



WATCHVIEW

Título proyecto: WATCHVIEW

Autor: Patricia Aguayo Escudero

Tutor: Javier Martín Rivero

Fecha de entrega: 21/05/2025

Convocatoria: 2024 2025

ÍNDICE

1. Introducción.....	1
2. Motivación.....	2
3. Abstract.....	2
4. Objetivos.....	3
5. Metodología.....	5
6. Tecnologías y herramientas.....	11
7. Estimación de recursos y planificación.....	12
8. Análisis.....	13
9. Diseño.....	37
10. Despliegue y pruebas.....	41
11. Conclusiones.....	48
12. Vías futuras.....	49
13. Glosario.....	51
14. Bibliografía.....	52
15. Anexos.....	53

1. Introducción

El principal motivo que impulsa la ejecución de esta aplicación es la necesidad de proporcionar a los usuarios de plataformas de streaming un espacio informativo eficiente y accesible sobre los próximos estrenos de contenido audiovisual. En la actualidad, se ha identificado que un gran número de consumidores no dispone del tiempo suficiente para llevar a cabo una búsqueda exhaustiva y precisa de los nuevos lanzamientos en dichas plataformas. Esta limitación temporal dificulta que los usuarios puedan mantenerse al día con las novedades del sector audiovisual, lo que puede repercutir negativamente en su experiencia de consumo.

Asimismo, se ha observado que la falta de campañas publicitarias efectivas y de estrategias de marketing en torno al estreno de determinados títulos ha provocado que muchas producciones pasen desapercibidas para el público. Esta invisibilidad ha resultado, en numerosas ocasiones, en índices de audiencia considerablemente bajos, lo que a su vez ha conducido a la cancelación prematura de series o películas, incluso cuando estas poseían una destacable calidad en términos de guion, producción o interpretación actoral.

En este contexto, se considera pertinente el desarrollo de una aplicación móvil que centralice y facilite el acceso a la información sobre los nuevos estrenos audiovisuales en diversas plataformas de streaming. Dicha aplicación busca proporcionar a los usuarios una herramienta intuitiva que les permita conocer de forma rápida y eficaz los próximos lanzamientos, sin que ello requiera una inversión significativa de tiempo en búsquedas independientes. Este planteamiento responde a la creciente necesidad de optimizar el consumo de contenido en un entorno digital cada vez más saturado de opciones.

Además de cumplir con este objetivo principal, la aplicación también incluirá funcionalidades adicionales que enriquecerán la experiencia del usuario. Entre estas se encuentra la integración de un apartado que mostrará el Top 10 de España en una o varias plataformas de streaming, permitiendo a los usuarios identificar fácilmente aquellos títulos que están siendo tendencia en el país. Asimismo, se incorporará una herramienta que funcionará como una wishlist, donde el usuario podrá guardar aquellas producciones que deseé ver en el futuro, facilitando así la organización de su consumo de contenido audiovisual.

2. Motivación

La concepción de este proyecto surge de un interés particular por el ámbito audiovisual y la necesidad de disponer de una herramienta que centralice de manera eficiente la información sobre los estrenos en plataformas de streaming. En un contexto en el que la oferta de contenido es cada vez más amplia y dispersa, resulta complejo para los usuarios mantenerse actualizados sobre las novedades del sector sin invertir una cantidad significativa de tiempo en la búsqueda de información en distintas fuentes.

Desde una perspectiva personal, la identificación de esta problemática ha sido el punto de partida para el desarrollo de esta aplicación. A lo largo del tiempo, se ha evidenciado que la dispersión informativa no solo afecta a nivel individual, sino que también impacta a un amplio sector de consumidores que, por diversas razones, no pueden realizar un seguimiento constante de los nuevos lanzamientos. Además, la falta de estrategias de promoción efectivas en determinadas producciones ha contribuido a que algunas de ellas no alcancen el reconocimiento que podrían merecer, lo que ha derivado en bajos índices de audiencia e incluso en la cancelación de títulos con gran potencial.

En este sentido, la motivación principal de este proyecto radica en el diseño de una solución que no solo optimice el acceso a la información sobre nuevos estrenos, sino que también favorezca la difusión de contenidos que, por diversos factores, pueden quedar relegados en el mercado audiovisual. La materialización de esta aplicación representa la oportunidad de contribuir a la mejora de la experiencia del usuario, facilitando la organización y planificación de su consumo de contenido. Asimismo, este proyecto permite abordar un desafío presente en la industria del entretenimiento digital, combinando la resolución de una necesidad detectada con el interés por ofrecer una herramienta funcional y accesible para una amplia comunidad de usuarios.

3. Abstract

This project arises from an interest in the audiovisual field and the need to develop a tool that centralizes information about new releases on streaming platforms. In a context where the content offering is vast and fragmented, users struggle to stay updated without

investing significant time in searching across various sources. The proposed application addresses this issue by offering an efficient, user-friendly, and accessible solution to help organize and plan content consumption. Beyond enhancing the individual user experience, the project also aims to promote visibility for lesser-known productions that, despite their quality, do not receive the attention they deserve due to limited marketing. This initiative not only responds to a real need in digital content consumption but also tackles a broader challenge in the entertainment industry: content oversaturation and unequal visibility. The application represents an opportunity to connect users with relevant releases, optimize their time, and support the promotion of promising titles within the audiovisual market.

4. Objetivos

Para abordar el desarrollo y análisis del proyecto se establecen una serie de objetivos generales (O.G.), que a su vez se desglosan en otros objetivos específicos (O.E.). Estos objetivos son los siguientes:

O.G.1. Explicar y diseñar la aplicación con un previo análisis del proyecto y desarrollo de la documentación esencial

El diseño de la aplicación requiere un análisis previo que exponga la problemática y posible solución sobre las tendencias de producciones audiovisuales en plataformas de streaming. Se establecerán los antecedentes y justificaciones del proyecto, definiendo su metodología y las tecnologías empleadas. Además, se elaborará la documentación técnica con los requerimientos y funcionalidades esenciales para su correcto desarrollo.

O.G.2. Crear y desarrollar una aplicación móvil capaz de mostrar y listar los próximos estrenos de una o varias plataformas de *streaming*

La aplicación debe centralizar la información sobre los estrenos en una o diversas plataformas, permitiendo la consulta de fechas de lanzamiento y tendencias de contenido. Se incorporará una función para la gestión de listas personalizadas y un motor de búsqueda avanzada con filtros específicos. También se integrará información detallada de cada título para mejorar la experiencia del usuario.

O.G.3. Analizar y evaluar el desarrollo del aplicativo, así como explorar el resultado de este y sus vías futuras

El rendimiento de la aplicación se evaluará mediante pruebas funcionales basadas en casos de uso, optimizando su funcionamiento y detectando posibles mejoras. Se analizará su viabilidad y su impacto en el mercado, proponiendo futuras líneas de desarrollo para ampliar sus funcionalidades y consolidar su utilidad en el ámbito del entretenimiento digital.

En la Tabla 4.1., que se muestra a continuación, se sintetizan los objetivos del proyecto, detallando también los apartados, que pueden ayudar a comprender su estructura.

Tabla 4.1. Objetivos del Trabajo Fin Ciclo Formativo del Grado Superior DAM

OBJETIVOS GENERALES	OBJETIVOS ESPECÍFICOS	EPÍGRAFES
O.G.1. Explicar y diseñar la aplicación con un previo análisis del proyecto y desarrollo de la documentación esencial.	O.E.1.1. Analizar y determinar la manera de consumir producciones audiovisuales de plataformas de streaming en base al desarrollo y avance tecnológico.	Justificación
	O.E.1.2. Indicar los antecedentes para explicar la elección de proyecto y temática.	Motivación
	O.E.1.3. Detallar la metodología y tecnologías utilizadas a lo largo del proyecto.	Metodología y Tecnologías y herramientas utilizadas
	O.E.1.4. Establecer las necesidades y finalidad del aplicativo a través de la documentación básica necesaria.	Análisis
O.G.2. Crear y desarrollar una aplicación móvil capaz de mostrar y listar los próximos estrenos de una o	O.E.2.1. Registrar las fechas de estreno de una o varias plataformas y su/sus top 10 en España.	Inicio

varias plataformas de streaming.	O.E.2.2. Establecer una lista personalizada con los estrenos y productos audiovisuales ya disponibles que se quieran ver.	Mi lista
	O.E.2.3. Detallar búsqueda de títulos por nombre u otros parámetros como el género.	Listado completo
	O.E.2.4. Detallar información básica de cada uno de los títulos.	Info series e Info pelis
O.G.3. Analizar y evaluar el desarrollo del aplicativo, así como exponer el resultado de este y sus vías futuras.	O.E.3.1. Determinar y registrar pruebas funcionales del aplicativo, basados en los casos de uso.	Despliegue y pruebas
	O.E.3.2. Advertir de la viabilidad del proyecto, así como de las vías de futuro.	Conclusiones y Vías futuras

Fuente: Elaboración propia (2025)

5. Metodología

Con el objetivo de evidenciar la aplicación práctica de los conocimientos adquiridos a lo largo del Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Multiplataforma (DAM), se ha elaborado la siguiente tabla, que recoge y relaciona los principales conceptos trabajados durante el desarrollo del proyecto con las distintas asignaturas del ciclo.

Tabla 5.1. Conceptos trabajados y su asociación con las asignaturas del Ciclo Formativo de Grado Superior DAM

Asignatura	Usos en el proyecto
Sistemas informáticos	<ul style="list-style-type: none">- Uso eficiente de los recursos del sistema para garantizar un buen rendimiento de la aplicación, evitando la sobrecarga del dispositivo y mejorando la velocidad de ejecución.
Bases de Datos	<ul style="list-style-type: none">- Implementación de una base de datos local mediante SQLite para el almacenamiento de información.- Diseño del modelo entidad-relación y normalización de datos, facilitando las operaciones de inserción, búsqueda, actualización y eliminación.
Programación	<ul style="list-style-type: none">- Desarrollo de la lógica de la aplicación utilizando funciones, métodos, variables y conexiones con la base de datos.- Creación del diagrama de clases, representando la estructura y relación entre clases, métodos y atributos.
Lenguajes de marcas y sistemas de gestión de información	<ul style="list-style-type: none">- Uso de archivos XML para el diseño visual y estructuración de la interfaz, así como para representar información de forma clara y organizada.
Entornos de desarrollo	<ul style="list-style-type: none">- Utilización de la herramienta Figma para el diseño del prototipo (Mockup) de la aplicación.

	<ul style="list-style-type: none">- Aplicación de metodologías ágiles como Scrum y Kanban, así como la técnica Planning Poker para estimar tiempos de desarrollo.- Identificación de requerimientos funcionales y no funcionales, elaboración del diagrama de casos de uso y sus respectivas plantillas de especificación.
Formación y Orientación Laboral	<ul style="list-style-type: none">- Aplicación de buenas prácticas en el entorno de trabajo: postura adecuada, ergonomía, uso responsable de los equipos, lo que favorece un desarrollo saludable y productivo del proyecto.
Acceso a datos	<ul style="list-style-type: none">- Lectura e integración de archivos JSON provenientes de la API Streaming Availability, utilizados para poblar la base de datos local con contenido multimedia.
Desarrollo de interfaces	<ul style="list-style-type: none">- Diseño de una interfaz de usuario intuitiva y atractiva.- Realización de pruebas funcionales con JUnit 5 y validaciones mediante pruebas de valores límite.
Programación de servicios y procesos	<ul style="list-style-type: none">- Gestión de hilos y colas en la actividad de inicio de sesión (LoginActivity) para la inserción eficiente de datos.- Implementación de procesos para el tratamiento de datos en operaciones de inserción y actualización.
Programación multimedia	<ul style="list-style-type: none">- Desarrollo de la aplicación móvil utilizando el

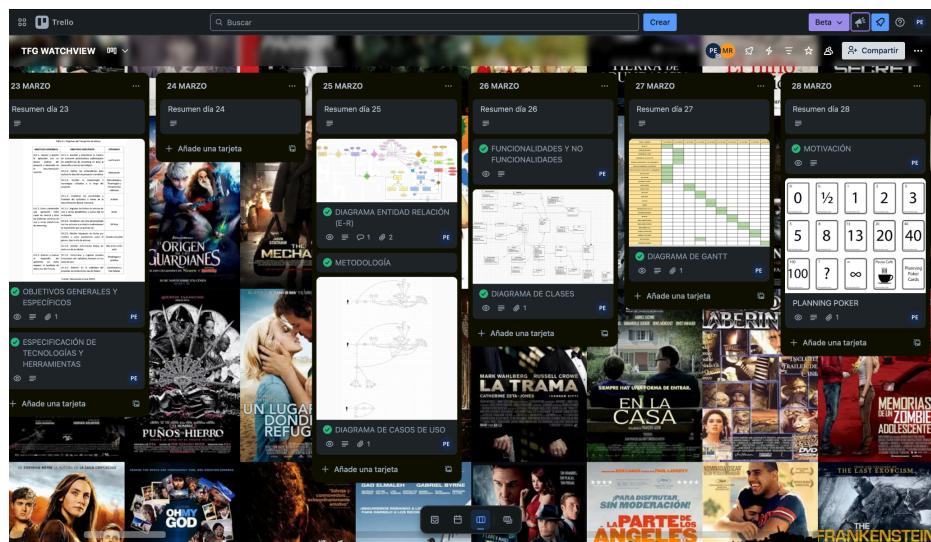
y dispositivos móviles	lenguaje Kotlin y el entorno de desarrollo Android Studio.
Sistemas de gestión empresarial	<ul style="list-style-type: none">- Creación de módulos administrativos que permiten gestionar aspectos como usuarios y fotografías de perfil, otorgando privilegios diferenciados.
Empresa e Iniciativa Emprendedora	<ul style="list-style-type: none">- Análisis del entorno y de las necesidades del consumidor para orientar el desarrollo de un producto innovador.- Consideración de los derechos de autor mediante el uso de una API gratuita con limitaciones controladas, evaluando futuras ampliaciones según el crecimiento del proyecto.
Horas de Libre Configuración	<ul style="list-style-type: none">- Interpretación y uso de documentación técnica en inglés relacionada con la API Streaming Availability, desarrollando competencias en el manejo de herramientas externas.

Fuente: Elaboración propia (2025)

Esta tabla tiene como propósito principal resaltar que los contenidos abordados en el Ciclo Formativo han sido puestos en práctica a lo largo de todo el desarrollo del proyecto. Cada asignatura ha contribuido de manera significativa al avance del trabajo, proporcionando las herramientas, habilidades y conocimientos necesarios para superar los distintos retos que han surgido en las etapas de diseño, programación, gestión, documentación y validación del producto. Gracias a esta integración, se refuerza el vínculo entre el aprendizaje teórico y su aplicación práctica, permitiendo al alumnado adquirir una visión global, profesional y transversal del proceso de desarrollo de software.

Con el fin de garantizar una gestión eficiente del proyecto y alcanzar los objetivos propuestos, se ha optado por la combinación de dos metodologías ágiles ampliamente utilizadas en el ámbito profesional: Scrum y Kanban. Su implementación se ha llevado a cabo mediante la plataforma Trello, una herramienta digital que permite la organización visual del trabajo a través de tableros, columnas y tarjetas adaptadas a las distintas fases del proyecto.

Imagen 5.1. Desarrollo Trello¹



Fuente: Elaboración propia (2025)

Scrum ha sido la metodología principal empleada para estructurar el trabajo mediante sprints, es decir, ciclos de desarrollo con un tiempo definido durante el cual se completan tareas específicas. Previo a cada entrega o revisión, se realizaron reuniones de seguimiento con el tutor del proyecto para evaluar avances, detectar posibles desviaciones y aplicar los ajustes necesarios. Uno de los elementos clave en la aplicación de Scrum fue el uso de la técnica Planning Poker, utilizada para estimar de manera colaborativa el esfuerzo y tiempo requerido para cada tarea, facilitando así una planificación más realista y participativa. La metodología Scrum también promueve la mejora continua, la adaptabilidad ante cambios y la retroalimentación constante, lo cual ha sido esencial para optimizar el proceso de desarrollo.

¹ Ver en mayor tamaño en Anexos

Por otro lado, Kanban ha sido integrado de forma complementaria, aportando una visión clara y en tiempo real del estado de cada tarea. A través de Trello, las actividades se clasificaron en columnas según su estado (pendientes, en proceso, finalizadas), permitiendo una gestión visual del flujo de trabajo. Este enfoque ha facilitado la identificación de cuellos de botella, la redistribución eficiente de tareas y una mayor transparencia en la ejecución del proyecto. La posibilidad de asignar responsables, establecer fechas de entrega y registrar tiempos estimados en cada tarjeta ha contribuido significativamente a la organización general.

La elección de combinar ambas metodologías responde a la necesidad de abordar el proyecto desde una perspectiva integral, uniendo la estructura y planificación estratégica de Scrum con la flexibilidad y seguimiento continuo de Kanban. Esta combinación ha permitido:

- Optimizar la organización del trabajo mediante la división en sprints y la representación visual del progreso.
- Facilitar la adaptabilidad ante imprevistos, gracias a la naturaleza flexible de Kanban.
- Mejorar la gestión del tiempo y la asignación de recursos, utilizando herramientas como Planning Poker y Trello.
- Fomentar la colaboración y la comunicación efectiva, con supervisión constante por parte del tutor y ajustes dinámicos según el avance del proyecto.

En conclusión, la aplicación conjunta de Scrum y Kanban ha permitido mantener un equilibrio entre planificación rigurosa y flexibilidad operativa, asegurando una mayor eficiencia, adaptabilidad y transparencia en el desarrollo del proyecto. Esta estrategia metodológica ha sido clave para cumplir de forma exitosa los objetivos establecidos y consolidar las competencias adquiridas durante el ciclo formativo.

6. Tecnologías y herramientas

El desarrollo de la aplicación se llevará a cabo en Android Studio, un entorno de desarrollo integrado (IDE) ampliamente utilizado para la creación de aplicaciones móviles en el ecosistema Android. El lenguaje de programación elegido es Kotlin, debido a su eficiencia, compatibilidad nativa con Android y capacidad para mejorar la seguridad y optimización del código. En cuanto a la gestión de datos, se implementará SQLite, una base de datos local

que permitirá almacenar, estructurar y gestionar la información recopilada sobre las distintas plataformas de streaming. Para la integración de estos datos, se empleará la API Streaming Availability, que proporcionará acceso en tiempo real a los estrenos y contenidos disponibles en diversos servicios de streaming, evitando la necesidad de ingresar información de manera manual.

El diseño de la interfaz de usuario y la experiencia de usuario (UI/UX) se trabajará en Figma, herramienta que facilitará la creación de prototipos interactivos y maquetas visuales antes de la implementación final. Esto permitirá definir de manera precisa la estructura y funcionalidad de la aplicación, asegurando una navegación intuitiva. Para la documentación gráfica del proyecto, se utilizará Canva, plataforma versátil que permitirá la elaboración de diagramas clave como el Diagrama de Entidad-Relación y el Diagrama de Casos de Uso, además de servir como apoyo en la presentación visual del proyecto. Asimismo, se recurirá a Draw.io para la creación del Diagrama de Clases, proporcionando una representación clara de la arquitectura del software y las relaciones entre sus componentes.

En términos de gestión y planificación del desarrollo, se aplicarán metodologías ágiles como Kanban y Scrum, con el fin de optimizar la organización del trabajo y mejorar la eficiencia del equipo. Para ello, se utilizarán herramientas como Trello, Google Sheets y Google Docs, que permitirán gestionar tareas, asignar responsabilidades y realizar un seguimiento del progreso del proyecto en tiempo real. Google Docs facilitará la colaboración en la documentación del proyecto, permitiendo la edición simultánea y el acceso compartido a información clave. Además, se utilizará el diagrama de Gantt como una herramienta fundamental para visualizar la planificación del proyecto, asegurando una mejor distribución del tiempo y un flujo de trabajo más dinámico.

El aspecto visual del aplicativo se reforzará con elementos gráficos obtenidos de Flaticon.es, recurso esencial para incorporar iconografía atractiva y coherente con el diseño de la interfaz. Por otro lado, el control de versiones se gestionará mediante Git y GitHub, lo que garantizará un desarrollo colaborativo seguro y permitirá un historial detallado de cambios a lo largo del proyecto. Finalmente, Google Drive servirá como plataforma de almacenamiento y respaldo de archivos, asegurando la sincronización de la documentación y el acceso remoto a los recursos necesarios para el desarrollo del aplicativo.

7. Estimación de recursos y planificación

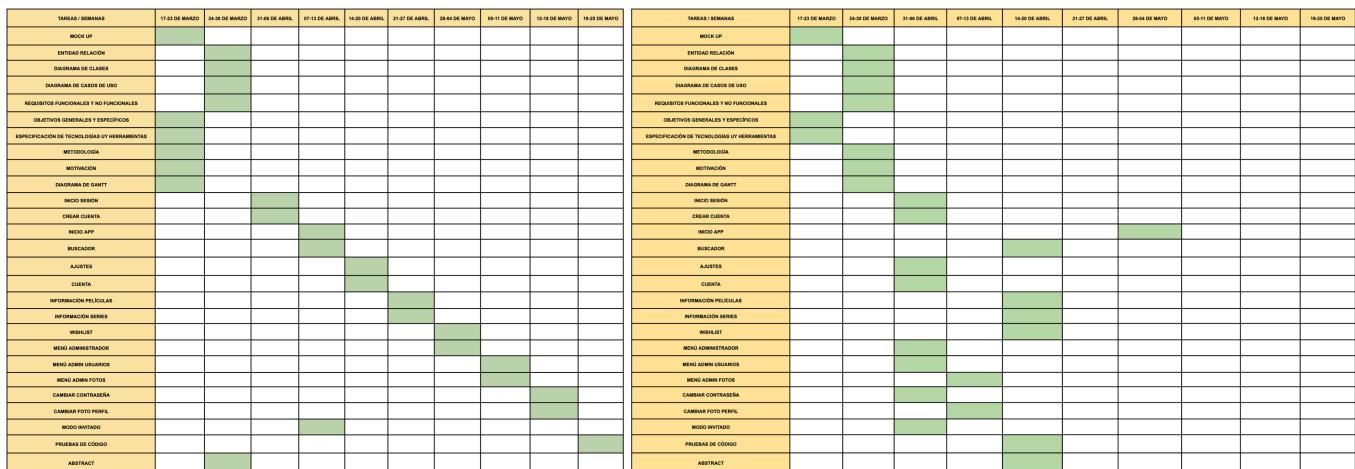
Para la planificación del proyecto se utilizó la herramienta del diagrama de Gantt, la cual permite organizar y visualizar de forma estructurada todas las tareas necesarias, así como el tiempo estimado para su ejecución. Dado que el proyecto cuenta con una duración limitada a unos pocos meses, se optó por una planificación semanal con el fin de tener un mayor control sobre el avance y facilitar el seguimiento del desarrollo.

La imagen de la izquierda corresponde al cronograma previsto al inicio del proyecto, mientras que la imagen de la derecha representa la ejecución real. A simple vista se observa que la planificación inicial distribuye las tareas de forma más espaciada a lo largo del tiempo, mientras que en la ejecución real muchas de ellas se concentraron en las primeras semanas.

Esta diferencia se debe a una planificación inicial prudente, cuyo objetivo era evitar la sobrecarga de trabajo y permitir un margen suficiente para cada tarea. Sin embargo, durante el desarrollo, varias de estas tareas se completaron en menos tiempo del estimado, ya sea por su menor complejidad o por los requerimientos de entregas parciales al tutor académico, como fue el caso de los distintos apartados del informe, incluyendo diagramas y análisis técnicos.

En resumen, la flexibilidad en la ejecución ha permitido una mayor eficiencia en la gestión del tiempo, ajustándose a las necesidades reales del proyecto.

Imagen 7.1. Comparativa entre el diagrama de Gantt planificado y el ejecutado²



² Ver en mayor tamaño en Anexos

Fuente: Elaboración propia (2025)

8. Análisis

Este apartado constituye una etapa fundamental dentro del ciclo de desarrollo de software, ya que permite establecer de manera precisa qué se espera del sistema, cómo debe comportarse y cuáles son sus principales componentes estructurales y funcionales. A través de este apartado se busca definir y documentar, con rigor técnico, los distintos elementos que orientarán tanto el diseño como la implementación de la solución propuesta.

Con el fin de garantizar una visión integral del sistema, se ha llevado a cabo un estudio detallado que abarca desde los requisitos del usuario hasta la representación estructurada de la base de datos y la arquitectura orientada a objetos. Cada uno de los siguientes subapartados incluye tanto la descripción teórica como el desarrollo práctico correspondiente, apoyado por listas de funcionalidades, tablas de requisitos, diagramas técnicos y fichas explicativas.

Los requisitos definen qué debe hacer el sistema y bajo qué condiciones debe operar. Para esta aplicación se han identificado los requisitos funcionales, estos describen las acciones concretas que el sistema debe permitir y los requisitos no funcionales, que establecen características de calidad del sistema.

REQUISITOS FUNCIONALES

-Creación cuenta de usuario: El sistema debe permitir que nuevos usuarios se registren proporcionando información básica como correo electrónico y contraseña. El registro podría implicar validación de correo mediante un código o enlace de verificación.

-Ver inicio: es la pantalla principal de la aplicación donde se pueden ver los top 10 de España y los próximos estrenos.

-Ir a inicio: el usuario podrá volver a inicio, ya sea que esté en Mi lista, Ajustes o el Buscador.

-Modo invitado: permite al invitado poder acceder a la aplicación sin registrarse pero sin poder ver todo el contenido.

-Introducir credenciales: Los usuarios deben poder iniciar sesión con sus credenciales (correo electrónico y contraseña). El sistema validará estas credenciales comparándolas con los datos almacenados en la base de datos.

-Cambiar contraseña: El sistema proporcionará un mecanismo para que los usuarios puedan cambiar su contraseña.

-Ver perfil: permite al usuario ver su perfil, poder ver las opciones para configurarlo y cerrar sesión.

-Cierre sesión: permite a los usuarios cerrar sesión de manera segura.

-Despliegue de opciones: menús desplegables que muestren distintas opciones en la interfaz de usuario.

-Botones para clicar las series y películas: Los botones permiten a los usuarios interactuar con los productos.

-Guardar título: una vez seleccionado se podrá añadir la película o la serie a la lista del usuario.

-Quitar título: una vez seleccionado se podrá eliminar la película o la serie de la lista del usuario.

-Añadir notificación: en el inicio cuando se muestran los próximos estrenos permite al usuario guardar esa serie o película a su lista y también en el momento que se estrene, enviar una notificación donde se avise del estreno de este título.

-Eliminar notificación: en el inicio cuando se muestran los próximos estrenos permite al usuario quitar esa serie o película de su lista y también se eliminará la notificación de su estreno.

-Lista de películas y series: el usuario podrá ver todas las películas y series disponibles en la aplicación.

-Filtrar títulos: permite al usuario buscar series o películas ya sea por género.

-**Buscar título:** permite buscar una película o serie por el nombre.

-**Ver título:** esto permite al usuario ver la información de una película o serie en específico.

-**Ver Iniciar Sesión:** Es la primera pantalla que visualizan los usuarios para entrar al aplicativo.

-**Ver ajustes:** Sección donde los usuarios y administrador pueden ver las opciones de ajustes y ver su perfil.

-**Modificar foto perfil:** permite al usuario poder cambiar la foto de perfil que tiene en la cuenta.

-**Mi lista:** permite a los usuarios ver los títulos guardados, la opción de quitarlos y ver su información.

-**Volver al menú administrador:** permite al administrador volver al menú desde el inicio, mi lista, ajustes y el buscador.

-**Ver menú administrador:** permite ver al administrador las opciones que tiene en la aplicación.

-**Ver menú fotos:** permite ver al administrador las opciones que tiene para editar las fotos de perfil.

-**Eliminar foto:** permite al administrador eliminar una foto de la aplicación.

-**Ver menú usuarios:** permite ver al administrador las opciones que tiene para editar los usuarios.

-**Añadir usuario:** permite al administrador añadir un usuario a la aplicación.

-**Eliminar usuario:** permite al administrador eliminar un usuario de la aplicación.

-**Modificar usuario:** permite al administrador modificar un usuario de la aplicación.

REQUISITOS NO FUNCIONALES

- Disponibilidad y confiabilidad:** La app debe estar disponible para los usuarios en todo momento y funcionar de manera estable sin fallos.
- Seguridad y privacidad:** Los datos personales y de pago de los usuarios deben estar protegidos para evitar filtraciones o accesos no autorizados.
- Rendimiento eficiente:** La app debe ser rápida y responder de forma ágil, incluso cuando hay muchos usuarios conectados.
- Actualizaciones continuas:** Debe ser posible hacer mejoras y corregir errores en la app sin interrumpir el servicio a los usuarios.
- Interfaz de usuario intuitiva:** La app debe tener un diseño fácil de entender, con menús y botones claros para que los usuarios no se confundan.
- Control de acceso a funciones:** Algunas funciones deben estar restringidas a ciertos usuarios (por ejemplo, solo administradores pueden acceder a configuraciones especiales).
- Entrada de texto:** Los usuarios deben poder introducir datos como nombres o títulos fácilmente y sin errores.
- Actualización diaria:** Proporcionar actualizaciones diarias de los próximos estrenos y los top 10 de España.

DIAGRAMA DE CASOS DE USO

A continuación, se presenta el diagrama de casos de uso del sistema, el cual permite identificar y representar gráficamente las principales funcionalidades desde la perspectiva de los diferentes actores que interactúan con la aplicación. En este caso, se distinguen tres perfiles clave: usuario invitado, usuario registrado y administrador, cada uno con distintos niveles de acceso y funcionalidades asociadas.

El diagrama tiene como objetivo mostrar de forma clara cómo cada actor interactúa con el sistema, resaltando las acciones que puede ejecutar y los módulos del sistema a los

que tiene acceso. Esta representación resulta especialmente útil para comprender el comportamiento esperado del sistema desde el punto de vista del usuario y sirve como base para las siguientes etapas de desarrollo, como la definición de requisitos, diseño técnico y pruebas.

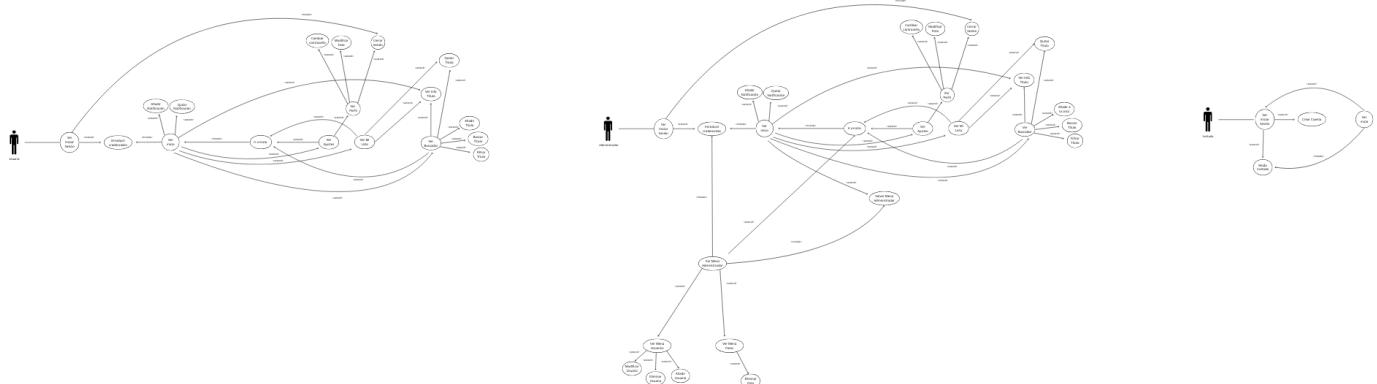
Perfiles y funcionalidades:

- **Usuario invitado:** Tiene acceso limitado, principalmente orientado a la visualización de contenidos sin necesidad de autenticación. Sus funcionalidades se centran en el acceso de la pestaña de inicio.
- **Usuario registrado:** Accede a funcionalidades personalizadas, como la creación y gestión de su cuenta, modificación de preferencias y gestión de listas personales (como "wishlist").
- **Administrador:** Tiene privilegios ampliados que le permiten gestionar tanto a los usuarios registrados como las fotos de perfil y la estructura del sistema, a través de un panel administrativo. Además de las funcionalidades descritas para el perfil de usuario registrado.

Este diagrama también facilita la identificación de posibles mejoras, extensiones o puntos críticos en la interacción usuario-sistema. Además, ayuda a establecer los límites del sistema, delimitando qué procesos son responsabilidad de cada actor y cuáles son automatizados o internos.

Imagen 8.1. Diagrama de Casos de Usos por perfil³

³ Ver en mayor tamaño en Anexos



Fuente: Elaboración propia (2025)

PLANTILLAS DE ESPECIFICACIONES

A continuación del diagrama de casos de uso, se incluyen las plantillas de especificación correspondientes, las cuales permiten describir con mayor profundidad cada una de las funcionalidades representadas gráficamente. Estas plantillas no solo detallan el objetivo de cada caso de uso, sino también los pasos implicados, los actores involucrados y las posibles excepciones que puedan surgir durante su ejecución.

Cada especificación se estructura de forma sistemática para facilitar su comprensión y análisis. Entre los elementos incluidos destacan: el nombre del caso de uso, una breve descripción, el flujo principal de eventos, condiciones de entrada y salida, y relaciones con otros casos de uso. Esta última sección es especialmente relevante, ya que permite visualizar cómo se conectan o dependen entre sí las distintas funcionalidades del sistema.

Este nivel de detalle resulta fundamental para el diseño técnico posterior, ya que ofrece una visión clara y completa del comportamiento esperado del sistema desde el punto de vista funcional.

Tabla 8.1. Tabla de especificación Ver Iniciar Sesión

Caso de uso 1	Ver Iniciar Sesión
Alias	
Actores	Usuario, Administrador e Invitado
Requisito funcional	Puerta de acceso al sistema de la aplicación

Descripción	Es la pantalla principal de la aplicación donde se puede ir a crear una cuenta, registrarse o entrar como invitado.
Referencias	Modo Invitado (4), Crear Cuenta (3), Introducir Credenciales (2) Ver Inicio (5) y Cerrar Sesión (11)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.2. Tabla de especificación Introducir Credenciales

Caso de uso 2	Introducir Credenciales
Alias	
Actores	Usuario y Administrador
Requisito funcional	Permite el acceso mediante credenciales
Descripción	Los usuarios deben poder iniciar sesión con sus credenciales (correo electrónico o nombre de usuario y contraseña). El sistema validará estas credenciales comparándolas con los datos almacenados en la base de datos.
Referencias	Ver Iniciar Sesión (1), Ver Inicio (5) y Ver Menú Administrador (36)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.3. Tabla de especificación Crear Cuenta

Caso de uso 3	Crear Cuenta
Alias	
Actores	Invitado

Requisito funcional	Crear cuenta para acceder a la aplicación
Descripción	Permite a los usuarios no registrados crearse una cuenta para poder acceder de forma completa a la aplicación.
Referencias	Ver Iniciar Sesión (1) y Ver Inicio (5)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.4. Tabla de especificación Modo Invitado

Caso de uso 4	Modo Invitado
Alias	
Actores	Invitado
Requisito funcional	Deja entrar a los usuarios no registrados en la aplicación
Descripción	Se encarga de permitir el acceso limitado a la aplicación a los usuarios que no se han registrado previamente en la aplicación.
Referencias	Ver Iniciar Sesión (1) y Ver Inicio (5)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.5. Tabla de especificación Ver Inicio

Caso de uso 5	Ver Inicio
Alias	
Actores	Invitado, Usuario y Administrador
Requisito funcional	Ver la pantalla de inicio
Descripción	Es la pantalla principal de la aplicación donde se pueden ver los top 10 en España y próximos estrenos de series y

Caso de uso 5	Ver Inicio
Alias	
	películas.
Referencias	Ver Iniciar Sesión (1), Introducir Credenciales (2), Modo Invitado (4), Ir a Inicio (6), Ver Ajustes (7), Ver mi Lista (12), Ver Buscador (13), Ver Información Título (18), Volver Menú Administrador (19), Añadir Notificación (27) y Eliminar Notificación (28)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.6. Tabla de especificación Ir a Inicio

Caso de uso 6	Ir a Inicio
Alias	
Actores	Usuario y Administrador
Requisito funcional	Volver a inicio
Descripción	Permite al usuario y administrador volver a inicio, ya sea que esté en mi lista, ajustes o el buscador.
Referencias	Ver inicio (5), Ver Ajustes (7), Ver mi Lista (12), Ver Buscador (13) y Ver Menú Administrador (20)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.7. Tabla de especificación Ver Iniciar Sesión

Caso de uso 7	Ver Ajustes
Alias	
Actores	Usuario y Administrador
Requisito funcional	Ver opciones de ajustes y ver su perfil.
Descripción	Sección donde los usuarios y administrador pueden ver las opciones de ajustes y ver su perfil.
Referencias	Ver inicio (5), Ir a Inicio (6) y Ver Perfil (8)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.8. Tabla de especificación Ver Perfil

Caso de uso 8	Ver Perfil
Alias	
Actores	Usuario y Administrador
Requisito funcional	Ver el perfil junto con sus opciones.
Descripción	Permite al usuario y administrador poder ver su perfil, las opciones para configurarlo y cerrar sesión.
Referencias	Ver Ajustes (7), Cambiar Contraseña (9), Modificar Foto (10) y Cerrar Sesión (11)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.9. Tabla de especificación Cambiar Contraseña

Caso de uso 9	Cambiar Contraseña
Alias	
Actores	Usuario y Administrador
Requisito funcional	Cambiar la contraseña de la cuenta.
Descripción	El sistema proporcionará un mecanismo para que los usuarios y administrador puedan cambiar su contraseña.
Referencias	Ver Perfil (8)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.10. Tabla de especificación Modificar Foto

Caso de uso 10	Modificar Foto
Alias	
Actores	Usuario y Administrador
Requisito funcional	Cambiar la foto de perfil de la cuenta.
Descripción	Permite al usuario y administrador poder cambiar la foto de perfil que tiene en la cuenta.
Referencias	Ver Perfil (8)
Comentarios	Ningún comentario

Fuente: Elaboración Propia (2025)

Tabla 8.11. Tabla de especificación Cerrar Sesión

Caso de uso 11	Cerrar Sesión
Alias	
Actores	Usuario y Administrador
Requisito funcional	Cerrar sesión en la aplicación.
Descripción	Permite a los usuarios y administrador cerrar sesión de manera segura.
Referencias	Ver Iniciar Sesión (1) y Ver Perfil (8)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.12. Tabla de especificación Ver mi Lista

Caso de uso 12	Ver mi Lista
Alias	
Actores	Usuario y Administrador
Requisito funcional	Ver los títulos guardados con opción de quitarlos y ver su información.
Descripción	Permite al usuario y administrador ver las series y películas guardadas, la opción de quitarlas y ver su información.
Referencias	Ver Inicio (5), Ir a Inicio (6), Quitar Título (14) y Ver Información Título (18)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.13. Tabla de especificación Ver Buscador

Caso de uso 13	Ver Buscador
Alias	
Actores	Usuario y Administrador
Requisito funcional	Ver todos los títulos disponibles en la aplicación.
Descripción	Permite al usuario y administrador ver todas las películas y series disponibles en la aplicación.
Referencias	Ver Inicio (5), Ir a Inicio (6), Quitar Título (14), Añadir Título (15), Buscar Título (16), Filtrar Título (17) y Ver Información Título (18)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.14. Tabla de especificación Quitar Título

Caso de uso 14	Quitar Titulo
Alias	
Actores	Usuario y Administrador
Requisito funcional	Quita un título de la lista del usuario.
Descripción	Permite al usuario y administrador eliminar una serie o película de su lista.
Referencias	Ver mi Lista (12) y Ver Buscador (13)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.15. Tabla de especificación Añadir Título

Caso de uso 15	Añadir Título
Alias	
Actores	Usuario y Administrador
Requisito funcional	Guarda un título a la lista del usuario.
Descripción	Permite al usuario y administrador añadir una película o serie a su lista.
Referencias	Ver Buscador (13)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.16. Tabla de especificación Buscar Título

Caso de uso 16	Buscar Título
Alias	
Actores	Usuario y Administrador
Requisito funcional	Buscar Título por el nombre.
Descripción	Permite al usuario y administrador buscar una película o serie por el nombre.
Referencias	Ver Buscador (13)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.17. Tabla de especificación Filtrar Título

Caso de uso 17	Filtrar Título
Alias	
Actores	Usuario y Administrador
Requisito funcional	Filtra los títulos por género.
Descripción	Permite al usuario y administrador buscar series o películas por género.
Referencias	Ver Buscador (13)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.18. Tabla de especificación Ver Información Título

Caso de uso 18	Ver Información Título
Alias	
Actores	Usuario y Administrador
Requisito funcional	Ver información básica de un título específico.
Descripción	Permite al usuario y administrador ver la información de una película o serie en específico.
Referencias	Ver Inicio (5), Ver mi Lista (12) y Ver Buscador (13)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.19. Tabla de especificación Volver Menú Administrador

Caso de uso 19	Volver Menú Administrador
Alias	
Actores	Administrador
Requisito funcional	Permite al administrador volver al menú.
Descripción	Permite al administrador volver al menú desde el inicio.
Referencias	Ver Inicio (5) y Ver Menú Administrador (20)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.20. Tabla de especificación Ver Menú Administrador

Caso de uso 20	Ver Menú Administrador
Alias	
Actores	Administrador
Requisito funcional	Ver las opciones del administrador.
Descripción	Permite ver al administrador las opciones que tiene en la aplicación.
Referencias	Introducir Credenciales (2), Ir a Inicio (6), Volver Menú Administrador (19), Ver Menú Fotos (21) y Ver Menú Usuarios (23)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.21. Tabla de especificación Ver Menú Fotos

Caso de uso 21	Ver Menú Fotos
Alias	
Actores	Administrador
Requisito funcional	Ver el menú de fotos y sus opciones.
Descripción	Permite ver al administrador las opciones que tiene para insertar las fotos de perfil.
Referencias	Ver Menú Administrador (20) y Eliminar Foto (22)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.22. Tabla de especificación Eliminar Foto

Caso de uso 22	Eliminar Foto
Alias	
Actores	Administrador
Requisito funcional	Permite eliminar una foto de perfil.
Descripción	Permite al administrador eliminar una foto de perfil de la aplicación.
Referencias	Ver Menú Fotos (21)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.23. Tabla de especificación Ver Menú Usuarios

Caso de uso 23	Ver Menú Usuarios
Alias	
Actores	Administrador
Requisito funcional	Son las opciones que tiene el administrador con respecto a los usuarios.
Descripción	Permite ver al administrador las opciones que tiene para editar los usuarios.
Referencias	Ver Menú Administrador (20) y Añadir Usuario (24), Eliminar Usuario (25) y Modificar Usuario (26)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.24. Tabla de especificación Añadir Usuario

Caso de uso 24	Añadir Usuario
Alias	
Actores	Administrador
Requisito funcional	El administrador puede añadir a un usuario.
Descripción	Permite al administrador añadir un usuario a la aplicación.
Referencias	Ver Menú Usuarios (23)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.25. Tabla de especificación Eliminar Usuario

Caso de uso 25	Eliminar Usuario
Alias	
Actores	Administrador
Requisito funcional	El administrador puede eliminar a un usuario.
Descripción	Permite al administrador eliminar un usuario de la aplicación.
Referencias	Ver Menú Usuarios (23)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.26. Tabla de especificación Modificar Usuario

Caso de uso 26	Modificar Usuario
Alias	
Actores	Administrador
Requisito funcional	El administrador puede modificar a un usuario.
Descripción	Permite al administrador modificar un usuario de la aplicación.
Referencias	Ver Menú Usuarios (23)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.27. Tabla de especificación Añadir Notificación

Caso de uso 27	Añadir Notificación
Alias	
Actores	Administrador y Usuario
Requisito funcional	El usuario puede añadir la notificación de un estreno.
Descripción	Permite al usuario guardar esa serie o película a su lista y también en el momento que se estrene, enviar una notificación donde se avise del estreno de este título.
Referencias	Ver Inicio (5)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

Tabla 8.28. Tabla de especificación Eliminar Notificación

Caso de uso 28	Eliminar Notificación
Alias	
Actores	Administrador y Usuario
Requisito funcional	El usuario puede quitar la notificación de un estreno.
Descripción	Permite al usuario quitar esa serie o película de su lista y también se eliminará la notificación de su estreno.
Referencias	Ver Inicio (5)
Comentarios	Ningún comentario

Fuente: Elaboración propia (2025)

DIAGRAMA DE ENTIDAD - RELACIÓN

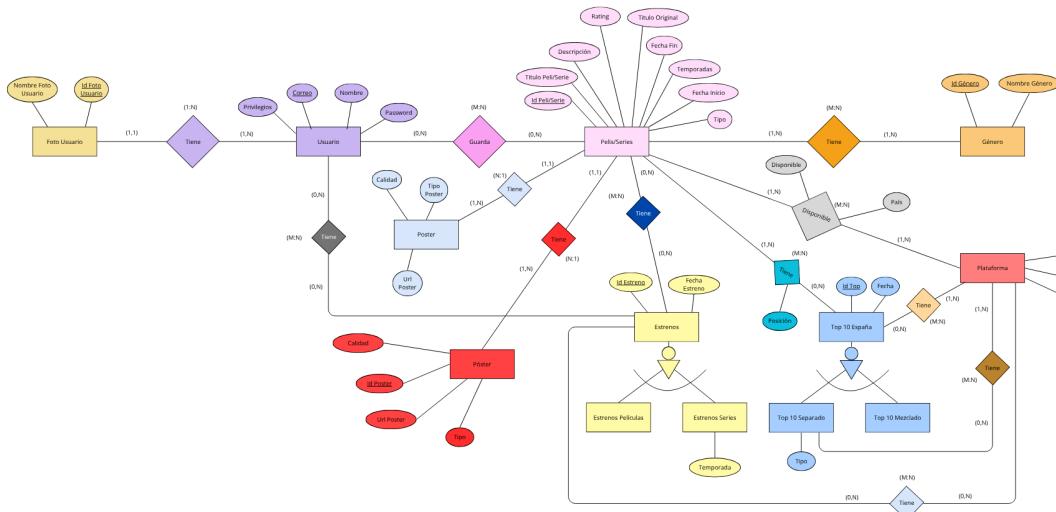
Una vez definidos los requisitos funcionales y no funcionales del sistema, así como los casos de uso que delimitan su comportamiento, se ha elaborado un Modelo Entidad-Relación (E-R) que representa de forma estructurada los datos que componen el sistema, así como las relaciones entre los mismos.

Este modelo constituye un paso fundamental en la fase de análisis, ya que permite establecer la base lógica para la posterior implementación del sistema de gestión de bases de datos. En él se identifican las principales entidades, sus atributos, y las relaciones que se producen entre ellas, recogiendo toda la información necesaria para cubrir las funcionalidades previstas.

En la imagen 8.2., se presenta el diagrama E-R elaborado. Como apoyo visual y metodológico, se ha optado por asignar colores diferenciados a cada conjunto de elementos para facilitar la lectura y comprensión del diagrama. Este recurso gráfico permite identificar rápidamente la función y el agrupamiento de cada componente:

- Entidades principales: representadas en distintos colores pastel, como lila, celeste, amarillo o naranja. Cada color agrupa entidades con funciones similares dentro del sistema.
- Atributos: se muestran como óvalos conectados a sus respectivas entidades, también coloreados de forma acorde al grupo al que pertenecen.
- Relaciones: aparecen en forma de rombos con un color distintivo respecto a las entidades, y están claramente etiquetadas con su nombre y cardinalidad.

El uso del color en el diagrama no responde únicamente a una cuestión estética, sino que tiene un propósito claramente funcional: ayudar a entender mejor la estructura del sistema. Al diferenciar visualmente las entidades, relaciones y atributos por colores, se facilita la identificación de los distintos componentes y se hace más accesible la lectura del modelo. Esta representación permite captar de forma rápida cómo se organizan los datos, qué elementos están relacionados entre sí y cómo se distribuyen dentro del sistema, lo que resulta especialmente útil tanto durante el análisis como en futuras fases de desarrollo.

Imagen 8.2. Diagrama de Entidad - Relación de WatchView⁴

Fuente: Elaboración propia (2025)

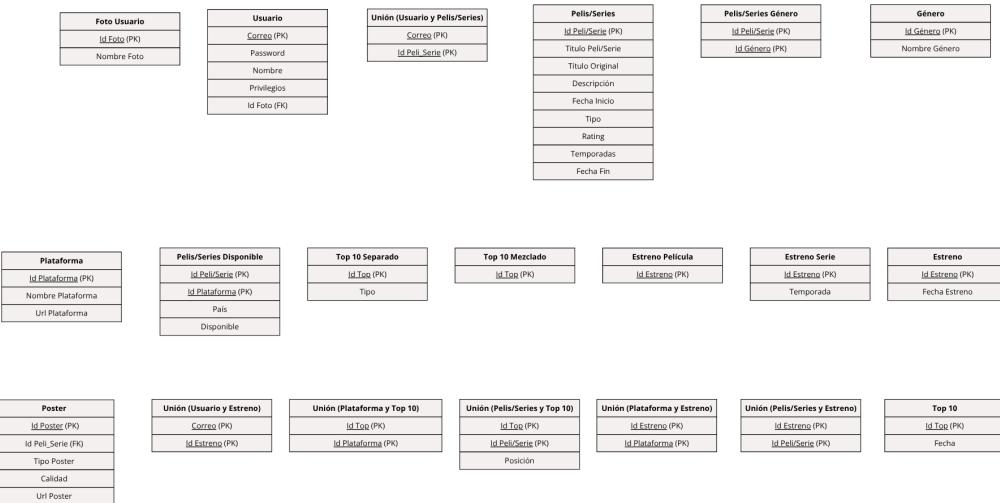
NORMALIZACIÓN

Antes de implementar el modelo físico, se ha aplicado el proceso de normalización al modelo entidad-relación con el objetivo de garantizar la coherencia, integridad y eficiencia de la base de datos. Esta técnica permite estructurar los datos correctamente, evitando redundancias y asegurando que las relaciones entre tablas mantengan su lógica funcional. Asimismo, la normalización pasa por tres fases:

- **Primera Forma Normal (1FN)**: Todos los atributos son atómicos y no existen grupos repetitivos. Segunda
- **Forma Normal (2FN)**: En las tablas con claves compuestas, los atributos dependen completamente de la clave primaria. No hay dependencias parciales.
- **Tercera Forma Normal (3FN)**: No existen dependencias transitivas entre atributos no clave. Todos dependen directamente de la clave primaria.

En este caso, el modelo ha sido normalizado hasta la Tercera Forma Normal (3FN), lo que asegura un diseño claro, optimizado y preparado para su implementación.

Imagen 8.3. Normalización de la Entidad - Relación⁵⁴ Ver en mayor tamaño en Anexos⁵ Ver en mayor tamaño en Anexos



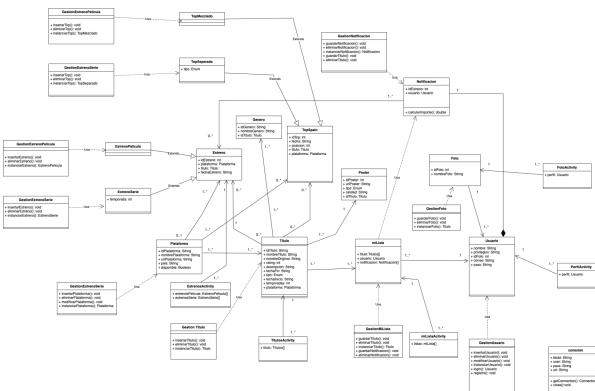
Fuente: Elaboración propia (2025)

La imagen 8.3. representa el modelo lógico resultante tras la normalización del esquema entidad-relación. En ella se muestran las distintas tablas que compondrán la base de datos, cada una con sus atributos y claves primarias (PK), subrayadas, y claves foráneas (FK) bien definidas. Se incluyen tanto entidades principales como Usuario, Pelis/Series o Plataforma, como también tablas intermedias que gestionan relaciones muchos a muchos, por ejemplo Pelis/Series Género o Unión (Usuario y Peli/Serie). Además, se reflejan especializaciones como Top 10 Mezclado o Estreno Película, que amplían el comportamiento de otras tablas generales. Este modelo permite una implementación clara, coherente y preparada para cualquier sistema gestor de bases de datos relacional.

En resumen, la normalización ha permitido transformar el modelo conceptual en una estructura lógica robusta, libre de inconsistencias y preparada para su implementación en un sistema de gestión de bases de datos relacional. Esta base sólida facilita la transición hacia el diseño orientado a objetos, donde el diagrama de clases permitirá representar la lógica del sistema desde una perspectiva más cercana a la programación y al desarrollo del software.

DIAGRAMA DE CLASES

Tras haber definido los requisitos y componentes funcionales del sistema, se presenta a continuación el diagrama de clases, el cual permite visualizar la estructura interna del sistema desde una perspectiva orientada a objetos. Este modelo facilita la comprensión de las entidades que intervienen, sus atributos, métodos principales y las relaciones que existen entre ellas.

Imagen 8.4. Diagrama de Clases de WatchView⁶

Fuente: Elaboración propia (2025)

Asimismo, la imagen 8.4, muestra una arquitectura organizada en torno a varios módulos clave. En primer lugar, se identifican las entidades relacionadas con los estrenos, diferenciando entre películas y series mediante herencia, y asociándolas a plataformas específicas. Por otro lado, se modela la entidad Título, que agrupa la información principal de cada contenido disponible (nombre, género, duración, etc.). También se contempla la gestión del usuario, incluyendo su perfil, listas personalizadas y notificaciones asociadas. Además, el sistema permite almacenar imágenes de perfil, gestionar favoritos, y ofrece funcionalidades para filtrar títulos mediante rankings o tops.

El diseño incorpora clases de gestión (GestionEstrenoPelícula, GestionUsuario, GestionTitulo, entre otras), lo cual favorece la separación de responsabilidades y una organización clara de la lógica de negocio.

En general, el modelo ofrece una visión completa y bien estructurada del dominio, permitiendo una implementación escalable y modular del sistema. Las relaciones y cardinalidades están correctamente planteadas y reflejan adecuadamente los vínculos entre las distintas clases.

9. Diseño

En esta sección se aborda la fase visual del proyecto, centrada en el diseño gráfico de la aplicación. El objetivo principal es dotar a *WatchView* de una identidad visual propia y coherente, que refleje su propósito y facilite el posterior desarrollo en código. Contar con un

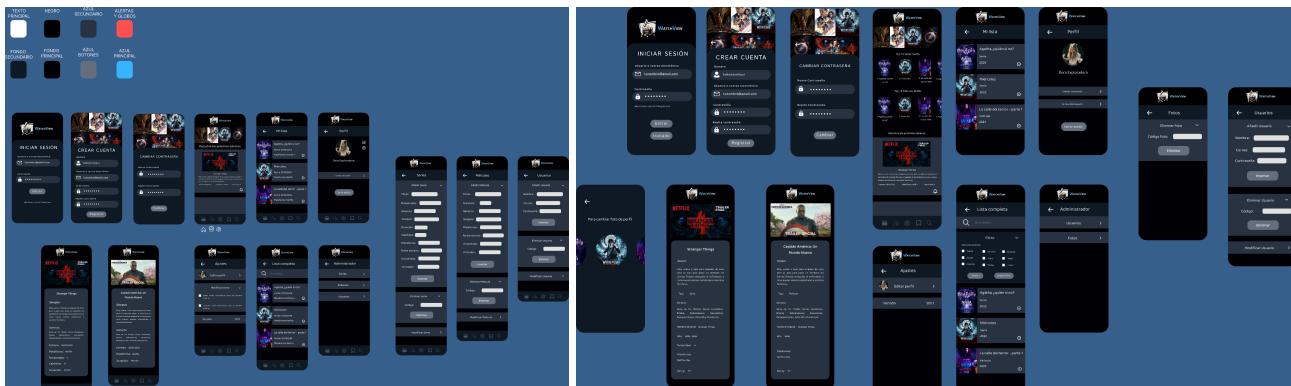
⁶ Ver en mayor tamaño en Anexos

diseño definido desde las primeras etapas permite agilizar la implementación, reducir errores de interpretación y mejorar la coordinación dentro del equipo de desarrollo.

Disponer de una referencia visual temprana también permite detectar posibles fortalezas y áreas de mejora en la experiencia de usuario, ya que es más sencillo ajustar elementos en esta fase que una vez iniciada la programación. Para ello, se ha elaborado un *mockup* que representa la estructura y funcionalidad de la interfaz gráfica propuesta. Como se muestra en la imagen 9.1, este diseño preliminar anticipa cómo será la interacción del usuario con la aplicación y sirve como guía para tomar decisiones relacionadas con la usabilidad, la navegación y la disposición de los elementos visuales.

La herramienta utilizada para la creación del mockup ha sido Figma, una plataforma especializada en diseño de interfaces y prototipos interactivos. Esta herramienta facilita un entorno de trabajo colaborativo y eficiente, lo que ha contribuido a mantener la coherencia visual del proyecto.

Imagen 9.1. Mock Up previo y final de la aplicación WatchView⁷



Fuente: Elaboración propia (2025)

Uno de los aspectos abordados durante esta etapa ha sido la definición de la paleta de colores que se mantendrá a lo largo de toda la aplicación. Como se observa en la imagen de la izquierda (dentro de la imagen 9.1), el diseño inicial ya contemplaba una estética oscura que se ha mantenido en el producto final. Esta elección se realizó de manera intencional, ya que muchas aplicaciones de entretenimiento son utilizadas principalmente en horarios nocturnos. Optar por colores oscuros reduce el deslumbramiento en condiciones de baja luz,

⁷ Ver en mayor tamaño en Anexos

mejorando así la comodidad del usuario. Este enfoque también es coherente con otras plataformas del sector, como Netflix, Prime Video o Max, que comparten una estética similar.

En cuanto a las modificaciones realizadas respecto al diseño inicial, cabe destacar los siguientes cambios:

- Desaparición de la barra de búsqueda: Se ha optado por una navegación más lineal, en la que el usuario entra y sale de secciones sin necesidad de utilizar una barra de herramientas fija.
- Ampliación de la pantalla de inicio: Se han añadido dos secciones que muestran el Top 10 de Netflix España, diferenciando entre películas y series, con el objetivo de ofrecer contenidos actualizados y relevantes.
- Nuevas opciones en el perfil de usuario: Se ha incorporado una pantalla adicional que permite seleccionar entre diferentes fotos de perfil.
- Simplificación de los ajustes: Se ha eliminado la opción de activar o desactivar notificaciones, ya que este permiso se solicita al iniciar la aplicación, haciéndola innecesaria dentro del menú.
- Mejoras en el buscador de títulos: Se ha añadido un desplegable para filtrar por géneros, facilitando una búsqueda más específica sin perder generalidad.
- Cambios en el menú de administración: Ya no se gestionan manualmente los títulos, puesto que estos se obtienen directamente desde una API externa. En su lugar, se ha habilitado una función para administrar las fotos de perfil, permitiendo eliminar imágenes no deseadas, con la única condición de mantener al menos una foto por usuario para evitar conflictos en la base de datos.

El *mockup* final presenta una interfaz moderna, intuitiva y visualmente limpia. El diseño se estructura en varias pantallas principales, que representan las funcionalidades clave de la aplicación:

- **Pantallas de autenticación:** Permiten al usuario iniciar sesión, registrarse o recuperar su contraseña. Se han diseñado con formularios simples, campos bien definidos y botones accesibles, para una experiencia intuitiva.
- **Pantalla de inicio (home):** Muestra los contenidos más populares, incluyendo tops de películas y series, estrenos recientes. Actúa como el punto de entrada principal para explorar el catálogo.
- **Información de los títulos:** Presenta información específica de cada título, como nombre, género, sinopsis, año de estreno, plataforma disponible y valoración. Esta sección ayuda al usuario a decidir qué ver.
- **Gestión del perfil:** Incluye opciones para cambiar la foto de perfil y contraseña y cerrar sesión.
- **Mi lista:** Permite al usuario acceder a los títulos guardados previamente, con opciones para filtrarlos y gestionarlos fácilmente.
- **Exploración de títulos:** Ofrece una vista más completa del catálogo, permitiendo aplicar filtro como género.
- **Menú de administración:** Destinado a usuarios con permisos especiales, permite gestionar cuentas y archivos relacionados.
- **Pantalla de ajustes:** Facilita la configuración de aspectos generales de la aplicación.

Este diseño actúa como una guía visual clara para el equipo de desarrollo y facilita la validación anticipada de decisiones relacionadas con la experiencia de usuario. De esta forma, se asegura una aplicación funcional, coherente con los objetivos del proyecto y atractiva para los usuarios.

10. Despliegue y pruebas

En esta sección se analizan los fragmentos de código más relevantes desarrollados durante la implementación de la aplicación *WatchView*, así como las pruebas realizadas para garantizar el correcto funcionamiento de ciertas funcionalidades críticas.

En primer lugar, en la imagen 10.1 se presenta un fragmento correspondiente al adaptador del buscador. En este caso, se ha diseñado una lógica para seleccionar el póster más adecuado para mostrar al usuario. Se ha optado por obtener la primera imagen de tipo vertical disponible, ya que este tipo suele tener menor peso y resolución, lo que optimiza el rendimiento de la aplicación sin comprometer la experiencia visual del usuario.

Imagen 10.1. Obtención del póster vertical en Adaptador Buscador⁸

```
// Cargar la imagen de póster si existe
val posterVertical = item.posters
    .filter { it.tipo == "vertical" }
    .first() // primero de la lista

if (posterVertical != null) {
    Glide.with(holder.imagen.context)
        .load(posterVertical.urlPoster)
        .apply(RequestOptions().centerCrop()) // caso normal: se recorta
        .thumbnail(
            Glide.with(holder.imagen.context)
                .load(posterVertical.urlPoster)
                .apply(RequestOptions().fitCenter()) // fallback visual
        )
        .into(holder.imagen)

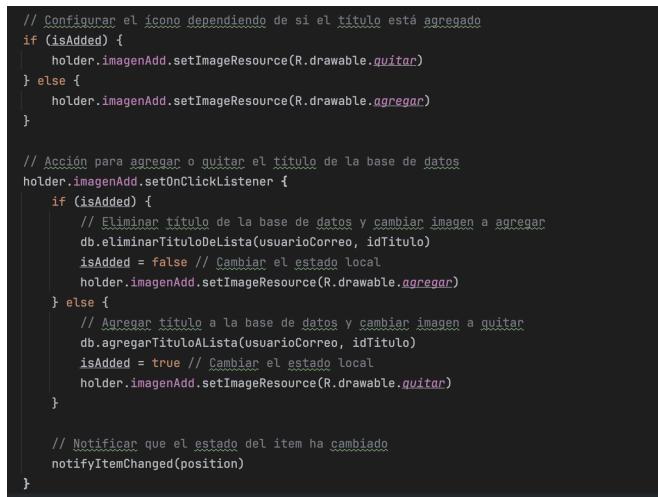
} else {
    holder.imagen.setImageResource(R.drawable.miercoles) // Default image if no poster
}
```

⁸ Ver en mayor tamaño en Anexos

Fuente: Elaboración propia (2025)

A continuación, en la imagen 10.2 se muestra la funcionalidad relacionada con la acción de añadir un título a la lista personalizada del usuario. Al hacerlo, la imagen del botón cambia dinámicamente entre los estados de “agregar” y “quitar”, permitiendo una retroalimentación visual inmediata que indica si el contenido ya ha sido añadido a la lista.

Imagen 10.2. Añadir título a lista y alternar entre imagen agregar y quitar⁹



Fuente: Elaboración propia (2025)

En la imagen 10.3 se observa cómo, desde el PerfilFragment, se accede a las imágenes de perfil almacenadas en la base de datos. Estas imágenes se encuentran dentro de la carpeta drawable, y son procesadas mediante un CenterCrop y RoundCorners para mejorar su presentación visual, ofreciendo un diseño más pulido y acorde a la estética general de la aplicación.

Imagen 10.3. Obtener y mostrar fotos de perfil de tipo drawable¹⁰



⁹ Ver en mayor tamaño en Anexos

¹⁰ Ver en mayor tamaño en Anexos

Fuente: Elaboración propia (2025)

Por otra parte, en la imagen 10.4 se muestra el fragmento que realiza la llamada a la API de *Streaming Availability*. Este fragmento está configurado para solicitar un máximo de 20 títulos disponibles en Netflix España, con el fin de evitar una sobrecarga de datos que pueda perjudicar el rendimiento del sistema. Esta limitación es especialmente relevante teniendo en cuenta que la API gestiona miles de contenidos, y el objetivo actual es el desarrollo de un prototipo funcional.

Imagen 10.4. Obtención de títulos con llamada a la Api Streaming Availability¹¹

```
CoroutineScope(Dispatchers.IO).launch {
    val client = OkHttpClient()
    val apiKey = context.getString(R.string.api_key)
    var hasMore = true
    var cursor: String? = null
    var totalGuardados = 0
    val maxTitulos = 20

    while (hasMore && totalGuardados < maxTitulos) {
        val urlBuilder = StringBuilder("https://streaming-availability.p.rapidapi.com/shows/search/filters")
        urlBuilder.append("?country=es&series_granularity=show&order_direction=asc&output_language=es&catalogs=netflix")
        if (cursor != null) urlBuilder.append("&show.cursor=${Uri.encode(cursor)})")

        val request = Request.Builder()
            .url(urlBuilder.toString())
            .get()
            .addHeader(name: "x-rapidapi-host", value: "streaming-availability.p.rapidapi.com")
            .addHeader(name: "x-rapidapi-key", apiKey)
            .build()

        try {
            val response = client.newCall(request).execute()
            if (response.isSuccessful) {
                val json = response.body?.string()
                val (next, more, guardadosEnEstaPagina) = leerJSONTitulos2(context, json, maxTitulos - totalGuardados)
                totalGuardados += guardadosEnEstaPagina
                cursor = next
                hasMore = more
            } else {
                hasMore = false
            }
        } catch (e: IOException) {
            Timber.e(e)
        }
    }
}
```

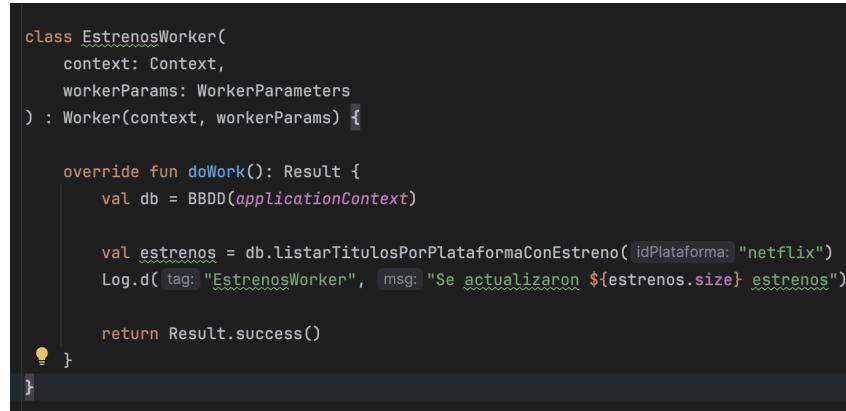
Fuente: Elaboración propia (2025)

Una parte fundamental del funcionamiento de la aplicación es la actualización automática de los datos. Para ello se han implementado dos workers, componentes diseñados para ejecutar tareas en segundo plano de forma periódica. En la imagen 10.5 se presenta el primero de ellos, cuya función es actualizar las tablas que contienen títulos con fecha de estreno, asegurando así que la sección de “estrenos recientes” se mantenga actualizada y refleje correctamente los títulos lanzados ese mismo día.

Imagen 10.5. Worker actualización listas¹²

¹¹ Ver en mayor tamaño en Anexos

¹² Ver en mayor tamaño en Anexos



```
class EstrenosWorker(
    context: Context,
    workerParams: WorkerParameters
) : Worker(context, workerParams) {

    override fun doWork(): Result {
        val db = BBDD(applicationContext)

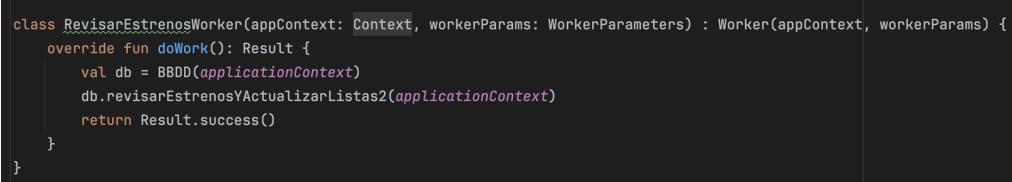
        val estrenos = db.listarTitulosPorPlataformaConEstreno(idPlataforma: "netflix")
        Log.d(tag: "EstrenosWorker", msg: "Se actualizaron ${estrenos.size} estrenos")

        return Result.success()
    }
}
```

Fuente: Elaboración propia (2025)

El segundo worker, mostrado en la imagen 10.6, se encarga no solo de actualizar la información en la base de datos, sino también de generar una notificación para el usuario en caso de que alguno de los títulos guardados en su lista haya sido estrenado. La función que lo acompaña, revisarEstrenosYActualizarListas2, compara la fecha actual con las almacenadas en la base de datos para verificar si se han producido nuevos estrenos.

Imagen 10.6. Worker actualizar listas y realizar notificación¹³



```
class RevisarEstrenosWorker(appContext: Context, workerParams: WorkerParameters) : Worker(appContext, workerParams) {
    override fun doWork(): Result {
        val db = BBDD(applicationContext)
        db.revisarEstrenosYActualizarListas2(applicationContext)
        return Result.success()
    }
}
```

Fuente: Elaboración propia

Como se detalla en la imagen 10.7, esta función central revisa si existen coincidencias entre la fecha actual y las fechas de estreno de los títulos. Si se encuentra una coincidencia, primero se genera una notificación mediante la función mostrarNotificaciónDeEstreno2, y posteriormente se actualiza la base de datos con actualizarTodosLosTitulosVistos y la función previamente mencionada de actualización de listas.

Imagen 10.7. Función de actualización y llamada a notificación¹⁴

¹³ Ver en mayor tamaño en Anexos

¹⁴ Ver en mayor tamaño en Anexos

```

fun revisarEstrenosYActualizarListas2(context: Context) {

    val formatoFecha = SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.getDefault())
    val fechaHoyDate = formatoFecha.parse(formatoFecha.format(java.util.Date()))
    val estrenos = obtenerEstrenos() // List<Estreno>

    this.writableDatabase.use { db -> // 'db' es la base de datos dentro de este bloque 'use'
        for (estreno in estrenos) {
            val idEstreno = estreno.idEstreno
            val idTitulo = estreno.idTitulo
            val fechaEstrenoDate = formatoFecha.parse(estreno.fechaEstreno)

            if (fechaEstrenoDate != null && fechaHoyDate != null) {
                if (fechaEstrenoDate == fechaHoyDate) {
                    obtenerNombreTitulo(idTitulo)?.let {
                        mostrarNotificacionDeEstreno2(context, it)
                        actualizarTodosLosTitulosVistos()
                        listarTitulosPorPlataformaConEstreno( idPlataforma: "netflix")
                    }
                }
            }

            if (fechaEstrenoDate <= fechaHoyDate) {
                val cursor = db.rawQuery(
                    sql: "SELECT correo FROM Estreno_Usuario WHERE idEstreno = ?",
                    array0f(idEstreno.toString())
                )
            }
        }
    }
}

```

Fuente: Elaboración propia

La función encargada de mostrar la notificación se encuentra en la imagen 10.8. En esta se configura el contenido del mensaje, incluyendo el nombre del título estrenado y el nombre de la aplicación, junto con el logotipo oficial de la misma. Esta notificación permite al usuario estar al tanto de los estrenos sin necesidad de acceder constantemente a la aplicación.

Imagen 10.8. Función para saltar notificación con logo de la App¹⁵

```

fun mostrarNotificacionDeEstreno2(context: Context, titulo: String) {
    val channelId = "estrenos_channel"
    val channelName = "Notificaciones de Estrenos"
    val notificationId = titulo.hashCode() // Para evitar duplicados

    val notificationManager = context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

    // Crear canal (obligatorio en Android 8+)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val channel = NotificationChannel(
            channelId,
            channelName,
            NotificationManager.IMPORTANCE_DEFAULT
        ).apply {
            description = "Notifica cuando se estrenan títulos"
        }
        notificationManager.createNotificationChannel(channel)
        Log.d( tag: "MostrarNotificacion", msg: "Canal de notificación creado con éxito")
    }

    // Cargar el ícono de la app desde los recursos
    val largeIcon = BitmapFactory.decodeResource(context.resources, R.drawable.logoapp2) // Reemplaza con tu ícono

    val builder = NotificationCompat.Builder(context, channelId)
        .setSmallIcon(R.drawable.logoapp2) // Ícono pequeño de la notificación (también puedes cambiarlo)
        .setContentTitle("¡Estreno disponible!")
        .setContentText("Ya está disponible: $titulo")
        .setLargeIcon(largeIcon) // Establece el ícono grande para la notificación
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setAutoCancel(true)

    // Mostrar la notificación
    notificationManager.notify(notificationId, builder.build())
}

```

Fuente: Elaboración propia

¹⁵ Ver en mayor tamaño en Anexos

Finalmente, en la imagen 10.9 se presentan tres funciones clave dentro de LoginActivity. La primera se encarga de programar el *worker* que realizará las tareas de actualización y notificación previamente descritas. La segunda calcula el tiempo restante hasta las 00:01 del día siguiente, momento en el que dicho *worker* debe ejecutarse. Por último, la tercera función solicita el permiso para recibir notificaciones, un requisito obligatorio a partir de Android 13.

Imagen 10.9. Funciones utilizadas para las notificaciones en LoginActivity¹⁶

```
class LoginActivity : AppCompatActivity() {  
    private fun programarWorkerDeEstrenos(context: Context) {  
        val initialDelay = calcularDelayHastaProxima12PM()  
  
        val request = PeriodicWorkRequestBuilder<RevisarEstrenosWorker>{ repeatInterval 1, TimeUnit.DAYS  
            .setInitialDelay(initialDelay, TimeUnit.MILLISECONDS)  
            .build()  
  
        WorkManager.getInstance(context).enqueueUniquePeriodicWork(  
            uniqueWorkName = "RevisarEstrenosDiario",  
            ExistingPeriodicWorkPolicy.KEEP,  
            request  
        )  
    }  
  
    private fun calcularDelayHastaProxima12PM(): Long {  
        val ahora = Calendar.getInstance()  
        val proxima3AM = Calendar.getInstance().apply {  
            set(Calendar.HOUR_OF_DAY, 0)  
            set(Calendar.MINUTE, 1)  
            set(Calendar.SECOND, 0)  
            set(Calendar.MILLISECOND, 0)  
  
            if (before(ahora)) {  
                add(Calendar.DAY_OF_MONTH, 1)  
            }  
        }  
        return proxima3AM.timeInMillis - ahora.timeInMillis  
    }  
  
    private fun solicitarPermisoNotificaciones() {  
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {  
            val permiso = Manifest.permission.POST_NOTIFICATIONS  
            if (ContextCompat.checkSelfPermission(context, permiso) != PackageManager.PERMISSION_GRANTED) {  
                ActivityCompat.requestPermissions(activity, arrayOf(permiso), requestCode: 1001)  
            }  
        }  
    }  
}
```

Fuente: Elaboración propia

Para garantizar que las funcionalidades básicas de la aplicación cumplen con los requisitos mínimos de calidad y seguridad, se han desarrollado una serie de pruebas orientadas a la validación de entradas del usuario. A continuación, se presenta en la tabla 10.1 con la especificación de cada una de estas pruebas:

¹⁶ Ver en mayor tamaño en Anexos

Tabla 10.1. Pruebas de código

Nº	ESPECIFICACIÓN DE PRUEBAS
1	<p>Correo en formato correo</p> <p>El formato válido es:</p> <ul style="list-style-type: none">- test@example.com- correo.valido_123@dominio.org <p>Ejemplos no válidos:</p> <ul style="list-style-type: none">- correoinvalido.com- invalido@com- otro@.com
2	<p>Nombre de usuario con primera letra en mayúscula</p> <p>El formato válido es:</p> <ul style="list-style-type: none">- Hola- A <p>Ejemplos no válidos:</p> <ul style="list-style-type: none">- hola- “ ” (vacío)
3	<p>Contraseña de usuario con primera letra en mayúscula, una minúscula, un número, un carácter especial y una longitud de 6 caracteres.</p> <p>El formato válido es:</p> <ul style="list-style-type: none">- Password1! <p>Ejemplos no válidos:</p> <ul style="list-style-type: none">- Pas1!- password1!- Password1

	- Password!
--	-------------

Estas pruebas se han diseñado para cubrir los aspectos básicos de validación en los formularios de registro e inicio de sesión. Con ello se pretende ofrecer una experiencia más segura al usuario, minimizando errores y posibles vulnerabilidades asociadas a entradas mal formateadas o inseguras.

11. Conclusiones

A lo largo del desarrollo de este trabajo se ha abordado la creación de la aplicación *WatchView* desde una perspectiva integral, abarcando todas las fases esenciales de un proyecto de software: análisis, diseño, desarrollo, pruebas e implementación. La planificación previa y la elaboración de documentación técnica resultaron fundamentales para estructurar de forma ordenada el desarrollo del proyecto y facilitar la posterior codificación. Si bien esta fase teórica demandó un esfuerzo considerable en tiempo y reflexión, permitió anticipar problemas, definir con claridad los objetivos funcionales y establecer una hoja de ruta eficaz para su ejecución.

Durante el proceso de implementación, el uso de referencias previas y fragmentos de código reutilizable ayudó a agilizar el desarrollo de la lógica de la aplicación. Comenzar con el desarrollo técnico desde etapas tempranas permitió identificar progresivamente los puntos más conflictivos, especialmente en aquellas pantallas con mayor nivel de complejidad funcional.

No obstante, la ejecución también presentó diversas dificultades técnicas que condicionaron el resultado final. Una de las principales complicaciones surgió en la gestión dinámica de la foto de perfil del usuario, cuyo cambio no se reflejaba de forma inmediata sin necesidad de navegar entre fragmentos. Este comportamiento afectó directamente a la experiencia de usuario.

Del mismo modo, la inserción y visualización de los títulos correspondientes al top 10 de Netflix supuso un reto constante, especialmente al combinar datos obtenidos a través de

una API con entradas manuales en la base de datos. A pesar de intentos por reorganizar la lógica de inserción, centralizar hilos de ejecución y mover procesos al inicio de sesión, la solución alcanzada resultó parcial, afectando la consistencia de los resultados mostrados en la aplicación.

Por otra parte, la funcionalidad relacionada con los estrenos y sus notificaciones, uno de los pilares conceptuales del proyecto, presentó limitaciones derivadas del funcionamiento de la API empleada. Esta no ofrecía información actualizada ni fiable sobre estrenos o rankings diarios, como inicialmente se preveía, lo que obligó a introducir manualmente los contenidos. Esta carencia afectó también a la lógica de las notificaciones, reduciendo su relevancia dentro del sistema.

Pese a estas dificultades, se ha logrado desarrollar una primera versión operativa de *WatchView* que permite explorar títulos, gestionar el perfil del usuario y, parcialmente, recibir notificaciones de nuevos contenidos. Si bien quedan aspectos por mejorar, la aplicación cumple con los objetivos fundamentales planteados al inicio del proyecto.

En definitiva, este trabajo ha supuesto una experiencia enriquecedora tanto en el plano académico como en el personal. Ha permitido consolidar conocimientos técnicos, enfrentarse a problemas reales de desarrollo y encontrar soluciones adaptadas al contexto. Aun con errores y limitaciones, se ha sentado una base sólida sobre la que se podrán construir futuras versiones más completas y pulidas. El proyecto demuestra el potencial de *WatchView* como aplicación orientada al usuario, y abre nuevas vías para su evolución tecnológica, funcional y visual en el futuro.

12. Vías futuras

Dado que esta aplicación se encuentra en una primera versión funcional, existen múltiples líneas de mejora que pueden explorarse en versiones posteriores para incrementar su robustez, usabilidad y funcionalidad.

Una de las principales áreas de mejora está relacionada con la actualización automática de contenidos, especialmente en lo referente a los estrenos y los rankings diarios. Aunque inicialmente se optó por utilizar la API de *Streaming Availability*, se comprobó que esta no cumplía con las expectativas descritas en su documentación. En lugar de ofrecer información actualizada sobre estrenos o top 10 diarios, la API proporcionaba cambios en los catálogos con fechas pasadas, lo cual limita seriamente su utilidad para el objetivo principal de la aplicación. Como consecuencia, tanto los estrenos

como los listados de top 10 de España han tenido que ser introducidos manualmente, con fechas fijas, sin posibilidad de actualización automática salvo que el programador intervenga directamente. Esto convierte a estas secciones en contenidos estáticos, perdiendo la capacidad de reflejar la realidad cambiante del mercado de streaming.

Derivado de este problema, la funcionalidad de notificaciones sobre estrenos pierde gran parte de su valor, ya que al no contar con datos actualizados de forma dinámica, el usuario no recibe información relevante ni oportuna. Este aspecto afecta a la propuesta de valor de la aplicación, cuyo objetivo era mantener informado al usuario sobre novedades relevantes de las plataformas. A ello se suma la dificultad general para acceder a esta clase de datos, dado que muchas plataformas de streaming mantienen su información de forma hermética, como parte de su estrategia comercial.

Cabe recordar que en el pasado, plataformas como Netflix ofrecieron APIs oficiales con datos básicos de sus títulos, lo que facilitaba el desarrollo de aplicaciones externas. Sin embargo, al observar que terceros comenzaban a lucrarse con estas integraciones, Netflix retiró su API pública, dificultando desde entonces el acceso a sus catálogos y rankings. Esta situación refleja un contexto más amplio en el que las grandes plataformas limitan el acceso a su información, lo que supone un reto técnico y legal para desarrolladores independientes.

A pesar de estas limitaciones, se considera como una vía futura prioritaria la búsqueda de nuevas fuentes de información, más fiables y actualizadas, que permitan restaurar y ampliar las funcionalidades estáticas. Entre las acciones a explorar se encuentran:

- Integrar fuentes alternativas de datos o APIs de terceros que, aunque de uso limitado, ofrezcan información parcial útil.
- Diseñar un sistema de scraping ético y controlado, si las condiciones legales lo permiten, para extraer información pública visible en los portales de las plataformas.
- Colaborar con servicios especializados o abrir vías de contacto con proveedores de datos que puedan facilitar el acceso a contenidos actualizados de forma segura y legal.

Además de estas mejoras, se mantienen otras líneas de desarrollo previamente mencionadas, como:

- La optimización del rendimiento de la aplicación, especialmente en lo relativo al uso de la base de datos y llamadas a la red.
- Mejoras en la interfaz y la experiencia de usuario.

- La incorporación de recomendaciones personalizadas y valoraciones.
- La preparación de la aplicación para su despliegue real en tiendas oficiales como Google Play.

En conjunto, estas propuestas buscan no solo solucionar los problemas actuales, sino también elevar el nivel de calidad, relevancia y competitividad de WatchView en futuras versiones.

13. Glosario

- Plataforma de streaming: Es un servicio digital que permite reproducir contenido multimedia —como series, películas o música— en tiempo real a través de internet, sin necesidad de descargarlo. Suelen funcionar bajo demanda y son accesibles desde múltiples dispositivos.
- Wishlist: Es una lista de deseos donde el usuario guarda productos, servicios o contenidos que le interesan para acceder a ellos más adelante.
- IDE: Es una herramienta que reúne en un solo programa todo lo necesario para desarrollar software, como editor de código, compilador, depurador y simulador de ejecución.
- Primary Key (PK): Es el atributo o conjunto de atributos que identifica de manera única cada fila de una tabla. No puede contener valores nulos ni repetidos.
- Foreign Key (FK): Es un atributo que establece una relación con la clave primaria de otra tabla. Sirve para mantener la integridad referencial entre tablas.

- Scraping ético: Es la práctica de extraer datos de sitios web respetando sus términos de uso, sin sobrecargar los servidores y sin vulnerar la privacidad o los derechos de los propietarios del contenido.
- API (Interfaz de Programación de Aplicaciones): Conjunto de reglas y funciones que permiten que diferentes programas o sistemas se comuniquen entre sí de forma estructurada.
- Mock Up: Representación visual estática del diseño de una aplicación o sitio web que muestra cómo se verá la interfaz antes de su desarrollo técnico.

14. Bibliografía

- movieofthenight. (2024). *Streaming Availability API*.
<https://docs.movieofthenight.com/>

15. Anexos

Trello

Resumen día 23

- ✓ Objetivos generales y específicos
- ✓ Especificación de tecnologías y herramientas

Resumen día 24

- + Añade una tarjeta

Resumen día 25

- ✓ DIAGRAMA ENTIDAD RELACIÓN (E-R)
- ✓ METODOLOGÍA
- ✓ DIAGRAMA DE CASOS DE USO

Resumen día 26

- ✓ FUNCIONALIDADES Y NO FUNCIONALIDADES
- ✓ DIAGRAMA DE CLASES

Resumen día 27

- ✓ DIAGRAMA DE GANTT
- + Añade una tarjeta

Resumen día 28

- ✓ MOTIVACIÓN
- + Añade una tarjeta

Diagrama de Gantt Previsto

TAREAS / SEMANAS	17-23 DE MARZO	24-30 DE MARZO	31-06 DE ABRIL	07-13 DE ABRIL	14-20 DE ABRIL	21-27 DE ABRIL	28-04 DE MAYO	05-11 DE MAYO	12-18 DE MAYO	19-25 DE MAYO
MOCK UP										
ENTIDAD RELACIÓN										
DIAGRAMA DE CLASES										
DIAGRAMA DE CASOS DE USO										
REQUISITOS FUNCIONALES Y NO FUNCIONALES										
OBJETIVOS GENERALES Y ESPECÍFICOS										
ESPECIFICACIÓN DE TECNOLOGÍAS Y HERRAMIENTAS										
METODOLOGÍA										
MOTIVACIÓN										
DIAGRAMA DE GANTT										
INICIO SESIÓN										
CREAR CUENTA										
INICIO APP										
BUSCADOR										
AJUSTES										
CUENTA										
INFORMACIÓN PELÍCULAS										
INFORMACIÓN SERIES										
WISHLIST										
MENÚ ADMINISTRADOR										
MENÚ ADMIN USUARIOS										
MENÚ ADMIN FOTOS										
CAMBiar CONTRASEÑA										
CAMBiar FOTO PERFIL										
MODO INVITADO										
PRUEBAS DE CÓDIGO										
ABSTRACT										

Diagrama de Gantt Real

TAREAS / SEMANAS	17-23 DE MARZO	24-30 DE MARZO	31-06 DE ABRIL	07-13 DE ABRIL	14-20 DE ABRIL	21-27 DE ABRIL	28-04 DE MAYO	05-11 DE MAYO	12-18 DE MAYO	19-25 DE MAYO
MOCK UP										
ENTIDAD RELACIÓN										
DIAGRAMA DE CLASES										
DIAGRAMA DE CASOS DE USO										
REQUISITOS FUNCIONALES Y NO FUNCIONALES										
OBJETIVOS GENERALES Y ESPECÍFICOS										
ESPECIFICACIÓN DE TECNOLOGÍAS Y HERRAMIENTAS										
METODOLOGÍA										
MOTIVACIÓN										
DIAGRAMA DE GANTT										
INICIO SESIÓN										
CREAR CUENTA										
INICIO APP										
BUSCADOR										
AJUSTES										
CUENTA										
INFORMACIÓN PELÍCULAS										
INFORMACIÓN SERIES										
WISHLIST										
MENÚ ADMINISTRADOR										
MENÚ ADMIN USUARIOS										
MENÚ ADMIN FOTOS										
CAMBiar CONTRASEÑA										
CAMBiar FOTO PERFIL										
MODO INVITADO										
PRUEBAS DE CÓDIGO										
ABSTRACT										

Diagrama de Casos de Uso (Usuario)

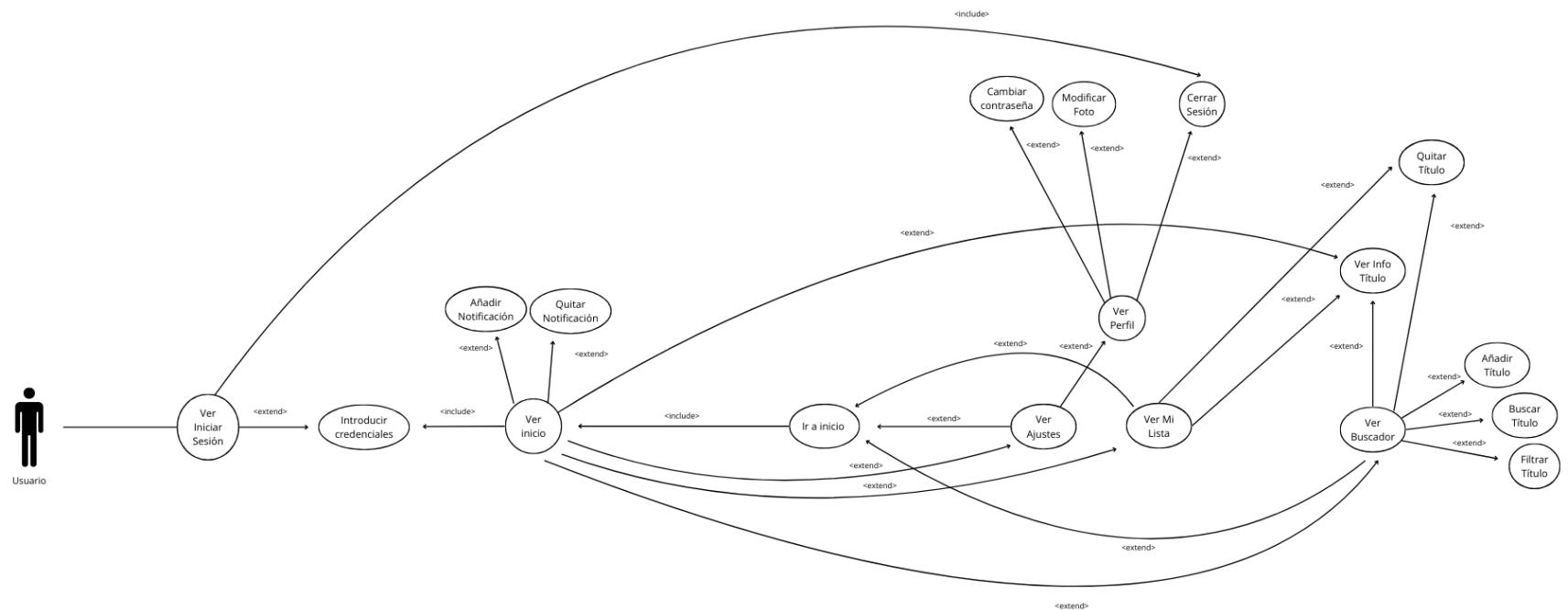


Diagrama de Casos de Uso (Administrador)

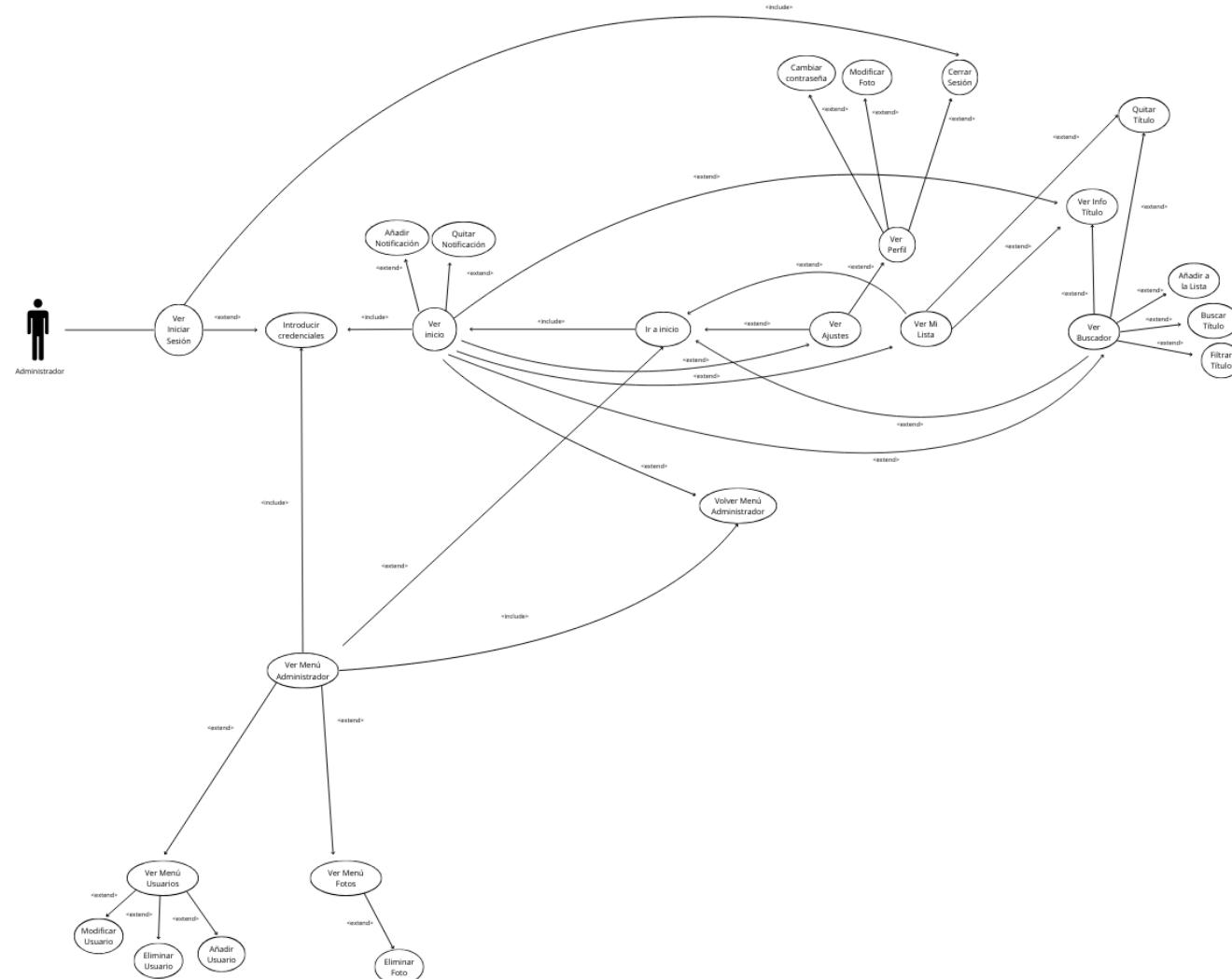


Diagrama de Casos de Uso (Invitado)

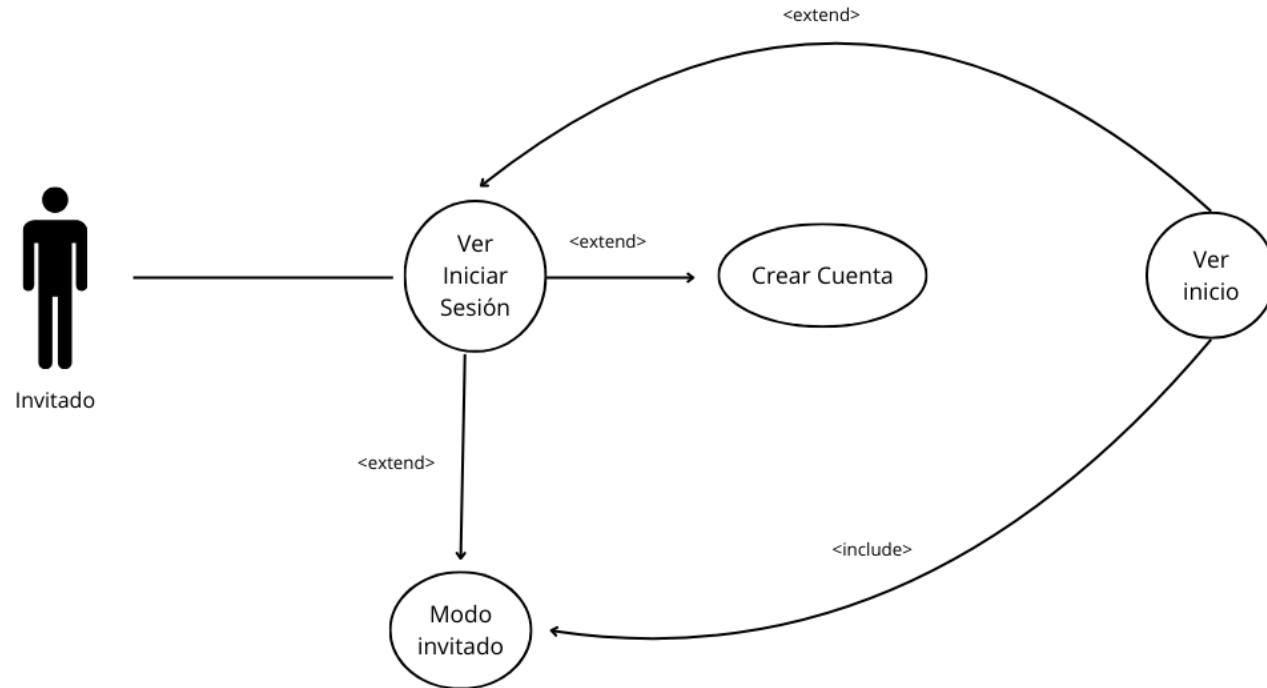
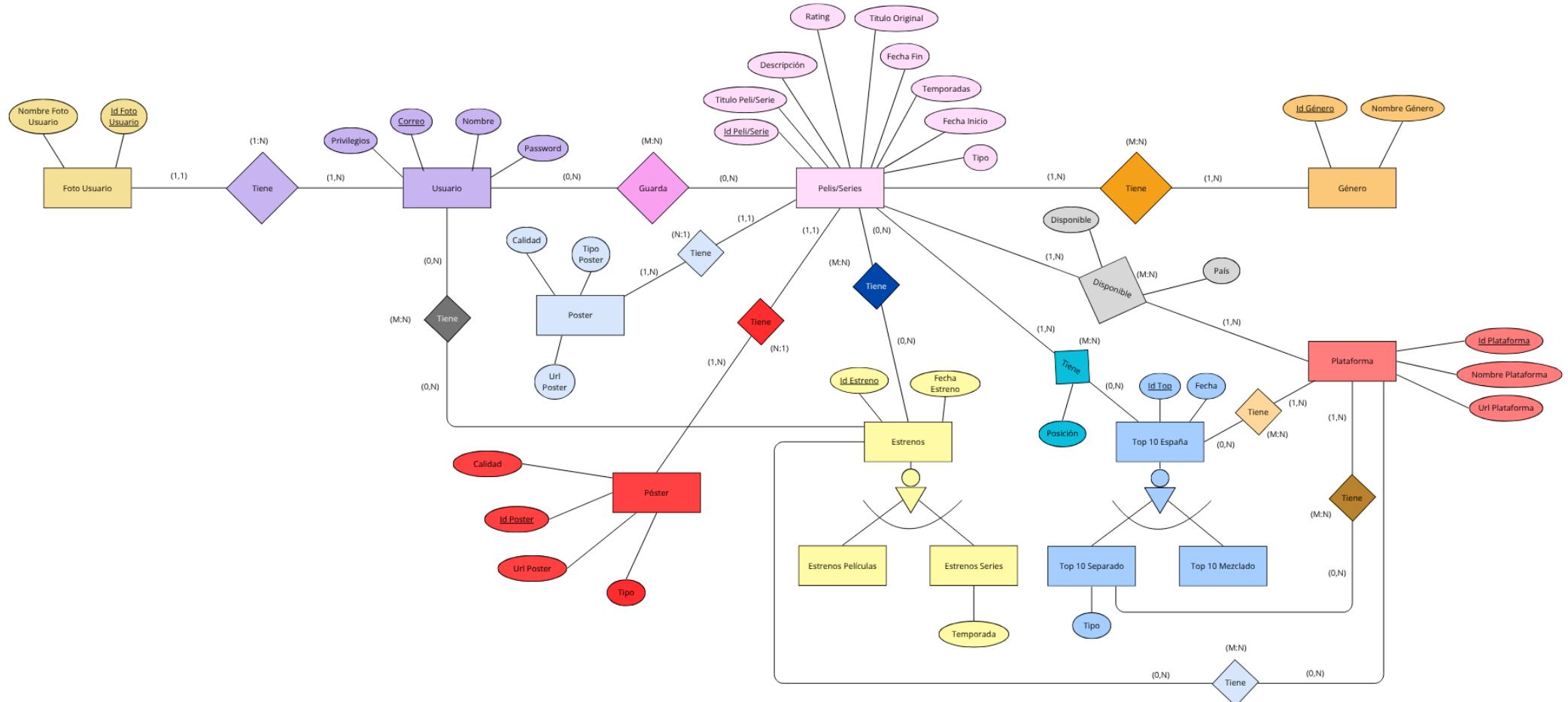


Diagrama Entidad - Relación



Normalización

Foto Usuario
<u>Id Foto</u> (PK)
Nombre Foto

Usuario
<u>Correo</u> (PK)
Password
Nombre
Privilegios
<u>Id Foto</u> (FK)

Unión (Usuario y Pelis/Series)
<u>Correo</u> (PK)
<u>Id Peli_Serie</u> (PK)

Pelis/Series
<u>Id Peli/Serie</u> (PK)
Titulo Peli/Serie
Título Original
Descripción
Fecha Inicio
Tipo
Rating
Temporadas
Fecha Fin

Pelis/Series Género
<u>Id Peli/Serie</u> (PK)
<u>Id Género</u> (PK)

Género
<u>Id Género</u> (PK)
Nombre Género

Plataforma
<u>Id Plataforma</u> (PK)
Nombre Plataforma
Url Plataforma

Pelis/Series Disponible
<u>Id Peli/Serie</u> (PK)
<u>Id Plataforma</u> (PK)
País
Disponible

Top 10 Separado
<u>Id Top</u> (PK)
Tipo

Top 10 Mezclado
<u>Id Top</u> (PK)

Estreno Película
<u>Id Estreno</u> (PK)

Estreno Serie
Temporada

Estreno
<u>Id Estreno</u> (PK)
Fecha Estreno

Poster
<u>Id Poster</u> (PK)
<u>Id Peli_Serie</u> (FK)
Tipo Poster
Calidad
Url Poster

Unión (Usuario y Estreno)
<u>Correo</u> (PK)
<u>Id Estreno</u> (PK)

Unión (Plataforma y Top 10)
<u>Id Top</u> (PK)
<u>Id Plataforma</u> (PK)

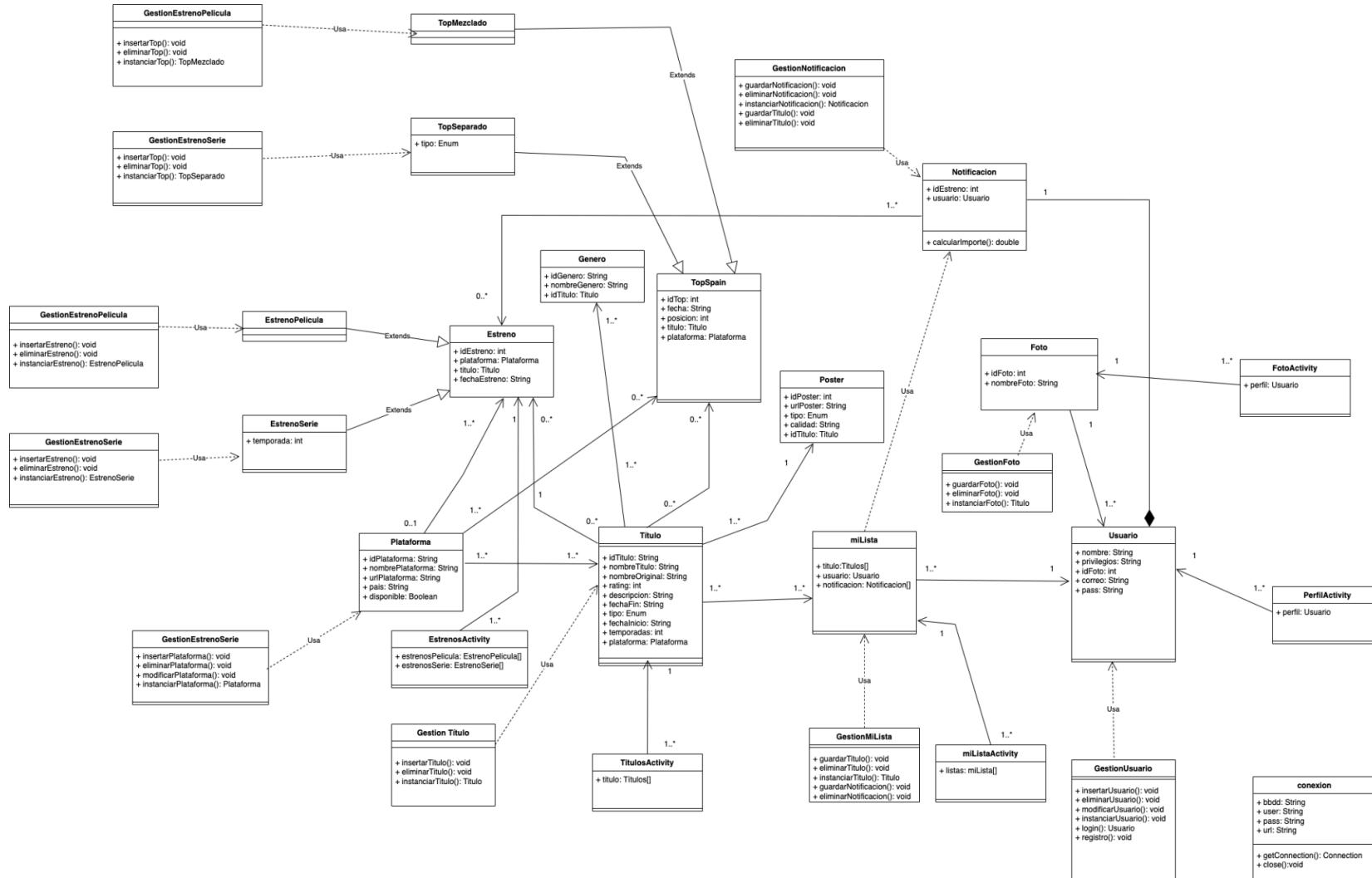
Unión (Pelis/Series y Top 10)
<u>Id Top</u> (PK)
<u>Id Peli/Serie</u> (PK)
Posición

Unión (Plataforma y Estreno)
<u>Id Estreno</u> (PK)
<u>Id Plataforma</u> (PK)

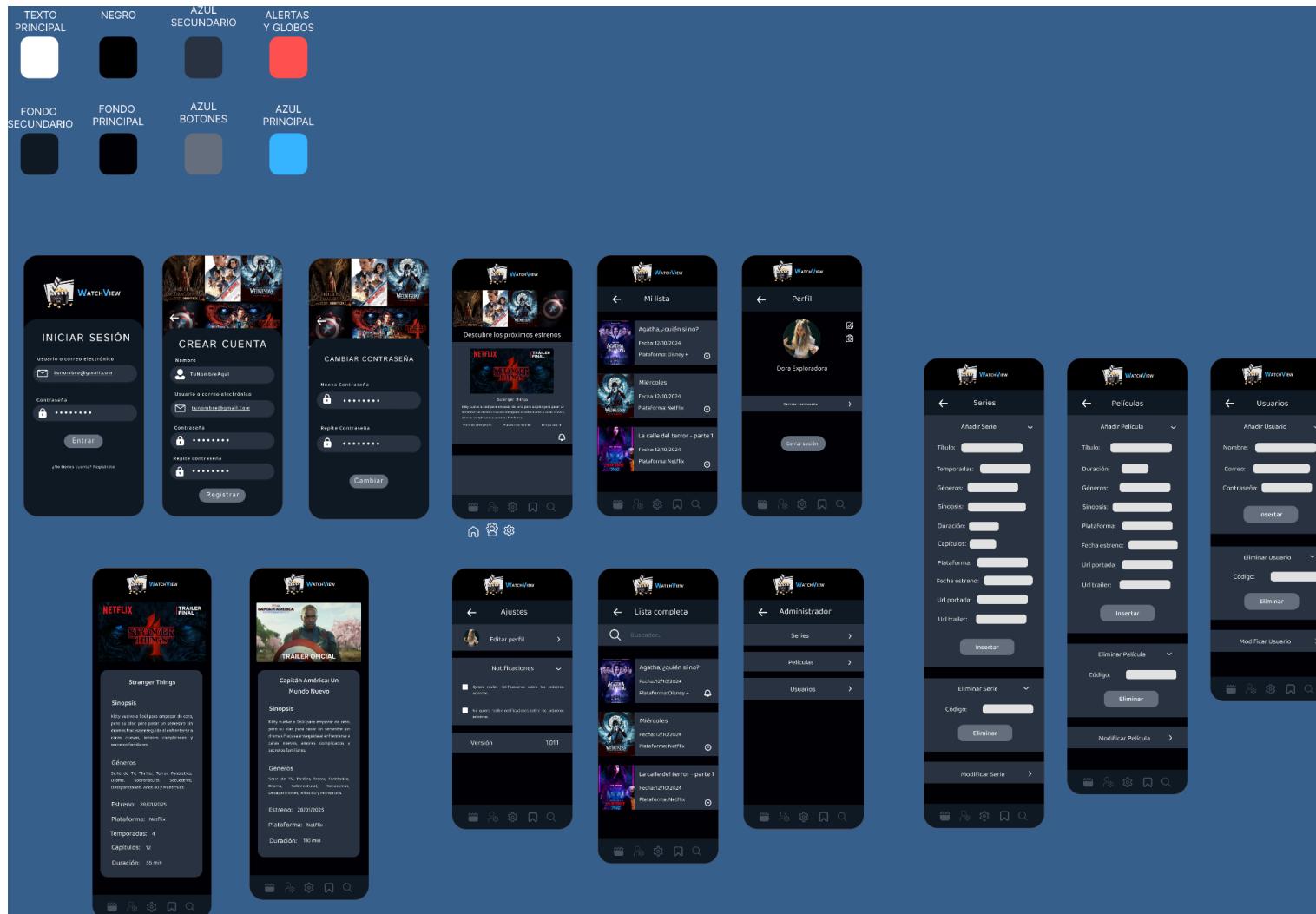
Unión (Pelis/Series y Estreno)
<u>Id Estreno</u> (PK)
<u>Id Peli/Serie</u> (PK)

Top 10
<u>Id Top</u> (PK)
Fecha

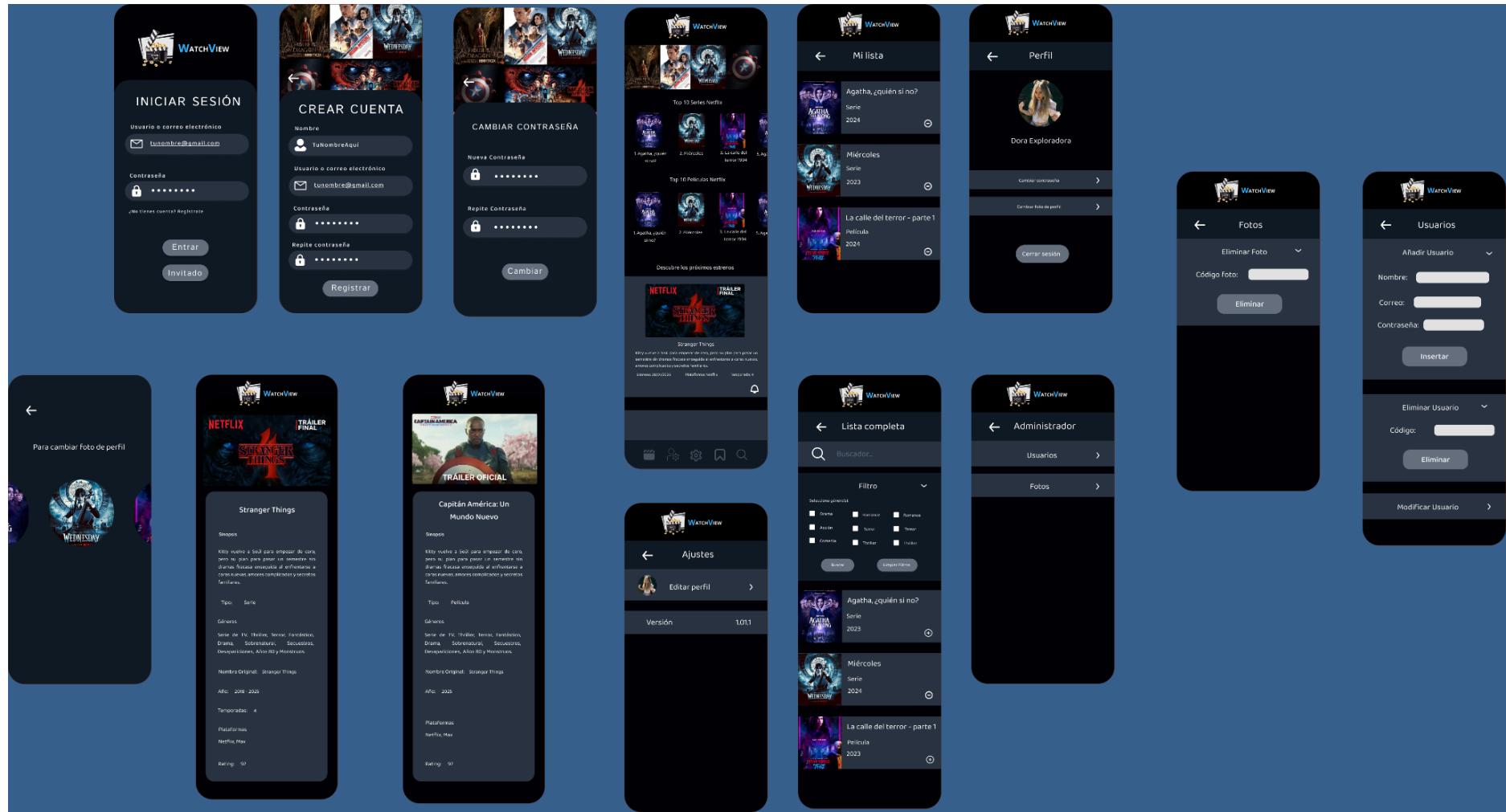
Diagrama de Clases



Mock Up Previo



Mock Up Final



Capturas de código (Obtención Póster Vertical en Adaptador Buscador)

```
// Cargar la imagen de póster si existe
val posterVertical = item.posters
    .filter { it.tipo == "vertical" }
    .first() // primero de la lista

if (posterVertical != null) {
    Glide.with(holder.imagen.context)
        .load(posterVertical.urlPoster)
        .apply(RequestOptions().centerCrop()) // caso normal: se recorta
        .thumbnail(
            Glide.with(holder.imagen.context)
                .load(posterVertical.urlPoster)
                .apply(RequestOptions().fitCenter()) // fallback visual
        )
        .into(holder.imagen)

} else {
    holder.imagen.setImageResource(R.drawable.miercoles) // Default image if no poster
}
```

Capturas de código (Aregar título a lista y variar imagen de agregar y quitar)

```
// Configurar el ícono dependiendo de si el título está agregado
if (isAdded) {
    holder.imagenAdd.setImageResource(R.drawable.quitar)
} else {
    holder.imagenAdd.setImageResource(R.drawable.agregar)
}

// Acción para agregar o quitar el título de la base de datos
holder.imagenAdd.setOnClickListener {
    if (isAdded) {
        // Eliminar título de la base de datos y cambiar imagen a agregar
        db.eliminarTituloDeLista(usuarioCorreo, idTitulo)
        isAdded = false // Cambiar el estado local
        holder.imagenAdd.setImageResource(R.drawable.agregar)
    } else {
        // Agregar título a la base de datos y cambiar imagen a quitar
        db.agregarTituloALista(usuarioCorreo, idTitulo)
        isAdded = true // Cambiar el estado local
        holder.imagenAdd.setImageResource(R.drawable.quitar)
    }

    // Notificar que el estado del item ha cambiado
    notifyDataSetChanged(position)
}
```

Capturas de código (Obtención foto de perfil de tipo drawable y visualización en Glide)

```
val fotoString= resources.getIdentifier(Usuario.fotoPerfil, defType: "drawable", requireContext().packageName)

Glide.with(requireContext())
    .load(if (fotoString != 0) fotoString else R.drawable.perfil1) // por si no se encuentra
    .transform(CenterCrop(), RoundedCorners(roundingRadius: 200))
    .into(fotoPerfil)
```

Capturas de código (Obtención títulos llamando a la Api Streaming Availability con un contador de títulos)

```
CoroutineScope(Dispatchers.IO).launch {
    val client = OkHttpClient()
    val apiKey = context.getString(R.string.api_key)
    var hasMore = true
    var cursor: String? = null
    var totalGuardados = 0
    val maxTitulos = 20

    while (hasMore && totalGuardados < maxTitulos) {
        val urlBuilder = StringBuilder( str: "https://streaming-availability.p.rapidapi.com/shows/search/filters")
        urlBuilder.append("?country=es&series_granularity=show&order_direction=asc&output_language=es&catalogs=netflix")
        if (cursor != null) urlBuilder.append("&show_cursor=${Uri.encode(cursor)}")

        val request = Request.Builder()
            .url(urlBuilder.toString())
            .get()
            .addHeader( name: "x-rapidapi-host", value: "streaming-availability.p.rapidapi.com")
            .addHeader( name: "x-rapidapi-key", apiKey)
            .build()

        try {
            val response = client.newCall(request).execute()
            if (response.isSuccessful) {
                val json = response.body?.string()
                val (next, more, guardadosEnEstaPagina) = leerJSONTitulos2(context, json, maxTitulos: maxTitulos - totalGuardados)
                totalGuardados += guardadosEnEstaPagina
                cursor = next
                hasMore = more
            } else {
                hasMore = false
            }
        }
    }
}
```

Capturas de código (Worker para actualizar listas donde aparecen los nuevos estrenos)

```
class EstrenosWorker(
    context: Context,
    workerParams: WorkerParameters
) : Worker(context, workerParams) {

    override fun doWork(): Result {
        val db = BBDD(applicationContext)

        val estrenos = db.listarTitulosPorPlataformaConEstreno( idPlataforma: "netflix")
        Log.d( tag: "EstrenosWorker", msg: "Se actualizaron ${estrenos.size} estrenos")

        return Result.success()
    }
}
```

Capturas de código (Worker para actualizar las listas y notificar estrenos)

```
class RevisarEstrenosWorker(appContext: Context, workerParams: WorkerParameters) : Worker(appContext, workerParams) {
    override fun doWork(): Result {
        val db = BBDD(applicationContext)
        db.revisarEstrenosYActualizarListas2(applicationContext)
        return Result.success()
    }
}
```

Capturas de código (Función que actualiza listas, notifica estrenos y lista los estrenos por plataforma)

```
fun revisarEstrenosYActualizarListas2(context: Context) {

    val formatoFecha = SimpleDateFormat(pattern: "yyyy-MM-dd", Locale.getDefault())
    val fechaHoyDate = formatoFecha.parse(formatoFecha.format(java.util.Date()))
    val estrenos = obtenerEstrenos() // List<Estreno>

    this.writableDatabase.use { db -> // 'db' es la base de datos dentro de este bloque 'use'
        for (estreno in estrenos) {
            val idEstreno = estreno.idEstreno
            val idTitulo = estreno.idTitulo
            val fechaEstrenoDate = formatoFecha.parse(estreno.fechaEstreno)

            if (fechaEstrenoDate != null && fechaHoyDate != null) {
                if (fechaEstrenoDate == fechaHoyDate) {
                    obtenerNombreTitulo(idTitulo)?.let {
                        mostrarNotificacionDeEstreno2(context, it)
                        actualizarTodosLosTitulosVistos()
                        listarTitulosPorPlataformaConEstreno( idPlataforma: "netflix")
                    }
                }
            }

            if (fechaEstrenoDate <= fechaHoyDate) {
                val cursor = db.rawQuery(
                    sql: "SELECT correo FROM Estreno_Usuario WHERE idEstreno = ?",
                    arrayOf(idEstreno.toString())
                )
            }
        }
    }
}
```

Capturas de código (Función para realizar la notificación visual del estreno con el logo de la App y mensaje personalizado)

```
fun mostrarNotificacionDeEstreno2(context: Context, titulo: String) {
    val channelId = "estrenos_channel"
    val channelName = "Notificaciones de Estrenos"
    val notificationId = titulo.hashCode() // Para evitar duplicados

    val notificationManager = context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

    // Crear canal (obligatorio en Android 8+)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val channel = NotificationChannel(
            channelId,
            channelName,
            NotificationManager.IMPORTANCE_DEFAULT
        ).apply {
            description = "Notifica cuando se estrenan títulos"
        }
        notificationManager.createNotificationChannel(channel)
        Log.d("MostrarNotificación", "Canal de notificación creado con éxito")
    }

    // Cargar el ícono de la app desde los recursos
    val largeIcon = BitmapFactory.decodeResource(context.resources, R.drawable.logoapp2) // Reemplaza con tu ícono

    val builder = NotificationCompat.Builder(context, channelId)
        .setSmallIcon(R.drawable.logoapp2) // Ícono pequeño de la notificación (también puedes cambiarlo)
        .setContentTitle("¡Estreno disponible!")
        .setContentText("Ya está disponible: $titulo")
        .setLargeIcon(largeIcon) // Establece el ícono grande para la notificación
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setAutoCancel(true)

    // Mostrar la notificación
    notificationManager.notify(notificationId, builder.build())
}
```

Capturas de código (Funciones para llamar al worker, establecer una hora de inicio y pedir permiso para las notificaciones)

```
class LoginActivity : AppCompatActivity() {

    private fun programarWorkerDeEstrenos(context: Context) {
        val initialDelay = calcularDelayHastaProxima12PM()

        val request = PeriodicWorkRequestBuilder<RevisarEstrenosWorker>(
            repeatInterval: 1, TimeUnit.DAYS)
            .setInitialDelay(initialDelay, TimeUnit.MILLISECONDS)
            .build()

        WorkManager.getInstance(context).enqueueUniquePeriodicWork(
            uniqueWorkName: "RevisarEstrenosDiario",
            ExistingPeriodicWorkPolicy.KEEP,
            request
        )
    }

    private fun calcularDelayHastaProxima12PM(): Long {
        val ahora = Calendar.getInstance()
        val proxima3AM = Calendar.getInstance().apply {
            set(Calendar.HOUR_OF_DAY, 0)
            set(Calendar.MINUTE, 1)
            set(Calendar.SECOND, 0)
            set(Calendar.MILLISECOND, 0)

            if (before(ahora)) {
                add(Calendar.DAY_OF_MONTH, 1)
            }
        }
        return proxima3AM.timeInMillis - ahora.timeInMillis
    }

    private fun solicitarPermisoNotificaciones() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            val permiso = Manifest.permission.POST_NOTIFICATIONS
            if (ContextCompat.checkSelfPermission(context, permiso) != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(activity, arrayOf(permiso), requestCode: 1001)
            }
        }
    }
}
```