Ilustración 1 Icono de la App

<- MEMORIA DEL PROYECTO-> FINAL DE CURSO

Anizeno - Android Compose App

IES San José Cortegana

Curso 2024-25

Autor/es	Tutor/es
Alejandro Soria Torres	José Antonio Ortega Hidalgo

ÍNDICE

RESUMEN

Anizeno es una aplicación móvil desarrollada en Android con Jetpack Compose, la idea inicial era construir una aplicación en local usando Compose para explorar series con un tema visual agradable, gestionar listas personalizadas además de añadir reseñas a las series. La aplicación integra la API de Jikan para obtener la información y utiliza tecnologías como Firebase Authentication y Room Database. Su diseño sigue principios de arquitectura MVVM y Material Design 3, proporcionando una experiencia de mantenimiento limpia y una UI moderna.

PALABRAS CLAVE

Android: Sistema operativo móvil desarrollado por Google.

Jetpack Compose: Framework moderno de Android para construir interfaces de usuario declarativas.

Firebase: Plataforma de Google para servicios en la nube: autenticación, base de datos en tiempo real, almacenamiento, hosting y más.

Room: Librería oficial de Android (parte de Jetpack) que facilita trabajar con **SQLite** mediante clases y anotaciones.

MVVM: Arquitectura recomendada por Google, se diferencia las partes de model (datos y repositorios), view (pantallas) y viewmodel (lógica, estados y comunicación entre vista y datos)

Anime: Estilo de animación japonesa

Jikan API: Servicio REST público que permite obtener información de anime desde MyAnimeList.

Kotlin: Lenguaje oficial de Android.

UI: La parte visible con la que interactúa el usuario: botones, textos, imágenes, navegación…

Material Design 3: Guía visual moderna de Google: colores dinámicos, formas suaves, tipografías claras y componentes accesibles.

INTRODUCCIÓN

El consumo de contenido anime ha aumentado considerablemente durante la última década. Plataformas como MyAnimeList, AniList o Kitsu se han popularizado al permitir a los usuarios gestionar listas, valorar series y descubrir nuevos títulos y hacer seguimiento de que vemos.

Sin embargo, estas plataformas suelen requerir registro online y conexión permanente, lo que puede suponer barreras para ciertos usuarios. Además, muchos aficionados buscan una herramienta minimalista, rápida y centrada únicamente en la gestión personal de su colección.

Anizeno nace para cubrir esa necesidad: una aplicación móvil de uso local, ligera, intuitiva y moderna. Permite gestionar listas de anime, ver información actualizada a través de la API pública de Jikan, crear un perfil de usuario y añadir reseñas locales. Todo ello desarrollando una arquitectura limpia, escalable y profesional.

ESTADO DEL ARTE

2.1 Plataformas existentes

MyAnimeList: la más conocida, con millones de usuarios, pero requiere registro y presenta una interfaz compleja.

AniList: Muy completa, con API potente pero orientada al uso online. **Kitsu:** Red social centrada en anime y manga, pero excesivamente orientada a interacción entre usuarios.

2.2 Oportunidades detectadas

- Falta una app sencilla y totalmente local.
- Interfaces complejas en apps existentes.
- Apps oficiales no presentan listas personalizadas cortas.
- Ninguna ofrece reseñas locales offline.

2.3 Conclusión

Existe un hueco para una aplicación móvil intuitiva, offline-first y enfocada exclusivamente a la gestión personal del anime. ANIZENO cubre este espacio.

ESTUDIO DE VIABILIDAD

3.1 DAFO

Fortalezas

- Aplicación ligera y rápida
- No necesita conexión constante
- Jetpack Compose permite alta productividad
- Arquitectura MVVM clara y mantenible

Debilidades

- Base de datos local limitada
- No hay sincronización en la nube
- Depende de API de terceros

Oportunidades

- Crecimiento global del consumo de anime
- Posible exportación/importación futura
- Potenciales usuarios minimalistas

Amenazas

- Cambios en la API de Jikan
- Competencia con grandes plataformas y APIs
- Limitaciones de Firebase gratuito

3.2 Estudio de mercado

Como este mercado está en pleno crecimiento y aunque contamos con plataformas complejas con muchos usuarios como MyAnimeList, para usuarios más minimalistas y que quieran un almacenamiento local es una app que puede ofrecer una experiencia limpia y sencilla además al no estar orientada a servicios web o conexión permanente a internet(offlline-first) o incluso el gran objetivo de las grandes apps con las que compite que es el conectar a sus usuarios y remarcar mucho la comunidad antes que la exp personal.

¿Es necesario nuestro producto?, diferenciándose de las grandes plataformas **sí**, habrá usuarios que busquen una opción orientada al usuario local y minimalista para llevar un seguimiento de sus series de anime favoritas, con acceso a internet limitado, sin necesitar estar conectado excesivamente con las comunidades de fans. Al final es una opción muy útil enfocado en ser privado, ligero y rápido.

¿Tiene hueco en el mercado?, si lo que se busca es una app privada, ligera y visualmente limpia muchos usuarios tienen esta alternativa además sin depender tanto de la red.

Viabilidad temporal

El proyecto ha sido desarrollado como proyecto final del curso, con una complejidad media dentro de la arquitectura MVVM y Kotlin Compose explorada, para un periodo de 11-12 semanas ha sido viable para un solo desarrollador, se ha seguido aprox unas fases de desarrollo comunes:

- Primera etapa de análisis para determinar que se espera de la app teniendo en cuenta las necesidades y el tiempo disponible para implementarlo usando las tecnologías de Compose, Kotlin y una API, validando las herramientas, cursos y requisitos para el proyecto.
- Segunda etapa de primer modelo, resolver dependencias y permisos, explorar las APIs disponibles que cumplan los requisitos y esquematizado de los componentes de la app y la UI (diagrama de flujo inicial),

Primera estructura de arquitectura para la app en Android Studio según lo recomendado para Compose.

- Tercera etapa desarrollo más interno (backend), implementar Firebase para la autentificación, conexión y modelo de datos según el formato que entrega la API de Jikan, definición de las vistas y sus modelos de datos para la API y la base de datos RoomDB
- Cuarta etapa donde se crean el aspecto visual de la app haciendo uso del nuevo Material 3 y Compose, conexión de las vistas con Room y la API y organización de la navegación dinámica con guardado de estados propia de Compose
- Quinta etapa de pruebas técnicas sobretodo orientado a la probar la nav. fluida de la app, posibles errores de recursos y ejecución relacionados con el servicio API, robustez de la DB y guardado de estados durante el cambio de ventanas.
- Etapa final para crear la documentación y el manual para el usuario, documentando el desarrollo y explicando todo el contenido de la app acompañado de imágenes y diagramas necesarios y por último presentación para la próxima exposición del proyecto.

Etapas	1	2	3	4	5	6	7	8	9	10	11	12
1° Análisis												
2° Modelado												
3° Desarrollo												
4° UI												
5° Testing												
6° Documentar												

Diagrama de Gantt

ANÁLISIS DE REQUISITOS

Se recogen las características de que debe tener la funcionalidad de la app, funcionales y no funcionales, es decir que debe de hacer la aplicación y como debe comportarse, tanto como a nivel técnico y de objetivos mínimos.

Requisitos funcionales

RF1: El usuario debe poder registrarse e iniciar sesión.

RF2: El usuario puede añadir un anime a Favoritos, Vistos o Me Interesa.

RF3: El usuario puede escribir una reseña para un anime.

RF4: La app debe poder buscar animes usando la API de Jikan.

RF5: La app debe mostrar el detalle de un anime.

Requisitos no funcionales

RNF1 (Rendimiento): La app debe cargar pantallas en menos de 2 segundos.

RNF2 (Compatibilidad): Debe funcionar desde Android 9 (SDK 28) en adelante.

RNF3 (Usabilidad): La interfaz debe ser intuitiva y accesible.

RNF4 (Seguridad): Las contraseñas no se guardarán en texto plano.

RNF5 (Internacionalización): Debe soportar varios idiomas (ES, EN, FR...).

RNF6 (Mantenibilidad): El código debe seguir arquitectura MVVM.

Requisitos técnicos

- Uso de Kotlin Compose para Android.
- Persistencia mediante base de datos Room.
- API de terceros para comunicarse.
- Control de versiones Git y Github.
- Versión de Android mínima para funcionar (Android 9)

Diagrama de casos de uso

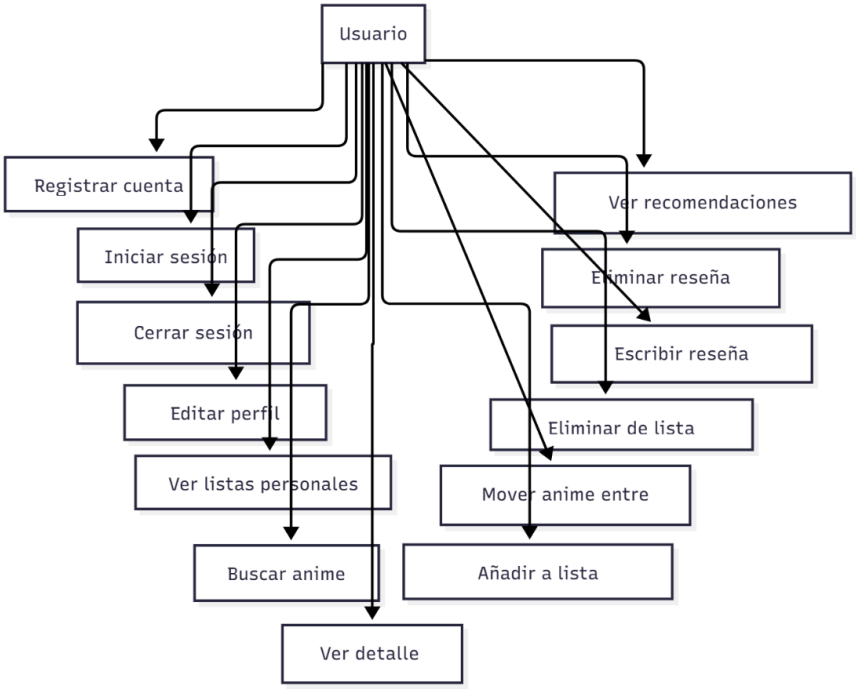

Ilustración 2 Diagrama de caso de uso completo

Caso de uso práctico

i. El usuario se registra en la app
ii. Explora las listas recomendadas en la pantalla principal
iii. Navega a los detalles de una serie de la lista
iv. Añade la serie a su lista de vistos
v. Escribe una reseña de la serie
vi. Elimina la reseña
vii. Cierra la app

DISEÑO

Para conectar las ideas y visualizar la estructura tanto de la base de datos como el funcionamiento de la app, se reúnen esquemas necesarios para comprender estos aspectos.

Diseño Conceptual Entidad-Relación

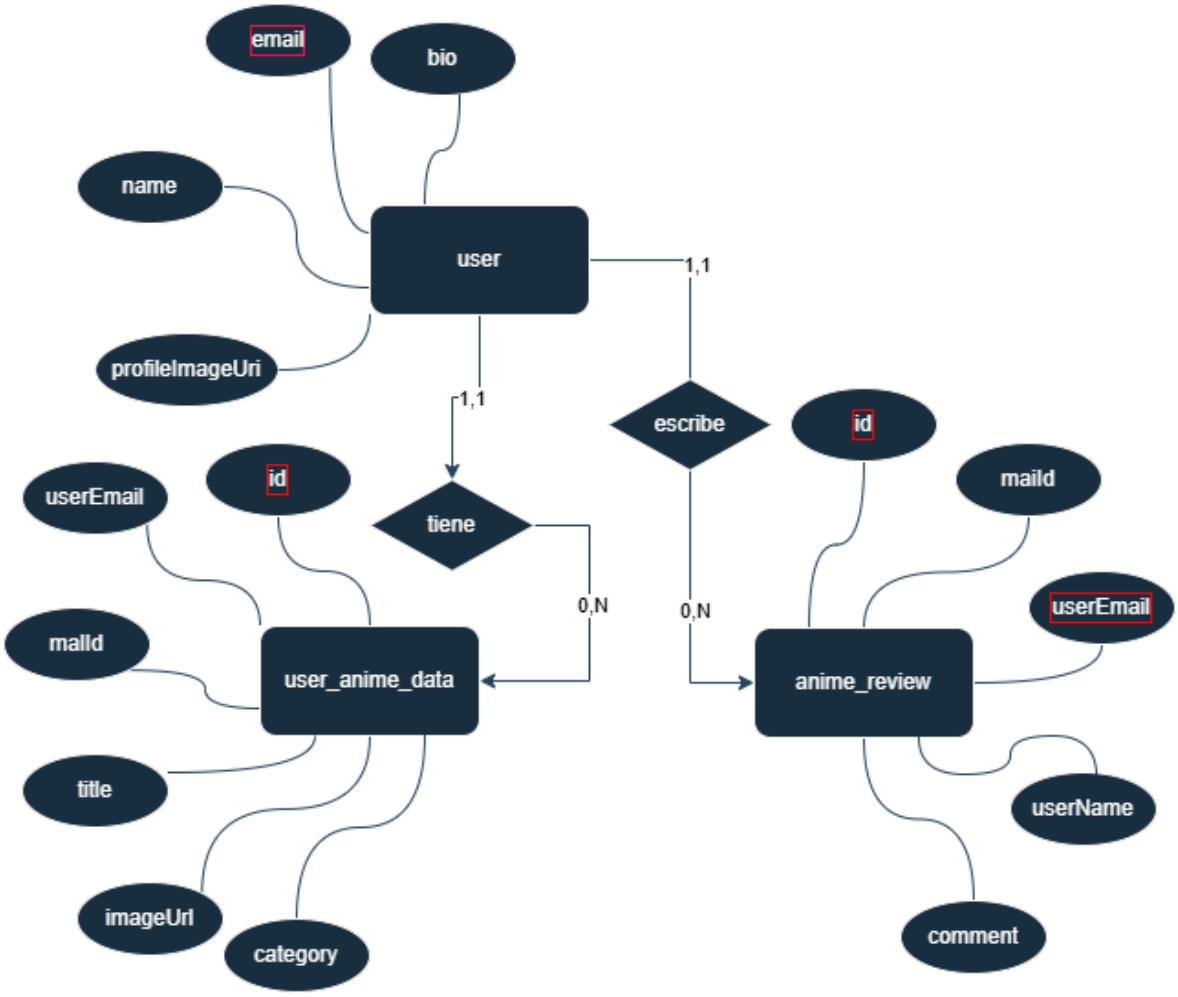

Ilustración 3 Diagrama Entidad-Relación conceptual

Descripción del modelo de datos

- El esquema recoge el modelo de datos necesarios para la base de datos de la aplicación y sus relaciones para almacenar los datos del usuario, sus series enlistadas y las reseñas del mismo.
- Cumple la 3° forma normal

La base de datos es guardada en el dispositivo donde se instala la app y es fácilmente removible cuando se borran los datos de app en Android, la librería Room (Object Relational Mapping u ORM) es la librería encargada de entenderse con una base de datos SQLite.

Sus tablas conceptuales son:

User: Guarda los datos del perfil del usuario, su nombre, imagen de perfil, y biografía, sobre todo.

Anime_review: Guarda los datos de la reseña vinculada a un usuario.

User_Anime_Data: Guarda los datos referentes a las series añadidas a las listas.

Sus relaciones son sencillas dado que:

- El usuario puede escribir 0 o varias reseñas aunque existe restricción de 1 por anime.
- El usuario tiene 0 o varias "user_anime_data" dado que está información es respectiva a la serie guardada en sus listas personales, con el objetivo de saber qué serie existe en que lista y de que usuario es.
- Los campos señalados en rojo son las respectivas claves primarias simples o compuestas.
- Se cumplen la 3° forma normal dado que cumple la 2° forma al no tener dependencias parciales ni existen dependencias transitivas.

Paso A tablas (diseño lógico relacional) - SQLite

```
CREATE TABLE USER (
    email TEXT PRIMARY KEY,
    name TEXT,
    bio TEXT,
    profileImageUri TEXT
);
```

Ilustración 4 Creación tabla Usuario

```
CREATE TABLE USER_ANIME_DATA (
    id INTEGER PRIMARY KEY auto_increment,
    userEmail TEXT NOT NULL,
    malId INTEGER NOT NULL,
    title TEXT NOT NULL,
    imageUrl TEXT NOT NULL,
    category TEXT NOT NULL,  -- fav, vista, me_interesa

    FOREIGN KEY (userEmail) REFERENCES USER(email),

    -- Evita duplicados: un usuario no puede tener dos veces el mismo anime en una lista
    UNIQUE(userEmail, malId, category)
);
```

Ilustración 5 Creación tabla Usuario-Anime

```
CREATE TABLE ANIME_REVIEW (
    id INTEGER PRIMARY KEY auto_increment,
    malId INTEGER NOT NULL,
    userEmail TEXT NOT NULL,
    userName TEXT NOT NULL,
    comment TEXT NOT NULL,

    FOREIGN KEY (userEmail) REFERENCES USER(email),

    -- Regla: solo una crítica por usuario y anime
    UNIQUE(malId, userEmail)
);
```

Ilustración 6 Creación de la reseña

El paso a tablas constituye la lógica de la base de datos al completo, que varía según el lenguaje SQLite o MySQL con los que se ha trabajado el proyecto, traduce los conceptos de relaciones con las claves foráneas y primarias, así como las restricciones como “UNIQUE”.

Diseño Físico (paso a tablas, optimizaciones) o diagrama MySQL

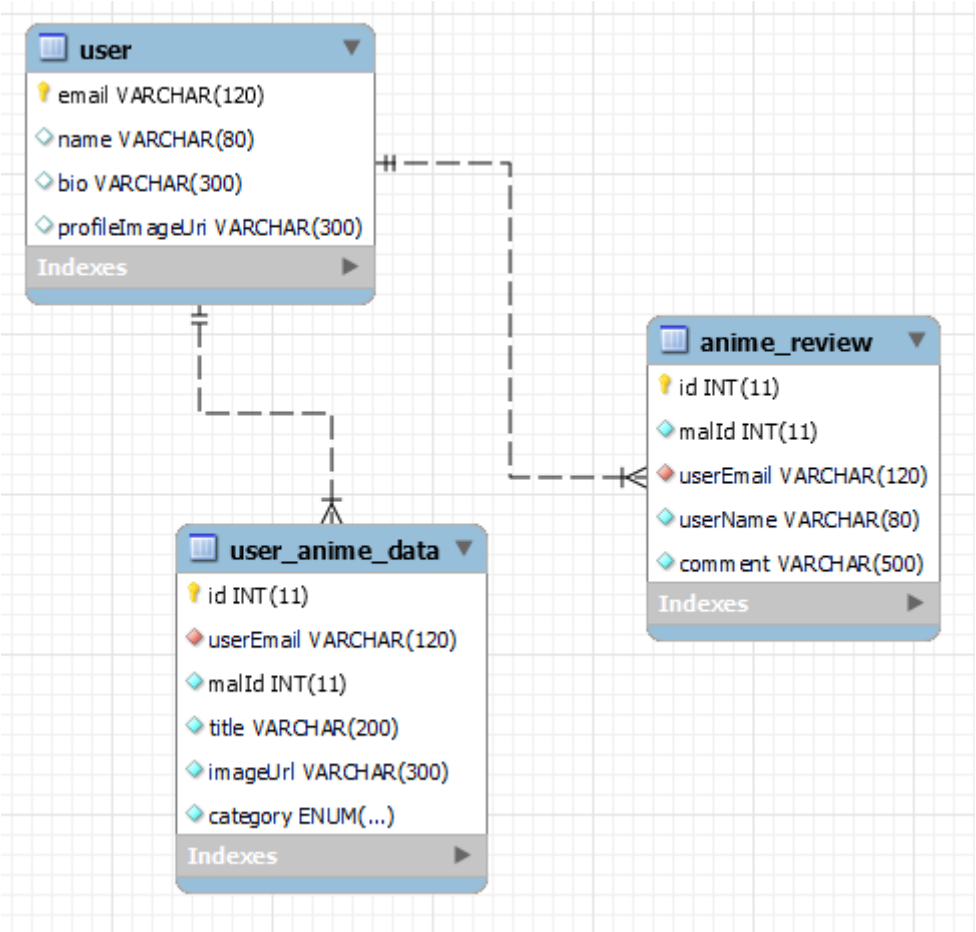

Ilustración 7 Diagrama MySQL

Después de crear la base de datos usando el diagrama físico, es decir, el código sql de creación que se ejecuta para SQLite, proveemos con ingeniería inversa el diagrama de tablas MySQL es este caso donde se usan lenguaje parecido pero diferente.

La base de datos que es almacenada en el dispositivo es modificada mediante el ORM Room, peticiones de consulta, insertar y eliminar que el usuario puede mandar desde la interfaz (UI), el famoso set de operaciones CRUD. La tabla refleja el tipo de dato que es cada campo de la tabla, así como su tamaño de información que son INT, VARCHAR y ENUM, instrucciones como FOREIGN KEY, PRIMARY KEY, NOT NULL y UNIQUE para garantizar entre otras que se creen las relaciones entre tablas, se apliquen restricciones para evitar duplicados o incongruencias en los datos.

El diseño de la DB para los requerimientos de la app se centra en ser sencillo teniendo en cuenta la idea del proyecto que consiste en manejar lo aprendido como Room, Kotlin, etc y trabajar usando el nuevo Jetpack Compose y Material 3 además de cara al tiempo disponible para realizarlo.

Se ha usado herramientas Como MySQL Workbench y Xampp para la base de datos además de un procesador de texto notepad++ para tipos los scripts sql.

Descripción de las tablas y campos.

User → Información básica del usuario

email (TEXT, PK): Identificador único del usuario.

n**ame** (TEXT): Nombre visible del usuario.

bio (TEXT): Biografía o descripción breve.

profileImageUri (TEXT): Ruta/URL de la foto de perfil.

User_Anime_Data → Registra los animes guardados por el usuario en sus listas personales

id (INTEGER, PK): Identificador único del registro.

userEmail (TEXT, FK): Usuario propietario del anime guardado.

malId (INTEGER): ID del anime en la API externa.

title (TEXT): Título del anime.

imageUrl (TEXT): Imagen del anime.

category (TEXT): Lista a la que pertenece (fav, vista, me_interesa).

Anime_Review → Registra los datos de la reseña del usuario

userEmail (TEXT, PK + FK): Usuario que escribe la reseña.

malId (INTEGER, PK): Anime al que corresponde la reseña.

userName (TEXT): Nombre del usuario en el momento de la reseña.

comment (TEXT): Texto de la crítica.

Orientación a objetos

Diagrama de clases

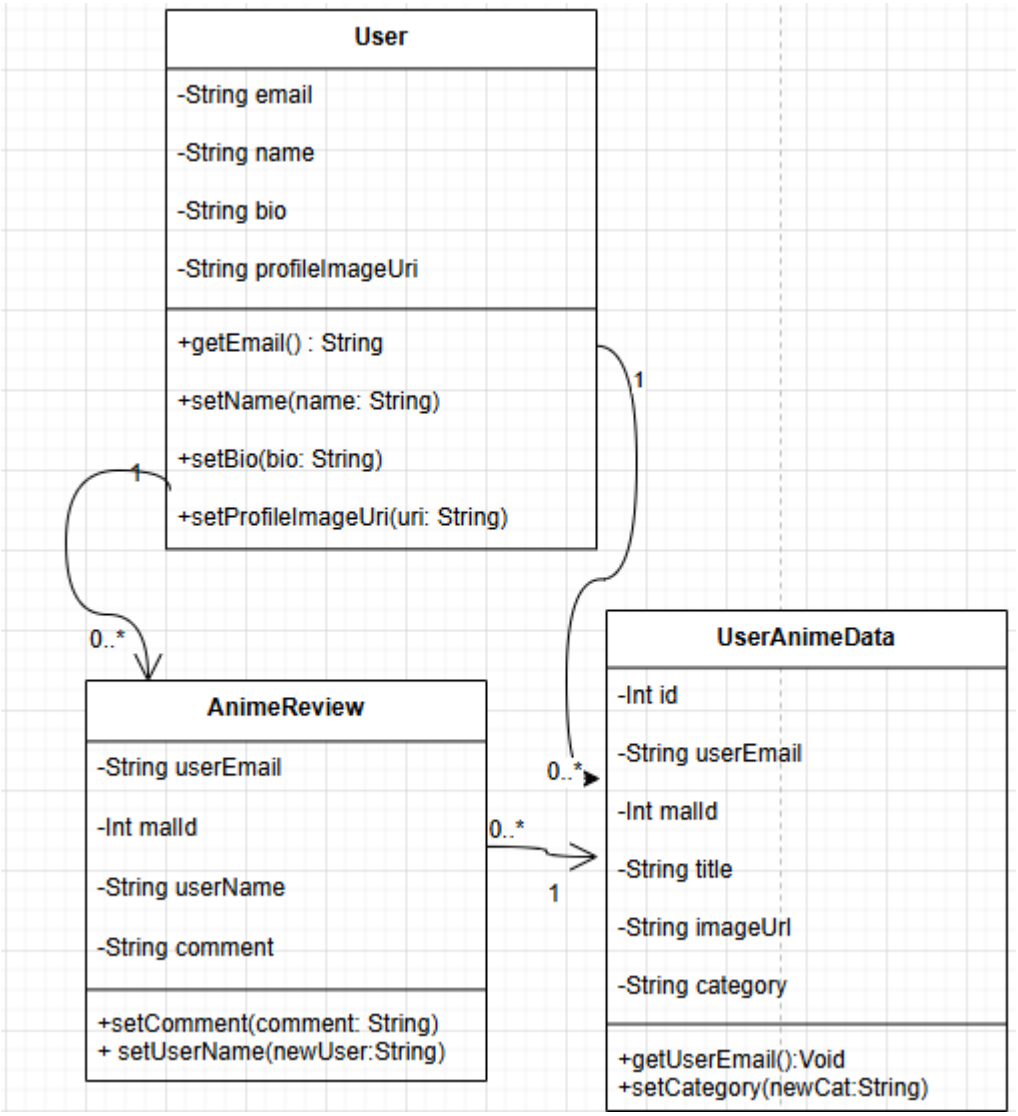

Ilustración 8 Diagrama Orientado a objetos

Se especifica sutilmente los atributos y métodos/funciones para cada clase.

Diagrama de secuencias

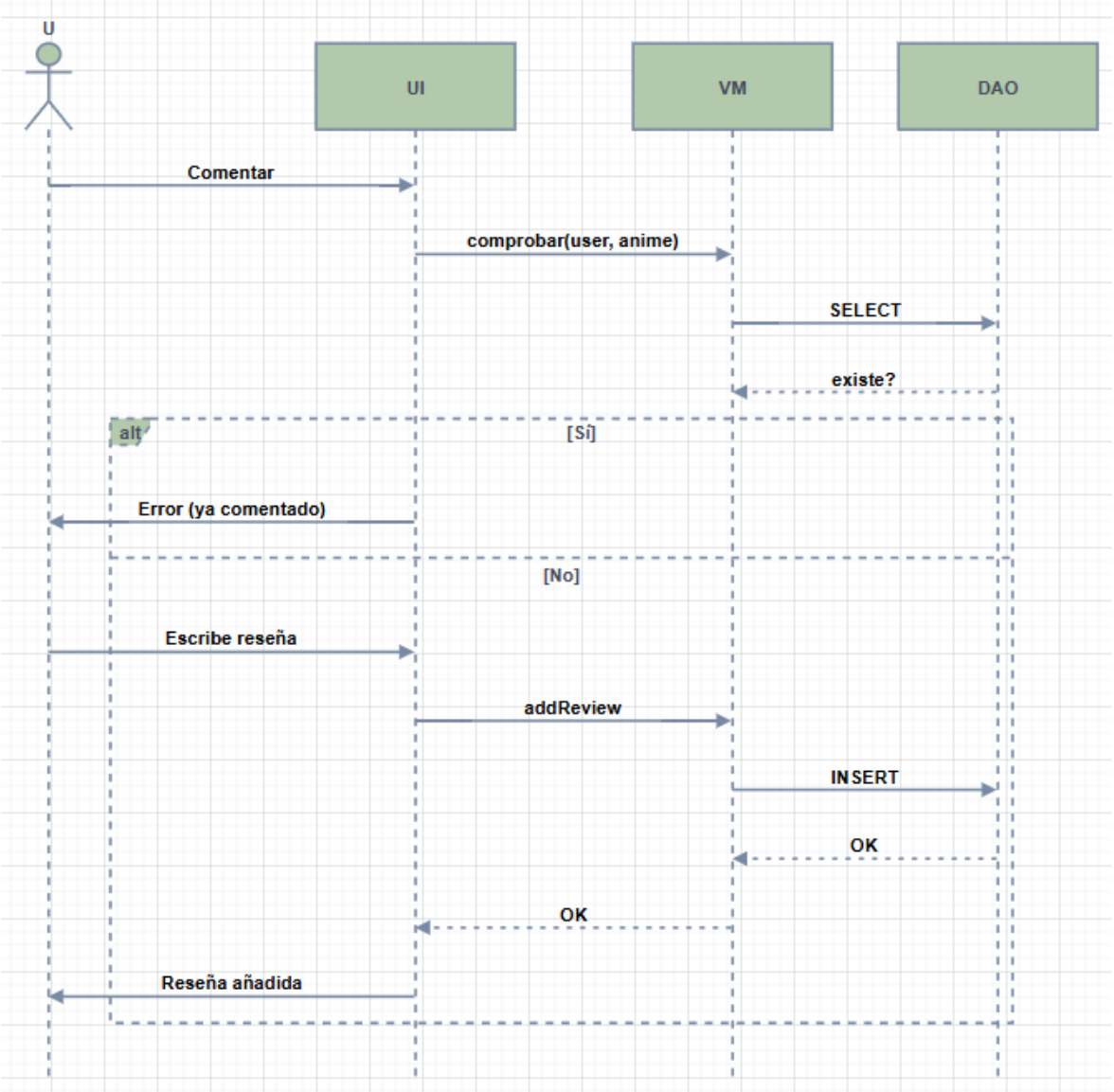

Ilustración 9 Diagrama de secuencia para escribir una reseña

Se describe la secuencia de escribir una reseña por el usuario, cuyos participantes:

interfaz → vista-modelo → repositorio → Room → SQLite

Proceso:

El usuario intenta comentar

Se comprueba si existe ya un comentario con ese usuario y serie

Si no existe comentario se inserta un nuevo comentario en la bd y se recibe una respuesta indicando al usuario que ha sido añadida la reseña, si no es alertado de que ya ha comentado y no se le permite volver a comentar mientras que no borre su reseña anterior.

Diagrama de actividad

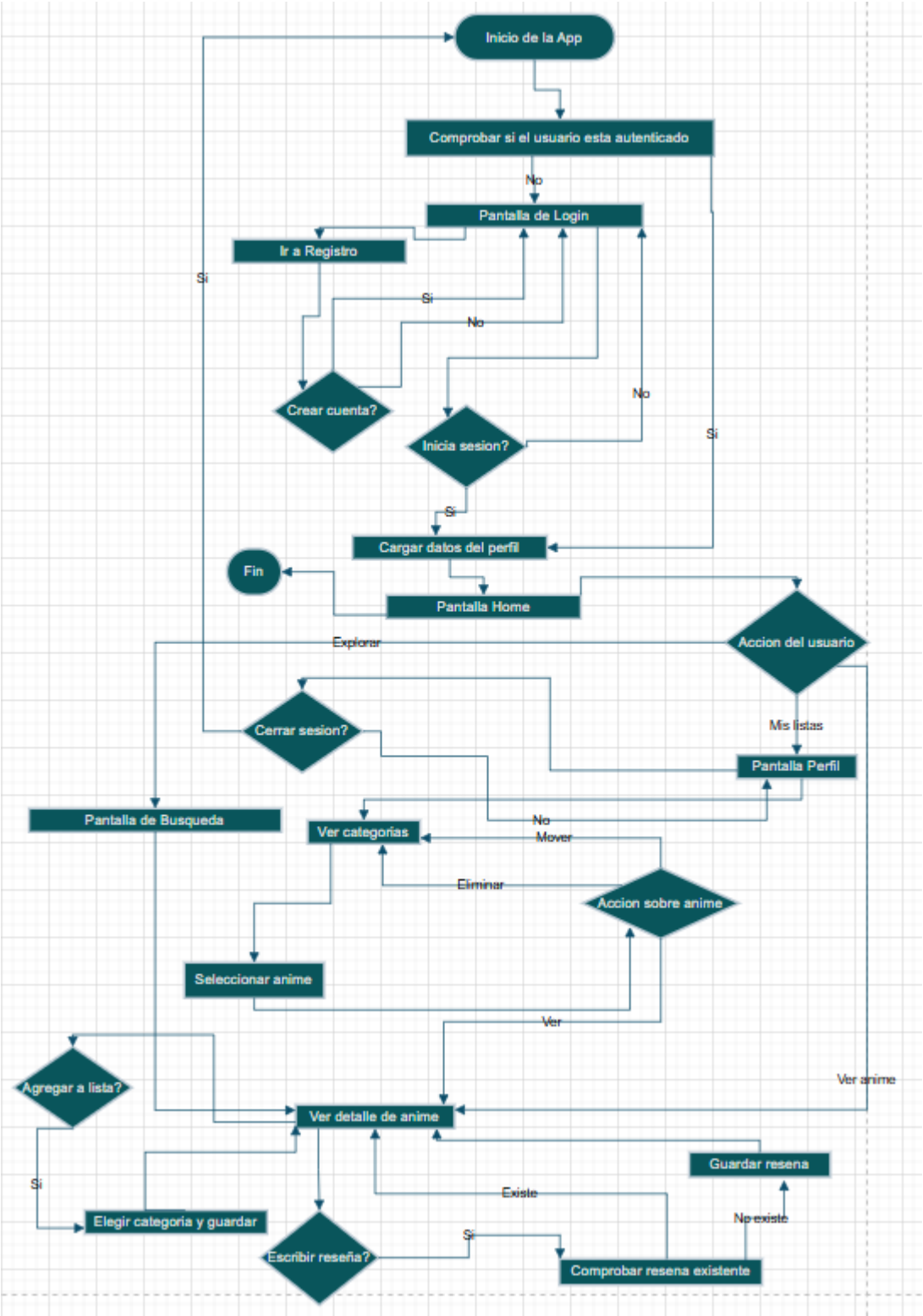

Ilustración 10 Diagrama de actividad general de la app

Este diagrama refleja toda la actividad posible durante el flujo de la app. La ejecución termina cuando el usuario cierra la app externamente o mediante la nav del dispositivo.

Diseño UX

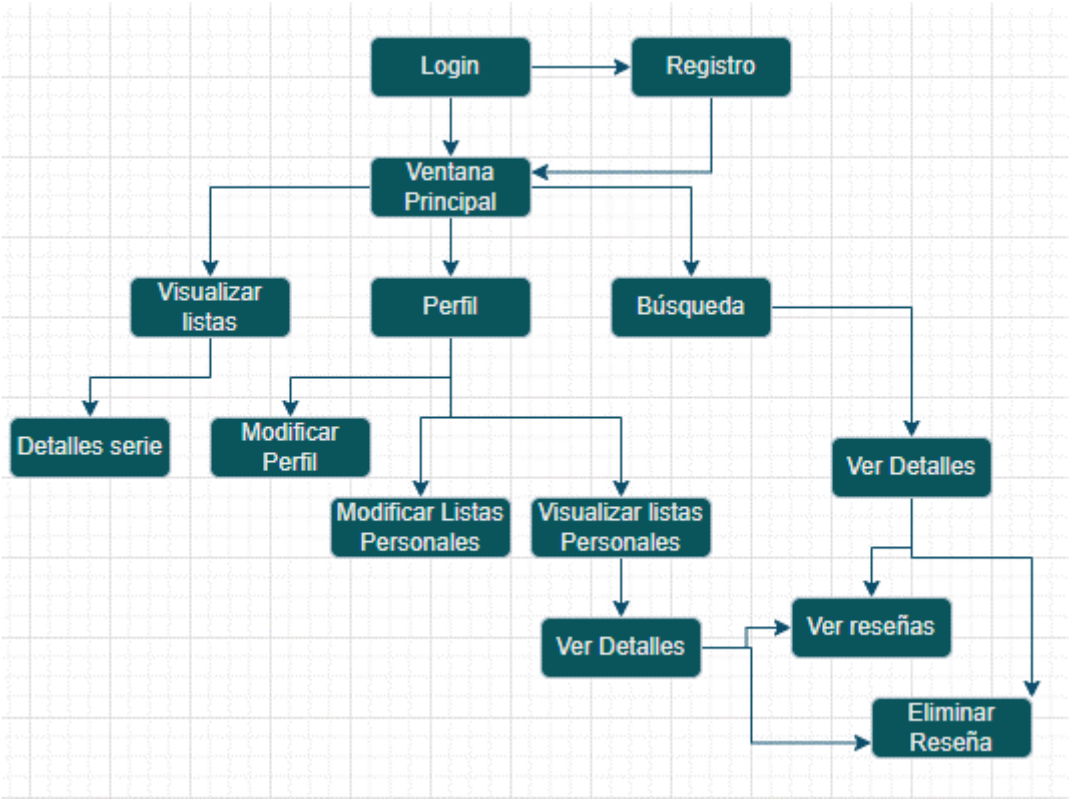

Ilustración 11 Diseño UX de Nav con Material 3 y NavHost

El diseño para la app se ha basado en la navegación con Navigation Compose con Rutas tipadas (seguras) mediante objetos serializables para las rutas, un mapa de rutas con NavHost que define los destinos: Login, Register, Home, Search, etc. La navegación garantiza seguridad y claridad, se usa además SavedStateHandle para enviar los objetos completos entre pantallas sin necesidad de serializar.

Mockups

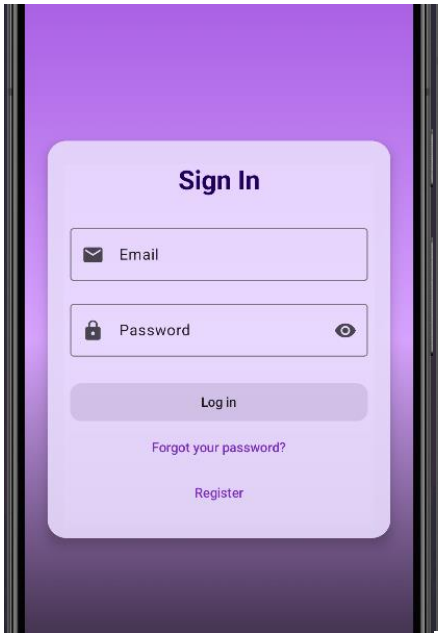

Ilustración 12 Pantalla Login

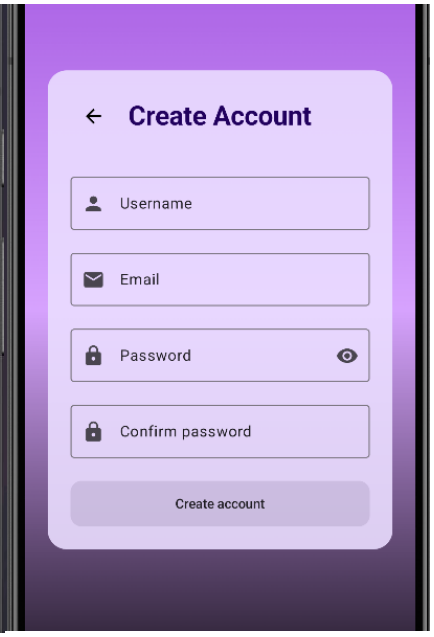

Ilustración 13 Pantalla de registro

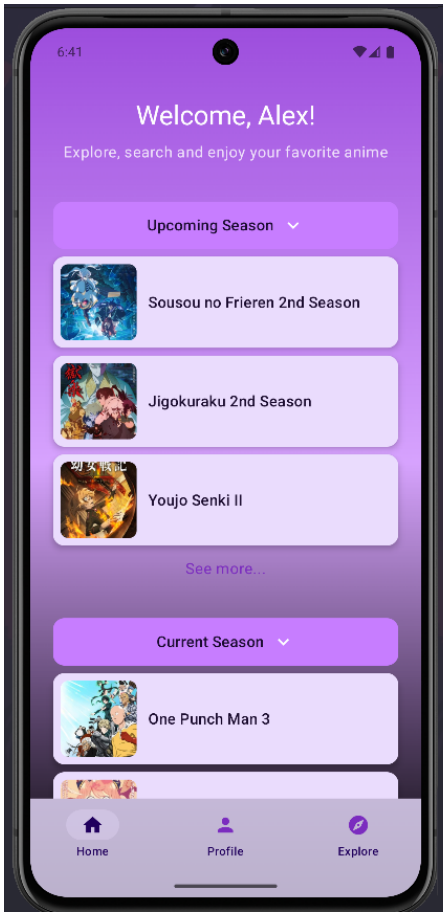

Ilustración 14 Pantalla Principal

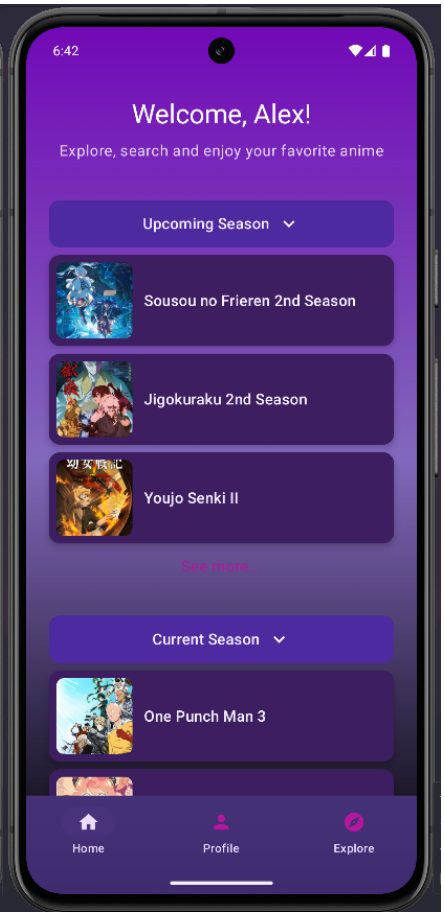

Ilustración 15 Pantalla Principal Tema Noche

Ilustración 16 Pantalla de Perfil *Ilustración 17 Pantalla de Búsqueda*

Estos diseños de ventana corresponden a todas las pantallas de la app basando la paleta de colores tanto como el modo día y el modo noche en todos violetas-rosas, aspectos más importantes son la navegación mediante botones, campos de texto redondeados, elementos colgantes y el énfasis cuando es necesario para destacar elementos principales.

CODIFICACIÓN

El desarrollo ha estado en manos del entorno de **Android Studio vanilla** ya que maneja sin añadidos Kotlin/Java siguiendo el aprendizaje obtenido para realizar el proyecto en **Jetpack Compose, Material 3 con Room y uso de API**. Añadir además una arquitectura limpia MVVM combinado con Clean Architecture.

La app sigue una estructura sencilla con la **separación de la lógica, las vistas y los modelos de datos**. Los componentes de la aplicación se separan en view, domain y data según las bases habituales de MVVM en el desarrollo Android con Kotlin, también los archivos del proyecto base para las

dependencias, el archivo **manifest** que especifica los componentes esenciales, declarar la actividad principal que se ejecuta antes de todo, los permisos esenciales para Android.

DOCUMENTACIÓN

La documentación del proyecto se simplifica en :

Documentación del código (Descripción del clases)

```
/**
 * LoginScreen muestra la pantalla de inicio de sesión.
 * Permite al usuario ingresar su correo electrónico y contraseña
 *
 * @param onLoginClick Callback que se llama cuando el usuario hac
 * @param onRegisterClick Callback que se llama cuando el usuario
 * @param onForgotPasswordClick Callback que se llama cuando el us
 * @param onBackClick Callback que se llama cuando el usuario hace
 *
 */
```

Memoria del proyecto (este mismo doc. donde se recoge todo los especificado para el proyecto)

Manual de usuario (instrucciones ilustradas de cómo usar adecuadamente la app y sus componentes)

DESPLIEGUE

La aplicación está lista para el despliegue, probadas las características de Auth, base de datos y comunicación con la API está lista para ser usada en dispositivos Android con los requisitos mínimos de versión, acceso a internet mínimo, etc.

Para su lanzamiento la app no necesita acceso a ninguna base de datos remota, usa Firebase ya configurado para funcionar para la autentificación y los permisos que debe tener la app están definidos. Como es común se usaría la tienda de Google llamada PlayStore para poner la app al alcance de los usuarios.

HERRAMIENTAS DE APOYO

Se ha contado con el entorno de desarrollo de **Android Studio,** lenguajes y librerías de Jetpack Compose, Room, Material 3 y Firebase además del servicio de la API, editores de texto, creación de diagramas con Draw.io, MySQL con Xampp para la base de datos y Word para la creación de la doc. Por último, GitHub y Github Desktop para el control de versiones y la visibilidad del proyecto en internet.

CONTROL DE VERSIONES

El control de versiones se ha basado en el uso de las herramientas de GitHub, la sincronización de los cambios del proyecto y buenas practicas comentando dentro del código.

SISTEMA DE INTEGRACIÓN CONTINUA

No se ha usado la integración continua en el proyecto hasta el momento, las releases actuales se verifican y compilan manualmente el por el programador y se lanzan en la plataforma de Github o en futuro lanzamiento en la tienda de Google. Para la posible implementación de este sistema se usará Github actions para publicar las modificaciones automáticamente en Github y que los usuarios tengan la última versión disponible o mediante la conexión con Google Cloud y Google Play para publicar versiones automáticamente pasando los test y verificaciones pertinentes.

GESTIÓN DE PRUEBAS

Aunque no se ha usado sistemas de testing como JUnit, mediante las herramientas de Android Studio (entre ellas las preview de UI) y verificando las respuestas de la API mediante Postman y web, se ha probado que la Auth, las respuestas de la API y la UI y funciones de la app no terminan en errores que el usuario no sepa interpretar o no sean solucionables.

CONCLUSIONES

La idea principal del proyecto era hacer una versión propia, una alternativa que a futuro pueda ser una buena opción compitiendo con las grandes plataformas para Android como MAL, implementar el desarrollo Android con Kotlin y Jetpack Compose que está en auge, los nuevos estilos visuales con Material 3 y abordar lo aprendido en el curso.

Siendo una versión inicial aún se pretende que sea ligera y no tan dependiente de internet aún las mejoras y expansiones de la app aumentaran el uso de la red al contar con más información para las series y nuevas funciones.

¿Que ha supuesto para mí? Al ser muy nuevo en el desarrollo Android he enfrentado la gran problemática que viene del continuo cambio de versiones, actualizaciones y nuevas librerías y tecnologías que suceden continuamente, el posible fallo de una versión u otra, que haya que tener en cuenta su uso para futuro, para que versiones sigue siendo compatible mi app, etc. El uso de una navegación moderna y fluida en Compose, el uso de estados para conservar la información en mi app y la comunicación entre las diferentes pantallas con ello. Con todos estos detalles que pueden ser un dolor de cabeza para el desarrollo incluso a pequeña escala como este, me aporta mi primera exp y aprendido lo que puede suponer un desarrollo más avanzado en una empresa y sus etapas.

¿Recorrido de la app? Además de las propuestas evolutivas que se tratan en el siguiente punto, la app continuará con un diseño estético claro, usa una base de datos en principio local y conoce la API para en un futuro mostrar más datos de las series y la autentificación ya es bastante fiable.

PROPUESTAS EVOLUTIVAS

Respetando la idea inicial de tener una app ligera con un diseño propio e implementando Compose con Material 3 y Room las partes claras de mejora son:

- Implementar una base de datos remota
- Expandir los detalles de las series usando la API
- Búsquedas por categorías, actores de doblaje, estudios, etc.
- Añadir amigos y chat para conectar a los usuarios y compartir info.
- Mejora en el visionado con la posibilidad de crear tus listas propias y hacer seguimiento de los capítulos.
- Distintos temas personalizados siguiendo Material 3 y el diseño propio de la app.

BIBLIOGRAFÍA

Draw.io. (n.d.). *Diagrams.net*. https://www.diagrams.net/

Google Firebase Documentation https://firebase.google.com/docs/android/setup?hl=es-419

JetPack Compose Developers

https://developer.android.com/develop/ui/compose/navigation?hl=es-419

Google. (n.d.). *Material Design 3*. https://m3.material.io/

GitHub. (n.d.). *GitHub Docs*. https://docs.github.com/

Navigation 3 Jetpack Compose curso https://www.youtube.com/watch?v=YCL1FfFaxwc

Recursos de la API Jikan https://docs.api.jikan.moe/

Recursos sobre JetPack Compose https://www.appcademy.dev/