Boceto del listado de mis listas

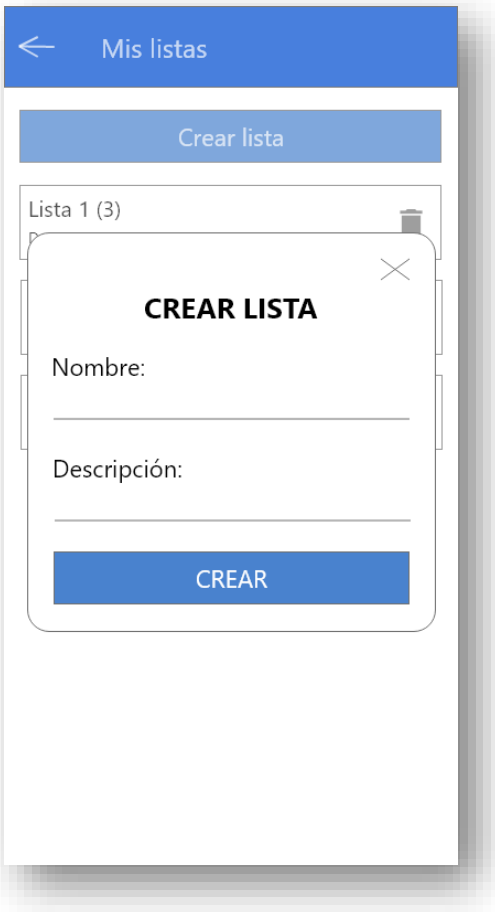

Figura 18: Boceto de la creación de una lista

5.3.9. Listado de películas/series de una de mis listas

La Figura 20 representa el listado de películas/series de una de las listas creadas. Cada uno de los elementos dispone del nombre, fecha de estreno, una estrella con la puntuación, y un botón que permite borrar cada una de las películas/series de la lista.

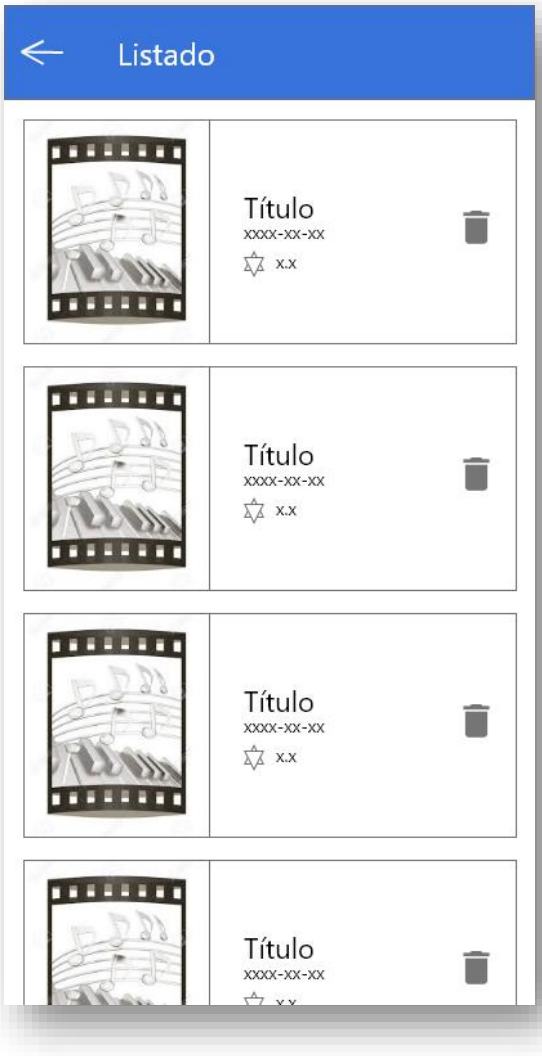

Figura 20: Boceto del listado de películas/series de una de mis listas

6. Tecnologías utilizadas

En este apartado se describen todas las tecnologías y herramientas utilizadas a lo largo de todo el proyecto, así como el entorno de desarrollo empleado.

6.1. Java

El lenguaje de programación empleado es Java [5]. Su icono se puede observar en la Figura 21. Java es un lenguaje de programación desarrollado por Sun Microsystems en 1995. Hoy en día, mediante el lenguaje Java, se puede crear cualquier tipo de aplicación. Tales como aplicaciones de escritorio multiplataforma, aplicaciones web, aplicaciones móviles para Android, etc.

Figura 21: Icono de Java

En la Máquina Virtual Java (JVM) es donde se ejecutan los programas Java, y su misión principal es brindar la portabilidad de las aplicaciones, es decir, su ejecución en cualquier dispositivo.

Es un lenguaje sencillo, rápido, seguro y orientado a objetos, y utiliza una sintaxis semejante a la de C++, pero reducida. Existe una gran cantidad de programadores Java dentro de la colectividad de programadores, lo que genera una gran cantidad de recursos actualizados.

6.2. Lucidchart

Lucidchart [6] es una herramienta de diagramación en la web que permite trabajar de forma colaborativa a los usuarios creando diagramas de clases, organigramas, diagramas de casos de uso, y muchos otros tipos de diagramas. Su icono se puede observar en la Figura 22.

Figura 22: Icono de Lucidchart

Esta herramienta se ha utilizado en el proyecto para realizar el diagrama de clases y el diagrama de casos de uso de la aplicación.

6.3. Github

Github [7] es un portal de trabajo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se ha utilizado para llevar un control de las distintas versiones del proyecto, y para volver a alguna versión de este en el caso de algún error o un estado indeseado. Su icono se puede observar en la Figura 23.

Figura 23: Icono de GitHub

Hoy en día GitHub es usada por más de 28 millones de desarrolladores en todo el mundo.

6.4. GitHub Desktop

Se ha utilizado la aplicación GitHub Desktop, la cual nos permite interactuar con GitHub de una manera más visual utilizando una interfaz de usuario gráfica (GUI) en vez de la línea de comandos o de un buscador web. Su icono se puede observar en la Figura 24.

Se pueden realizar la mayoría de los comandos de Git con esta aplicación, por ejemplo: subir, extraer y clonar repositorios remotos.

Figura 24: Icono de GitHub Desktop

6.5. Postman

Se ha utilizado la herramienta Postman [8], una plataforma que permite a los desarrolladores diseñar y comprobar sus APIs. En el proyecto se ha utilizado para visualizar y comprobar las respuestas de las peticiones a la API. Su icono se puede observar en la Figura 25.

Se ha utilizado principalmente esta aplicación, y no otra, por dos motivos. El primero es que ya se ha empleado durante la carrera en diferentes asignaturas, y eso facilita su uso. Y el segundo motivo es porque constituye el centro de API público más grande del mundo.

Figura 25: Icono de Postman

6.6. Adobe XD

Adobe XD [9] es una aplicación para diseñar interfaces, tanto para páginas web como para aplicaciones. Sus siglas corresponden a Adobe Experience Design. En la actualidad forma parte de Adobe Creative Cloud, un servicio de Adobe que ofrece a los usuarios acceso a diferentes programas. Su icono se puede observar en la Figura 26.

Se ha utilizado especialmente para crear los bocetos de la aplicación y las diferentes interacciones entre los elementos de estos.

Figura 26: Icono de Adobe XD

6.7. Android Studio

El entorno de desarrollo que se ha empleado para el desarrollo de la aplicación es Android Studio [10]. Se trata del entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android, y está basado en el IDE IntelliJ IDEA [11]. Un dato curioso es que remplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones móviles para Android. Su icono se puede observar en la Figura 27.

Al ser un entorno enfocado a aplicaciones móviles para Android, Android Studio ofrece una gran variedad de funciones que aumentan la comodidad y productividad en el desarrollo. Entre las diversas funciones que ofrece podemos destacar las siguientes:

Figura 27: Icono de Android Studio

- Integración con GitHub.
- Variedad de marcos de trabajo y herramientas de prueba.
- Aplicación de cambios, para poder agregar nuevo código, cambios en el mismo, o nuevos recursos a la aplicación en ejecución sin reiniciarla.

- Herramientas para identificar problemas de rendimiento y usabilidad.

7. Implementación de la solución

En este apartado se describe el desarrollo de la solución propuesta, indicando cómo se ha pasado de la propuesta inicial a la solución final, las dificultades encontradas, y las decisiones que se han tenido que tomar para llegar a la solución final.

7.1. Pasos iniciales

Lo primero que se ha realizado antes de empezar a desarrollar es preparar el entorno de desarrollo. Se ha escogido un dispositivo virtual de Android, concretamente un Pixel 2 en la versión Android 12, y se han incluido las dependencias que se han creído oportunas para realizar el proyecto. Algunas de las dependencias se han actualizado durante el proyecto, y otras se han añadido en el mismo.

7.2. Bibliotecas utilizadas

Durante la realización de la aplicación se han utilizado algunas bibliotecas para poder realizar ciertas funcionalidades complejas, y así avanzar hacia la solución propuesta. A continuación, se explican de forma resumida todas las bibliotecas empleadas:

7.2.1. Retrofit

Retrofit [12] se trata de una biblioteca para Android y Java desarrollada por Square [13]. Permite hacer peticiones al servidor tipo: GET, POST, PUT, PATCH, DELETE y HEAD. En este proyecto solo nos han hecho falta las peticiones tipo GET, POST y DELETE. De forma automática pasa las respuestas a un tipo de datos.

7.2.2. Room

Room [14] es una biblioteca de bases de datos creada en Google por el equipo de Android. Permite trabajar con bases de datos SQLite [15] de una forma sencilla, eficaz y rápida. SQLite es un sistema gestor de bases de datos relacionales. Y simplemente en unos minutos, se puede tener una base de datos lista para ser usada.

En la Figura 28 se aprecian los 3 componentes en los que se divide Room:

- Room Database: soporte de la base de datos.
- Data Access Objects (DAO): son los objetos que permiten acceder a la base de datos (a través de sus métodos).
- Entities: clases o entidades que representan las tablas de la base de datos.

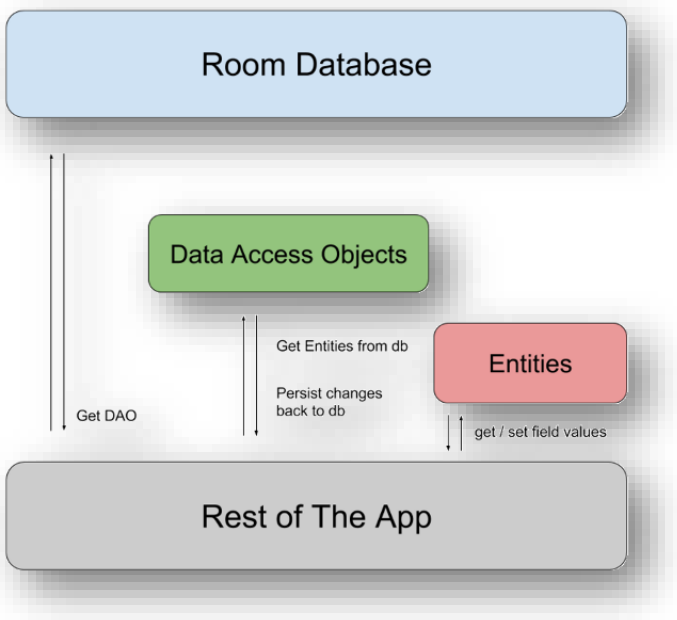

Figura 28: Componentes Room [16]

Se ha utilizado para guardas las películas/series favoritas y en seguimiento en la base de datos local.

7.2.3. Glide

Glide [17] es una biblioteca Android para cargar imágenes, vídeos y GIFs animados. Se ha empleado esta biblioteca para cargar las imágenes que proporciona la API. El servidor proporciona las imágenes a través de URL, entonces mediante esta biblioteca se cargan las imágenes de una forma sencilla. En la Figura 29 se aprecia un ejemplo de su uso.

```
Glide.with(holder.itemView.getContext())
        .load( string: "https://image.tmdb.org/t/p/w500/" + movie.getPoster_path())
        .into(holder.mediaImage);
```

Figura 29: Ejemplo de uso de la librería Glide

7.2.4. Paging

Paging [18] es una biblioteca que permite trabajar con listas de datos de una manera más eficiente. Ayuda a cargar y mostrar páginas de datos de un conjunto de datos más grande. Es más eficiente ya que hay un almacenamiento en caché en memoria para los datos paginados.

En este proyecto el gran el conjunto de datos es obtenido del servidor, y los mismos se utilizan para representar en la aplicación los diferentes listados de películas/series.

7.2.5. Hilt

Hilt [19] es una biblioteca Android de inyección de dependencias. Permite reducir el trabajo reiterativo de inyectar dependencias de forma manual en el proyecto.

Se ha utilizado para obtener las instancias que necesitan las entidades a través de los constructores, en vez de que creen sus propias instancias. Esto trae algunas ventajas:

- Se pueden realizar tests que prueben los componentes de forma aislada.
- No hay dudas de las dependencias, ya que quedan explícitas en el constructor.

7.3. Estructura del proyecto

En este apartado se va a mostrar la estructura final del proyecto Android desarrollado, explicando por encima aquellas partes que se consideran más relevantes. Se puede observar la estructura en la Figura 30.

Los directorios más importantes que conforman el proyecto son el directorio manifests, el directorio java, y el directorio res.

El primer de ellos es el que contiene un archivo llamado AndroidManifest.xml, que describe la información esencial de la aplicación. En él se declaran todas las actividades que componen la aplicación y los permisos que necesita la aplicación para su correcto funcionamiento.

El directorio java contiene todo el código fuente de la aplicación, y se encuentra estructurado a su vez por directorios.

Por último, el directorio res (Resources) contiene todos los recursos sin código del proyecto. Es donde se encuentran las vistas en formato XML, iconos, strings de IU y estilos.

Los dos últimos directorios se van a explicar de una forma más detallada, ya que son paquetes bastante grandes y con mucha información.

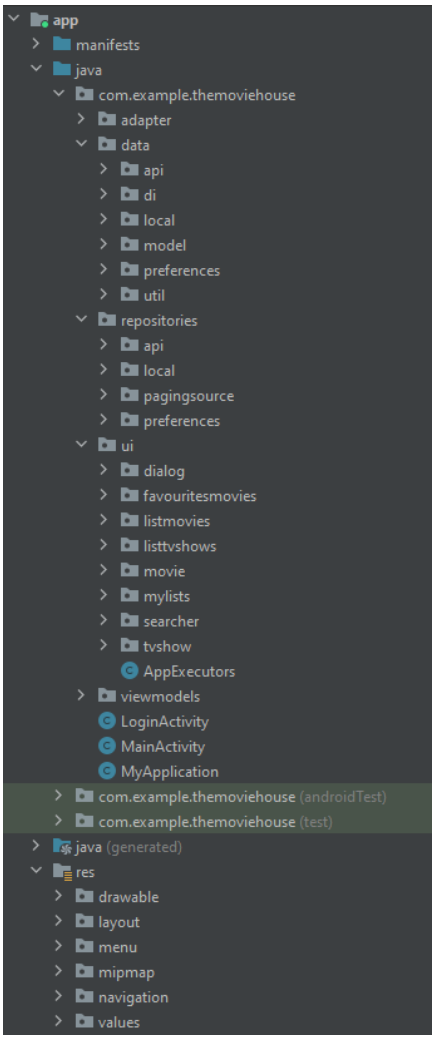

Figura 30: Estructura del proyecto

7.3.1. Directorio java

Como se ha mencionado anteriormente, este directorio contiene todo el código fuente de la aplicación. Está estructurado en varios directorios, los cuales se describen a continuación:

- **adapter**: contiene todos los adapters empleados en la aplicación. Un adapter es un mecanismo que hace de puente entre los datos de la aplicación y las vistas [20].

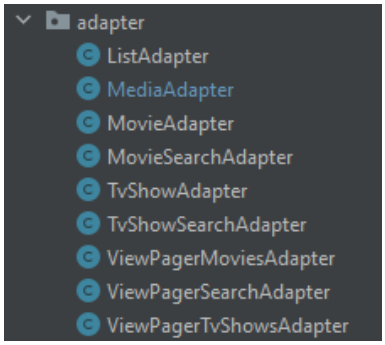

Figura 31: Directorio adapter

Se han utilizado varios adapters al tener diferentes listas con distintos datos en cada una de ellas. Por ejemplo, existe una lista de películas con determinados datos, y otra lista de series con otros datos.

- **data**: contiene todos los objetos java definidos en la aplicación, y todas las clases que representan los DataSource de la aplicación. Un DataSource es una fuente de datos que la aplicación consume.

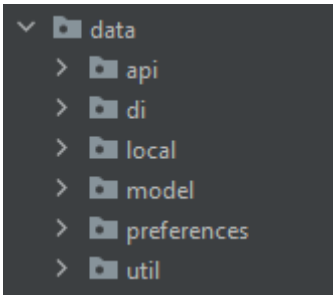

Figura 32: Directorio data

En este proyecto se tienen tres DataSources. Posteriormente se explicarán los DataSource utilizados en el proyecto.

En el paquete model se encuentran todos los objetos java definidos, y en el paquete util todos las clases que sirven de utilidad (como podrían ser enumerados).

Y finalmente, en el paquete di se encuentra el módulo que contiene las diferentes dependencias. Se trata de una clase que a través de unos métodos provee esas dependencias. Estamos hablando de inyección de dependencias.

- **repositories**: contiene todas las clases que actúan como repositorios. Las clases Repository son las que se encargan de gestionar todas las operaciones de persistencia contra los DataSource. A través de ellas podemos obtener, añadir o modificar los datos de las diferentes fuentes de datos.

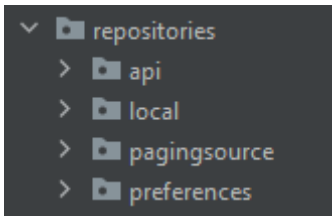

Figura 33: Directorio data

- **ui**: contiene todos los Fragment utilizados en la aplicación. Cada paquete agrupa a un conjunto de fragments, esto es debido a que cada uno de los paquetes representa un menú del desplegable lateral principal.

- **viewmodels**: contiene todos los ViewModel empleados en la aplicación. La clase MediaViewModel se ha centrado en administrar los datos relacionados con la vistas que tienen listado de películas/series, la clase LoginViewModel se ha centrado con la vista de iniciar sesión, la clase SearcherViewModel con la vista que contiene los buscadores, y la clase MediaDetailViewModel con la vista de los detalles de las películas/series.

  Se ha dividido de esta forma para tener más organizado el código, y no todo en una clase.

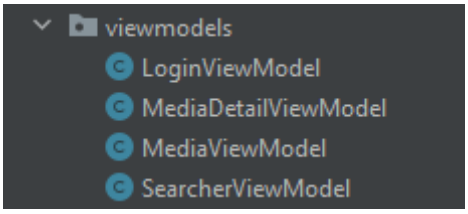

Figura 34: Directorio viewmodels

- Finalmente encontramos las dos activities principales de la aplicación y la clase MyApplication. La primera Activity es para la pantalla de iniciar sesión, y la segunda para todas las demás pantallas de la aplicación. Para ello, se están combinando y/o reutilizando bastantes fragments en la Activity llamada MainActivity.

  La clase MyApplication es una clase que extiende de la clase Application, clase predefinida en Android Studio. Sirve para inicializar los componentes que van a ser necesarios durante el proyecto.

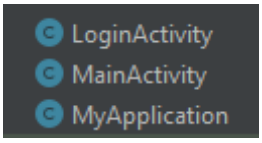

Figura 35: Activities y clase MyApplication

7.3.2. Directorio res

Como se ha mencionado anteriormente, este directorio contiene todos los recursos sin código del proyecto. Está estructurado por varios directorios, se describen los más importantes a continuación:

- **drawable**: contiene todos los iconos utilizados en la aplicación. Todos estos iconos son generados por una herramienta incluida en Android Studio, llamada Image Asset Studio.

- **layout**: contiene todas las vistas de la aplicación. Son archivos XML utilizados por los fragments y activities durante toda la aplicación.

- **menu**: contiene un fichero XML donde se encuentran los diferentes menús del desplegable lateral izquierdo de la aplicación.

- **navigation**: contiene un fichero XML en el que se encuentra el componente Navigation. Su función es representar en un gráfico todas las rutas de navegación de la aplicación. Está diseñado para las aplicaciones que tienen una actividad principal con varios destinos de fragmentos, como es el caso de la actividad MainActivity.

- **values**: contiene archivos que definen los estilos y las cadenas de texto usadas en la aplicación. Se puede destacar el fichero strings.xml, donde se ubican todas las cadenas de texto que aparecen en la aplicación. En este mismo fichero se realizan las traducciones a otros idiomas. Con solo cambiar el idioma del dispositivo, la aplicación lo detectará y usará las cadenas oportunas. La aplicación actualmente soporta el español y el inglés.

7.4. Implementación de la arquitectura

En este apartado se describe como se ha implementado la arquitectura MVVM [21], explicando los diferentes componentes que han intervenido.

Empezando desde la capa más baja, se tienen los DataSources. Recordando, un DataSource es una fuente de datos del repositorio. En el proyecto se tienen tres fuentes de datos posibles:

- **API**, llamada MoviesDataSource. Tiene el acceso al servidor The Movie Database y usa Retrofit para realizar las peticiones.

- **Local**, llamada LocalDataSource. Tiene el acceso a la base de datos y está implementada con Room.

- **Preferencias**, llamada PreferencesDataSource. Tiene el acceso a las preferencias en Android. Se guardan en un archivo XML.

A continuación, se muestran parte de la implementación de los distintos DataSource que tiene el proyecto.

```
@Override
public void getTopRatedMovies(ListMovieDataSourceCallBack listMovieDataSourceCallBack) {
    Call<ListMovie> call = retrofitInterface.getTopRatedMovies();

    call.enqueue(new Callback<ListMovie>() {
        @Override
        public void onResponse(Call<ListMovie> call, Response<ListMovie> response) {
            listMovieDataSourceCallBack.receivedMediaMovies(response.body());
        }

        @Override
        public void onFailure(Call<ListMovie> call, Throwable t) {
            listMovieDataSourceCallBack.onFailedMediaMovies();
        }
    });
}
```

Figura 36: Fragmento de código MoviesDataSourceImpl

En la Figura 36 se puede apreciar un método que realiza una llamada al servidor para obtener las películas mejor valoradas. En Android, el hilo principal de ejecución es el que se encarga de dibujar la interfaz gráfica y atender las acciones del usuario. Al tratarse de una llamada asíncrona, no puede bloquearse el hilo principal esperando la respuesta del servidor, por lo que se encola y se lanza en un hilo secundario. Debe definirse un callback que será invocado cuando se reciba la respuesta del servidor. Finalmente, retorna los datos correctos en la función “onResponse” y un mensaje de error en la función “onFailure” si la petición no ha tenido éxito.

```
@Override
public void addFavouriteMovie(FavouriteMovie favouriteMovie) {
    new Thread(() -> movieDAO.insertFavouriteMovie(favouriteMovie)).start();
}
```

Figura 37: Fragmento de código LocalDataSourceImpl

La Figura 37 representa un método que añade una película a favoritos. A través del DAO de la base de datos (movieDAO) se realiza la adición. Al igual que antes, el acceso a BD no puede realizarse en el hilo principal, porque podría requerir mucho tiempo de ejecución y bloquearía la interfaz. Por ello, se ejecuta en un hilo secundario.

```
@Override
public void getSessionId(SessionIdDataSourceCallback callback) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            String sessionId = sharedPreferences.getString( s: "sessionId",  s1: "");
            callback.receivedSessionId(sessionId);
        }
    }).start();
}
```

Figura 38: Fragmento de código PreferencesDataSourceImpl

Y en la Figura 38 se puede visualizar un método que obtiene un objeto String almacenado en el archivo de preferencias de Android Studio. El acceso a ficheros también debe realizarse en hilo secundario porque puede requerir mucho tiempo y podría bloquear la interfaz.

Por otro lado, se tienen los repositorios. Es una capa que abstrae a los ViewModel de los datos. De esta forma, el ViewModel podría solicitar la obtención de las películas favoritas, por ejemplo, y el Repository sería el encargado de obtenerlas del servidor, o de la base de datos local. Así, es el Repository el responsable de recuperar los datos de las diferentes fuentes disponibles de forma transparente al ViewModel.

En el proyecto se tienen tres clases Repository: MoviesRepository, LocalRepository y PreferencesRepository. Estos repositorios realizan las consultas a los servicios usando el DataSource correspondiente. Inmediatamente retornan los datos a las consultas hechas.

Una capa por encima se sitúa el ViewModel. Se dispone de cuatro ViewModel, explicados el uso de cada uno de ellos en el apartado anterior. Se encargan de mantener el estado de la vista y solicitar el listado de películas/series, u otros datos provenientes de los DataSource. Para ello, solicitan los datos a través de los repositorios.

```
public void getMediaFromAList(int listId) {
    moviesRepository.getMediaFromAList(mediaOfMyListsRepositoryCallBack, listId);
}
```

Figura 39: Fragmento de código MediaViewModel

Se puede apreciar en la Figura 39 la clase MediaViewModel haciendo uso de un Repository para solicitar datos, en este caso las listas creadas por el usuario que ha iniciado sesión.

Finalmente se tiene la capa de interfaz de usuario (UI Layer), donde se encuentran las clases que tienen conexión directa con la interfaz de usuario. Estas clases son las actividades y los fragmentos. A través de la Figura 40 se va a explicar la relación que tiene esta capa con los ViewModel.

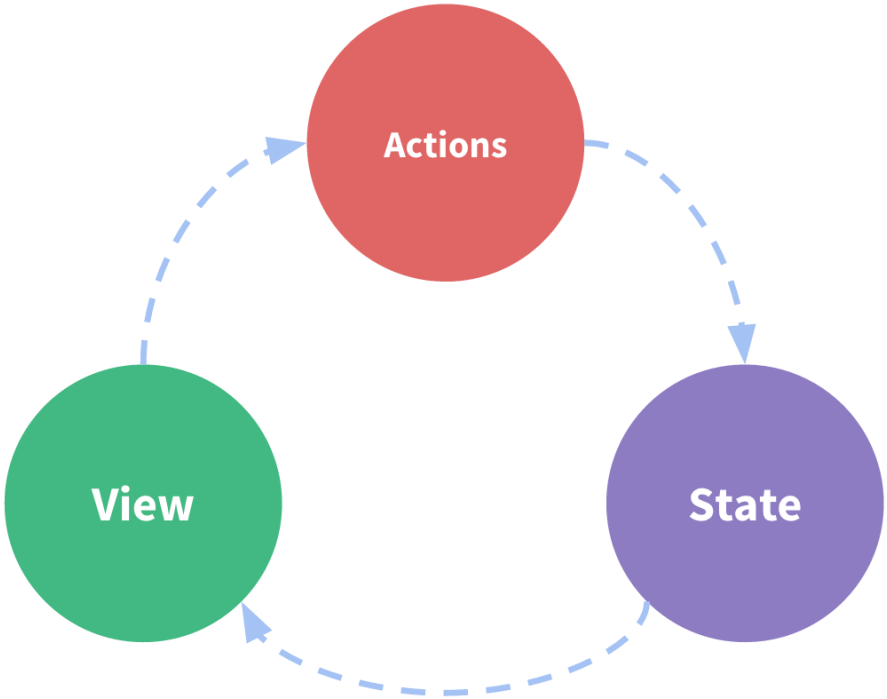

Figura 40: Relación View-ViewModel [22]

Se trata de un patrón UDF (Unidirectional Data Flow) definido para Redux [22], y que se caracteriza porque la información siempre fluye en una única dirección (vista -> acciones -> estado -> vista --> …).

En primer lugar, se van a explicar las tres partes que actúan en esta relación.

- **State**: se trata del estado de la vista en un momento determinado. Los ViewModel se encargan de guardar esos estados.

- **View**: se trata de una descripción de la interfaz basada en el estado actual de la aplicación.

- **Actions**: se tratan de eventos que ocurren en la aplicación basados en entradas de usuario, y que desencadenan actualizaciones en el estado.

De forma más explicativa se puede decir que se dispone de un conjunto de vistas (View). Y la información mostrada en esas vistas es proporcionada por el ViewModel, ya que se encarga de guardar el estado de la vista. Además, el usuario puede interactuar con cada una de las vistas, y al hacerlo puede llegar a desencadenar cambios en los valores de los datos que mantiene el ViewModel. Si llega a ocurrir eso, la vista actualiza la información mostrada de forma inmediata al estar observando el estado actual de la vista.

En la Figura 41 se puede observar parte de la clase ListFragment, perteneciente a la capa de interfaz de usuario. Se puede apreciar como a través del ViewModel, llamado MediaViewModel,

observa el estado de la variable listItems. Eso indica que, al haber una actualización de esa variable por parte de un evento realizado por el usuario, la vista asociada al Fragment ListFragment se verá actualizada de forma inmediata con los nuevos datos de la variable.

```
mediaViewModel.listItems.observe(getViewLifecycleOwner(), new Observer<List<ListItem>>() {
    @Override
    public void onChanged(List<ListItem> listItems) {
        if (listItems != null) {
            adapter.updateList(listItems);
        }
    }
});
```

Figura 41: Fragmento de código de la clase ListFragment

7.5. Principio de inversión de dependencias

Este principio hace que el código sea testable y mantenible. Además, gracias a este principio se puede hacer que la lógica de negocio no dependa de detalles de implementación.

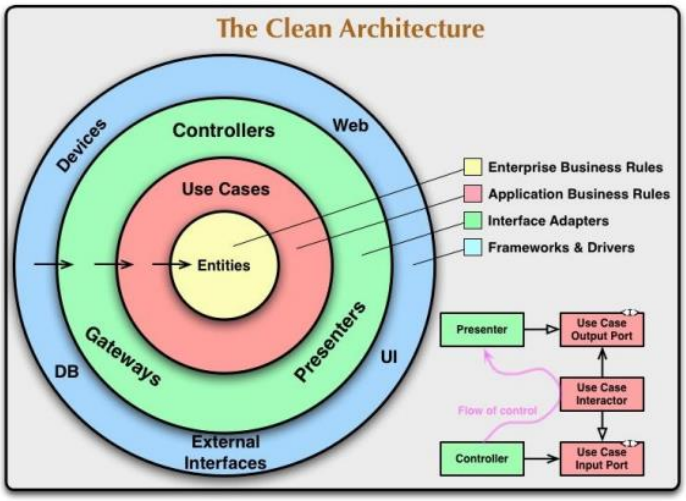

Figura 42: Uncle Bob's Clean Architecture [23]

En la Figura 42 se puede apreciar el esquema de la arquitectura Clean de Uncle Bob [23]. Lo que quiere representar esta arquitectura es que las capas internas (lógica de negocio) no pueden depender de las capas externas (las implementaciones concretas). En caso contrario, cualquier cambio en la implementación nos llevaría también a hacer cambios en la lógica. Por este motivo, se realiza la inversión de dependencias.

Lo que se hace es definir interfaces que detallan qué métodos debe exponer cada uno de los componentes de las diferentes capas. Cada uno de estos componentes deberá implementar la interfaz correspondiente y, de esta forma, se ocultarán los detalles de la implementación, ya que los componentes utilizarán las interfaces definidas. Esto permite cambiar fácilmente la implementación de los componentes de las diferentes capas mientras respeten las interfaces definidas. De esta manera el código no tendrá que conocer cuál es la implementación real para funcionar.

A continuación, se va a presentar un ejemplo de inversión de dependencias en el proyecto desarrollado.

```
public MoviesRepositoryImpl() {
    this.moviesDataSource = new MoviesDataSourceImpl();
}
```

Figura 43: Código sin inversión de dependencias

En la Figura 43 se puede apreciar un fragmento de código que incumple el principio de inversión de dependencias. El problema es que MoviesRepositoryImpl es un objeto de una capa interior y el DataSource es un objeto de una capa exterior (es de una plataforma, tecnología o implementación concreta). Una capa interna no puede depender de una externa.

Ahora en la Figura 44 se ha mejorado un poco. Ahora ya se ha aplicado la inversión de dependencias, pero se puede mejorar.

```
public MoviesRepositoryImpl(MoviesDataSourceImpl moviesDataSource) {
    this.moviesDataSource = moviesDataSource;
}
```

Figura 44: Código con inversión de dependencias (1)

Ahora no tenemos que crear siempre la instancia, ya que se nos pasa por parámetro de entrada. El problema aquí es que recibimos una implementación particular. Todos los métodos de la clase MoviesDataSourceImpl acceden a un DataSource, que provee datos de la API. En la Figura 45 se aprecia uno de los métodos de esa clase.

```
@Override
public void getTopRatedTvShows(ListTvShowDataSourceCallBack listTvShowDataSourceCallBack) {
    Call<ListTvShow> call = retrofitInterface.getTopRatedTvShows();

    call.enqueue(new Callback<ListTvShow>() {
        @Override
        public void onResponse(Call<ListTvShow> call, Response<ListTvShow> response) {
            listTvShowDataSourceCallBack.receivedMediaTvShows(response.body());
        }

        @Override
        public void onFailure(Call<ListTvShow> call, Throwable t) {
            listTvShowDataSourceCallBack.onFailedMediaTvShows();
        }
    });
}
```

Figura 45: Método getTopRatedTvShows() de la clase MoviesDataSourceImpl

Si ahora se quiere cambiar la fuente de origen de los datos y, por ejemplo, traer los datos de la base de datos local, se debe cambiar la implementación pasada por el constructor por la implementación correspondiente. No es una buena técnica, y lo que se busca es no depender de la implementación, no depender de la tecnología.

```
public ApiRepositoryImpl(MoviesDataSource moviesDataSource) {
    this.moviesDataSource = moviesDataSource;
}
```

Figura 46: Código con inversión de dependencias (2)

Ahora, en la Figura 46 se puede apreciar como ya no dependemos de una implementación particular. Esto es debido a que se ha creado una interfaz (MoviesDataSource) que define el comportamiento que se necesita, sin centrarse en una implementación concreta. En la Figura 47 se puede apreciar un fragmento de la interfaz MoviesDataSource.

```
public interface MoviesDataSource {

    void createRequestToken(RequestTokenDataSourceCallBack requestTokenDataSourceCallBack);

    void validateRequestToken(ValidateTokenDataSourceCallBack validateTokenDataSourceCallBack, String username, String password,
                              String request_token);

    void createSession(SessionDataSourceCallBack sessionDataSourceCallBack, String request_token);

    void getPopularMovies(ListMovieDataSourceCallBack listMovieDataSourceCallBack);

    void getPopularMoviesByPage(ListMovieDataSourceCallBack listMovieDataSourceCallBack, Integer page);

    ListenableFuture<ListMovie> getPopularMoviesByPage(Integer page);

    void getTopRatedMovies(ListMovieDataSourceCallBack listMovieDataSourceCallBack);
```

Figura 47: Fragmento de código de la interfaz MoviesDataSource

Esa interfaz se pasa como parámetro del constructor, y alguien se encargará de proveer la dependencia. Ese mismo es Hilt, un inyector de dependencias que anteriormente se ha explicado.

Ahora se puede crear una implementación que obtenga los datos de otra fuente de datos y que implemente esa interfaz. Gracias a eso podemos recibir como parámetro cualquier implementación que implemente esa interfaz, eliminando así la dependencia de una implementación única.

Finalmente, se ha conseguido invertir las dependencias y no depender de una implementación en concreto, mejorando así la escalabilidad y mantenibilidad del proyecto.

7.6. Paging

Se va a presentar como se ha adaptado la biblioteca Paging a la arquitectura de la aplicación. En la Figura 48 se muestran los componentes de la biblioteca, agrupados en las capas que operan.

Como punto de partida, se tiene que explicar que son los datos paginados y para que se necesita esta biblioteca en el proyecto. Los datos paginados se tratan de datos que están agrupados por páginas con un número fijo de elementos. Y esta biblioteca lo que nos permite es trabajar con esas páginas de datos para mostrarlas cuando hagan falta, y no con todos los datos como si fueran un conjunto. De esta forma reducimos el uso de la red y los recursos del sistema.

A continuación, se va a describir los componentes y cómo funcionan juntos para cargar y mostrar datos paginados.

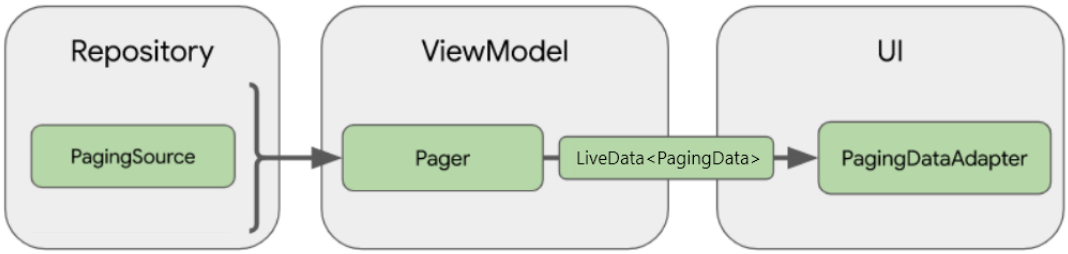

Figura 48: Biblioteca Paging adaptada a la arquitectura MVVM

En la capa del repositorio se encuentra el componente PagingSource. Un objeto PagingSource es capaz de utilizar una fuente de datos que ya existe y encapsular los datos que proporciona para suministrarlos como una página. En la aplicación se dispone de dos objetos que extienden de PagingSource: MoviePagingSource y TvShowPagingSource. El primero es para obtener listas de películas paginadas, y el segundo para obtener listas de series paginadas.

En la capa ViewModel se tiene el componente Pager. Este componente se encarga de indicar a PagingSource que debe obtener la página siguiente o la página anterior conforme el usuario consume la información contenida en la página actual. Además, nos proporciona una API para crear instancias de PagingData. Un objeto PagingData es un contenedor donde se almacenan los datos paginados. Y este mismo objeto es el que conecta la capa ViewModel con la UI (User Interface). Cada instancia de PagingSource se utiliza para cargar páginas de datos para una instancia de PagingData. En la Figura 49 se puede apreciar los dos PagingData.

```
private LiveData<PagingData<Movie>> pagingData;
private LiveData<PagingData<TvShow>> pagingData2;
```

Figura 49: Variables PagingData

Los datos almacenados en el PagingData van creciendo a medida que nos vamos deslizando por las listas de películas/series.

PagingData obtiene una instantánea de los datos disponibles que se almacena en cache para reutilizarlos (en caso de que el usuario navegue hacia delante o hacia atrás). Por ello, si estos datos cambian es necesario invalidar toda la información recogida y generar un nuevo PagingData que muestre la información actualiza de la página actual.

Por último, en la capa de la interfaz de usuario tenemos el componente PagingDataAdapter. Es un adaptador especial cuya función es mostrar en forma de lista datos paginados. En el proyecto se tiene dos adaptadores de este tipo: MovieAdapter y TvShowAdapter. Una para los listados de películas y el otro para los listados de series. Una cosa importante que se agrega en estos adaptadores es la clase DiffUtil. Es una clase de utilidad que calcula la diferencia entre dos listas y emite unas operaciones que convierten la primera en la segunda.

```
private static DiffUtil.ItemCallback<Movie> DIFF_CALLBACK = new DiffUtil.ItemCallback<Movie>() {
    @Override
    public boolean areItemsTheSame(@NonNull Movie oldItem, @NonNull Movie newItem) {
        return oldItem.equals(newItem);
    }

    @Override
    public boolean areContentsTheSame(@NonNull Movie oldItem, @NonNull Movie newItem) {
        return oldItem.equals(newItem);
    }
};
```

Figura 50: Fragmento de la clase MovieAdapter - DiffUtil

En la Figura 50 se puede ver el uso de la clase DiffUtil en el adaptador MovieAdapter. A modo de ejemplo, cuando una película es eliminada del listado se realizan estos métodos para calcular la diferencia entre el listado anterior a la eliminación y el actual. Al comprobar que hay diferencias convierten el primer listado en el segundo. Y se muestra actualizado en la interfaz del usuario.

Ambos métodos hacen uso del método equals(), pero en el primero se comprueba si dos películas son la misma, y en el segundo se comprueba si son dos películas distintas, pero todo el contenido es igual.

8. Pruebas

En este apartado se describen las pruebas llevadas a cabo que demuestran que la aplicación funciona con fidelidad a lo establecido. Se han realizado tres tipos de pruebas: pruebas funcionales, pruebas de calidad y pruebas de usabilidad.

8.1. Pruebas funcionales

En este tipo de pruebas se determina cómo funciona la aplicación con respecto a las necesidades establecidas. La idea principal es verificar que todos los casos de uso se cumplen en la aplicación desarrollada.

Tabla 16: Pruebas de funcionalidad

Casos de uso	¿Cumple?	Notas
UC-01	Si	La aplicación realiza la búsqueda de películas/series con éxito.
UC-02	Si	La aplicación muestra todos los listados de las películas/series con éxito.
UC-03	Si	La aplicación añade películas/series a la lista de favoritos con éxito.
UC-04	Si	La aplicación elimina películas/series de la lista de favoritos con éxito.
UC-05	Si	La aplicación añade películas/series a la lista de seguimiento con éxito.
UC-06	Si	La aplicación elimina películas/series de la lista de seguimiento con éxito.
UC-07	Si	La aplicación filtra películas/series por algunos géneros, y ordena por un criterio (popularidad, fecha de lanzamiento o promedio de votos) con éxito.
UC-08	Si	La aplicación crea una lista de películas/series con éxito.
UC-09	Si	La aplicación elimina una lista de películas/series con éxito.
UC-10	Si	La aplicación muestra los detalles de las películas/series con éxito.
UC-11	Si	La aplicación permite iniciar sesión en la misma con éxito.
UC-12	Si	La aplicación permite cerrar sesión en la misma con éxito.
UC-13	Si	La aplicación añade una película/serie a una de las listas creadas con éxito.
UC-14	Si	La aplicación elimina una película/serie de una de las listas creadas con éxito.
UC-15	Si	La aplicación permite registrar usuarios en la misma con éxito.

A raíz de la Tabla 16, se puede observar que la aplicación realiza todos los casos de uso con éxito. Eso quiere decir que todas las funcionalidades realizadas cumplen con lo establecido en la propuesta inicial.

8.2. Pruebas de calidad

Este tipo de pruebas se encargan de verificar que se cumplen los criterios definidos por Google para la calidad de las aplicaciones móviles Android [24]. Cuantos más criterios se cumplan mayor calidad de las aplicaciones.

A continuación, se muestran un conjunto de los criterios principales de calidad y las pruebas que se han realizado para comprobar que se cumplen estos:

Tabla 17: Pruebas de calidad - Experiencia visual

Área	Criterio	Test
Navegación	Admite navegación estándar mediante el botón Atrás.	Correcto
	Admite navegación por gestos.	Correcto
	Preserva y restaura correctamente el estado del usuario o la app, por ejemplo, al hacer una navegación hacia atrás.	Correcto
Notificaciones	Las notificaciones siguen las recomendaciones de Material Design.	No aplica
	Las notificaciones brindan compatibilidad con la acción de respuesta directa.	No aplica
IU y gráficos	Admite tanto la navegación horizontal como la vertical.	Incorrecto, solo admite navegación vertical
	Utiliza toda la pantalla en ambas orientaciones.	Incorrecto, solo en la orientación vertical
	Controla correctamente las transiciones rápidas entre las orientaciones de la pantalla sin perder el estado.	Incorrecto, solo en la orientación vertical
Calidad visual	Muestra imágenes, gráficos, texto y otros elementos sin distorsión ni pixelado.	Correcto
	Muestra el texto y los bloques de texto de forma aceptable, con suficiente espacio entre el texto y los elementos que los rodean	Correcto
	Admite el tema oscuro.	Correcto

Las pruebas de calidad respecto a la experiencia visual se pueden observar en la Tabla 17. Se cumplen con los criterios de calidad menos los orientados con la orientación. A pesar de que en la guía de Android se indica que esto debería hacerse, se ha decidido fijar la orientación del dispositivo en modo vertical porque es más cómodo.

Tabla 18: Pruebas de calidad - Funcionalidad

Área	Criterio	Test
Audio	Cuando la aplicación regresa a primer plano la reproducción de audio continúa como estaba antes.	No aplica
	Admite la reproducción en segundo plano si la reproducción de audio es una función principal.	No aplica

	Cuando el usuario inicie la reproducción de un audio la aplicación comenzará la reproducción o avisará de que los datos se están preparando	No aplica
	Solicita el foco de audio cuando el audio comience a reproducirse y lo abandona cuando se detenga.	No aplica
	Controla las solicitudes de foco de audio de otras aplicaciones.	No aplica
Multimedia	Cuando se reproduce música en segundo plano se crea una notificación con estilo MediaStyle.	No aplica
	Admite la reproducción en pantalla completa cuando se reproducen videos.	No aplica
	Codifica contenido de video mediante el estándar de comprensión de videos HEVC.	No aplica
Se comparte	Usa Android Sharesheet [25] cuando comparta contenido.	No aplica
Servicio en segundo plano	Siempre y cuando sea posible evitar que se ejecuten servicios en segundo plano.	No aplica
	Admite correctamente las funciones de administración de energía que se introdujeron en Android 6.0.	No aplica

La aplicación no utiliza en ningún momento ni audio ni contenido multimedia, y no contiene servicios en segundo plano. Lo único que sí tiene son las tareas en segundo plano, pero no servicios.

Tabla 19: Pruebas de calidad - Rendimiento y estabilidad

Área	Criterio	Test
Estabilidad	No falla ni bloquea el subproceso de IU que provoca errores ANR.	Correcto
Rendimiento	Carga rápidamente.	Correcto
	Renderiza los programas cada 16 ms a efectos de alcanzar 60 fotogramas por segundo.	Correcto
	Cuando se activa StrictMode [26] no se ven destellos rojos en la pantalla.	Correcto
SDK	Se ejecuta en la última versión pública de la plataforma de Android sin que se produzcan fallos.	Correcto
	Se orienta al SDK más reciente mediante la configuración del valor **targetSdk**.	Correcto
	Se compila la app con el último SDK estableciendo el valor **compileSdk**.	Correcto
	Todos los SDK de terceros usados están actualizados.	Correcto
	No usa interfaces que no pertenecen al SDK.	Correcto
Batería	Admite correctamente las funciones de administración de energía que se introdujeron en Android 6.0	No aplica

Al probar la aplicación en el emulador de Android Studio hay ciertos criterios de la Tabla 19 que no se cumplían. Al probarlo en un dispositivo Android sí que se cumplen todas las pruebas de rendimiento.

Tabla 20: Pruebas de calidad - Privacidad y seguridad

Área	Criterio	Test
Permisos	Solicita solo la cantidad mínima absoluta de permisos que necesita.	Correcto
	Solo debe de solicitar permiso de acceso a datos sensibles si están relacionados con los casos de uso principales.	No aplica
	Solicita permisos de tiempo de ejecución en contexto en lugar de hacerlo durante el inicio de la aplicación.	No aplica
	Transmite de forma clara el motivo por el que se necesitan ciertos permisos.	No aplica
Datos y archivos	Todos los datos sensibles se almacenan en el almacenamiento interno de la app.	No aplica
	Los datos personales o sensibles de los usuarios no se registran en ningún registro de la app.	Correcto
	No usa ID de hardware que no se puedan restablecer (IMEI) para fines de identificación.	No aplica
Identidad	Brinda sugerencias para autocompletar información sensible.	No aplica
	Integra One Tap [27] para optimizar varios tipos de credenciales.	No aplica
	Integra la autentificación biométrica para proteger la información sensible y transacciones financieras.	No aplica
Componentes de la aplicación	Solo se exportan los componentes de la aplicación que comparten datos con otras aplicaciones o con los que otras aplicaciones necesitan comunicarse.	No aplica
	Todos los intents y transmisiones siguen las prácticas recomendadas por Android.	Correcto
	Todos los proveedores de contenido que comparten contenido entre tus aplicaciones usan android:protectionLevel="signature" para los permisos personalizados.	No aplica
Redes	Todo el tráfico de red se envía mediante SSL.	Correcto
	Declara una configuración de red.	No aplica
	Si utiliza los Servicios de Google Play se inicializará el proveedor de seguridad cuando lo haga la aplicación.	No aplica
Bibliotecas	Están actualizados los SDK, las dependencias y las bibliotecas.	Correcto
	No incluye bibliotecas de depuración en la app de producción.	No aplica
WebViews **[28]**	Para acceder al contenido local usa WebViewAssetLoader [29].	No aplica.
	Las WebViews no deben usar addJavaScriptInterface() con contenido que no sea de confianza.	No aplica.
Ejecución	No carga código dinámicamente desde fuera del APK de la aplicación. No modifica el comportamiento de la aplicación en tiempo de ejecución.	Correcto
Criptografía	Usa un generador de números aleatorios y algoritmos criptográficos fuertes.	No aplica

En la Tabla 20 se puede apreciar las pruebas de calidad con respecto a la privacidad y seguridad. La aplicación solo solicita los permisos de acceso a internet. Además, la aplicación no dispone de datos sensibles, por tanto, muchos de los criterios mencionados sobre los datos y los permisos no se aplican. Se ha comprobado que la aplicación dispone de todas las bibliotecas y SDK actualizados.

La aplicación no utiliza WebViews, unos componentes que permiten que las aplicaciones puedan mostrar contenido web sin necesidad de abrir un navegador externo. Por ello, todos los criterios relacionados con esta área tampoco se aplican.

Tabla 21: Pruebas de calidad - Google Play

Área	Criterio	Test
Políticas	Cumple estrictamente con los términos de la Política de Contenido para Desarrolladores de Google Play	No aplica
	En función de las recomendaciones de Clasificación del Contenido se establece el nivel de madurez de la aplicación de forma correcta.	No aplica
Página de detalles de la aplicación	El directorio de la aplicación incluye un gráfico central de alta calidad, y dicho gráfico no se parece a un anuncio publicitario.	No aplica
	Todas las capturas y videos de la aplicación hacen referencia solo a dispositivos Android.	No aplica
	Todas las capturas y videos de la aplicación no representan el contenido ni la experiencia que ofrece la aplicación de forma confusa.	No aplica
Asistencia para el usuario	Los erros comunes que sean informados en la pestaña Reseñas de la página de Google Play se abordan cuando ocurren en muchos dispositivos diferentes.	No aplica

En la Tabla 21 se puede apreciar las pruebas de calidad con respecto a Google Play. Como la aplicación todavía no está en fase de producción no se ha subido a Google Play, por lo tanto, estos criterios no aplican. Si se subiera la aplicación a Google Play se debería cumplir con esos criterios.

8.3. Pruebas de usabilidad

Se ha realizado una prueba de usabilidad para comprobar el funcionamiento de la aplicación. Este tipo de pruebas muchas veces nos ayudan a encontrar fallos en las funcionalidades.

Se ha creado un guion con los pasos que deben seguir los usuarios para comprobar las funcionalidades principales de la aplicación. Deben seguir todos los pasos en orden y rellenar un formulario al terminar. El formulario contiene preguntas sobre el aspecto y uso de la aplicación, y se puede encontrar en el Anexo 2: Cuestionario usabilidad.

Guion de testeo:

1. Iniciar la aplicación.
2. Registrarse (usar un correo válido).
3. Iniciar sesión.
4. Posicionarse en el listado de mis listas de películas/series.

5. Crear una lista de películas/series.
6. Buscar la película “Batman” en el buscador de películas.
7. Seleccionar el primer elemento de la búsqueda.
8. Agregar la película a favoritos.
9. Agregar la película a la lista de seguimiento (watchlist).
10. Agregar la película a la lista creada anteriormente.
11. Posicionarse sobre lista de favoritos y observar que aparece la película que hemos agregado antes.
12. Elimina la película de favoritos.
13. Posicionarse sobre la lista de seguimiento y observar que aparece la película que hemos agregado antes.
14. Elimina la película de la lista de seguimiento.
15. Posicionarse en el listado de mis listas de películas/series de nuevo y observar si se ha añadido en la lista correctamente.
16. Filtra películas que traten el tema “Desigualdad de género”.
17. Agrega algunas cuantas a favoritos.
18. Posicionarse sobre lista de favoritos y observar que aparece la película que hemos agregado antes.
19. Finalmente cierra sesión en la aplicación.

La aplicación la han probado 10 personas y en las Figura 51, Figura 52, Figura 53, Figura 54, Figura 55, Figura 56, Figura 57, Figura 58, Figura 59 y Figura 60 se pueden observar los resultados obtenidos.

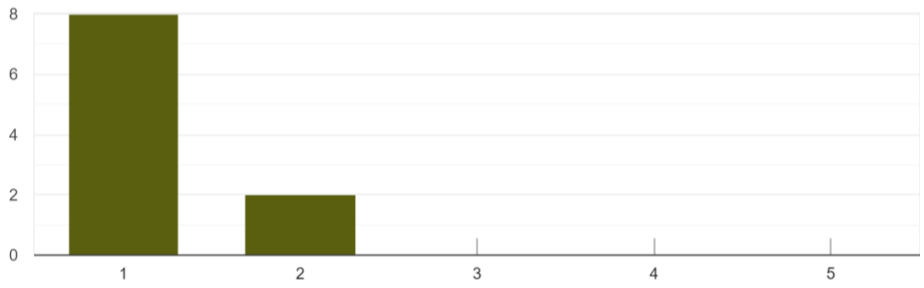

Figura 51: Resultados de "Fue simple de usar esta App"

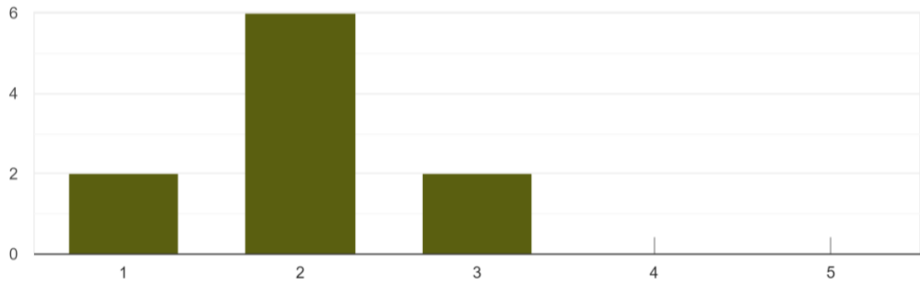

Figura 52: Resultados de "Es fácil encontrar en la App la información que necesito"

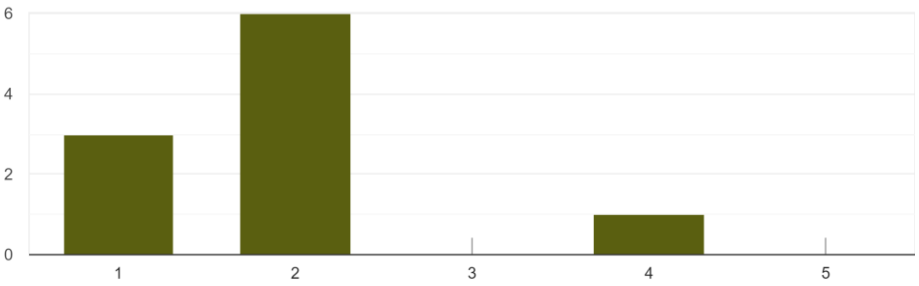

Figura 53: Resultados de "Pude completar eficazmente las tareas y los escenarios utilizando esta aplicación"

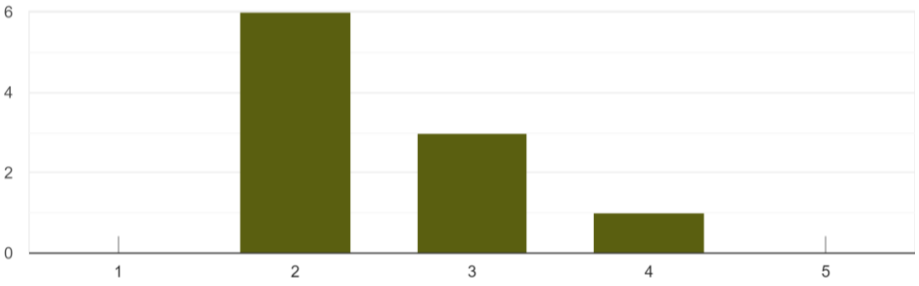

Figura 54: Resultados de "La interfaz de la App es agradable"

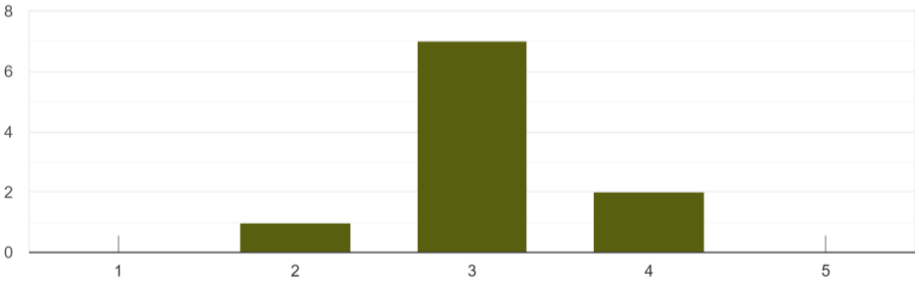

Figura 55: Resultados de "El sistema daba mensajes de error que me indicaban claramente cómo solucionar los problemas"

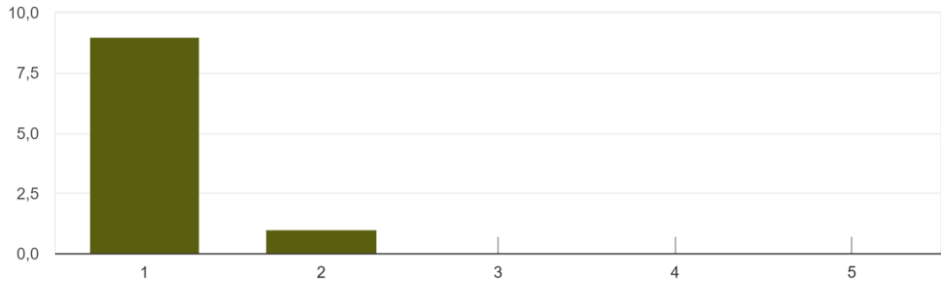

Figura 56: Resultados de "Fue fácil aprender a utilizar esta aplicación"

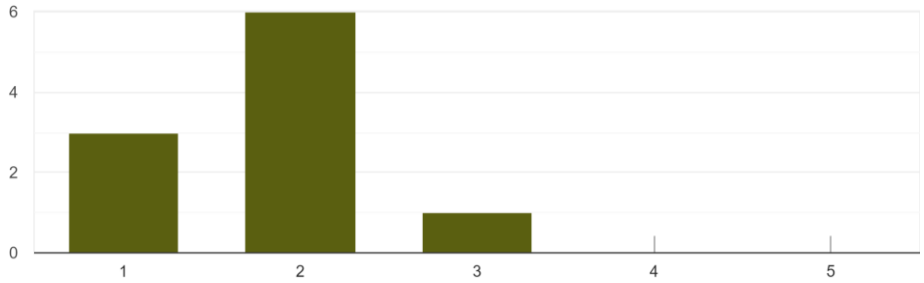

Figura 57: Resultados de "La organización de la información de la App fue clara"

Observando las preguntas que se respondían usando una escala de calificación, se puede apreciar que la aplicación fue simple de usar y fácil aprender a utilizarla. Además, también se puede ver que todos han llegado a completar con éxito las tareas menos una persona. Esto último se aprecia en la Figura 53.

Por otro lado, también hay usuarios a los que no les ha gustado del todo la interfaz de la aplicación. Eso se tiene anotado como posible mejora para una futura versión.

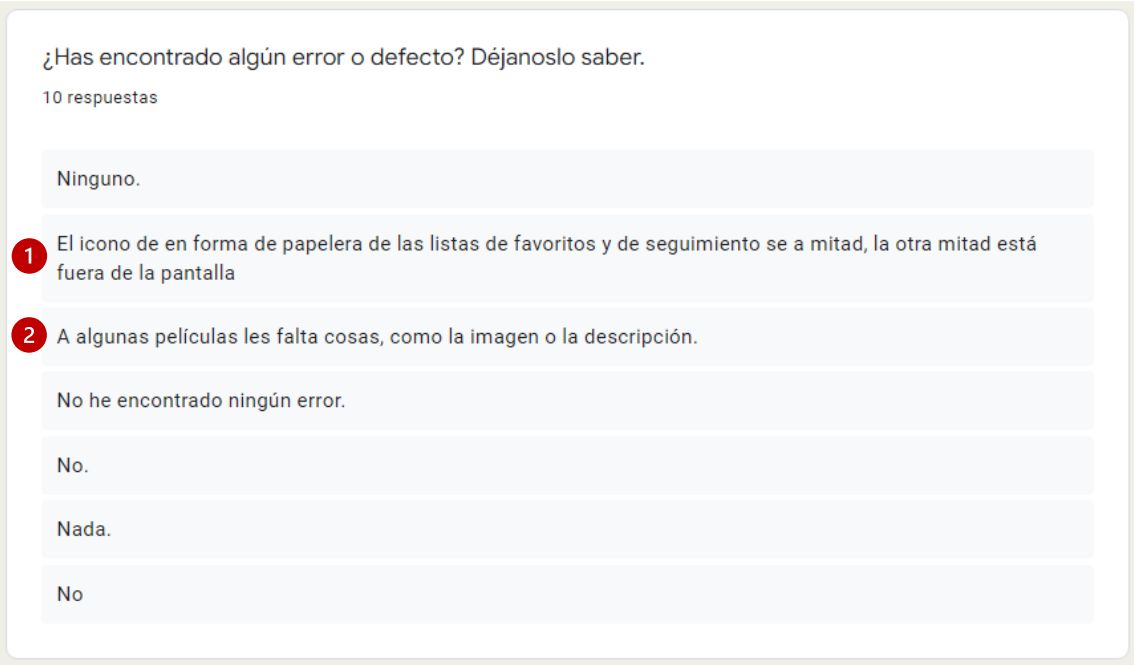

Figura 58: Resultados de "¿Has encontrado algún error o defecto? Déjanoslo saber"

Con respecto a la Figura 58, se han encontrado algunos errores que se han solucionado con éxito. Los errores son los dos marcados con los número en rojo. El primero de ellos se ha solucionado bastante rápido, ya que solo había que adaptar unos detalles de la vista asociada. Para resolver el segundo se ha estado depurando la aplicación para comprobar realmente lo que fallaba, y se ha llegado a identificar cual es el motivo de que no aparezcan ciertos detalles de las películas. Es debido al servidor, ya que no dispone de todos los datos de las películas y series. Como no es por un fallo de la aplicación no se puede llegar a resolver.

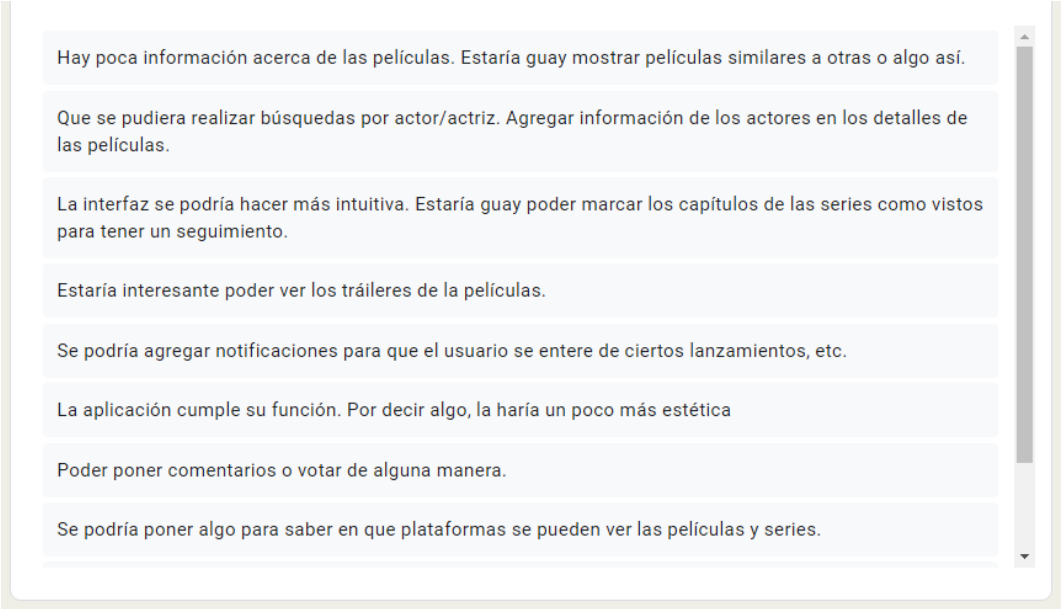

Figura 59: Resultados de " ¿Qué aspectos de la aplicación mejorarías? ¿Qué funcionalidades te parecerían interesantes que estuvieran en la aplicación? Déjanoslo saber." (1)

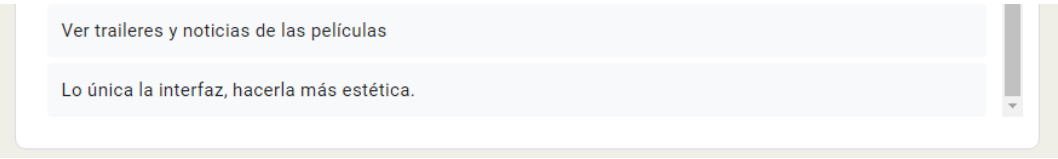

Figura 60: Resultados de " ¿Qué aspectos de la aplicación mejorarías? ¿Qué funcionalidades te parecerían interesantes que estuvieran en la aplicación? Déjanoslo saber." (2)

En las Figura 59 y Figura 60 se pueden apreciar las mejoras que han sugerido los usuarios. Todas estas recomendaciones se tendrán en cuenta para las siguientes versiones de la aplicación para mejorar sus funcionalidades, así como para incorporar nuevas.

9. Resultado final

En el apartado actual se presentan las capturas de las interfaces de usuario de la aplicación.

9.1. Interfaces de usuario

A continuación, se presentan las interfaces de usuario resultantes después de todo el desarrollo de la aplicación.

9.1.1. Inicio de sesión

En la Figura 61 se muestra el inicio de sesión (UC-11 Iniciar sesión) con el logo de la aplicación. Debajo del logo se puede ver un texto de bienvenida y los campos para introducir el nombre de usuario y contraseña. Finalmente, en la parte inferior de la captura se aprecia el botón para iniciar sesión y la opción de registrarse (UC-15 Registrarse), en el caso de no tener cuenta.

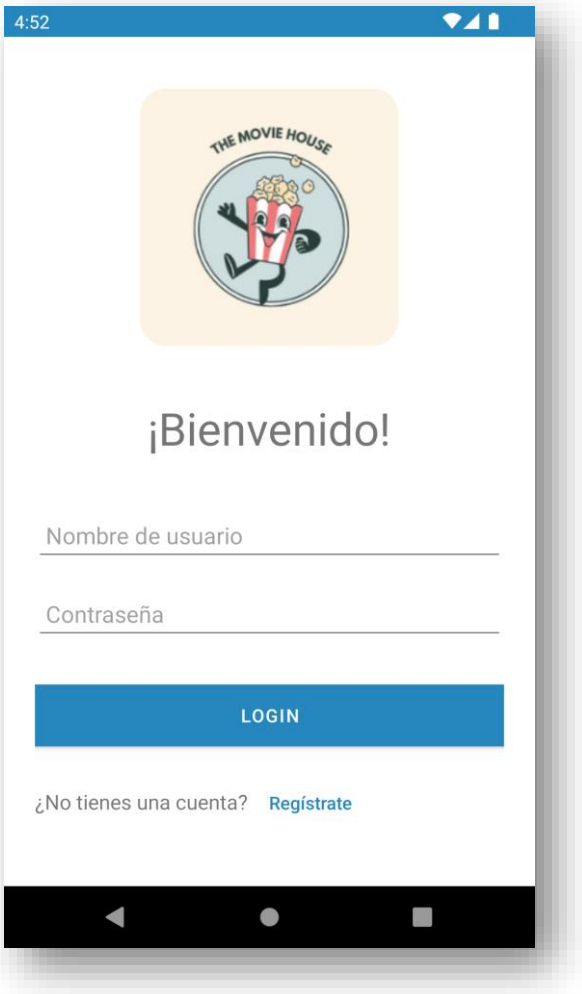

Figura 61: Inicio de sesión

9.1.2. Listado de películas/series

En las Figura 63 y Figura 62 se muestran capturas de pantalla de dos listados de la aplicación. Representan el caso de uso UC-02 Mostrar lista de películas/series. El primero corresponde al listado de películas populares, y el segundo al listado de series populares. Se ha decidido mostrar ambos listados, películas y series, para que se pueda apreciar la similitud entre ellos.

Un detalle por destacar es que los datos que se muestran están siempre sincronizados con los del servidor. También hay que señalar que las películas se van cargando poco a poco cuando se desliza hacia abajo, gracias a la biblioteca Paging.

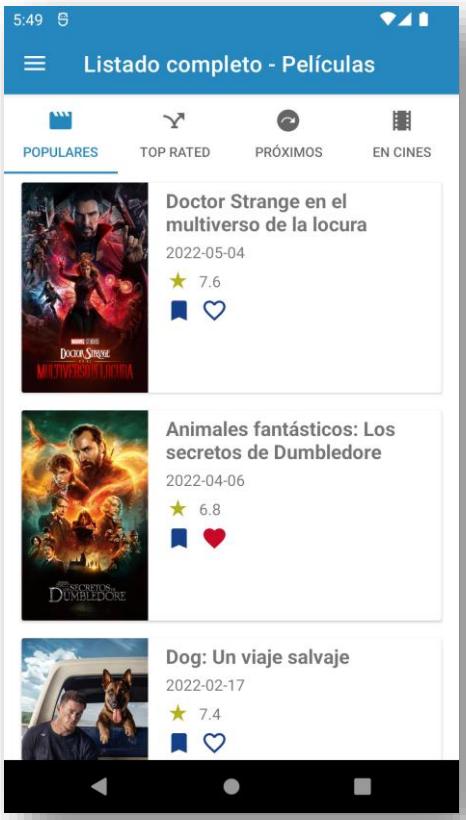

Figura 63: Listado de películas populares

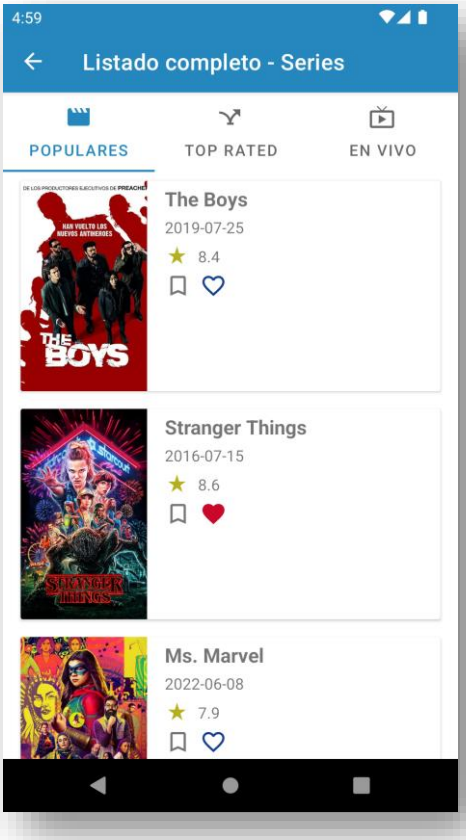

Figura 62: Listado de series populares

9.1.3. Listado de películas/series favoritas y en seguimiento

Se muestran en las Figura 64 y Figura 65 las capturas de pantalla de los listados de películas/series favoritas y en seguimiento.

Para añadir una película/serie a favoritos se ha de pulsar sobre el elemento en forma de corazón desde alguno de los listados de la aplicación (UC-03 Añadir película/serie a favoritos), o desde los detalles de esta. Para quitar una de favoritos se puede hacer desde el mismo sitio en el que se añade, o desde el propio listado de favoritos (UC-04 Quitar película/serie de favoritos).

El caso de añadir y quitar películas/series de la lista de seguimiento sigue la misma dinámica, pero se trata de otro listado y el elemento a pulsar esta vez es en forma de marcador. Corresponde con el caso de uso UC-05 Añadir película/serie a lista de seguimiento (watchlist) y UC-06 Quitar película/serie de lista de seguimiento (watchlist).

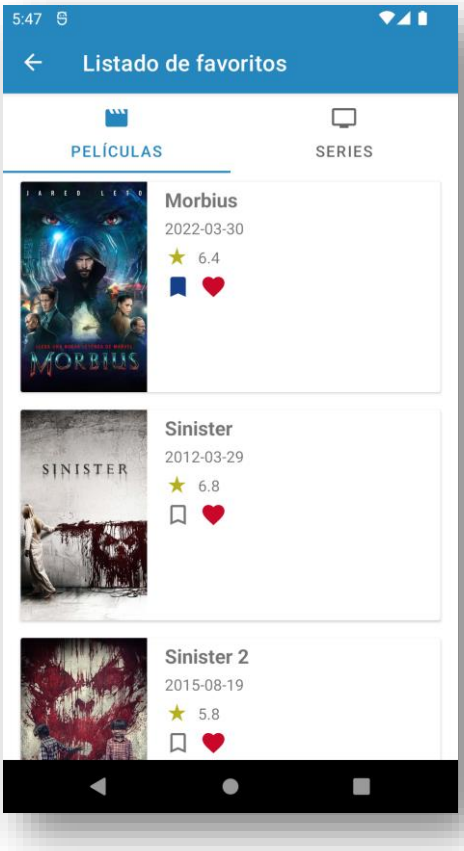

Figura 64: Listado de películas/series favoritas

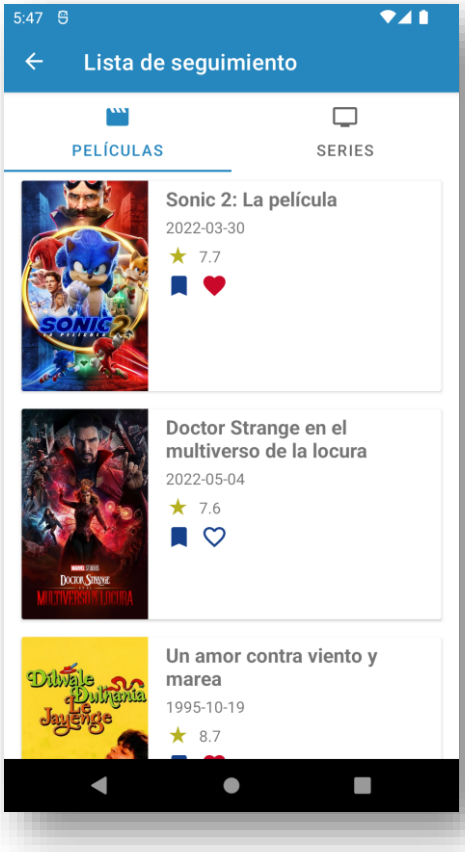

Figura 65: Listado de películas/series en la lista de seguimiento

9.1.4. Navegación principal

En la Figura 66 se muestra la navegación principal de la aplicación. Se trata de un desplegable lateral en el que se muestra también el logo de la aplicación y el nombre de esta. Así mismo, debajo del nombre se puede ver los distintos menús agrupados por secciones. Se puede apreciar en la última sección el menú "Cerrar sesión". La funcionalidad de este botón representa el caso de uso UC-12 Cerrar sesión.

Al iniciar sesión siempre se accederá por defecto al menú "Listado completo - Películas" de la sección "Películas".

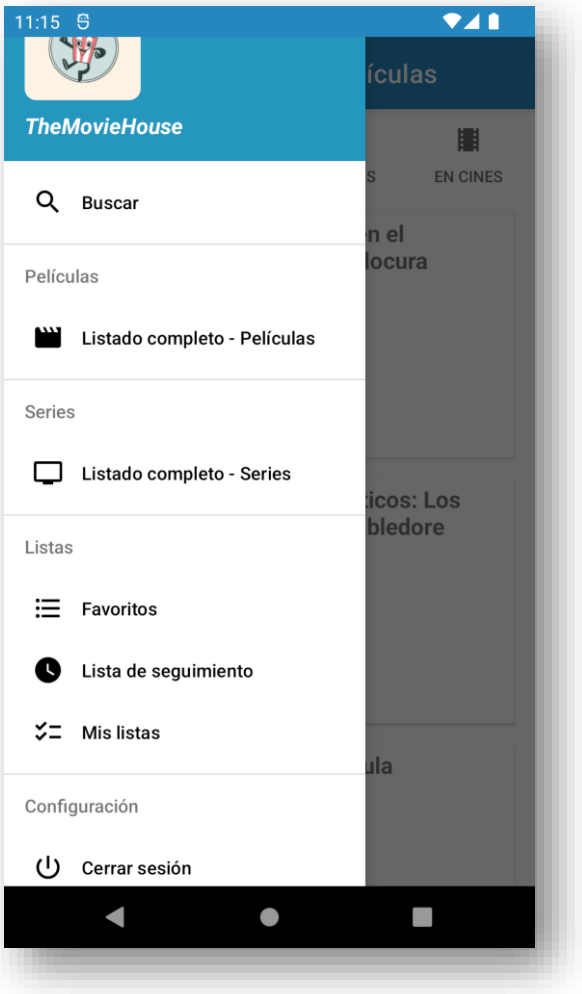

Figura 66: Navegación principal

9.1.5. Detalles de la película/serie

En la Figura 67 se muestra la captura de pantalla de los detalles de la película/serie, que representa el caso de uso UC-10 Mostrar detalles de una película/serie. Se puede apreciar los diferentes botones para agregar la película/serie a una lista existente, a la lista de seguimiento, o a la lista de favoritos.

Si se pulsa sobre el primer botón se muestra el diálogo de la Figura 68, donde se representa la funcionalidad que permite agregar una película/seria a una lista existente (UC-13 Añadir una película/serie a una de mis listas creadas).

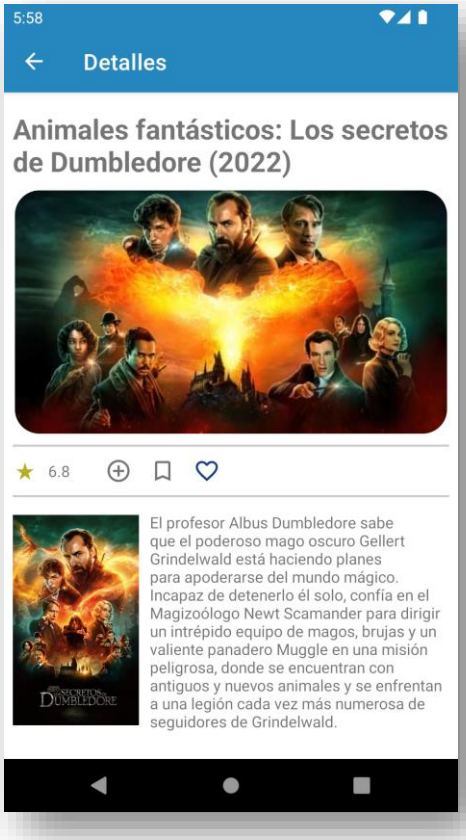

Figura 67: Detalles de la película/serie

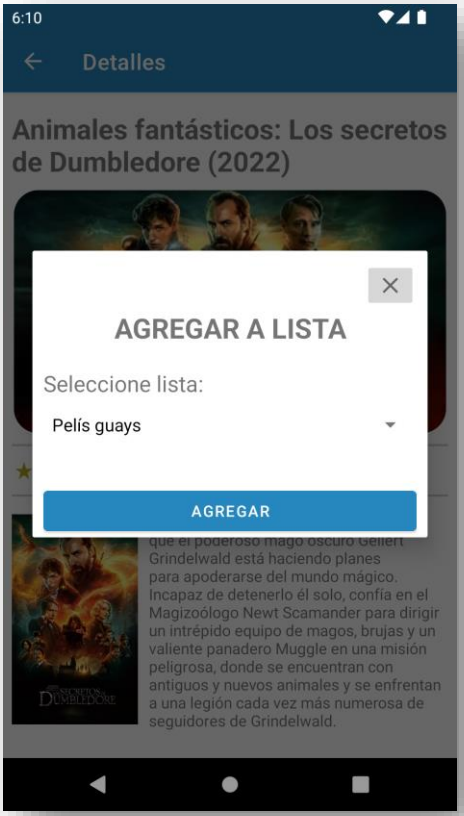

Figura 68: Agregar película/serie a lista existente

9.1.6. Buscador de películas/series

En la Figura 69 se representa el buscador de películas/series (UC-01 Buscar películas/series). Para mostrar un ejemplo de búsqueda se ha decidido utilizar el buscador de películas y buscar una película en concreto.

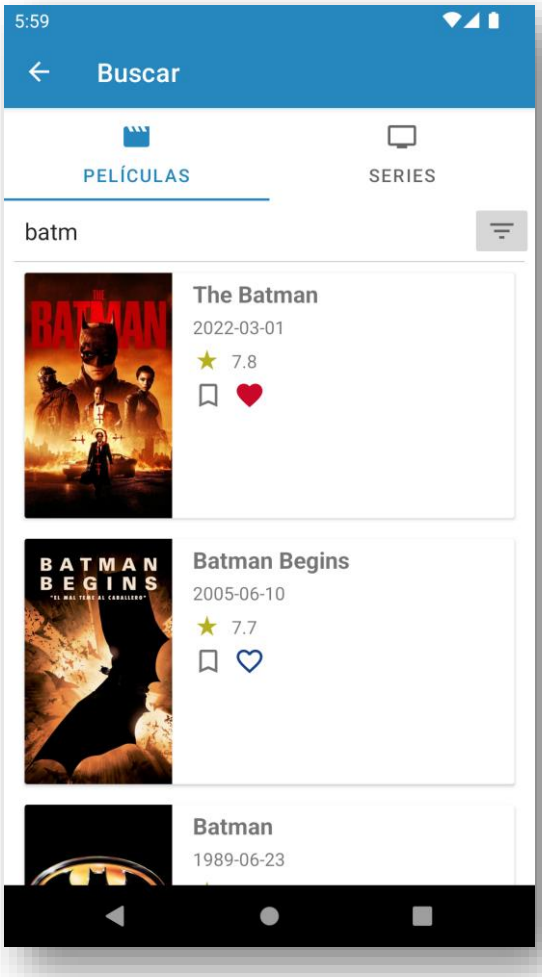

Figura 69: Buscador de películas/series

9.1.7. Filtro de películas/series

Se muestra en la Figura 71 la representación del caso de uso UC-07 Filtrar películas/series (género, popularidad, …). Finalmente, se ha decidido filtrar por dos géneros, populares y relevantes. Se pueden apreciar ambos en las capturas de abajo.

Cuando se cambia de “Géneros populares” a “Géneros relevantes” se muestran debajo los géneros que corresponda, y viceversa. Los géneros relevantes son géneros con un trasfondo social que apoyan al Objetivo de Desarrollo Sostenible número 10.

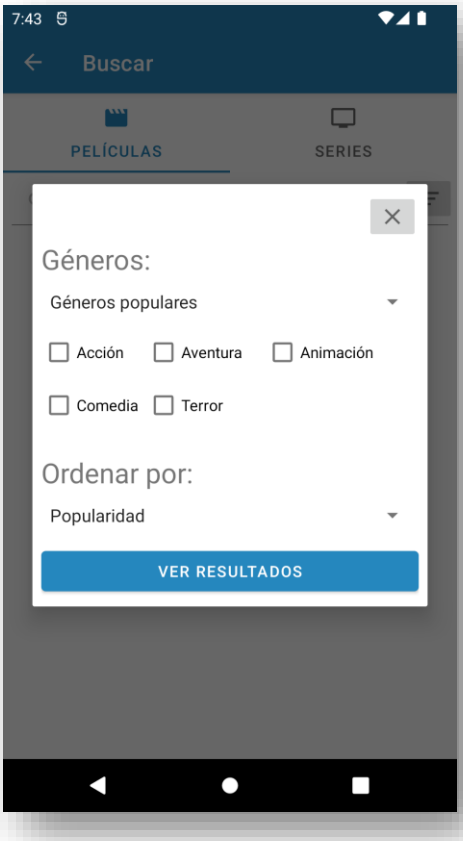

Figura 71: Filtro de películas/series (2)

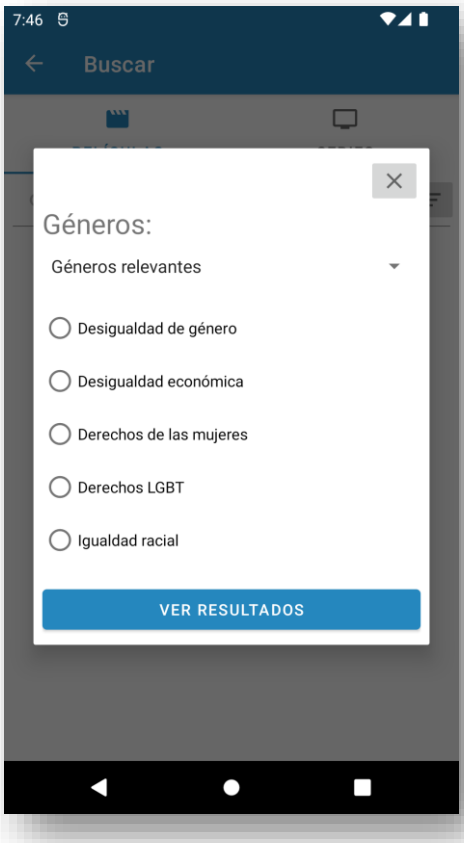

Figura 70: Filtro de películas/series (1)

9.1.8. Listado de mis listas

Se muestran en las Figura 73 y Figura 72 el listado de mis listas y la funcionalidad que permite crear una lista nueva (UC-08 Crear una lista de películas/series). Además, en la primera figura se representa la funcionalidad que permite eliminar una lista pulsando sobre el botón con forma de papelera (UC-09 Eliminar una lista de películas/series).

Finalmente, estas listas solo pueden contener películas, y no series. Esto es debido a una limitación del servidor, ya que en su versión 3 solo permite cargar películas y no series.

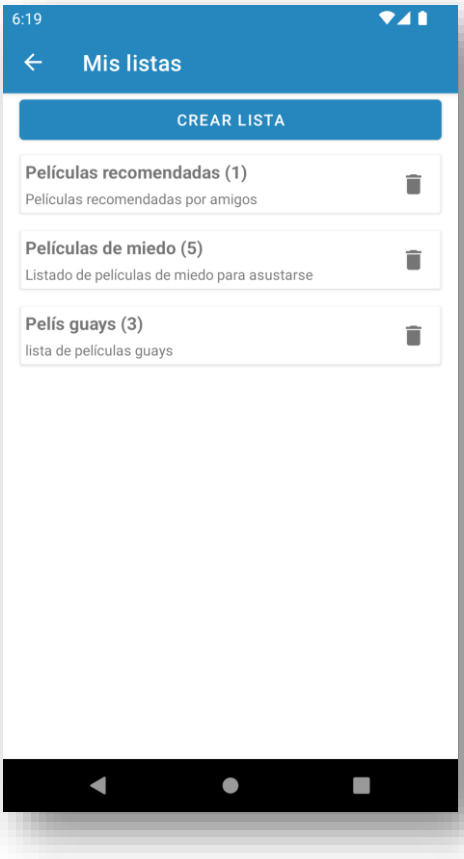

Figura 73: Listado de mis listas

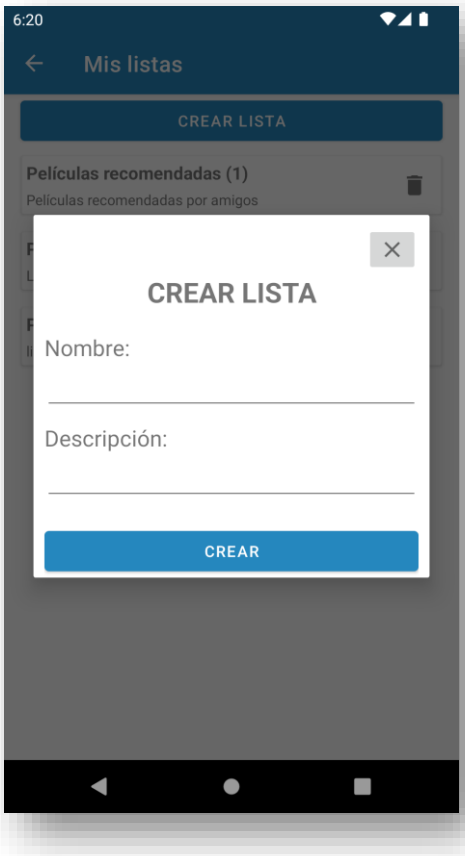

Figura 72: Crear una lista de películas y series

9.1.9. Listado de películas/series de una de mis listas

En la Figura 74 se muestra el listado de películas/series de la lista “Películas de miedo”, mostrada en una figura del punto anterior.

Se puede quitar una película de la lista pulsando sobre el icono con forma de papelera. Esta funcionalidad corresponde al caso de uso UC-14 Quitar una película/serie de una de mis listas creadas.

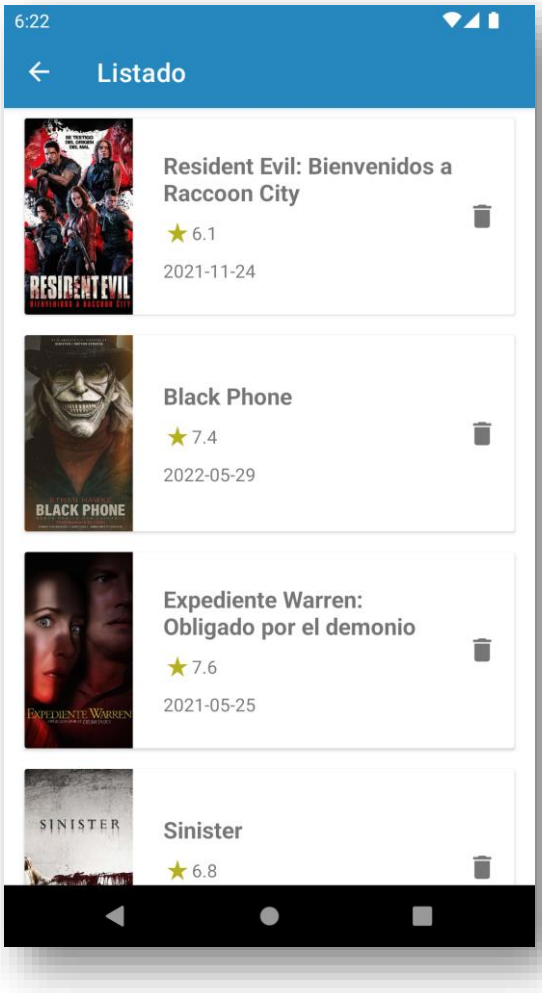

Figura 74: Listado de películas/series de una de mis listas

10. Conclusiones

En este apartado se analiza si se ha llegado a alcanzar todos los objetivos planteados en el apartado 1.2 de este documento. También se describe la relación de los estudios realizados con el proyecto realizado, y los futuros trabajos que se van a realizar para mejorar las funcionalidades de la aplicación.

Se puede decir que el trabajo desarrollado ha cumplido todos los objetivos propuestos. Se ha llegado a desarrollar una aplicación que cumple con todas las funcionalidades propuestas, y esto se puede validar mirando el resultado final de la aplicación en el apartado 9.

Al principio del proyecto surgían bastantes problemas y dificultades que impedían el avance de este. Esto era debido a no estar familiarizado con alguna de las tecnologías y por poca experiencia realizando proyectos para dispositivos móviles. Sin embargo, todas estas dificultades van desapareciendo poco a poco a medida que avanza el proyecto puesto que vas familiarizándote con las tecnologías. Un ejemplo de ello es Android Studio, entorno de desarrollo que no había utilizado antes de la realización del proyecto.

A nivel personal, este proyecto me ha ayudado a aprender más sobre el mundo del cine, un mundo que me apasiona y que consumo casi a diario. Así mismo, me ha servido para aprender a desarrollar aplicaciones móviles desde cero, ya que con anterioridad no sabía nada sobre ello. También he aprendido a compaginar el desarrollo de este proyecto con las demás obligaciones que tenía.

Y a nivel profesional, he aprendido una cosa muy importante, y es que nuestra profesión nos exige un aprendizaje constante durante toda nuestra vida. Siempre nos vamos a encontrar con barreras que nos impiden avanzar hacia una meta, y adquiriendo conocimientos logramos romper esas barreras.

10.1. Relación con los estudios cursados

Como punto de partida, hay que señalar que el desarrollo del proyecto no habría sido posible sin los conocimientos adquiridos durante la carrera universitaria.

Gran parte de las asignaturas de la rama de la ingeniería del software han sido de mucha ayuda. De las más relevantes para este proyecto han sido la asignatura AER (Análisis y especificación de requisitos) y DADM (Desarrollo de aplicaciones para dispositivos móviles). En la primera de ella se ha aprendido a analizar y especificar los requisitos del proyecto, y en la segunda se ha aprendido a realizar una aplicación móvil desde cero.

Junto a estas asignaturas también hay que mencionar las demás que tienen relación con el proyecto:

- Ingeniería del software
- Gestión de proyectos
- Interfaces persona computador
- Calidad de software
- Diseño de software
- Mantenimiento y evolución de software
- Proceso de software
- Proyecto de ingeniería de software

10.2. Futuros trabajos

La aplicación desarrollada no se trata de una aplicación definitiva. Se trata de una aplicación abierta a la que se le pueden agregar una gran variedad de funcionalidades y mejorar las existentes. Así pues, unas cuantas funcionalidades que se pueden implementar para mejorar la aplicación serían:

- **Proporcionar información de los actores**: se trata de una funcionalidad que se tenía pensada hacer desde un inicio, y no se realizó porque había otras más prioritarias por encima. Se puede considerar como una mejora en las siguiente versiones.

- **Introducir comentario**: una funcionalidad que está bastante interesante es la posibilidad de agregar comentarios a las películas/series. Al leer los comentarios se facilita al usuario la tarea de escoger las películas/series para ver.

- **Marcar/Desmarcar película/capítulo de serie como visto**: se trata de otra funcionalidad bastante interesante para el usuario, ya que le permite tener un seguimiento de las películas/capítulos que ha visto.

- **Mejorar la estética de la aplicación**: se ha decido usar un diseño sencillo e intuitivo, pero no llega a ser muy estético. Se podría cambiar el estilo de los botones, hacer un cambio en los colores, o quizás algún cambio radical en alguna de las interfaces.

- **Ver las plataformas disponibles**: esto permitirá a los usuarios de la aplicación saber en qué plataformas están disponibles las películas/series, y de este modo evitar que vayan una o por una buscándolas.

- **Notificaciones**: otra mejora a añadir son las notificaciones que permitan saber cuándo se estrena un nuevo capítulo de nuestra serie, que películas se estrenan en el día de hoy en el cine, o que recomienden películas según las que se hayan visto.

- **Multiplataforma**: la aplicación solo se puede usar en Android. Una mejora bastante innovadora sería que se pudiese utilizar en cualquier dispositivo móvil existente en el mercado.

Finalmente, todas estas posibles mejoras están muy bien ya que mejoran la funcionalidad de la aplicación. Pero, también habrá que tener en cuenta las opiniones y sugerencias de los usuarios para asegurar el progreso continuo de la aplicación. Algunos ejemplos se han podido extraer del cuestionario realizado, y son los siguientes:

- Añadir tráileres y noticas de las películas.

- Agregar más detalles a las películas/series.

11. Referencias

[1] Kanban Tool, «¿Por qué utilizar la metodología Kanban?,» [En línea]. Available: https://kanbantool.com/es/metodologia-kanban. [Último acceso: 13 Abril 2022].

[2] E. Rodríguez, «MVVM – Qué es y como funciona.,» 21 Diciembre 2020. [En línea]. Available: https://inmediatum.com/blog/ingenieria/mvvm-que-es-y-como-funciona/. [Último acceso: 30 Junio 2022].

[3] Academia Android, «Activity y Fragments,» 26 Agosto 2014. [En línea]. Available: https://academiaandroid.com/activity-y-fragments/. [Último acceso: 28 Mayo 2022].

[4] Ionos, 12 Febrero 2019. [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagramas-de-clases-con-uml/. [Último acceso: 26 Junio 2022].

[5] Oracle, «Oracle Java,» [En línea]. Available: https://www.oracle.com/es/java/. [Último acceso: 21 Mayo 2022].

[6] Lucidchart, [En línea]. Available: https://www.lucidchart.com/. [Último acceso: 10 Abril 2022].

[7] GitHub, [En línea]. Available: https://github.com/. [Último acceso: 12 Abril 2022].

[8] Postman, [En línea]. Available: https://www.postman.com/. [Último acceso: 12 Abril 2022].

[9] Adobe, [En línea]. Available: https://www.adobe.com/es/products/xd.html. [Último acceso: 12 Abril 2022].

[10] Android Developers, [En línea]. Available: https://developer.android.com/studio. [Último acceso: 12 Abril 2022].

[11] Jet Brains, «IntelliJ IDEA,» [En línea]. Available: https://www.jetbrains.com/es-es/idea/. [Último acceso: 30 Junio 2022].

[12] Square, «Retrofit,» [En línea]. Available: https://square.github.io/retrofit/. [Último acceso: 30 Junio 2022].

[13] Square, «Herramientas comerciales para los que hacen,» [En línea]. Available: https://squareup.com/es/es. [Último acceso: 30 Junio 2022].

[14] A. Leiva, «Room, la librería de Base de datos de Android,» [En línea]. Available: https://devexperto.com/room-la-libreria-de-base-de-datos-de-android/. [Último acceso: 30 Junio 2022].

[15] Academia Android, «SQLite en Android: creación y acceso base de datos e inserción de registros,» 6 Octubre 2016. [En línea]. Available:

https://academiaandroid.com/sqlite-android-creacion-acceso-base-datos-insercion/. [Último acceso: 30 Junio 2022].

[16] Android Developers, «Cómo guardar contenido en una base de datos local con Room,» [En línea]. Available: https://developer.android.com/training/data-storage/room?hl=es-419. [Último acceso: 30 Junio 2022].

[17] Glide, «Glide,» [En línea]. Available: https://bumptech.github.io/glide/. [Último acceso: 17 Mayo 2022].

[18] Android Developers, «Descripción general de la biblioteca de Paging,» [En línea]. Available: https://developer.android.com/topic/libraries/architecture/paging/v3-overview?hl=es-419. [Último acceso: 23 Junio 2022].

[19] Android Developers, «Inyección de dependencias con Hilt,» [En línea]. Available: https://developer.android.com/training/dependency-injection/hilt-android?hl=es-419. [Último acceso: 14 Junio 2022].

[20] Universitat Politècnica de València, «Adaptadores para bases de datos,» 2017. [En línea]. Available: http://www.androidcurso.com/index.php/56-mooc-introduccion/496-adaptadores-para-bases-de-datos. [Último acceso: Junio 2022].

[21] F. M. L. Jurado, «Architecture Components: Repository y DataSources,» 7 Marzo 2019. [En línea]. Available: https://betabeers.com/blog/architecture-components-repository-datasources-androidmeetskotlin-395/. [Último acceso: Junio 2022].

[22] Redux, «Redux Fundamentals, Part 2: Concepts and Data Flow,» [En línea]. Available: https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow. [Último acceso: 20 Junio 2022].

[23] U. Bob, «The Clean Architecture,» 13 Agosto 2012. [En línea]. Available: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. [Último acceso: 26 Junio 2022].

[24] Android Developers, «Calidad básica de las apps,» 10 Febrero 2021. [En línea]. Available: https://developer.android.com/docs/quality-guidelines/core-app-quality. [Último acceso: 25 Junio 2022].

[25] Android Developers, «Cómo enviar datos simples a otras apps,» [En línea]. Available: https://developer.android.com/training/sharing/send. [Último acceso: 29 Junio 2022].

[26] Android Developers, «StrictMode,» [En línea]. Available: https://developer.android.com/reference/android/os/StrictMode. [Último acceso: 29 Junio 2022].

[27] Android Developers, «One Tap for Android,» [En línea]. Available: https://developers.google.com/identity/one-tap/android. [Último acceso: 29 Junio 2022].

[28] Xataka móvil, «Qué es WebView, la app del sistema que hace que se te cierren solas algunas aplicaciones en tu móvil Android,» [En línea]. Available: https://www.xatakamovil.com/aplicaciones/que-webview-app-sistema-que-hace-que-se-te-cierren-solas-algunas-aplicaciones-tu-movil-

android#:~:text=WebView%20es%20un%20componente%20del,que%20abramos%20un%20navegador%20externo.. [Último acceso: 29 Junio 2022].

[29] Android Developers, «WebViewAssetLoader,» [En línea]. Available: https://developer.android.com/reference/androidx/webkit/WebViewAssetLoader. [Último acceso: 29 Junio 2022].

[30] Naciones Unidas, «Objetivos de Desarrollo Sostenible,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Último acceso: 30 Junio 2022].

[31] Naciones Unidas, «Objetivo 10: Reducir la desigualdad en y entre los países,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/inequality/. [Último acceso: 28 Junio 2022].

[32] Android Developers, «Descripción general de ViewModel,» [En línea]. Available: https://developer.android.com/topic/libraries/architecture/viewmodel?hl=es-419. [Último acceso: 30 Junio 2022].

Anexo 1: ODS

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**				**X**
ODS 4. **Educación de calidad.**				**X**
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.		**X**		
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**			**X**	
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Los Objetivos de Desarrollo Sostenible (ODS) [30] se tratan de 17 objetivos globales, que aplican a todas las naciones, que constituyen una visión del futuro que queremos. Todos estos objetivos se aprobaron el año 2015, y se estableció un plan para alcanzarlos en 15 años. Cada vez más se están intensificando las iniciativas a favor del desarrollo sostenible.

Estos objetivos incorporan unas metas o desafíos a los que nos enfrentamos día a día, como pueden ser la pobreza, el hambre, la desigualdad y la justicia.

En un proyecto como el desarrollado podría llegar a haber datos muy variados. Eso es debido a que se puede llegar a encontrar películas y series de distintos géneros y estilos. Por ello, este proyecto podría estar relacionado con muchos Objetivos de Desarrollo Sostenible en función del tipo de película y serie.

Sin embargo, los ODS que se pueden relacionar principalmente con este Trabajo Fin de Grado son el número 10 y el 13, conocidos por el nombre de “Reducción de las desigualdades” y “Acción por el clima”.

Por un lado, en relación con el objetivo número 10, la aplicación dispone en el filtro de búsqueda de las películas unos géneros con un trasfondo social, llamados “Géneros relevantes” en la aplicación. Actualmente se encuentran los géneros “Desigualdad de género”, “Desigualdad económica”, “Derecho de las mujeres”, “Derechos LGBT” e “Igualdad racial”. Si te interesan películas de alguno de estos temas se puede filtrar por alguno de ellos y ver que películas tratan esos temas. Gracias a eso se está ayudando a dar más visibilidad a estos temas.

Con esta funcionalidad agregada se está fomentando y dando visibilidad a temas muy importantes con el fin de disfrutar de un mundo más equitativo frente a las diferencias existentes. Concretamente está relacionado con la meta 10.2 del ODS 10, la cual se puede observar a continuación:

> ***10.2*** *De aquí a 2030, potenciar y promover la inclusión social, económica y política de todas las personas, independientemente de su edad, sexo, discapacidad, raza, etnia, origen, religión o situación económica u otra condición. [31]*

Por otro lado, el proyecto está relacionado un poco con el ODS 13 debido a que las aplicaciones en dispositivos móviles consumen mucho menos que los ordenadores. De esta manera se están incorporando medidas, aunque sea en bajo grado, para reducir el consumo y minimizar el impacto en el medio ambiente.

Finalmente, el trabajo realizado tiene como meta apoyar y proteger el planeta, haciendo especialmente hincapié en el clima y en las desigualdades existentes en el mismo.

Anexo 2: Cuestionario usabilidad

En este anexo se muestra el formulario realizado para realizar las pruebas de usabilidad. Se ha realizado a través de Google Forms.

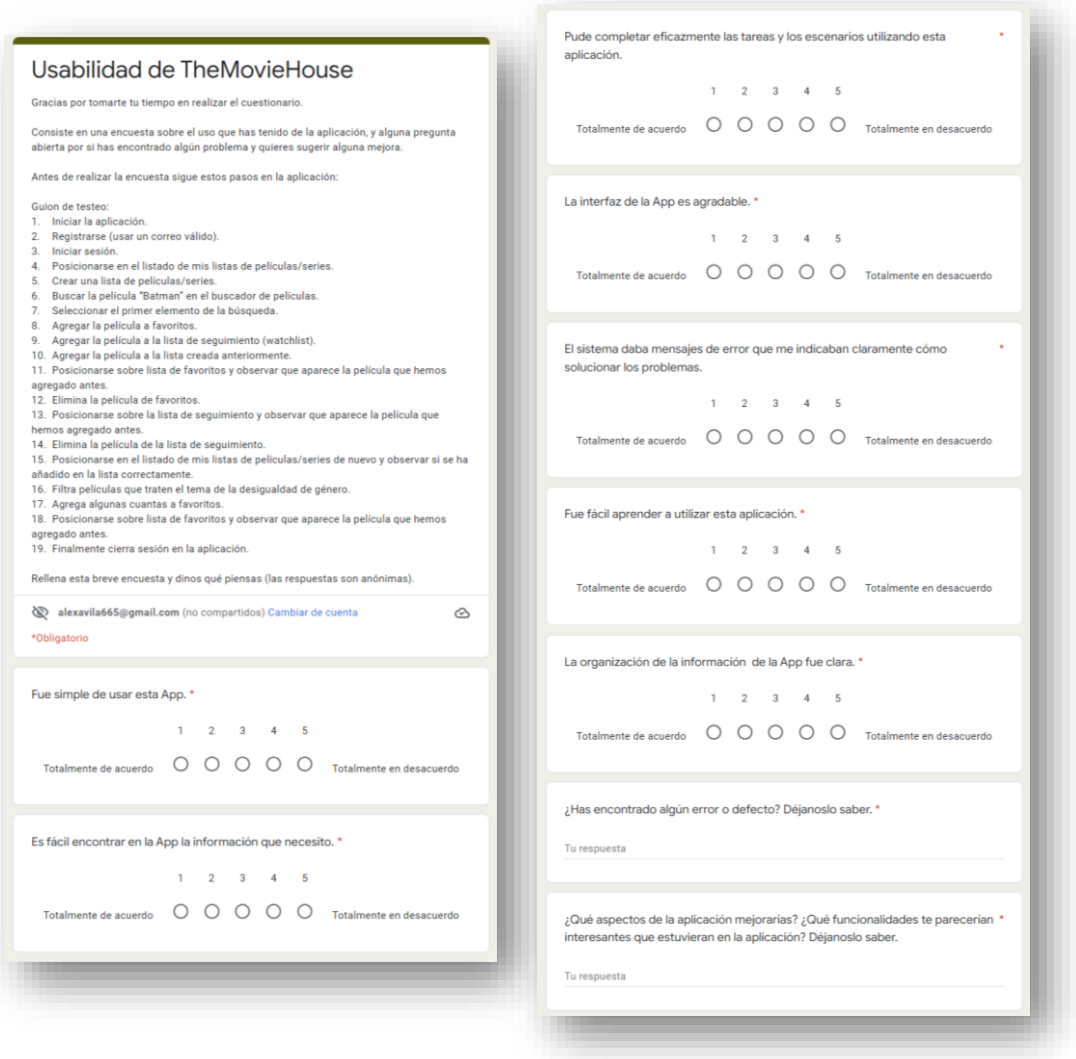

Figura 75: Cuestionario usabilidad