



Grado Superior en Desarrollo de Aplicaciones Multiplataforma (2º Vespertino)

C.P.I.F.P Los Enlaces

Tutor: Rufino Esteban

08/01/2024

Proyecto de Fin de Ciclo de Desarrollo de Aplicaciones Multiplataforma



Índice de contenido

1. Descripción del proyecto	4
1.1 Contexto del proyecto	4
1.1.1 Ámbito y entorno	4
1.1.2 Análisis de la realidad	5
1.1.3 Solución y justificación de la solución propuesta	6
1.1.4 Destinatarios de la aplicación	7
1.2 Objetivo del proyecto	8
1.3 Project goal	9
1.4 Marco legal	10
2. Acuerdo del proyecto	11
2.1 Requisitos funcionales y no funcionales	11
2.1.1 Requisitos funcionales	12
2.1.2 Requisitos no funcionales	13
2.2 Definición de tareas	14
2.3 Metodología a seguir para la realización del proyecto	15
2.4 Planificación temporal de tareas	16
2.5 Presupuesto	18
2.6 Licencia de distribución	19
2.7 Análisis de riesgos	20
3. Análisis y diseño	21
3.1. Análisis y diseño de la arquitectura de la aplicación	21
3.1.1 Tecnologías/Herramientas usadas y descripción de las mismas	21
3.1.2 Arquitectura de componentes de la aplicación	23
3.2 Modelado de datos	25
3.3 Análisis y diseño del sistema funcional	30
3.4 Análisis y diseño de la interfaz de usuario	32
4. Documento de implementación e implantación del sistema	39
4.1 Implementación	39
4.2 Pruebas	43
Pruebas unitarias	43
Pruebas de integración	44
Pruebas de usabilidad	44
Pruebas de rendimiento y estabilidad	45
Pruebas de seguridad	46
4.3 Instalación y configuración	47
5. Documento de cierre	48



5.1. Documento de instalación y configuración	48
5.2. Manual de usuario	49
5.3. Resultados obtenidos y conclusiones	53
5.4. Cuaderno de bitácora	55
5.5. Bibliografía	56



1. Descripción del proyecto

El proyecto en cuestión consiste en crear una aplicación nativa para Android utilizando el lenguaje de programación **Kotlin** y aprovechando las ventajas de **Jetpack Compose** para el diseño de la interfaz de usuario.

El objetivo principal de la aplicación es proporcionar a los usuarios acceso a una amplia base de datos de videojuegos, permitiéndoles buscar, explorar y obtener información detallada sobre sus títulos favoritos.

1.1 Contexto del proyecto

1.1.1 Ámbito y entorno

Este proyecto no está dirigido a una empresa o cliente concretos, sino que forma parte del trabajo final de la titulación del grado superior en desarrollo de aplicaciones multiplataforma.

Dentro de este contexto académico, el objetivo es demostrar una **sólida comprensión** de las tecnologías necesarias y los conceptos clave para desarrollar aplicaciones móviles de alta calidad. El entorno tecnológico actual en el que se desarrolla este proyecto es muy **dinámico y competitivo**.

Las aplicaciones móviles se han convertido en una parte esencial de la vida cotidiana, proporcionando a los usuarios acceso a una amplia gama de servicios y opciones de entretenimiento.

Como resultado, la capacidad de desarrollar aplicaciones móviles de forma eficiente y eficaz se ha convertido en algo crucial para los desarrolladores.

1.1.2 Análisis de la realidad

En el contexto actual del desarrollo de aplicaciones móviles, es crucial realizar un **examen exhaustivo** de las circunstancias actuales para comprender el entorno en el que funcionará la aplicación. Este examen abarca múltiples factores que influyen en el desarrollo, la aplicación y la recepción por parte de los usuarios.

- **Tendencias:** Se observa un **crecimiento constante** en el uso de aplicaciones móviles, abarcando desde servicios cotidianos hasta entretenimiento. Este crecimiento impulsa la necesidad de desarrollar aplicaciones atractivas, funcionales y eficientes que cumplan con las expectativas de los usuarios.
- **Competencia y diferenciación:** El mercado de las aplicaciones se caracteriza por una gran competencia. Para destacar, es crucial ofrecer características distintivas y una experiencia de usuario única. El acceso a una base de datos de videojuegos puede ser un elemento **diferenciador** si se enfoca en brindar información detallada y relevante.
- **Experiencia y diseño centrado en el usuario:** Los usuarios valoran la facilidad de uso y una interfaz atractiva. Jetpack Compose, al permitir un **diseño flexible** y centrado en el usuario, se convierte en un activo para crear una experiencia atractiva y adaptable.
- **Seguridad y rendimiento:** En un entorno donde la seguridad y el rendimiento son prioritarios, el uso de Retrofit API debe ser respaldado por **buenas prácticas** de seguridad y un rendimiento eficiente para garantizar la confianza del usuario y la respuesta ágil de la aplicación.
- **Interés y demanda en videojuegos:** El sector de los videojuegos cuenta con una amplia base de seguidores. Proporcionar acceso a **información detallada** y actualizada sobre videojuegos podría satisfacer una demanda latente en los usuarios interesados en esta industria.



1.1.3 Solución y justificación de la solución propuesta

Basándonos en el análisis de la realidad del proyecto y las necesidades identificadas, las soluciones propuestas y su justificación podrían ser las siguientes:

Experiencia y diseño centrado en el usuario:

- **Solución:** Utilizar Jetpack Compose para el diseño de la interfaz de usuario, centrándose en la estética y la usabilidad.
- **Justificación:** Jetpack Compose ofrece un enfoque declarativo para diseñar interfaces de usuario interactivas y dinámicas. Permite una mayor flexibilidad en la personalización de la apariencia y el comportamiento de la aplicación. Esto se alinea con la necesidad de proporcionar una experiencia atractiva y funcional.

Acceso a información detallada sobre videojuegos:

- **Solución:** Integrar la API de RAWG utilizando Retrofit para acceder a una amplia base de datos de videojuegos, permitiendo obtener información detallada y actualizada sobre los títulos.
- **Justificación:** La elección de RAWG a través de Retrofit garantiza un acceso eficiente a una gran base de datos de videojuegos, lo que permitirá a los usuarios obtener información completa y actualizada sobre los juegos. Esta solución aborda la necesidad de proporcionar datos detallados para satisfacer las demandas de los aficionados a los videojuegos.

Optimización de rendimiento y seguridad:

- **Solución:** Implementar buenas prácticas de seguridad y optimización del rendimiento en la comunicación con la API, garantizando tiempos de carga rápidos y una experiencia segura.
- **Justificación:** En un entorno donde la seguridad y el rendimiento son prioritarios, esta solución asegura que la comunicación con la API se realice de manera segura.

1.1.4 Destinatarios de la aplicación

El proyecto en cuestión está dirigido a una audiencia diversa, pero principalmente orientado a:

- **Amantes de los videojuegos/Gamers:** El público principal de la aplicación serán los entusiastas de los videojuegos. Aquellas personas que sienten una gran pasión por los videojuegos y están deseosas de descubrir, profundizar y adquirir conocimientos exhaustivos sobre una amplia gama de títulos. Estos usuarios pueden tener interés en descubrir nuevos lanzamientos, leer críticas, localizar información sobre el desarrollo de un videojuego, etc.
- **Buscadores de información sobre videojuegos:** Usuarios que buscan información específica sobre videojuegos, ya sea por razones de entretenimiento, investigación o interés profesional. Esto podría incluir desde desarrolladores de juegos en busca de datos técnicos hasta críticos y analistas que necesitan información detallada y actualizada para sus análisis.
- **Profesionales y estudiantes:** La aplicación también puede ser útil para estudiantes o profesionales que deseen estudiar el uso de tecnologías específicas, como Jetpack Compose, Kotlin y la integración de APIs mediante Retrofit, en un contexto práctico.

1.2 Objetivo del proyecto

El objetivo del proyecto es desarrollar una aplicación multiplataforma para dispositivos móviles, centrándose en el sistema operativo Android.

Esta aplicación busca ofrecer a los usuarios acceso a una amplia base de datos de videojuegos, permitiéndoles buscar, explorar y obtener información detallada sobre distintos títulos.

La meta principal no solo es proporcionar una experiencia atractiva a los entusiastas de los videojuegos, sino también afianzar y aplicar los conocimientos adquiridos a lo largo del grado superior en desarrollo de aplicaciones multiplataforma.

Además, como desarrollador Android, el proyecto trata de fortalecer y consolidar habilidades laborales en el ámbito del desarrollo de aplicaciones móviles, al implementar tecnologías clave como Kotlin, Jetpack Compose, Retrofit API y RAWG para acceder a bases de datos de videojuegos.



1.3 Project goal

The objective of the project is to develop a multiplatform application for mobile devices, focusing on the Android operating system.

This application seeks to offer users access to a large database of video games, allowing them to search, explore and obtain detailed information about different titles.

The main goal is not only to provide an engaging experience for video game enthusiasts, but also to consolidate and apply the knowledge acquired throughout the advanced degree in multiplatform application development.

In addition, as an Android developer, the project seeks to strengthen and consolidate job skills in the field of mobile application development, by implementing key technologies such as Kotlin, Jetpack Compose, Retrofit API and RAWG to access video game databases.

1.4 Marco legal

En el desarrollo de una aplicación móvil, especialmente aquellas que manipulan datos de usuarios o acceden a información de terceros, es esencial considerar y cumplir con las normativas legales y de protección de datos correspondientes. Algunos aspectos legales a tener en cuenta podrían ser:

- **Reglamento General de Protección de Datos (RGPD):** Si la aplicación maneja información personal de usuarios, es fundamental asegurar el cumplimiento del RGPD en términos de recopilación, almacenamiento, procesamiento y protección de datos personales.
- **Derechos de autor y propiedad intelectual:** Si la aplicación muestra información, imágenes o contenido relacionado con videojuegos (tales como logotipos, imágenes, descripciones), es importante respetar los derechos de autor y asegurarse de contar con los permisos necesarios para mostrar este contenido.
- **Política de privacidad y términos de uso:** La aplicación debería contar con una política de privacidad clara que informe a los usuarios sobre cómo se manejan sus datos y un conjunto de términos y condiciones de uso que establezca las reglas para la utilización de la aplicación.
- **Normativas de la tienda de aplicaciones:** Es importante conocer y cumplir con las normativas específicas de las tiendas de aplicaciones, como Google Play Store, que establecen reglas sobre el tipo de contenido permitido, la presentación de la aplicación, entre otros aspectos.

Es fundamental realizar una revisión exhaustiva para asegurarse de que la aplicación cumpla con todas las normativas legales pertinentes, garantizando la protección de datos de los usuarios y el cumplimiento de las regulaciones establecidas.

2. Acuerdo del proyecto

2.1 Requisitos funcionales y no funcionales

Requisitos funcionales

Los requisitos funcionales describen las funciones y las acciones específicas que el sistema o software debe ser capaz de realizar. Estos requisitos definen lo que el sistema debe hacer en términos de comportamiento y operaciones, centrándose en las acciones que el usuario puede llevar a cabo y cómo el sistema responde a estas acciones. Por ejemplo, la capacidad de buscar juegos, mostrar detalles de un juego específico, permitir comentarios de usuarios, entre otros, son ejemplos de requisitos funcionales. Estos requisitos suelen ser verificables y medibles.

Requisitos no funcionales

Los requisitos no funcionales se centran en las características que definen la calidad, el rendimiento y las restricciones del sistema en lugar de sus funciones específicas. Estos requisitos abordan aspectos como la seguridad, la usabilidad, el rendimiento, la escalabilidad, la disponibilidad, la compatibilidad, entre otros. A diferencia de los funcionales, los requisitos no funcionales no se refieren a acciones específicas del sistema, sino a cómo debería comportarse o qué características debería tener el sistema en términos de calidad, eficiencia y restricciones técnicas.

2.1.1 Requisitos funcionales

ID	Función	Descripción
RF-01	Búsqueda de juegos	Permite a los usuarios buscar juegos por nombre, género, plataforma, desarrollador, etc.
RF-02	Detalles del juego	Muestra información detallada sobre un juego específico, incluyendo descripción y puntuación.
RF-03	Lista de favoritos	Los usuarios pueden marcar juegos como favoritos para acceder fácilmente a ellos más tarde.
RF-04	Lista de tendencias	Muestra los juegos más populares o tendencias en un período específico.
RF-05	Comentarios y reseñas	Posibilidad para los usuarios de dejar comentarios y reseñas sobre los juegos.
RF-06	Filtrado por plataforma	Permite a los usuarios ver juegos específicos disponibles en una plataforma seleccionada.
RF-07	Integración con la API RAWG	Conexión a la base de datos RAWG para obtener información actualizada sobre juegos y detalles.
RF-08	Autenticación de usuario	Registro e inicio de sesión para personalizar la experiencia del usuario y guardar preferencias.

2.1.2 Requisitos no funcionales

ID	Criterio	Descripción
RNF-01	Usabilidad	La interfaz de usuario debe ser intuitiva y fácil de navegar para usuarios nuevos y experimentados.
RNF-02	Accesibilidad	La aplicación debe cumplir con estándares de accesibilidad para permitir su uso por personas con discapacidades visuales o motoras.
RNF-03	Rendimiento	Tiempo de carga de la aplicación y tiempos de respuesta de las acciones deben ser óptimos y ágiles, incluso en condiciones de red variables.
RNF-04	Escalabilidad	La aplicación debe ser capaz de manejar un aumento gradual de usuarios y datos sin degradación significativa del rendimiento.
RNF-05	Estructura de la aplicación	El código debe seguir principios de diseño limpio y arquitectura bien estructurada, como separación de capas (UI, lógica de negocio, acceso a datos).
RNF-06	Manuales y documentación	Disponibilidad de manuales de usuario y documentación técnica clara y completa para futuros desarrolladores o mantenimiento.

2.2 Definición de tareas

ID	Tarea	Duración (h)	Descripción
T-01	Análisis del proyecto	20	Definición de requerimientos, investigación de la API RAWG, análisis de competencia, definición de casos de uso.
T-02	Diseño	30	Diseño de la base de datos, creación de diagramas de arquitectura, diseño de interfaces de usuario con Jetpack Compose.
	Diseño de la base de datos	10	
	Diseño de las interfaces Jetpack Compose	20	
T-03	Implementación	100	Configuración del entorno de desarrollo, desarrollo del front-end, integración con Retrofit para la conexión a la API, lógica de negocio.
T-04	Pruebas	25	Pruebas unitarias, pruebas de integración, pruebas de interfaz, pruebas de rendimiento.
T-05	Documentación	15	Creación de documentación técnica, elaboración de manuales de usuario, documentación de código.
T-06	Revisión y ajustes	10	Revisión general del proyecto, ajustes según feedback, correcciones y optimizaciones.
T-07	Despliegue y entrega final	5	Preparación de la aplicación para el despliegue, empaquetado y entrega final.
	Total	205	

2.3 Metodología a seguir para la realización del proyecto

Utilizando metodologías ágiles como Scrum o Kanban:

Fase de análisis y diseño (Cascada):

- **Análisis de requerimientos:** Se realizará una fase inicial de análisis de requerimientos donde se definirán las necesidades y funcionalidades clave de la aplicación en colaboración con el cliente.
- **Diseño y planificación:** Basado en los requerimientos definidos, se diseñarán los elementos clave como la arquitectura de la aplicación, la estructura de la base de datos y las interfaces de usuario. Se establecerá una planificación inicial del proyecto.

Interacción con el cliente (Ágil):

- **Prototipos interactivos:** Se diseñarán prototipos interactivos al inicio del proyecto para permitir al cliente visualizar y validar las funcionalidades clave antes del desarrollo completo.

Fase de implementación (Ágil):

- **Desarrollo iterativo:** Se priorizan las funcionalidades y se desarrollan en ciclos cortos, permitiendo la adaptación a cambios y una mayor flexibilidad.
- **Demostraciones periódicas:** Se realizarán demostraciones al cliente al final de cada iteración para obtener feedback y ajustar el desarrollo según las necesidades cambiantes.

Fase de pruebas y documentación (Cascada y Ágil):

- **Pruebas continuas:** Se llevarán a cabo pruebas a lo largo del proceso de desarrollo, tanto unitarias como de integración, de manera continua.
- **Documentación:** La documentación técnica y de usuario se irá elaborando a medida que se avanza en el proyecto, manteniéndola actualizada en cada iteración.

Fase de revisión y entrega (Cascada):

- **Revisión final:** Al finalizar el desarrollo, se llevará a cabo una revisión general para asegurar que se cumplen todos los requisitos y estándares establecidos.
 - **Entrega y despliegue:** Una vez completadas las revisiones, se procederá con la entrega final y el despliegue de la aplicación.
-



2.4 Planificación temporal de tareas

1. Análisis y diseño (T-01 y T-02)

- Análisis de requerimientos (Semana 1): Identificación de necesidades y funcionalidades clave.
- Diseño de prototipos (Semana 2): Creación de prototipos interactivos para validar conceptos.
- Diseño de arquitectura (Semana 3): Definición de la estructura general de la aplicación.

2. Implementación (T-03)

- Configuración del entorno (Semana 4): Preparación del entorno de desarrollo.
- Desarrollo front-end (Semanas 5-8): Creación de la interfaz de usuario con Jetpack Compose.
- Integración con la API RAWG (Semanas 6-9): Conexión y obtención de datos de la API.
- Desarrollo del Back-end (Semanas 7-10): Implementación de la lógica de negocio y acceso a datos.

3. Pruebas (T-04)

- Pruebas unitarias (Semana 11): Verificación de funciones a nivel de unidad.
- Pruebas de interfaz (Semana 12): Validación de la interfaz de usuario.
- Pruebas de rendimiento (Semanas 12-13): Evaluación del rendimiento de la aplicación.

4. Documentación (T-05)

- Documentación técnica (Semanas 13-14): Creación de documentación técnica detallada.
- Manuales de usuario (Semana 14): Elaboración de manuales para los usuarios finales.

5. Revisión y entrega (T-06)

- Revisión general (Semana 15): Evaluación global del proyecto.
 - Entrega final y despliegue (Semana 16): Preparación y distribución de la aplicación.
-

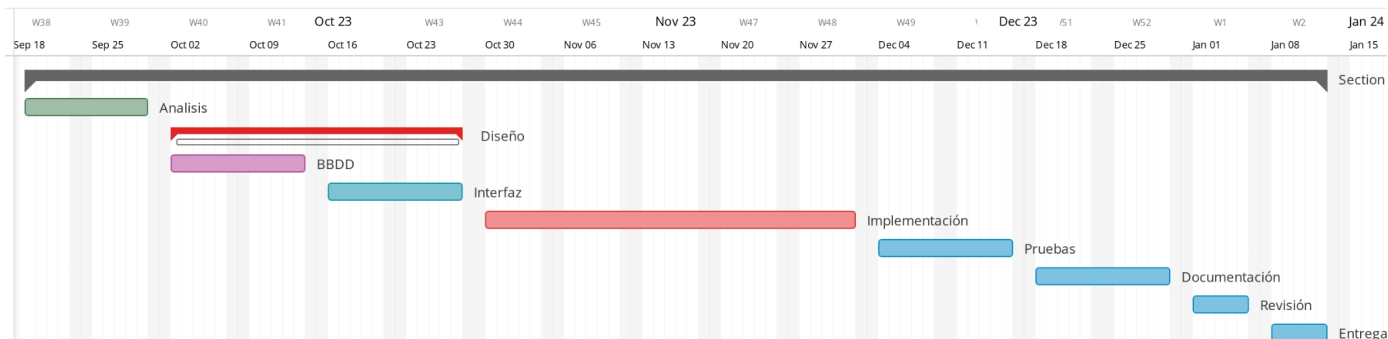


Diagrama de Gantt

2.5 Presupuesto

Partida	Unidades	Costo por Unidad	Subtotal
Ordenadores	1	1300€	1300€
Software	1	145€	145€
Material y Equipamiento			1445€
Licencia API RAWG	4 meses	150€/mes	600€
Licencia Google Play Store	1	25€	25€
Licencias y Servicios			625€
Programador	140 horas	25€/hora	3500€
Diseñador	30 horas	20€/hora	600€
Analista	20 horas	28€/hora	560€
Personal			4660€
Alquiler	4 meses	800€/mes	3200€
Gastos internet	4 meses	160€/mes	640€
Gastos luz	4 meses	60€/mes	240€
Gastos Indirectos			4080€
Total			10785€

2.6 Licencia de distribución

Objetivo y alcance de la licencia:

- Define claramente qué derechos otorga la licencia a los usuarios y cómo pueden utilizar el software.

Condiciones de uso:

- Establece los términos bajo los cuales los usuarios pueden utilizar la aplicación. Por ejemplo, si es para uso personal, comercial, restricciones de modificación, redistribución, etc.

Restricciones:

- Especifica claramente las acciones o usos que no están permitidos, como la descompilación, la redistribución sin autorización, la modificación sin consentimiento, etc.

Propiedad intelectual:

- Detalla quién posee los derechos sobre la aplicación y cómo pueden los usuarios utilizar el contenido generado por la aplicación.

Garantías y responsabilidades:

- Explica el alcance de las garantías ofrecidas, si las hay, y la responsabilidad del desarrollador en caso de fallos o problemas con la aplicación.

Actualizaciones y mantenimiento:

- Describe cómo se manejan las actualizaciones del software y si se proporcionará soporte técnico.

Terminación de la licencia:

- Especifica las condiciones bajo las cuales la licencia puede ser terminada, ya sea por incumplimiento de los términos o por decisión del desarrollador.

Leyes y jurisdicción:

- Indica la jurisdicción y las leyes aplicables en caso de disputas legales.

El acuerdo de licencia completo se encuentra en el [Anexo](#) Acuerdo de Licencia y condiciones de uso para referencia detallada.

2.7 Análisis de riesgos

Debilidades (D)	Amenazas (A)
<ul style="list-style-type: none">• Posibles problemas de integración: Dificultad para integrarse con la API RAWG o incompatibilidad con versiones futuras.• Curva de aprendizaje: Con una tecnología relativamente nueva, el equipo puede tardar algún tiempo en adaptarse por completo.	<ul style="list-style-type: none">• Competencia en el mercado: La posibilidad de que otras aplicaciones similares ofrezcan mejor funcionalidad o rendimiento.• Cambios en las API externas: Dependencia de la API RAWG y posibilidad de cambios inesperados que afecten la funcionalidad.
Fortalezas (F)	Oportunidades (O)
<ul style="list-style-type: none">• Tecnología de vanguardia: Utiliza las últimas tecnologías como Kotlin, Jetpack Compose y Retrofit para brindar una experiencia de usuario mejorada.• Conexión a la API RAWG: Acceso a una gran base de datos de videojuegos actualizada.• Flexibilidad de desarrollo: Los métodos mixtos permiten adaptarse a los cambios y recibir comentarios de los clientes.	<ul style="list-style-type: none">• Mercado en crecimiento: Gran demanda de videojuegos y aplicaciones relacionadas con el entretenimiento.• Oportunidades de mejora: Posibilidad de agregar nuevas funciones según los comentarios de los clientes.

3. Análisis y diseño

3.1. Análisis y diseño de la arquitectura de la aplicación

3.1.1 Tecnologías/Herramientas usadas y descripción de las mismas

- **Kotlin (1.9.10):** Es un lenguaje de programación moderno desarrollado por JetBrains y adoptado oficialmente por Google desde 2017 como el lenguaje de programación preferido para el desarrollo de aplicaciones Android. Algunas de sus características son:
 - **Concisión y legibilidad:** Kotlin está diseñado para ser conciso, lo que significa que se requiere menos código para lograr las mismas funcionalidades que en Java. Esto mejora la legibilidad del código y reduce la posibilidad de errores.
 - **Seguridad:** ofrece un sistema de tipos que elimina muchos errores comunes asociados con el código nulo (null pointer exceptions). Esto se logra mediante la distinción clara entre tipos que pueden contener valores nulos y los que no.
 - **Compatibilidad:** Kotlin es compatible con el código existente de Java, lo que permite a los desarrolladores utilizar bibliotecas y frameworks de Java sin problemas. También se puede integrar fácilmente en proyectos existentes.
 - **Interoperabilidad:** Puede interactuar sin problemas con código Java existente. Esto significa que los desarrolladores pueden migrar gradualmente sus proyectos de Java a Kotlin.
 - **Desarrollo de aplicaciones Android:** Kotlin se ha convertido en el lenguaje de programación preferido para el desarrollo de aplicaciones Android. Google ha brindado un fuerte respaldo a Kotlin, lo que ha llevado a un aumento significativo en su adopción por parte de la comunidad de desarrollo de Android.
 - **Jetpack Compose:** Biblioteca de interfaz de usuario moderna y nativa para simplificar y acelerar el desarrollo de interfaces en aplicaciones Android.
 - **Retrofit:** Biblioteca de cliente HTTP para Android y Java que simplifica el consumo de API REST al convertir endpoints en interfaces Java.
-

- **Android Studio:** Entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android. Proporciona herramientas para el diseño, codificación y depuración de apps.
 - **Firebase:** Plataforma de desarrollo de aplicaciones móviles de Google que ofrece una variedad de herramientas, como Firebase Authentication, Cloud Firestore, Cloud Functions, entre otros, para facilitar el desarrollo y la gestión de aplicaciones.
 - **Git:** Sistema de control de versiones distribuido para rastrear cambios en el código fuente durante el desarrollo del proyecto y permitir la colaboración entre equipos.
 - **GanttProject:** es un programa de código abierto con licencia GPL escrito en Java con la biblioteca Swing, su objetivo es la administración de proyectos usando el diagrama de Gantt.
 - **Draw.io:** es un software de dibujo gráfico multiplataforma desarrollado en HTML5 y JavaScript. Su interfaz permite crear diagramas como diagramas de flujo, wireframes, diagramas UML, organigramas y diagramas de red.
 - **Room:** Es una biblioteca de persistencia de datos que proporciona una capa de abstracción sobre SQLite para permitir una manipulación más fácil y robusta de la base de datos en aplicaciones Android. Algunas de sus características clave son:
 - Simplicidad en su uso: Room simplifica la interacción con SQLite al proporcionar anotaciones que permiten definir fácilmente esquemas de base de datos y consultas SQL.
 - Abstracción de SQLite: Ofrece una capa de abstracción que maneja gran parte de la complejidad subyacente de SQLite, permitiendo a los desarrolladores centrarse en la lógica de la aplicación.
 - Verificación en tiempo de compilación: Room realiza verificaciones en tiempo de compilación de SQL queries.
 - Integración con LiveData: Room se integra fácilmente con LiveData de Jetpack, permitiendo una actualización automática de la interfaz de usuario cuando los datos de la base de datos cambian.
-

3.1.2 Arquitectura de componentes de la aplicación

Para la arquitectura de componentes de la aplicación, voy a seguir un **patrón de diseño MVVM** (Modelo-Vista-Vista Modelo) para organizar la parte lógica y física de la app.

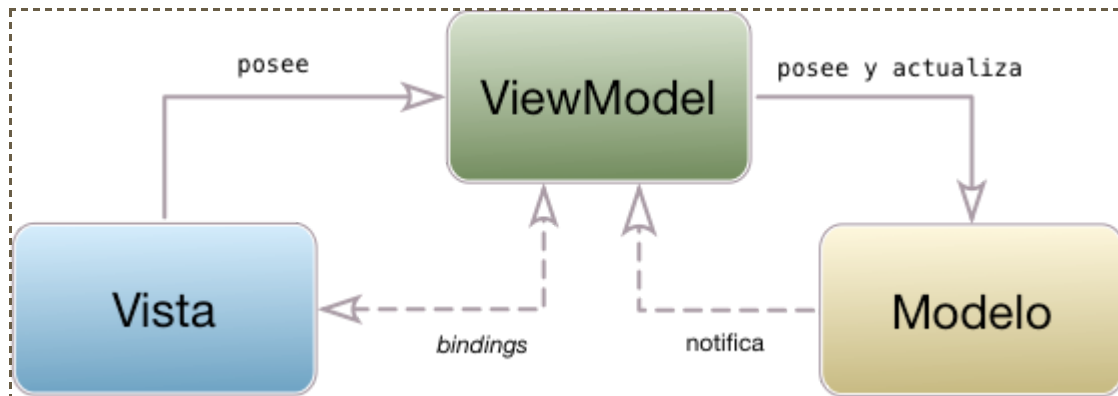
Arquitectura MVVM:

- **Organización Lógica:**

- **Modelo (Model):** Representa los datos y la lógica de negocio de la aplicación. Aquí se **manejan los datos y se realizan operaciones**.
- **Vista (View):** **Es la interfaz de usuario**. Se encarga de la presentación visual y la interacción con el usuario.
- **Vista Modelo (ViewModel):** Actúa como un intermediario entre el Modelo y la Vista. Provee los **datos** de manera que la vista pueda mostrarlos y se encarga de manejar la **lógica de presentación**.

- **Organización Física:**

- Para la arquitectura física, la aplicación estará diseñada para **dispositivos Android**. Los requisitos mínimos del dispositivo estarán alineados con las versiones de las herramientas y tecnologías empleadas.
 - **Versión de Android:** Se requerirá una versión mínima de Android compatible con las bibliotecas y herramientas utilizadas. Por ejemplo, **Android 14** (Android **SDK 34**, es la utilizada en la app) o superior para garantizar la compatibilidad con las bibliotecas de Jetpack y Kotlin.
-



Patrón de diseño MVVM

Este esquema representa la relación entre los componentes de la arquitectura MVVM, donde el modelo maneja la lógica de negocio y los datos, el Vista modelo se encarga de exponer y gestionar estos datos para la vista, y la vista se enfoca en la presentación y la interacción con el usuario.

Esta arquitectura proporciona su capacidad para separar de forma limpia la presentación de una aplicación determinada y la lógica del negocio de su interfaz de usuario y facilita el desarrollo, la prueba y el mantenimiento de la aplicación al proporcionar una estructura clara y escalable.

3.2 Modelado de datos

Datos de entrada:

- Inicio de sesión:
 - Correo electrónico o nombre de usuario
 - Contraseña
- Selección de plataformas:
 - Elección de la plataforma de juego
- Lista de juegos por plataforma:
 - Búsqueda de juegos por nombre
- Detalle del juego:
 - Información específica del juego: nombre, género, descripción, puntuación, comentarios.

Datos de Salida:

- Información detallada del juego:
 - Todos los detalles sobre un juego específico, incluyendo descripción, calificación, comentarios, etc.
- Lista de Favoritos:
 - Juegos marcados como favoritos por el usuario.
- Comentarios y Reseñas:
 - Opiniones y comentarios de usuarios sobre juegos individuales.

Almacenamiento de los datos:

Los datos se almacenarán en bases de datos, ya sea relacional (como **Room/SQLite**) o no relacional (como **Firestore**). Para cada tipo de dato, se utilizarán colecciones o tablas específicas para organizar la información de manera coherente y accesible para la aplicación.

Estos datos de entrada, salida y almacenamiento definen la interacción del usuario con la aplicación, los resultados que obtendrán y cómo se organizarán y gestionarán estos datos en el backend de la aplicación.

En el desarrollo de la capa de persistencia de datos, se ha empleado la librería Room, que ofrece una capa de abstracción sobre SQLite para manejar la base de datos local de la aplicación. A continuación se detalla la implementación realizada:

GameDAO: Interfaz de Acceso a Datos

kotlin

Copy code

@Dao

```
interface GameDAO {  
    @Query("SELECT * FROM favorite_games WHERE userId = :userId")  
    fun getAll(userId : String): List<GameEntity>  
  
    @Insert(onConflict = OnConflictStrategy.IGNORE)  
    fun insertAll(vararg games: GameEntity)  
  
    @Delete  
    fun delete(game: GameEntity)  
  
    @Query("SELECT COUNT(*) FROM favorite_games WHERE id = :gameId AND userId = :userId")  
    fun isGameFavorite(gameId: Long, userId : String): Int  
}
```

La interfaz GameDAO define métodos para operaciones CRUD en la tabla favorite_games. Las anotaciones como @Query, @Insert, @Delete se utilizan para realizar consultas SQL y operaciones de base de datos.

GameDatabase: Configuración de la Base de Datos

kotlin

Copy code

```
@Database(entities = [GameEntity::class], version = 11)  
@TypeConverters(GenreListConverter::class)  
abstract class GameDatabase : RoomDatabase() {  
    abstract fun gameDao(): GameDAO  
}
```

La clase GameDatabase es una clase abstracta que extiende RoomDatabase y define la versión de la base de datos y las entidades que la componen mediante la anotación @Database. Además, se declara un método abstracto gameDao() que proporciona acceso al DAO GameDAO.

Controles de entrada de datos:

- **Botón (Button):**
 - Los botones permiten a los usuarios interactuar y realizar acciones. Pueden activar eventos cómo enviar formularios, iniciar procesos o realizar una acción específica en la aplicación.
 - **Botón de favoritos:**
 - Este botón, representado comúnmente por un corazón, permite a los usuarios marcar contenido como favorito o agregarlo a una lista especial. Al hacer clic en este botón, se añade o elimina el elemento de la lista de favoritos.
 - **Campos de texto (Text Fields):**
 - Permiten a los usuarios ingresar texto. Pueden ser campos de texto de una sola línea (para ingresar nombres, contraseñas, etc.) o campos de texto de varias líneas (para comentarios, descripciones, etc.).
 - **Selectores (Dropdowns/Spinners):**
 - Estos controles muestran una lista de opciones desplegables para que el usuario elija una. Pueden ser desplegables (Dropdowns) o seleccionadores (Spinners).
 - **Listas desplegables (Dropdown Lists):**
 - Similar a los selectores, muestran una lista de opciones cuando se hace clic en un área específica. Estas listas desplegables son útiles cuando se necesita espacio para mostrar varias opciones.
 - **Casillas de verificación (Checkboxes):**
 - Permiten a los usuarios seleccionar una o varias opciones de una lista de elementos. Son útiles cuando se quiere permitir al usuario elegir múltiples opciones.
 - **Botones de selección única (Radio Buttons):**
 - Permiten al usuario seleccionar una única opción de entre varias. Son útiles cuando se necesita que el usuario elija una sola opción de un conjunto de opciones.
 - **Barra de búsqueda (Search Bar):**
 - Al hacer clic en este ícono, se abre un campo de búsqueda donde los usuarios pueden introducir términos de búsqueda para encontrar contenido específico dentro de la aplicación. Este ícono puede representarse como una lupa y es fundamental para permitir a los usuarios buscar información relevante.
-

Diagrama de clases:

- **Clase Juego (Game):**

- id: Identificador único del juego.
- metacritic: Puntuación del juego en Metacritic.
- name: Nombre del juego.
- slug: Nombre simplificado para URL.
- name_original: Nombre original del juego.
- platforms: Lista de plataformas en las que está disponible el juego.
- genres: Lista de géneros a los que pertenece el juego.
- background_image: URL de la imagen de fondo del juego.
- background_image_additional: URL de una imagen de fondo adicional del juego.
- description: Descripción del juego.
- description_raw: Descripción sin procesar del juego.
- released: Fecha de lanzamiento del juego.
- rating: Puntuación del juego.

- **Clase Género (Genre):**

- id: Identificador único del género.
- name: Nombre del género.
- slug: Nombre simplificado para URL.
- games_count: Cantidad de juegos asociados a este género.

- **Clase Plataforma (Platform):**

- id: Identificador único de la plataforma.
 - name: Nombre de la plataforma.
 - slug: Nombre simplificado para URL.
 - games_count: Cantidad de juegos asociados a esta plataforma.
 - image_background: URL de la imagen de fondo de la plataforma.
 - description: Descripción de la plataforma.
-

- **Clase Plataformas (Platforms):**

- platform: Objeto de tipo Platform que contiene la información de una plataforma en particular.

- **Clase Desarrollador (Developer):**

- id: Identificador único del desarrollador.
- name: Nombre del desarrollador.
- slug: Nombre simplificado para URL.
- gameCount: Cantidad de juegos asociados a este desarrollador.
- imageBackgroundURL: URL de la imagen de fondo del desarrollador.
- description: Descripción del desarrollador.

- **Clase Distribuidor (Publisher):**

- id: Identificador único del editor.
- name: Nombre del editor.
- slug: Nombre simplificado para URL.
- gameCount: Cantidad de juegos asociados a este editor.
- imageBackgroundURL: URL de la imagen de fondo del editor.
- description: Descripción del editor.

- **Clase RawgData:**

- Está diseñada para manejar respuestas de la API RAWG.
 - Al utilizar la anotación `@SerializedName`, se especifica el nombre real de los campos en el **JSON** que se espera recibir de la API, permitiendo así que **Gson** (o alguna otra librería de serialización/deserialización) mapee correctamente los datos JSON a los campos de la clase Kotlin.
 - count: Representa la cantidad total de resultados en la respuesta de la API.
 - next: URL de la próxima página de resultados.
 - prev: URL de la página anterior de resultados.
 - result: Contiene los datos principales de la respuesta de la API. Este campo es genérico y puede ser de cualquier tipo `<T>`.
-

3.3 Análisis y diseño del sistema funcional

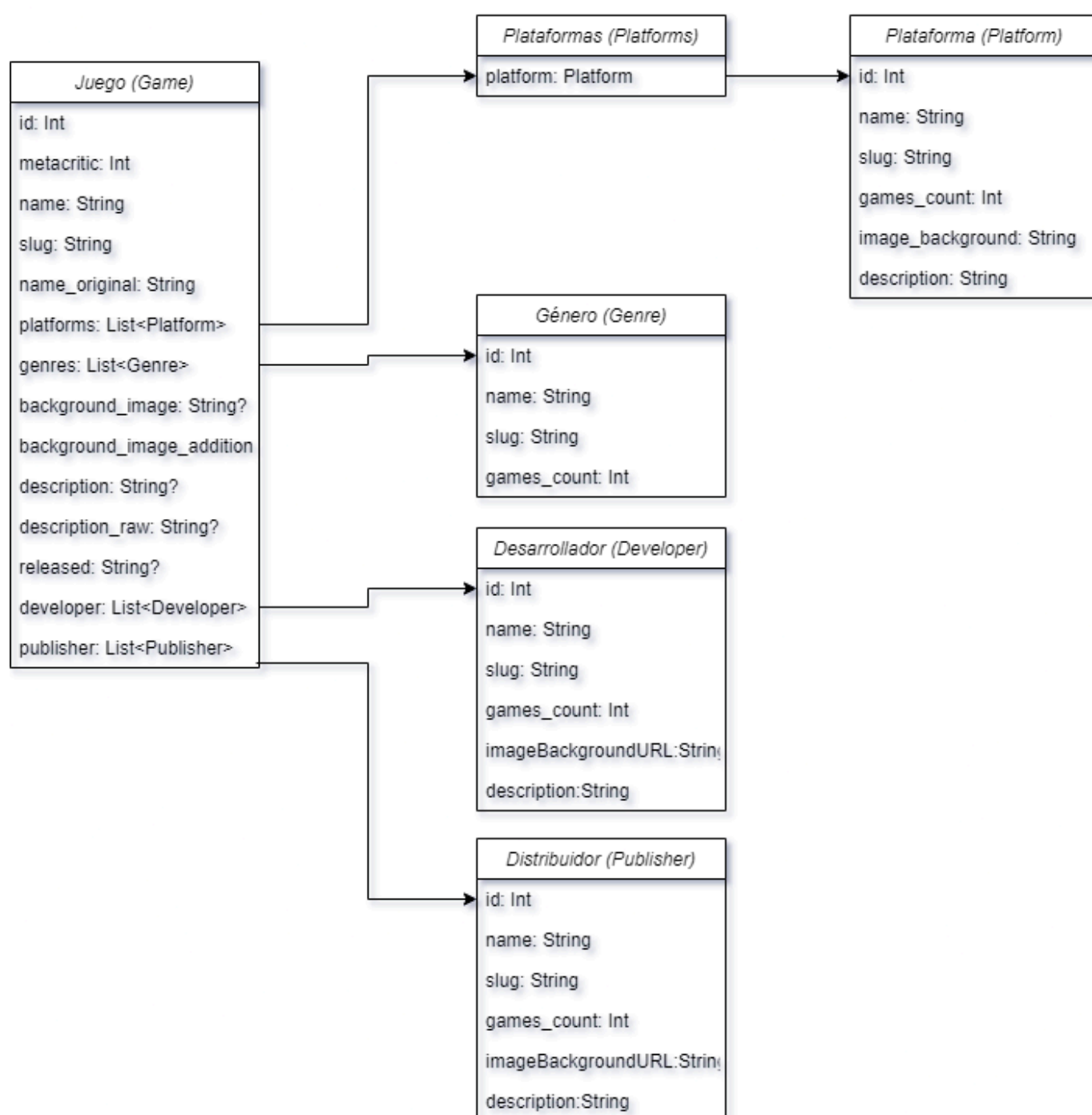
Para abordar el análisis y diseño del sistema funcional, se pueden considerar las siguientes estrategias y elementos:

Diagramas UML:

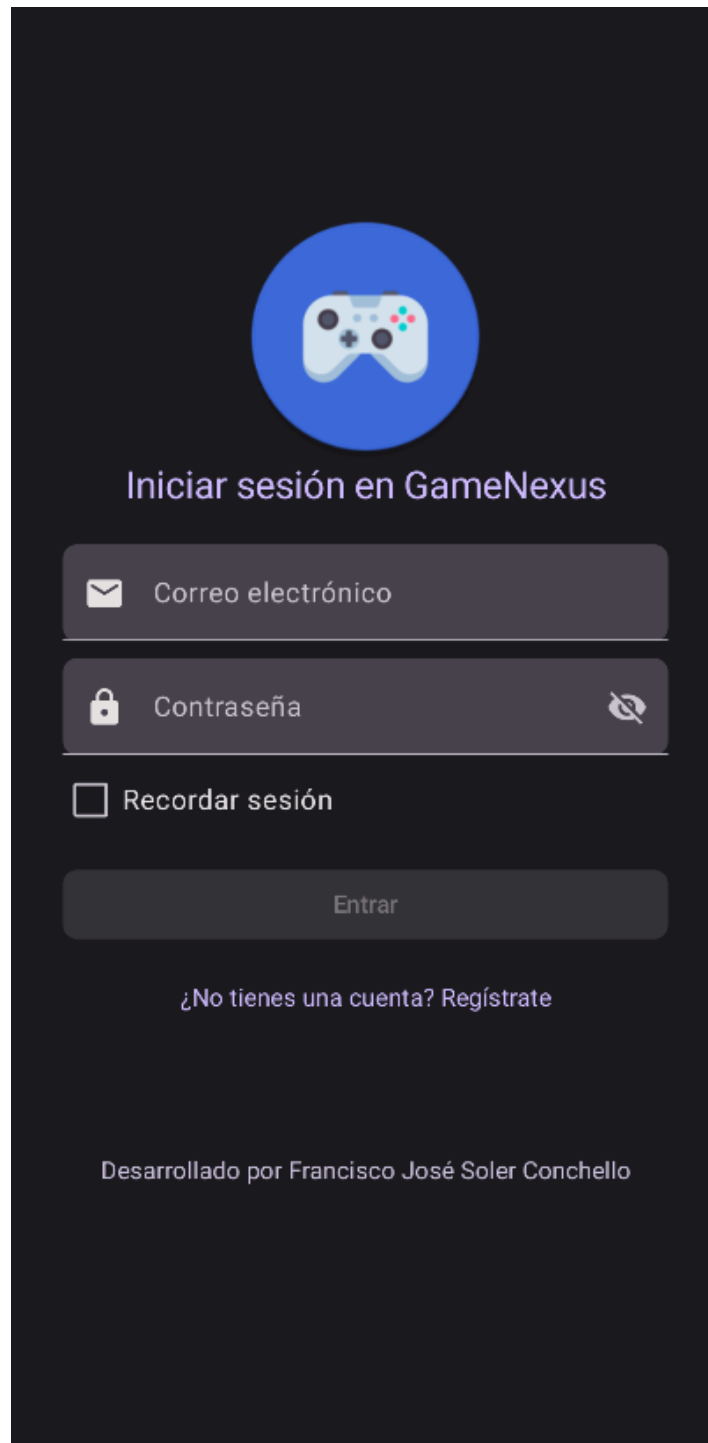
- Diagrama de Casos de Uso: Muestra los actores y los diferentes casos de uso de la aplicación.
 - Diagrama de Clases: Representa las clases del sistema y sus relaciones.
 - Diagrama de Secuencia: Describe la interacción entre diferentes componentes o actores del sistema a lo largo del tiempo.
 - Diagrama de Actividad: Muestra el flujo de actividades y acciones del sistema.
-

<i>RawgData<T></i>
count: Int
next: String
prev: String
result: T

<i>DatosUsuario</i>
username: String
pwd: String
remember: Boolean

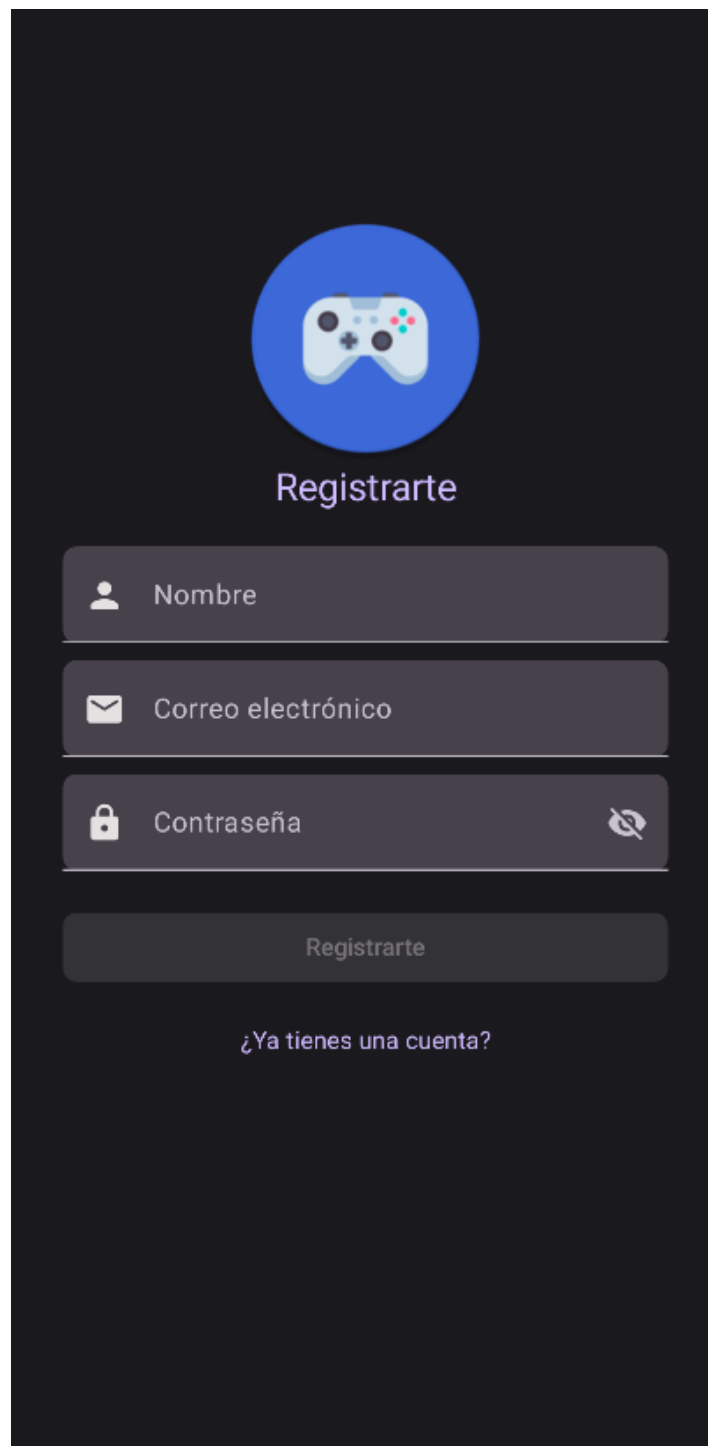


3.4 Análisis y diseño de la interfaz de usuario



Mockup 1: Ventana de inicio de sesión

Esta pantalla muestra los campos de entrada para iniciar sesión con el correo electrónico y la contraseña.



Registrarte

Nombre

Correo electrónico

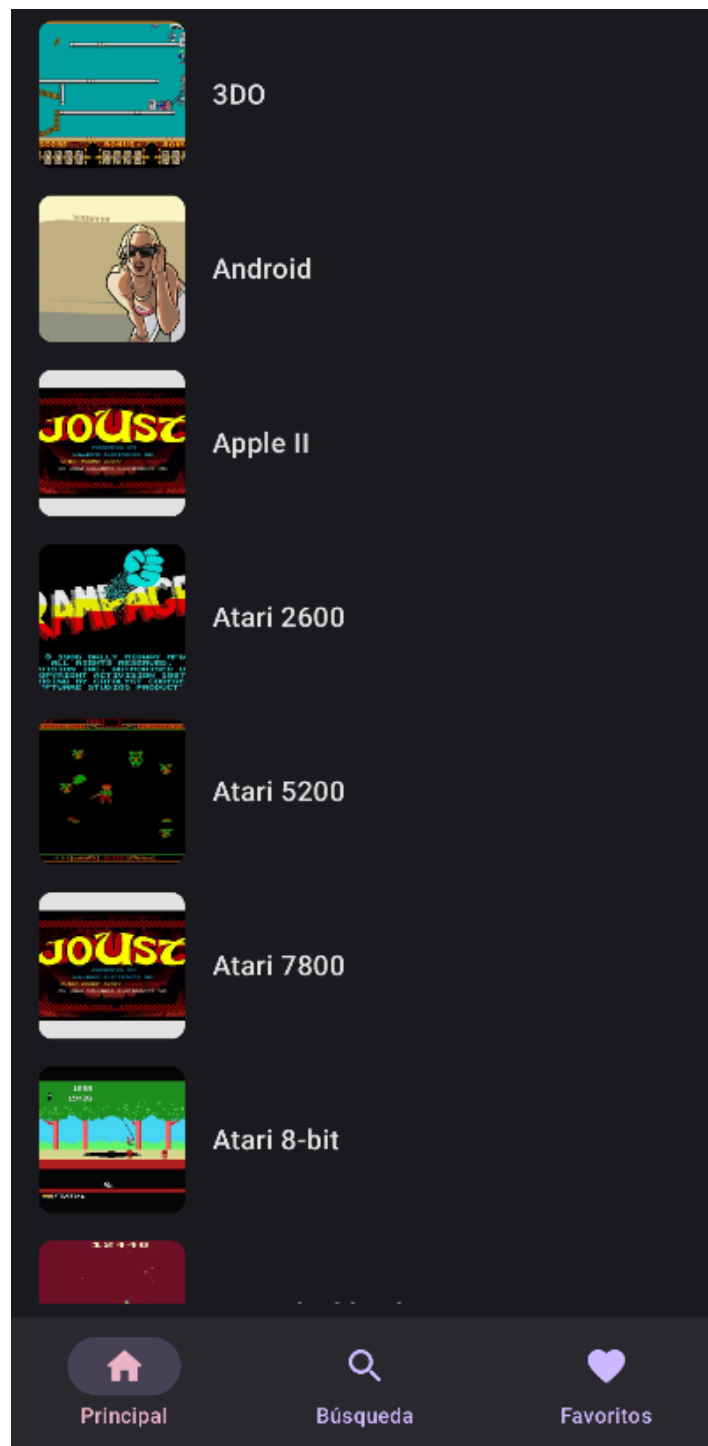
Contraseña

Registrarte

[¿Ya tienes una cuenta?](#)

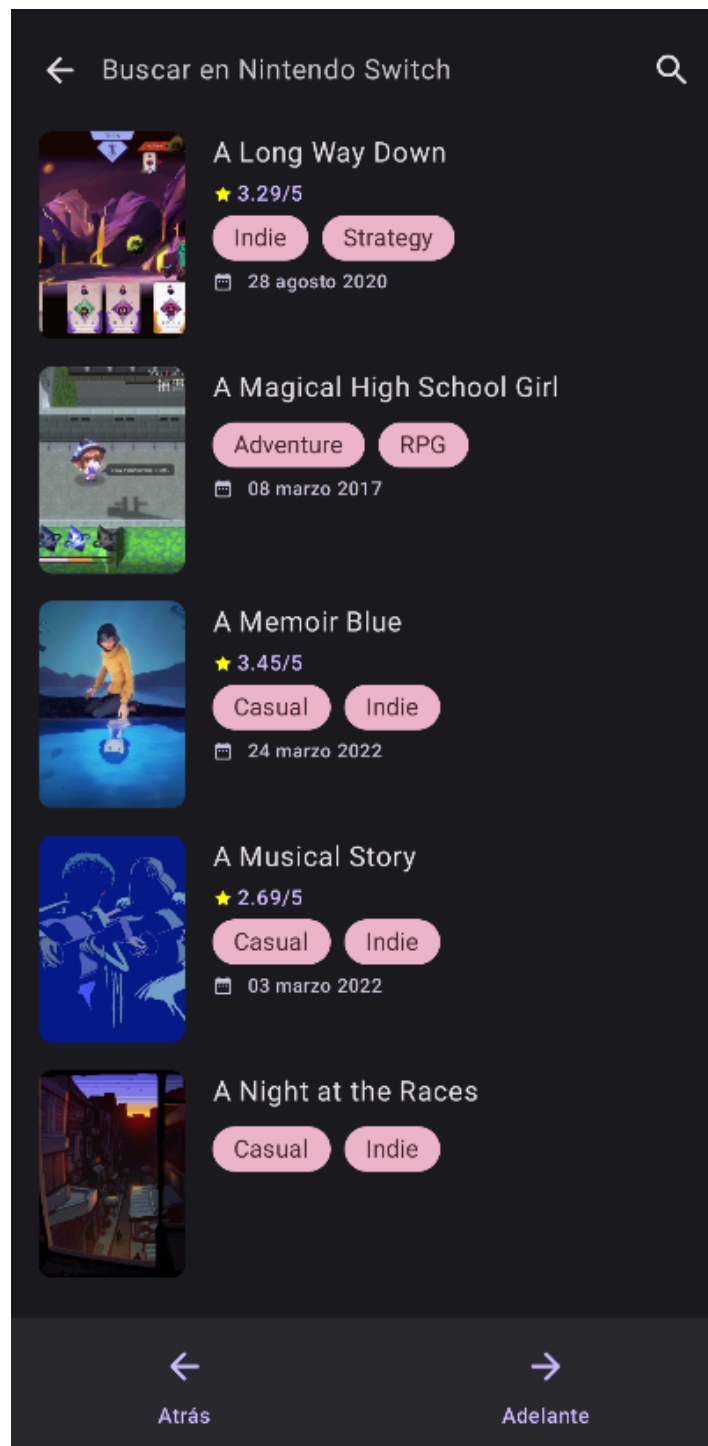
Mockup 2: Ventana de registro de usuario

Para registrar una nueva cuenta, el usuario deberá introducir su nombre, correo electrónico y una contraseña.



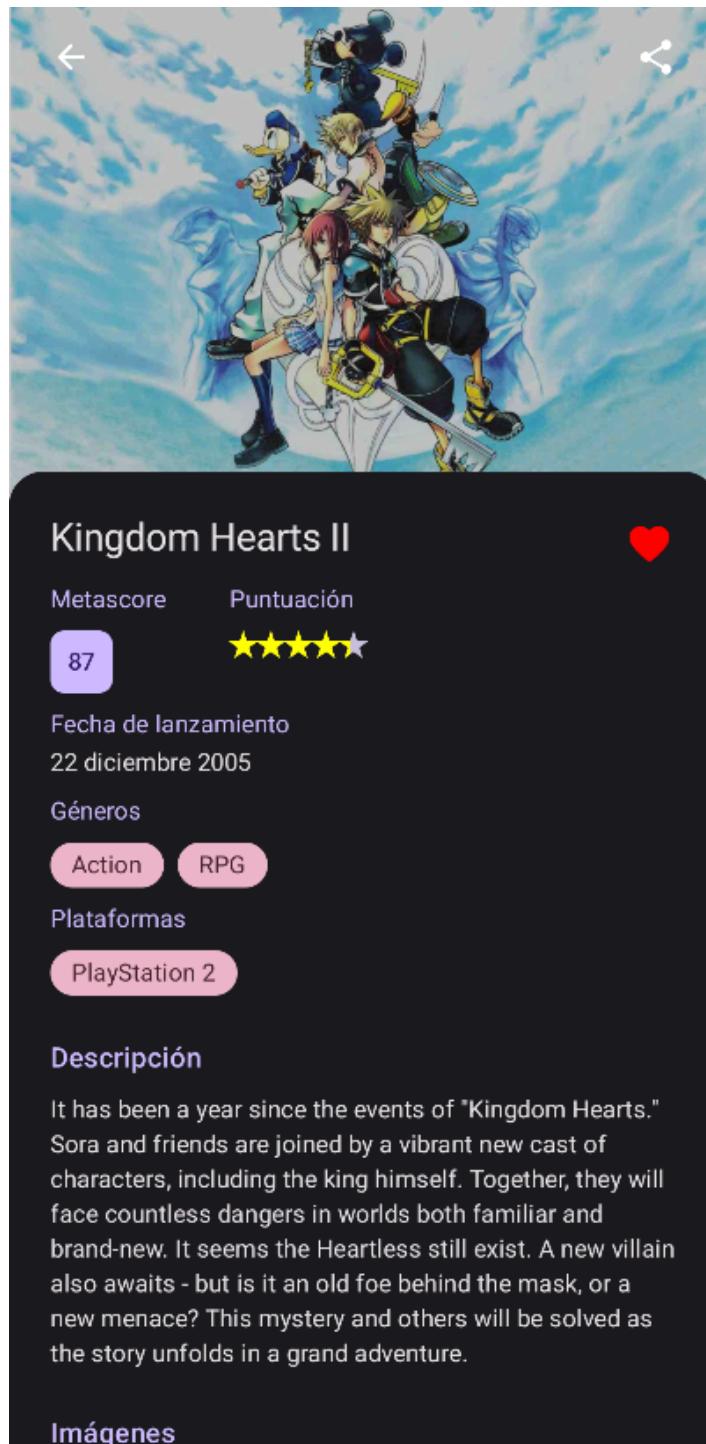
Mockup 3: Selección de plataformas

Presenta una lista de plataformas disponibles para que el usuario elija una.



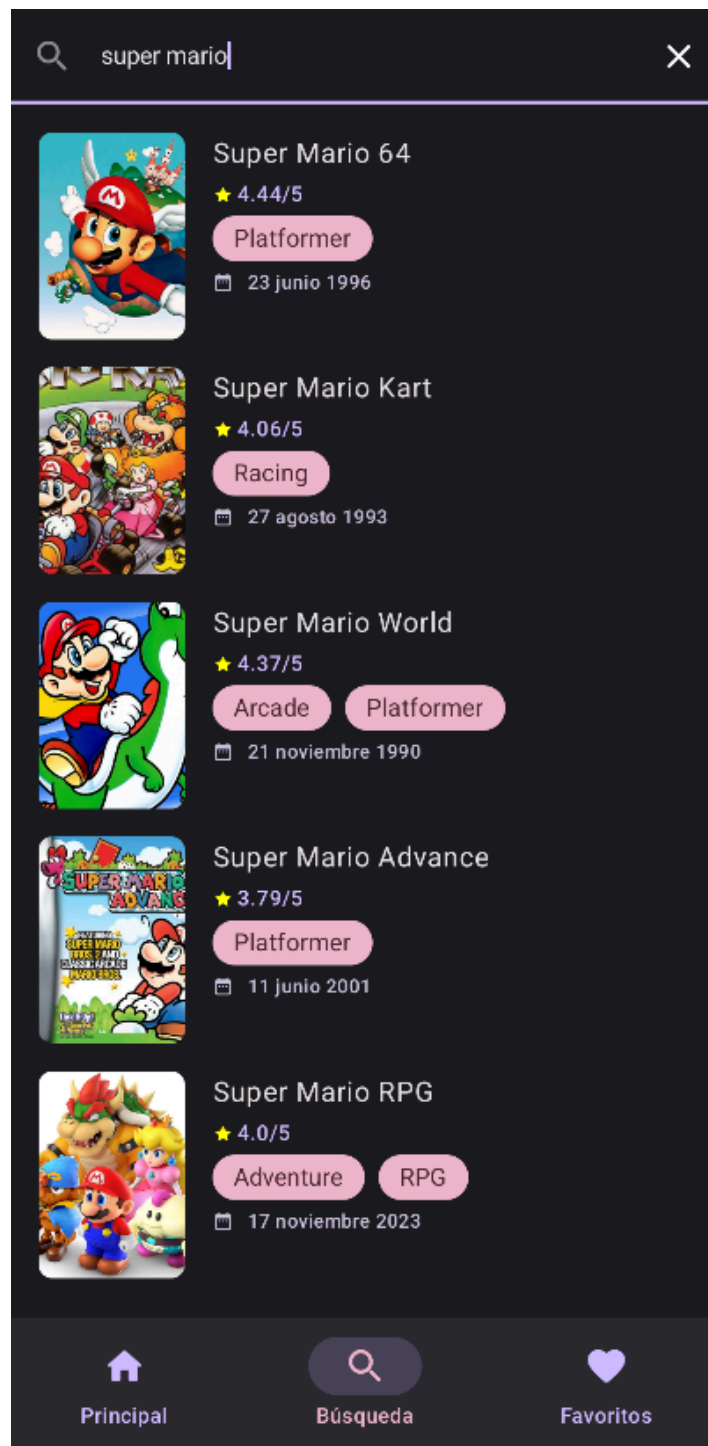
Mockup 4: Lista de juegos por plataforma

Muestra una lista de juegos disponibles para la plataforma seleccionada.



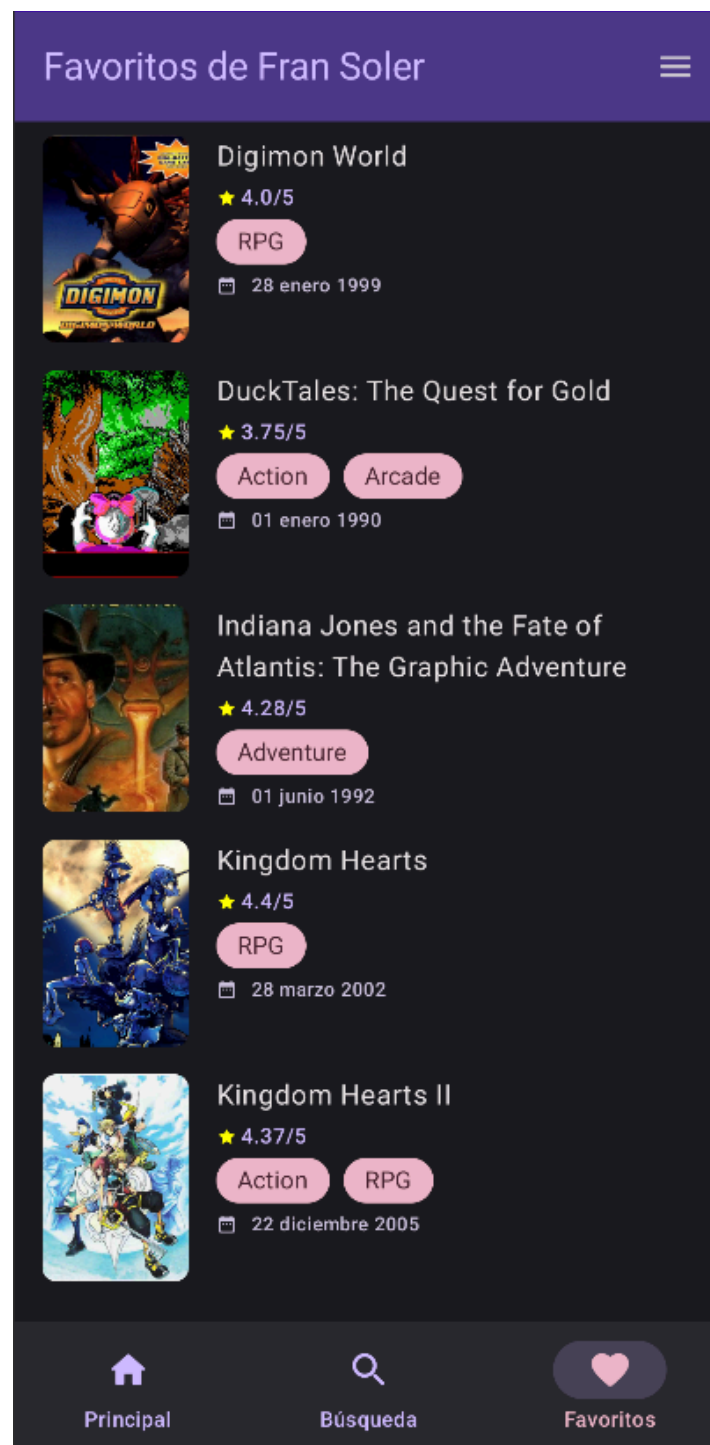
Mockup 5: Detalles del juego

Al seleccionar un juego de la lista, se abre esta pantalla con detalles completos del juego, como descripción, fecha de lanzamiento, géneros, plataformas, puntuación, etc.



Mockup 6: Búsqueda de juegos

En esta ventana se permite la búsqueda de juegos relevantes por nombre.

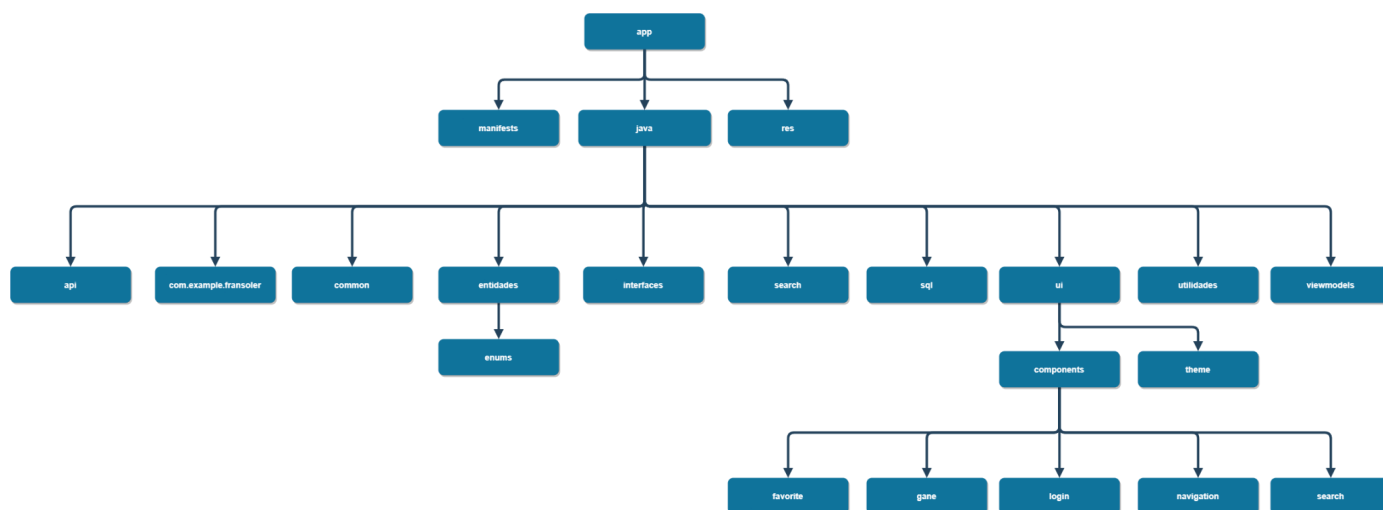


Mockup 7: Lista de favoritos

En esta ventana se muestran los juegos almacenados en la lista de favoritos del usuario.

4. Documento de implementación e implantación del sistema

4.1 Implementación



Esquema de la aplicación Android: GameNexus

El sistema está estructurado siguiendo una arquitectura de aplicación Android, con un conjunto de módulos y paquetes que se distribuyen de la siguiente manera:

- api

Este directorio almacena las clases y utilidades relacionadas con la comunicación de la aplicación con servicios web externos, proporcionando capas de abstracción para la obtención y manipulación de datos.

- com

En este directorio, se encuentra la estructura del paquete de la aplicación, que incluye componentes como la lógica de la interfaz de usuario, modelos de datos y otros submódulos.

- com.example.fransoler

Este paquete contiene la lógica específica de la aplicación, incluyendo actividades principales, fragmentos, adaptadores y archivos de diseño.

- **MainActivity.kt:** Controla la actividad principal de la aplicación, gestiona la navegación entre fragmentos y elementos principales de la interfaz.
- **GameListActivity.kt:** Encargado de mostrar una lista de juegos disponibles, interactuando con la capa de datos para mostrar la información adecuada.



- **GameInfoActivity.kt:** Presenta los detalles de un juego específico cuando se selecciona de la lista.
- **LoginActivity.kt:** Controla el flujo de inicio de sesión y autenticación de usuarios en la aplicación.
- **SearchActivity.kt:** Maneja la búsqueda de juegos dentro de la aplicación.

- entidades

Contiene las entidades del dominio de la aplicación, como juegos, usuarios, géneros, etc. Estas entidades representan los datos principales manipulados por la aplicación y se utilizan para interactuar con la capa de persistencia y servicios externos.

- interfaces

Incluye interfaces y contratos que definen las interacciones entre diferentes componentes de la aplicación, como interfaces de usuario, interfaces de datos y autenticación.

- **GameInfoInterface.kt**

Contiene interfaces relacionadas con la obtención o manipulación de información específica de un juego.

- **GameListInterface.kt**

Contiene interfaces relacionadas con la gestión de listas de juegos.

- **LoginInterface.kt**

Contiene interfaces relacionadas con el proceso de inicio de sesión en la aplicación.

- **PlatformInterface.kt**

Contiene interfaces relacionadas con la gestión de plataformas de juegos.

- sql

Contiene archivos y utilidades relacionadas con la capa de acceso a datos utilizando SQLite o alguna otra base de datos local.

- ui

Este paquete contiene componentes de interfaz de usuario reutilizables, como widgets personalizados, adaptadores y otros elementos visuales.

- viewmodels

En el directorio viewmodels, se encuentran diferentes archivos de ViewModel, los cuales manejan la lógica de presentación y la interacción entre la interfaz de usuario y el repositorio de datos.

- **GameInfoViewModel.kt**

Está asociado a la vista de detalles de un juego específico.

- **GameListViewModel.kt**

Está vinculado a la lista de juegos disponibles en la aplicación.

- **GameScreenshotsViewModel.kt**

Maneja la lógica relacionada con la visualización y presentación de capturas de pantalla de juegos.

- **PlatformInfoViewModel.kt**

Asociado a la información detallada de una plataforma específica.

- **PlatformListViewModel.kt**

Maneja la lógica relacionada con la lista de plataformas disponibles. Gestiona la obtención y presentación de la lista de plataformas.

- **common**

En el directorio común, se encuentran diferentes compartidos por la aplicación.

- **Constant.kt**

Almacena constantes que se utilizan de manera global en la aplicación, como valores estáticos, claves API, o cualquier otro tipo de información constante y compartida.

- **entidades**

Aquí se almacenan archivos que definen distintas entidades o modelos que representan datos específicos dentro del contexto de la aplicación de juegos, tales como representaciones de objetos como juegos, géneros, plataformas, desarrolladores, entre otros.

- **api**

Aquí se almacenan archivos de configuración y conexión con la API de RAWG.

- **API.kt**

Contiene la lógica relacionada con las llamadas a la API de RAWG, como funciones para realizar solicitudes HTTP, manejo de respuestas, definición de rutas o endpoints, entre otras operaciones relacionadas con la interacción con la API en la que se basa la aplicación.

- **HeaderInterceptor.kt**

Define un interceptor para encabezados de solicitud, el cual puede ser utilizado para agregar o manipular encabezados en las solicitudes HTTP que la aplicación realiza hacia el servidor.

Esta es una descripción general de la estructura y los componentes principales de la aplicación, con el fin de proporcionar una visión clara de cómo se organizan y las funciones que desempeñan los distintos módulos y paquetes.

4.2 Pruebas

Pruebas unitarias

Prueba	Descripción	Sprint	Precondiciones	Datos de Entrada	Datos de Salida
RF-08: Registro de Usuarios	Verificar el registro de nuevos usuarios	Sprint 1	Aplicación instalada y abierta	Datos de registro (nombre, email, contraseña)	Confirmación de registro exitoso
RF-08: Inicio de Sesión	Comprobar la autenticación de usuarios	Sprint 1	Usuario registrado previamente	Credenciales de inicio de sesión (email, contraseña)	Acceso a la interfaz principal
RF-01: Búsqueda de Juegos	Validar la búsqueda de juegos en la app	Sprint 2	Base de datos de juegos cargada	Término de búsqueda (nombre de juego)	Listado de juegos coincidentes
RF-03: Agregar Juegos	Verificar la capacidad de añadir juegos	Sprint 2	Usuario autenticado	Detalles del juego (nombre, plataforma, género)	Confirmación de juego añadido
RF-03: Lista de favoritos	Confirmar la funcionalidad	Sprint 3	Juegos agregados por el usuario	Selección de juego para la lista de favoritos	Confirmación de juego añadido a la lista de favoritos

Pruebas de integración

Prueba	Descripción	Sprint	Precondiciones	Datos de Entrada	Datos de Salida
RF-07: Integración con API RAWG	Verificar la conexión e integración con la API RAWG	Sprint 2	Acceso a la API RAWG	Solicitud de datos de juegos desde la API RAWG	Respuesta exitosa con datos de juegos integrados
RNF-01: Integración Front-end y Back-end	Comprobar la comunicación entre el front-end y back-end	Sprint 2	Ambos sistemas funcionando	Interacción del usuario con la interfaz de usuario	Respuestas coherentes de la lógica del servidor

Pruebas de usabilidad

Prueba	Descripción	Sprint	Precondiciones	Datos de Entrada	Datos de Salida
RNF-01: Navegación y Flujo de Usuario	Evaluar la facilidad y lógica del flujo de usuario	Sprint 3	Aplicación instalada	Interacción del usuario con la app	Interfaz intuitiva y navegación coherente
RNF-01: Diseño Responsivo	Verificar la adaptabilidad y usabilidad en diferentes dispositivos	Sprint 3	Dispositivos variados	Acceso desde diferentes dispositivos	Interfaz adaptada y legible en distintas pantallas



Pruebas de rendimiento y estabilidad

Prueba	Descripción	Sprint	Precondiciones	Datos de Entrada	Datos de Salida
RNF-03: Carga y tiempo de respuesta	Evaluar el tiempo de carga y respuesta de la aplicación	Sprint 4	Aplicación bajo carga normal	Acceso a la aplicación	Tiempos de carga aceptables y respuestas rápidas
RNF-04: Estabilidad bajo estrés	Verificar el comportamiento bajo condiciones de estrés	Sprint 4	Generar carga en la aplicación	Alta demanda de solicitudes simultáneas	Estabilidad sin fallos ni cuellos de botella



Pruebas de seguridad

Prueba	Descripción	Sprint	Precondiciones	Datos de Entrada	Datos de Salida
RF-08: Seguridad de datos sensibles	Verificar el manejo seguro de datos sensibles	Sprint 4	Sistema con datos sensibles	Inserción de datos sensibles	Almacenamiento seguro y encriptado de los datos
RF-07: Prevención de inyección de Código	Comprobar la prevención de inyecciones de código	Sprint 4	Sistema con protección	Intento de inyección de código	Bloqueo de intento de inyección de código
RF-08: Autenticación y gestión de sesiones	Validar la autenticación segura y gestión de sesiones	Sprint 4	Sistema con acceso autenticado	Inicio y cierre de sesión	Gestión segura de sesiones y datos autenticado

4.3 Instalación y configuración

Tipo de instalador

Se proporcionará un archivo **APK** (Android Package) para la instalación en dispositivos Android. El procedimiento de instalación será estándar, similar al de otras aplicaciones descargadas desde fuentes externas o la tienda de aplicaciones de Google Play Store.

Configuración para uso personal

Dado que la aplicación es de uso personal, no se necesitarán configuraciones especiales ni requerirá acceso a información restringida o peculiaridades específicas de entornos empresariales. Los usuarios sólo necesitarán seguir los pasos típicos de instalación en sus dispositivos Android.

Proceso de instalación

- Descargar el archivo APK en el dispositivo Android.
- Habilitar la opción "Fuentes desconocidas" en la configuración del dispositivo para permitir la instalación desde fuentes externas a la Play Store (si es necesario).
- Ejecutar el archivo APK y seguir las instrucciones de instalación que aparecerán en pantalla.
- Una vez completada la instalación, la aplicación estará lista para su uso personal en el dispositivo.

En resumen, al ser una aplicación para uso personal, el proceso de instalación será directo y no requerirá consideraciones especiales más allá de las configuraciones estándar de seguridad del dispositivo.

5. Documento de cierre

5.1. Documento de instalación y configuración

Este documento proporciona las instrucciones necesarias para instalar y configurar la aplicación desarrollada como parte del proyecto. Se detallan los requisitos del sistema, el proceso de instalación y las configuraciones iniciales para garantizar una implementación exitosa.

Requisitos del Sistema

Para utilizar la aplicación en dispositivos Android, se requiere:

- Versión mínima de Android: 10 o superior.
- Al menos 20 MB de espacio de almacenamiento disponible.
- Conexión a internet para acceder a funcionalidades que requieren conexión.

Proceso de Instalación

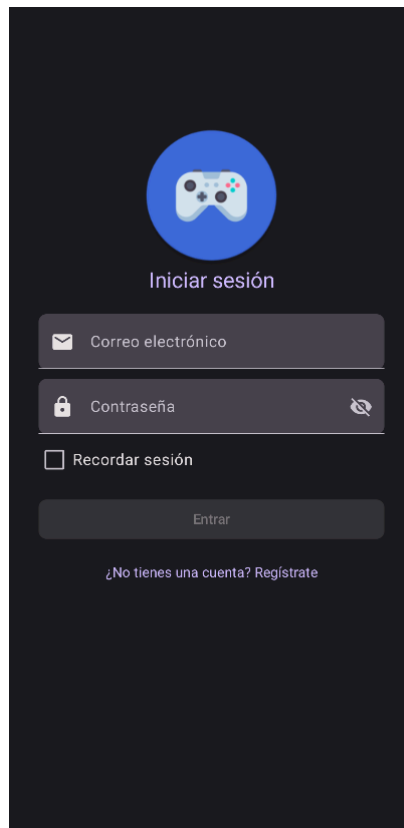
- Descargar el archivo APK:
 - Acceder al enlace proporcionado para descargar el archivo APK de la aplicación.
- Habilitación de fuentes desconocidas:
 - Ajustes > Seguridad en el dispositivo.
 - Habilitar la opción "Fuentes desconocidas" para permitir la instalación desde fuentes externas a la Play Store, si es necesario.
- Instalación de la aplicación:
 - Abrir el archivo APK descargado.
 - Seguir las instrucciones en pantalla para completar la instalación.

Configuración Inicial

Después de la instalación, seguir estos pasos iniciales:

- Inicio de sesión:
 - Ingresar las credenciales o crear una cuenta nueva si es necesario.
-

5.2. Manual de usuario



¡Bienvenido a GameNexus! Con esta aplicación, tendrás acceso a un amplio catálogo de juegos. Descubre los títulos más destacados y gestiona tu lista de juegos favoritos de manera sencilla.

- 1. Inicio de sesión/registro de usuario

Para comenzar a utilizar la aplicación, es necesario iniciar sesión si ya tienes una cuenta o registrarte para crear una nueva:

● Iniciar sesión

- Abre la aplicación e introduce tu correo electrónico y contraseña en los campos correspondientes.
- Haz clic en el botón "Entrar".
- Si los datos son correctos, accederás a la pantalla principal de la aplicación.

Regístrate

Nombre

Correo electrónico

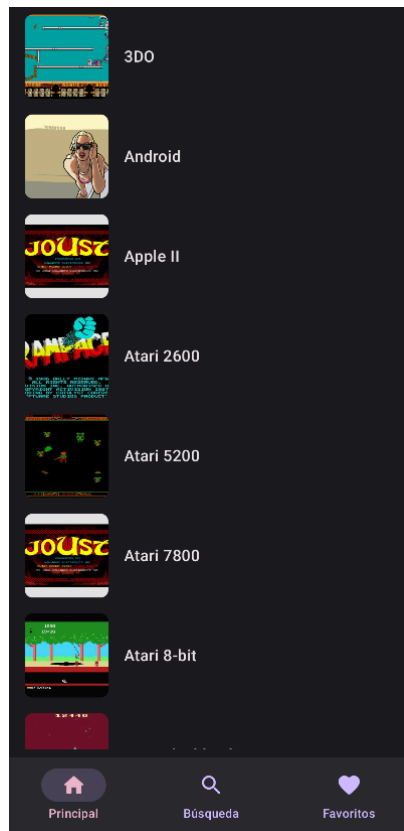
Contraseña

Regístrate

¿Ya tienes una cuenta?

- **Registro de usuario**

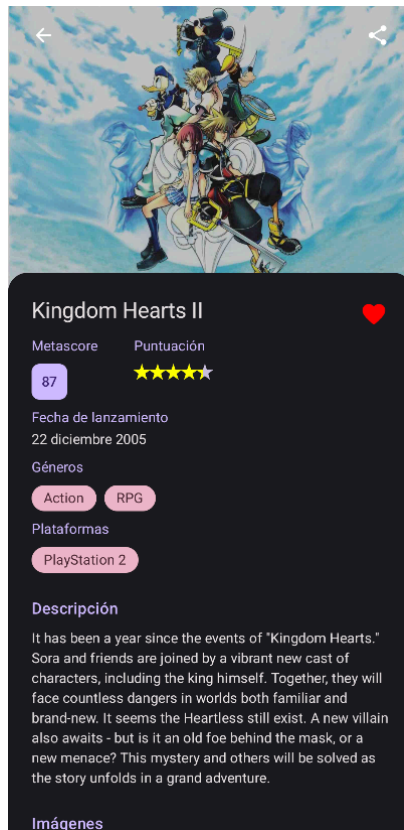
- Si eres nuevo, selecciona la opción "Registrarse" en la pantalla de inicio de sesión.
- Completa los campos requeridos: nombre, correo electrónico y contraseña.
- Haz clic en "Regístrate" para crear tu cuenta.
- Una vez registrado, podrás acceder a la aplicación con tus datos.



- 2. Navegación por el menú

Una vez dentro de la aplicación, encontrarás las siguientes funcionalidades en el menú principal:

- **Búsqueda de juegos por plataforma**
 - Selecciona la opción "Principal" en el menú principal.
 - Elige la plataforma deseada (por ejemplo: PC, PlayStation, Xbox, etc.).
 - Se mostrará una lista de juegos disponibles para esa plataforma.
- **Pantalla de búsqueda**
 - Utiliza la barra de búsqueda para encontrar un juego específico.
 - Ingresa el nombre del juego que estás buscando.
 - Se desplegará una lista con los juegos coincidentes con tu búsqueda.
- **Lista de favoritos**
 - Navega a la sección "Favoritos" en el menú principal.
 - Aquí puedes navegar por los juegos en tu lista de favoritos.
 - Selecciona el juego deseado y utiliza la opción "Agregar a Favoritos".



- 3. Información detallada de un juego

Dentro de cada juego, encontrarás información detallada que incluye:

- **Título del juego:** Nombre del juego en cuestión.
- **Fecha de lanzamiento:** Fecha de estreno del juego.
- **Puntuación:** Valoración o puntuación asignada al juego.
- **Géneros:** Categorías a las que pertenece el juego.
- **Plataformas:** Dispositivos en los que está disponible el juego.
- **Descripción:** Breve resumen o descripción del juego.
- **Galería de imágenes:** Muestra visual del juego a través de imágenes.

Acciones disponibles

Dentro de la visualización de un juego, tendrás acceso a dos acciones adicionales:

- **Compartir en redes sociales**
 - Haz clic en el botón "Compartir" para difundir la información del juego en redes sociales.
- **Añadir/Eliminar de favoritos**
 - Utiliza el botón en forma de corazón para guardar el juego en tu lista de favoritos.
 - Si el juego ya está en tus favoritos, puedes eliminarlo volviendo a hacer clic.

5.3. Resultados obtenidos y conclusiones

Durante el desarrollo de este proyecto, he logrado alcanzar varios hitos significativos que resaltan la funcionalidad y la experiencia del usuario en mi aplicación de búsqueda de juegos.

1. Implementación del sistema de registro y autenticación

Se logró desarrollar un sistema de registro seguro y una interfaz de autenticación eficiente para que los usuarios accedan a la aplicación.

2. Integración exitosa de la API

Se alcanzó la integración exitosa de la API RAWG, permitiendo la obtención de datos detallados sobre una amplia variedad de juegos, incluyendo información relevante como título, fecha de lanzamiento, géneros y plataformas.

3. Desarrollo de funcionalidades de búsqueda avanzada

Se implementó un sistema de búsqueda avanzada que permite a los usuarios buscar juegos por múltiples criterios, como plataforma, género o fecha de lanzamiento, proporcionando una experiencia de búsqueda flexible y personalizada.

4. Gestión de favoritos y almacenamiento de datos

Se logró crear un sistema de gestión de favoritos que permite a los usuarios guardar y gestionar su lista personalizada de juegos favoritos, asegurando un acceso rápido y sencillo.

5. Interfaz de usuario intuitiva y atractiva

Se diseñó una interfaz de usuario intuitiva y atractiva que facilita la navegación y el uso de la aplicación, mejorando la experiencia del usuario y garantizando una interacción más amigable.

Logros y funcionalidades destacadas

He implementado con éxito funciones clave, como el registro de usuarios, la búsqueda detallada de juegos por plataforma y la capacidad de administrar una lista de juegos favoritos. La visualización detallada de cada juego, junto con la posibilidad de compartir en redes sociales y la gestión de favoritos, se ha convertido en un punto fuerte de mi aplicación.

En cuanto a mis objetivos iniciales, he logrado cumplir la mayoría de ellos, proporcionando una experiencia de usuario intuitiva y funcional.

Problemas resueltos

Durante el desarrollo, me enfrenté a desafíos técnicos relacionados con la integración de la API y la optimización de la experiencia del usuario. Pude superar estos obstáculos y mejorar la estabilidad y el rendimiento de la aplicación usando Kotlin y Jetpack Compose.

Pruebas y validación

Las pruebas realizadas han validado la mayoría de las funciones implementadas, destacando la solidez de la funcionalidad de búsqueda y la estabilidad del sistema. Sin embargo, he identificado áreas para futuras mejoras, especialmente en la optimización del rendimiento en dispositivos de menor capacidad.

Conclusiones

Lecciones aprendidas

Este proyecto me ha enseñado la importancia de la planificación detallada y la flexibilidad durante el desarrollo. Aprendí a resolver problemas complejos y a valorar la importancia de las pruebas exhaustivas en cada etapa del proceso.

Valoración general del proyecto

Considero que el proyecto ha sido exitoso en términos de cumplir mis objetivos principales. La aplicación ofrece una experiencia satisfactoria para los usuarios, aunque reconozco la necesidad de mejoras continuas para optimizar aún más su rendimiento y funcionalidades.

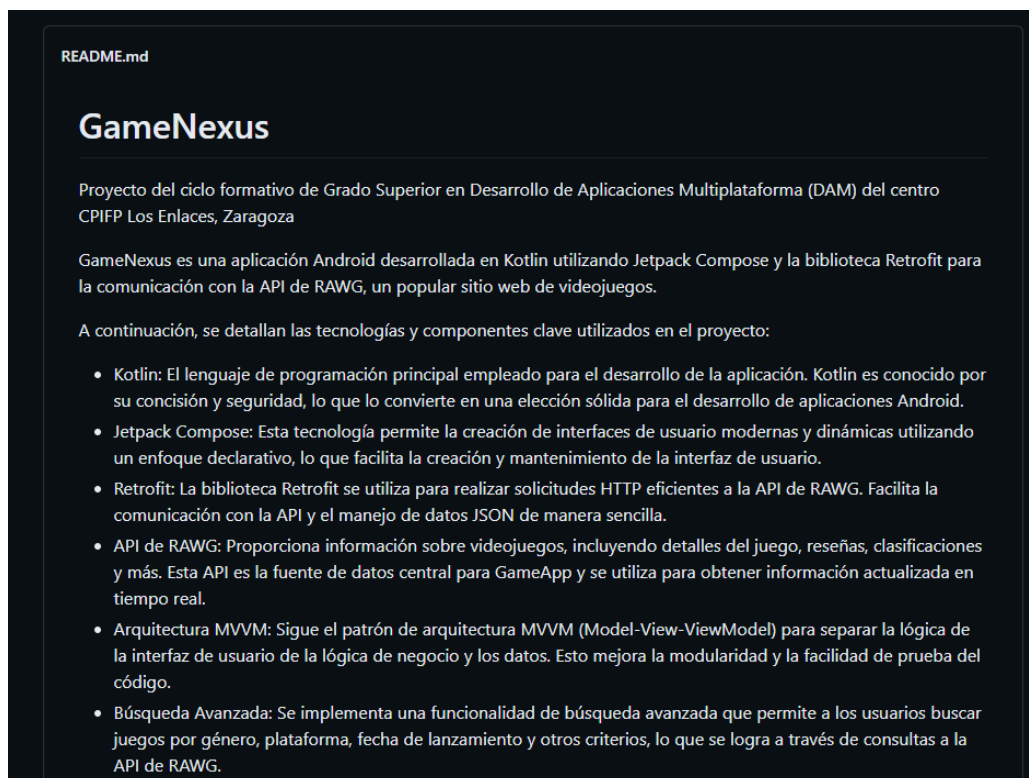
5.4. Cuaderno de bitácora

En cuanto a la planificación inicial del proyecto, se han seguido las pautas en su mayoría. Sin embargo, algunas tareas planificadas han quedado pendientes:

Tareas pendientes:

- Optimización de carga de imágenes, se planea resolver esto en futuras actualizaciones.
- Revisión de estilo en la interfaz de usuario, la revisión se detuvo temporalmente para enfocarse en la implementación de nuevas características. Se retomará para mejorar la estética y la coherencia visual en versiones posteriores.
- Limpieza de código, refactorización, se considera una tarea continua en este aspecto para mantener un código limpio y eficiente a medida que evoluciona el proyecto.
- Scroll infinito en la carga de la lista de juegos. Esta característica se aplazó debido a prioridades de implementación. Se contempla su inclusión en una actualización próxima para mejorar la experiencia de navegación del usuario.

El progreso diario, las decisiones clave y las reflexiones sobre el desarrollo de este proyecto se registran en mi cuenta de github:



<https://github.com/franciscosoler90/GameNexus>



5.5. Bibliografía

Documentación oficial de Android:

<https://developer.android.com/jetpack/compose/components?hl=es-419>

<https://developer.android.com/training/data-storage/room?hl=es-419>

Documentación RAWG API:

<https://api.rawg.io/docs/>

<https://rawgthedocs.orels.sh/api/#basics>

Tutoriales:

<https://www.youtube.com/@mouredev/videos>

<https://devexpert.io/jetpack-compose-que-es/>

<https://cursokotlin.com/curso-jetpack-compose-desde-cero/>
