



Programación de Aplicaciones Móviles Nativas

GameCritic

Autores

David Miranda Campos
Alejandro Guerra Jimenez
Esteban Trujillo Santana

Índice general

1. Introducción	1
2. Descripción de la aplicación	2
3. Diseño	4
4. Arquitectura	13
5. Modelo de Datos	15
6. Desarrollo	18
7. Interfaces	20
8. Conclusiones	22

Índice de figuras

3.1. Log In Screen	5
3.2. Sing Up Screen	6
3.3. Home Screen	7
3.4. Explore Screen	8
3.5. Search Screen	9
3.6. Following Screen	10
3.7. Profile Screen	11
3.8. Profile Screen	12
4.1. Diagrama de arquitectura simplificado de GameCritic.	14

Índice de tablas

5.1. Campos de la colección <code>users</code>	15
5.2. Campos de la colección <code>videogames</code>	16
5.3. Campos de la colección <code>comments</code>	16
5.4. Campos de la entidad <code>RecentSearch</code>	17

Capítulo 1

Introducción

En este documento se recoge el desarrollo de **GameCritic**, una aplicación móvil pensada para organizar, consultar y valorar videojuegos. La idea surge de la necesidad de tener una plataforma sencilla donde los usuarios puedan descubrir nuevos títulos, comparar características y compartir opiniones con otros jugadores.

GameCritic busca ser una herramienta útil y fácil de usar, permitiendo buscar juegos por género o nombre, y acceder rápidamente a información relevante. Además, fomenta la participación de la comunidad, ya que cada usuario puede dejar valoraciones y comentarios, ayudando así a otros a decidir qué jugar.

Para el desarrollo se han utilizado tecnologías actuales como *Kotlin* para la programación en Android, *Jetpack Compose* para crear interfaces modernas y reactivas, *Room* para gestionar datos locales, y *Firebase* para gestionar los datos en la nube y la autenticación de usuarios. Estas herramientas permiten que la app sea robusta, fácil de mantener y preparada para crecer en el futuro.

Durante todo el proceso se ha dado mucha importancia a la usabilidad y a que la experiencia de usuario sea lo más cómoda posible, siguiendo las recomendaciones de diseño de Android. También se ha cuidado la calidad del código y la estructura del proyecto, para que sea sencillo de entender y modificar.

En las siguientes secciones se explican los objetivos del proyecto, la metodología seguida, las decisiones técnicas más importantes y los resultados obtenidos. El objetivo de esta memoria es dejar constancia de todo el trabajo realizado y reflexionar sobre el proceso de crear una app móvil en el mundo de los videojuegos.

Capítulo 2

Descripción de la aplicación

La aplicación **GameCritic** surge de la idea de crear un espacio donde los aficionados a los videojuegos puedan encontrar todo lo que necesitan en un solo lugar. Muchas veces, buscar información sobre un juego, decidir si merece la pena probarlo o simplemente descubrir algo nuevo puede ser un proceso largo y poco intuitivo, sobre todo si tienes que saltar entre diferentes webs, foros o apps. Por eso, con **GameCritic** hemos querido reunir en una sola plataforma la posibilidad de explorar, organizar y valorar videojuegos de forma sencilla y cómoda desde el móvil.

El objetivo principal de la app es que cualquier usuario, tanto si es un jugador experimentado como si acaba de empezar en este mundo, pueda navegar fácilmente entre títulos, buscarlos por géneros o por nombre, y acceder a la información de cada uno. Además, la aplicación está pensada para que la comunidad tenga un papel protagonista: cada usuario puede dejar su valoración, escribir comentarios y ver lo que opinan los demás, creando así una comunidad en constante crecimiento, alimentada por las experiencias de todos.

Uno de los puntos que más nos ha motivado a desarrollar **GameCritic** es la falta de aplicaciones que combinen una buena experiencia de usuario con una comunidad activa y herramientas útiles para descubrir juegos. Muchas apps se quedan cortas en algún aspecto: o bien tienen una interfaz poco intuitiva, o la información está desactualizada, o no permiten interactuar con otros usuarios. Por eso, desde el principio hemos puesto mucho énfasis en que la app sea fácil de usar, visualmente atractiva y que funcione de manera fluida, sin complicaciones ni menús innecesarios.

A nivel técnico, hemos apostado por tecnologías modernas como *Kotlin* y *Jetpack Compose* para el desarrollo en Android, lo que me ha permitido crear interfaces más dinámicas y adaptables. Para la gestión de datos en la nube y la autenticación de usuarios, hemos utilizado *Firebase*, que ofrece una solución robusta y escalable. Para algunos datos locales hemos utilizado *Room*. Todo esto hace que **GameCritic** no solo sea una app útil hoy, sino que esté preparada para incorporar nuevas funciones y adaptarse a lo que los usuarios vayan pidiendo en el futuro.

En definitiva, **GameCritic** es una apuesta por mejorar la forma en la que buscamos, descubrimos y compartimos videojuegos, poniendo siempre al usuario y a la comunidad en el centro. La idea es que cada vez que alguien quiera saber más sobre un juego, encontrar recomendaciones o simplemente compartir su opinión, sepa que puede hacerlo de forma rápida y sencilla desde su móvil.

Objetivos

- **Facilitar la exploración de videojuegos por géneros y categorías.**

Uno de los pilares de **GameCritic** es que sea muy sencillo encontrar juegos que se adapten a los gustos de cada usuario. Para ello, la app organiza los títulos en diferentes géneros y categorías, y permite filtrarlos y ordenarlos según distintos criterios, como popularidad, fecha de lanzamiento o valoración media. Además, se incluyen carruseles temáticos y recomendaciones personalizadas, para que descubrir nuevos juegos sea algo natural y entretenido, sin tener que perder tiempo buscando en diferentes sitios.

- **Fomentar la participación y el intercambio de opiniones en la comunidad.**

Otro objetivo fundamental es que la comunidad tenga un papel activo dentro de la aplicación. Cada usuario puede dejar su valoración sobre los juegos que ha probado, escribir comentarios y responder a las opiniones de otros. De esta forma, se crea un espacio donde compartir experiencias, debatir sobre los puntos fuertes y débiles de cada título y ayudar a otros a decidir qué jugar. Además, las valoraciones y comentarios influyen en la visibilidad de los juegos dentro de la app, haciendo que las recomendaciones sean más relevantes y personalizadas.

- **Ofrecer una experiencia de usuario agradable y una base técnica sólida.**

Desde el principio, hemos querido que **GameCritic** sea una app que apetezca usar, con una interfaz clara, moderna y adaptada a todo tipo de dispositivos. Para conseguirlo, hemos seguido las guías de diseño de Android y hemos utilizado tecnologías que permiten crear interfaces reactivas y adaptables. Además, la estructura técnica de la app está pensada para que sea fácil de mantener y ampliar, utilizando buenas prácticas de desarrollo y patrones de arquitectura que facilitan la incorporación de nuevas funciones y la corrección de errores.

- **Permitir la gestión y organización personalizada de videojuegos.**

La aplicación no solo permite consultar información, sino también organizar los juegos favoritos, crear listas personalizadas y llevar un seguimiento de los títulos jugados o pendientes. Así, cada usuario puede adaptar la app a sus necesidades y tener siempre a mano la información que más le interesa.

- **Asegurar la escalabilidad y la evolución futura de la aplicación.**

Por último, uno de los objetivos a largo plazo es que **GameCritic** pueda crecer y adaptarse a las nuevas tendencias del sector. Por eso, la app está preparada para incorporar nuevas funcionalidades, como integración con redes sociales, notificaciones personalizadas o incluso la posibilidad de organizar eventos y torneos dentro de la comunidad. Todo esto, manteniendo siempre una base técnica robusta y una experiencia de usuario de calidad.

Capítulo 3

Diseño

El diseño de *GameCritic* está pensado para que cualquier usuario, independientemente de su experiencia previa con apps de videojuegos, pueda moverse por la aplicación de forma cómoda y agradable. Se ha apostado por una interfaz moderna, con una paleta de colores suaves pero con suficiente contraste para que todo sea legible incluso en condiciones de poca luz. Los degradados y las esquinas redondeadas no solo aportan un toque actual, sino que también ayudan a que la navegación sea más fluida y visualmente atractiva.

La organización de los contenidos por géneros y el uso de carruseles temáticos hacen que explorar el catálogo de videojuegos sea intuitivo y entretenido. Además, se ha cuidado mucho la jerarquía tipográfica y los espaciados, para que la información esté bien estructurada y sea fácil de localizar de un vistazo. Todo esto contribuye a que la experiencia de usuario sea clara, directa y sin distracciones innecesarias.

Patrones y decisiones

A nivel de arquitectura y patrones de diseño, se han seguido las recomendaciones de Jetpack Compose, aprovechando componentes como **Scaffold** para estructurar las pantallas y organizar los elementos principales (barra superior, contenido, navegación inferior, etc.). La composición de la interfaz se realiza mediante filas y columnas, lo que permite adaptar fácilmente el diseño a diferentes tamaños de pantalla y orientaciones.

La navegación entre pantallas se gestiona con **NavController**, lo que garantiza transiciones suaves y un desacoplamiento adecuado entre las distintas secciones de la app. Esto facilita tanto el mantenimiento como la incorporación de nuevas funcionalidades en el futuro.

Para la presentación de los videojuegos, se ha optado por un patrón de datos agrupados, mostrando los títulos organizados por categorías y géneros. Esto no solo ayuda a comparar y descubrir nuevos juegos, sino que también hace que la experiencia sea más personalizada y relevante para cada usuario. El uso de degradados en la cabecera y barras de color en los títulos refuerza la identidad visual de la aplicación y ayuda a diferenciar claramente cada sección.

Pantallas

A continuación se muestra una descripción breve de las pantallas principales del proyecto:

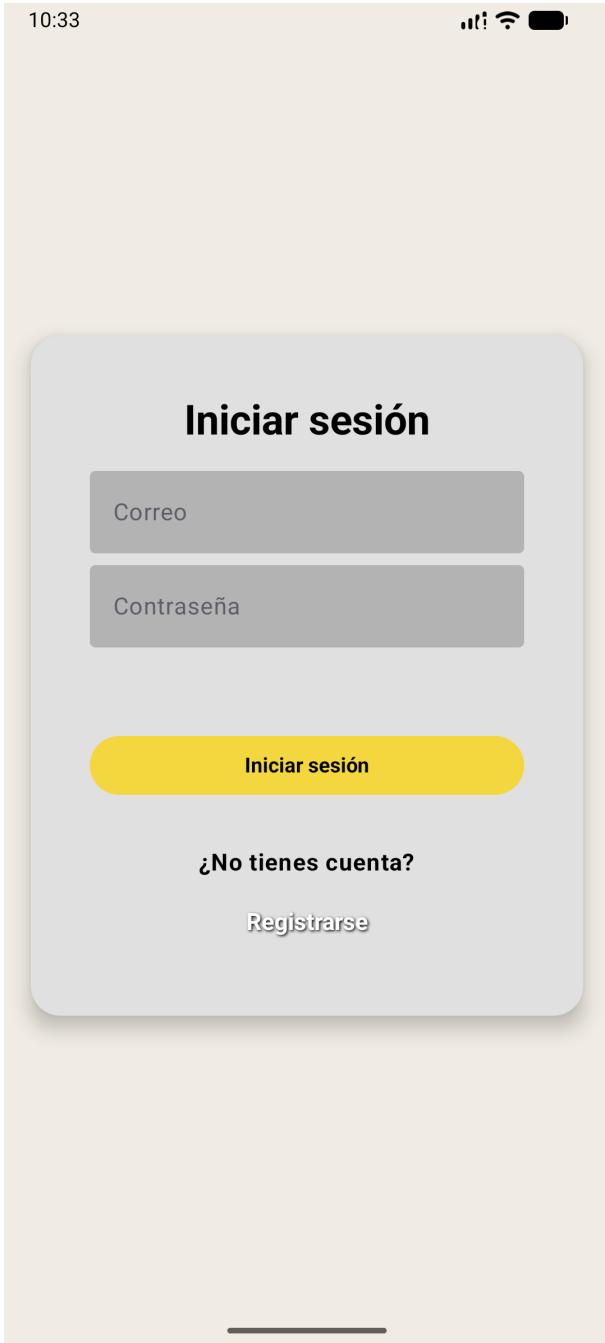


Figura 3.1: Log In Screen

- **Pantalla de inicio de sesión (Log in screen)**

- **Propósito:** Iniciar sesión con un usuario ya existente en la base de datos.
- **Componentes principales:** Un formulario en el que el usuario deberá introducir la información de la cuenta existente.
- **Flujo:** punto de entrada tras ejecutar la aplicación por primera vez, también accesible desde "User Profile Screen" al pulsar en "Cerrar sesión" y desde "Sign Up Screen" al pulsar en "Iniciar sesión".

- Permite navegar a *"Sign Up Screen"* y a *"Home Page Screen"*.

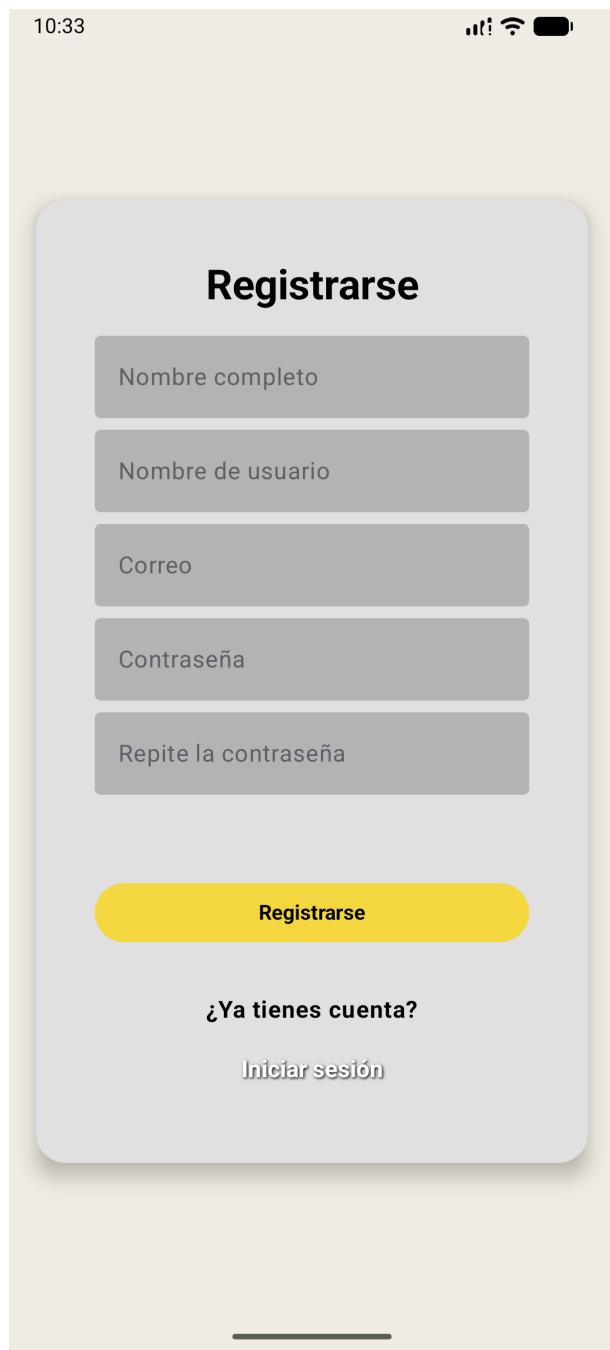


Figura 3.2: Sing Up Screen

- **Pantalla para registrarse (Sing up screen)**

- **Propósito:** Registrar un nuevo usuario en la base de datos.
- **Componentes principales:** Un formulario en el que el usuario deberá aportar la información correspondiente necesaria para crear un nuevo usuario.
- **Flujo:** Accesible desde *"Log In Screen"* pulsando en *"Registrate"*.
 - Permite navegar a *"Log In Screen"* y *"Home Page Screen"*.

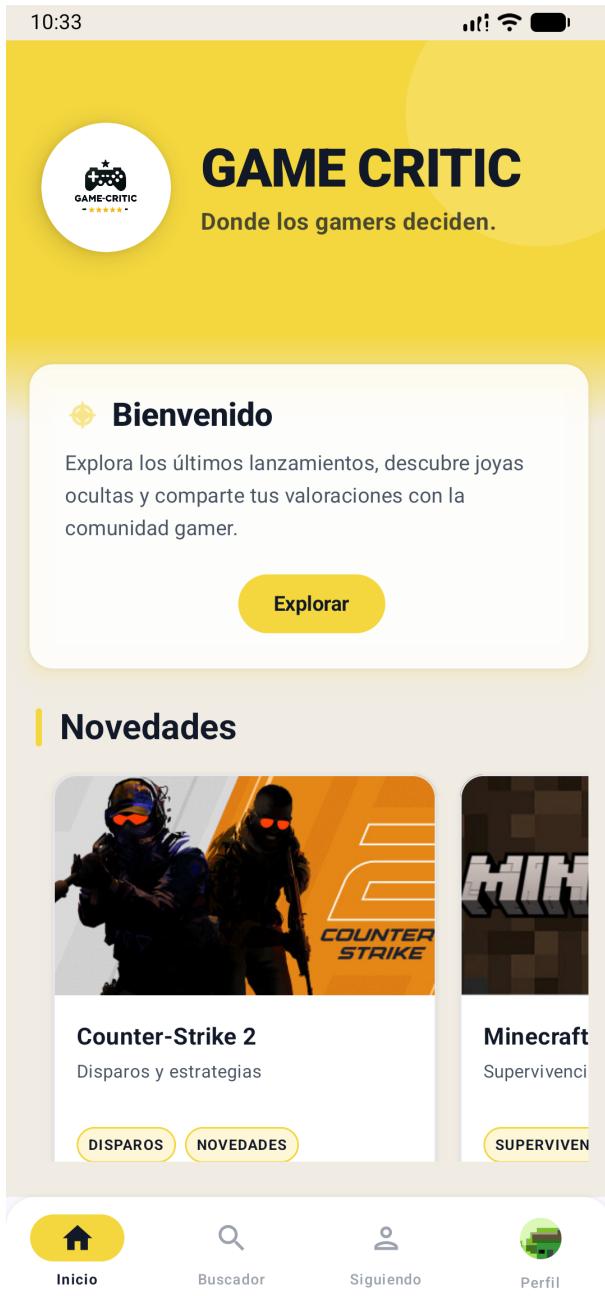


Figura 3.3: Home Screen

- **Pantalla de inicio (Home screen)**

- **Propósito:** hacer una breve presentación a la página y ofrecer recomendaciones al usuario.
- **Componentes principales:** carrusel de juegos destacados, buscador rápido y accesos directos a categorías.
- **Flujo:** punto de entrada tras la autenticación y accesible desde cualquier "Screen" en el footer *Inicio*.
- Permite navegar a "*Explore Screen*" y a las "*Screens*" en el **footer**.

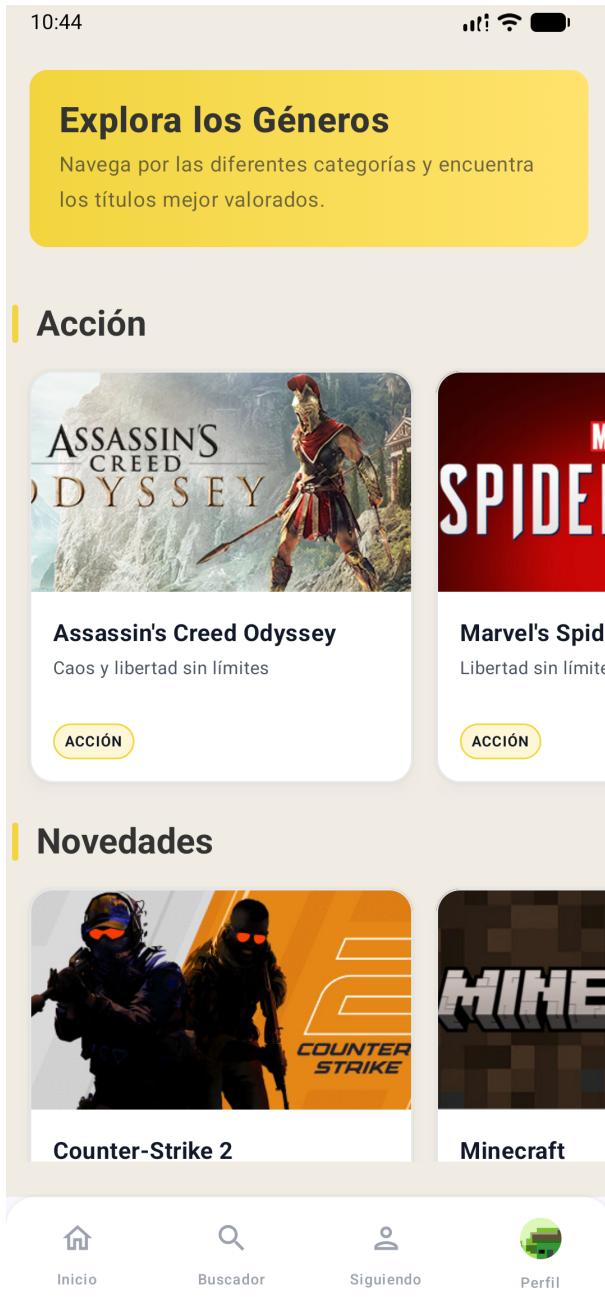


Figura 3.4: Explore Screen

- Pantalla de exploración (Explore screen)
 - **Propósito:** presentar novedades y recomendaciones al usuario por categorías.
 - **Componentes principales:** carruseles de juegos destacados por categorías.
 - **Flujo:** accesible al pulsar "Explorar" en "Home Page Screen".
 - Permite navegar a "Videogame Detail" y a las "Screens" en el footer.

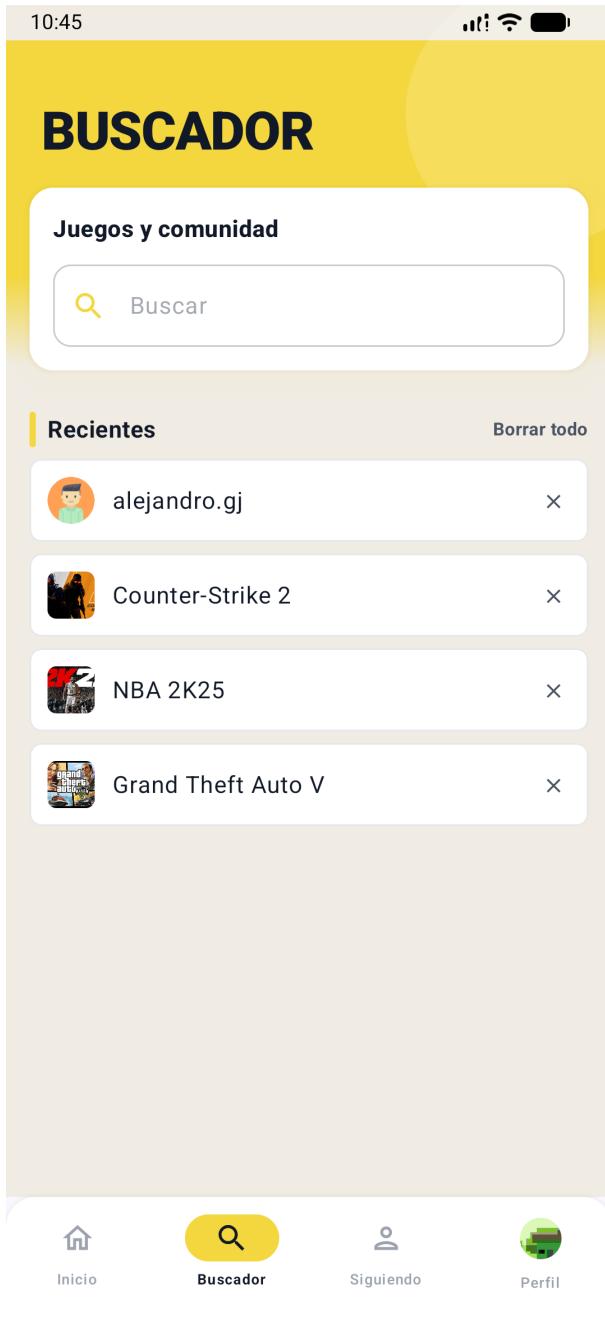


Figura 3.5: Search Screen

- **Pantalla de búsqueda (Search screen)**
 - **Propósito:** buscar videojuegos y aplicar filtros.
 - **Componentes principales:** campo de búsqueda, filtros por plataforma y género, lista de resultados.
 - **Flujo:** accesible desde cualquier "Screen" en el footer al pulsar "Buscar".
 - Permite navegar a "Videogame Detail" y a las "Screens" en el footer.

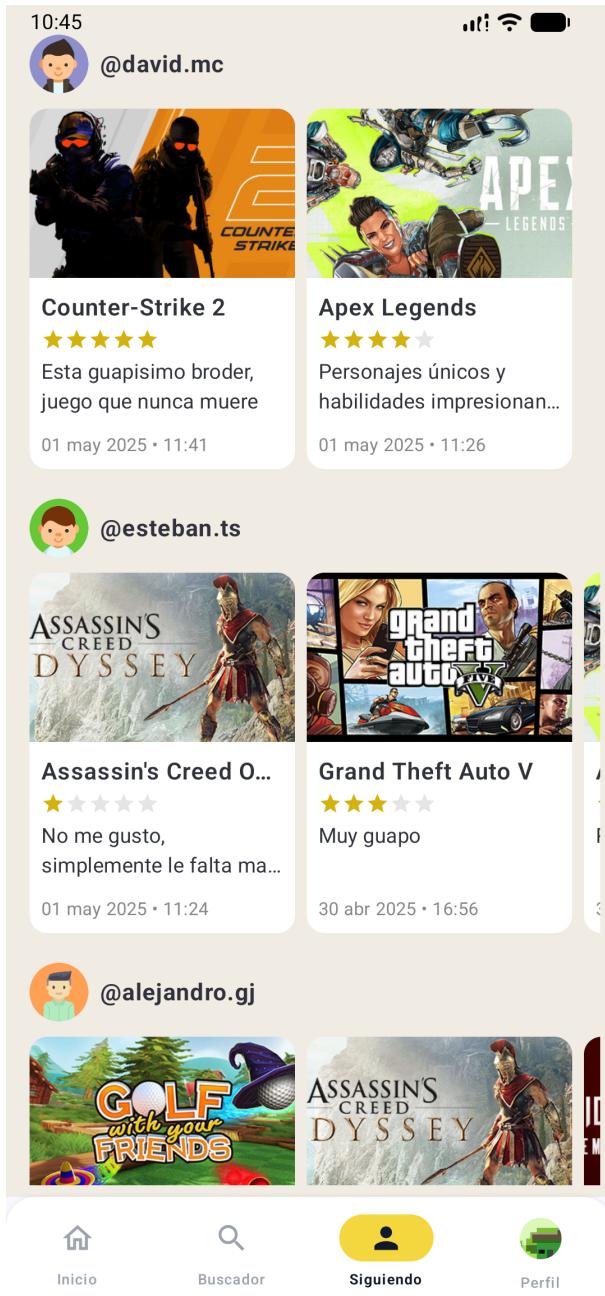


Figura 3.6: Following Screen

- **Pantalla de usuarios seguidos (Following screen)**
 - **Propósito:** gestionar los datos del usuario (información personal, avatar, preferencias).
 - **Componentes principales:** datos del usuario, historial de actividad y accesos a ajustes.
 - **Flujo:** accesible desde cualquier "Screen" en el footer al pulsar "Siguiendo".
 - Permite navegar a "Settings Screen" o gestionar la autenticación.

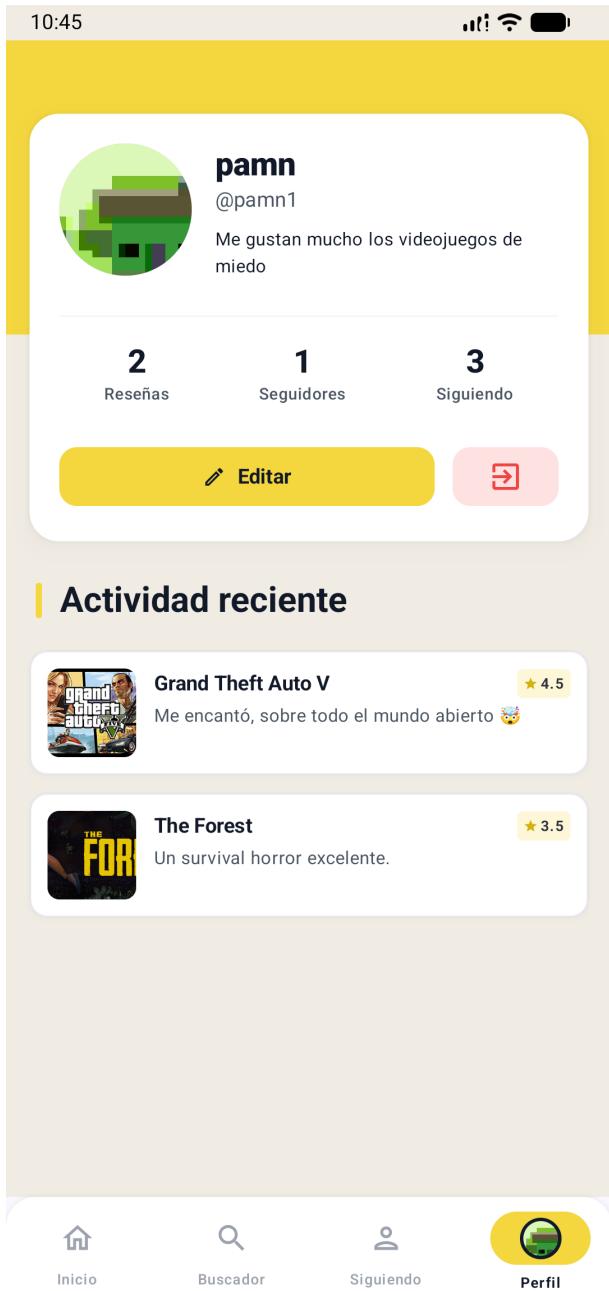


Figura 3.7: Profile Screen

- **Pantalla de perfil (Profile screen)**
 - **Propósito:** gestionar los datos del usuario (información personal, avatar, preferencias).
 - **Componentes principales:** datos del usuario, historial de actividad y accesos a ajustes.
 - **Flujo:** accesible desde cualquier "Screen" en el footer al pulsar "Perfil".
 - Permite navegar a "Settings Screen" o gestionar la autenticación.

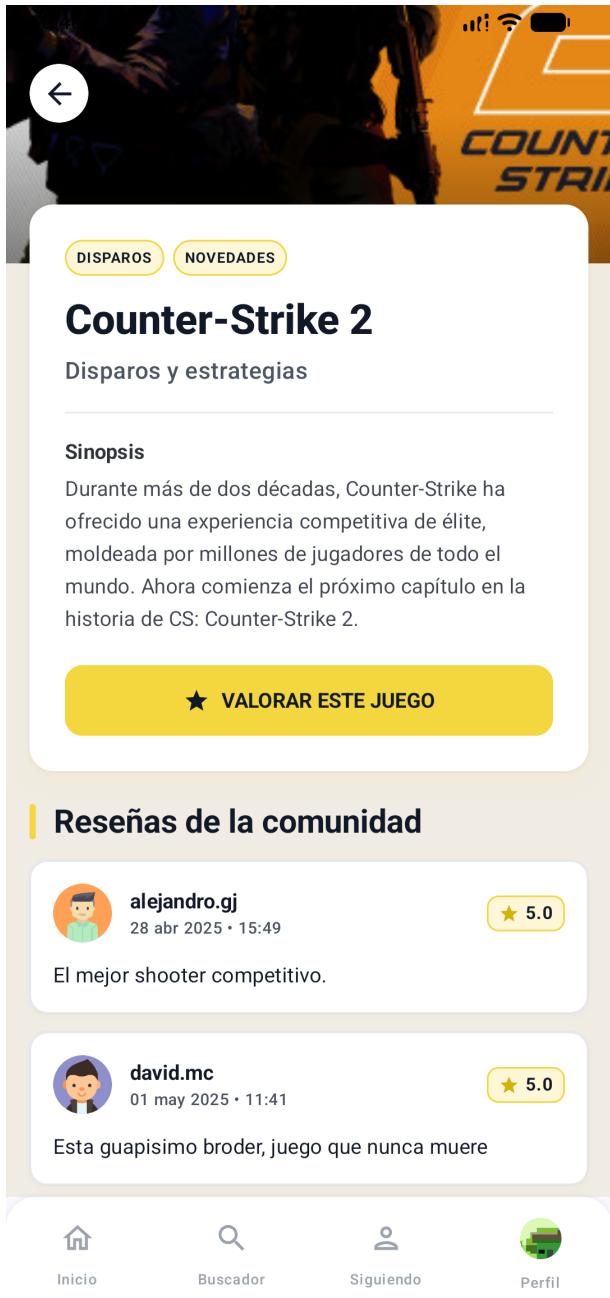


Figura 3.8: Profile Screen

■ Pantalla de Perfil de Videojuego (Videogame Profile Screen)

- **Propósito:** mostrar toda la información relevante sobre un videojuego concreto, incluyendo su descripción, género, desarrollador, fecha de lanzamiento, puntuaciones y valoraciones de la comunidad.
- **Componentes principales:** imagen de portada del juego, ficha técnica (género, plataforma, desarrollador, etc.), resumen o descripción, valoraciones y comentarios de los usuarios, y botones de interacción (añadir a favoritos, valorar, comentar).
- **Flujo:** se accede a esta pantalla al seleccionar un videojuego desde cualquier listado o carrusel de la app.
 - Desde aquí, el usuario puede ver detalles ampliados, leer opiniones, dejar su propia valoración o comentario, y añadir el juego a sus listas personalizadas.

Capítulo 4

Arquitectura

GameCritic sigue un enfoque modular por capas que separa responsabilidades para facilitar escalabilidad, mantenibilidad y pruebas. Se adapta el patrón MVVM: las vistas (View) muestran la UI, los ViewModels orquestan la presentación y la lógica de interacción, y el modelo (Model) agrupa entidades, casos de uso y acceso a datos.

Capa de presentación (UI) — la View en MVVM

- **Responsabilidad:** construir la interfaz y manejar la navegación entre pantallas.
- **En el proyecto:** paquete ui (pantallas como HomeScreen, ExploreScreen, VideogameDetailScreen), components (composables reutilizables) y theme (tema, colores, tipografías).
- **Relación MVVM:** observa el estado expuesto por los ViewModels y reacciona a eventos de usuario, sin contener lógica de negocio.

ViewModels — la capa ViewModel en MVVM

- **Responsabilidad:** exponer estado, transformar datos para la UI y gestionar acciones (cargas, refrescos, navegación lógica).
- **En el proyecto:**.viewmodel (p. ej. VideogameCarouselViewModel, VideogameDetailViewModel).
- **Relación MVVM:** actúan como puente entre la UI y el Model; contienen lógica de presentación pero no acceso directo a fuentes remotas/locales.

Capa de dominio — parte del Model en MVVM

- **Responsabilidad:** modelos de negocio y reglas puras; casos de uso que representan operaciones de alto nivel.
- **En el proyecto:** domain (entidades como Videogame, Comment o User).
- **Relación MVVM:** representa el núcleo lógico que los ViewModels consumen para cumplir requisitos funcionales.

Esta organización (capas + MVVM) garantiza que la interfaz permanezca desacoplada de la lógica y de las fuentes de datos, mejorando la mantenibilidad, testabilidad y la posibilidad de crecer funcionalmente en el futuro.

Diagrama de arquitectura

A continuación se presenta un diagrama de alto nivel que ilustra la estructura y las tecnologías empleadas:

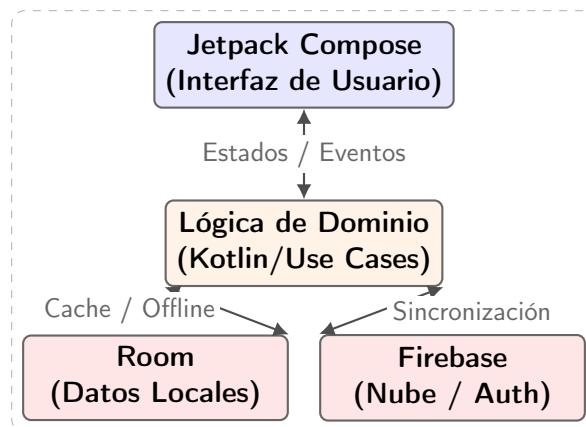


Figura 4.1: Diagrama de arquitectura simplificado de GameCritic.

Capítulo 5

Modelo de Datos

El modelo de datos de la aplicación se apoya en tres colecciones principales en Firestore: `users`, `videogames` y `comments`. Cada colección está diseñada para almacenar la información esencial de la plataforma, facilitando la escalabilidad y la consulta eficiente.

Colección `users`

La colección `users` almacena la información de cada usuario registrado. El identificador principal es el UID proporcionado por Firebase Authentication. Además de los datos básicos, se incluyen listas de seguidores y seguidos para gestionar relaciones sociales dentro de la plataforma.

Tabla 5.1: Campos de la colección `users`

Campo	Tipo	Descripción
<code>id</code>	string	UID del usuario (clave primaria)
<code>name</code>	string	Nombre completo
<code>username</code>	string	Nombre de usuario
<code>email</code>	string	Correo electrónico
<code>profileicon</code>	string	URL o ruta del icono de perfil
<code>description</code>	string	Descripción personal
<code>createdAt</code>	string	Fecha de creación (ISO 8601)
<code>followers</code>	array(string)	Lista de UIDs de seguidores
<code>following</code>	array(string)	Lista de UIDs de seguidos

Colección `videogames`

La colección `videogames` contiene la información de cada videojuego disponible en la plataforma. Cada documento incluye datos descriptivos, imágenes y una lista de categorías o géneros, almacenados como un array de cadenas.

Tabla 5.2: Campos de la colección `videogames`

Campo	Tipo	Descripción
id	string	Identificador único del videojuego
title	string	Título del juego
subtitle	string	Subtítulo o frase descriptiva
description	string	Descripción detallada
category	array(string)	Lista de géneros/categorías
imagecarousel	string	URL de la imagen para carrusel
imageprofile	string	URL de la imagen de perfil

Colección `comments`

La colección `comments` almacena las valoraciones y comentarios que los usuarios realizan sobre los videojuegos. Cada comentario está vinculado tanto a un usuario como a un videojuego mediante sus identificadores. Se almacena la puntuación, el contenido textual y la fecha de creación.

Tabla 5.3: Campos de la colección `comments`

Campo	Tipo	Descripción
id	string/number	Identificador único del comentario
userId	string	UID del usuario que comenta
videogameId	string	ID del videojuego comentado
rating	number	Puntuación numérica
content	string	Texto del comentario
createdAt	string	Fecha de creación (ISO 8601)

Para mejorar la experiencia de usuario y la rapidez en las búsquedas, se utiliza una base de datos local implementada con Room. En este caso, Room no replica las entidades principales de Firestore, sino que se emplea exclusivamente para almacenar las búsquedas recientes realizadas por el usuario. Esto permite mostrar sugerencias rápidas y mantener un historial local, incluso sin conexión a Internet.

Entidad `RecentSearch` (Room)

La entidad `RecentSearch` almacena cada búsqueda reciente, incluyendo información relevante para su visualización y filtrado.

Tabla 5.4: Campos de la entidad `RecentSearch`

Campo	Tipo	Descripción
id	Long?	Clave primaria autogenerada
itemId	String	ID del elemento buscado (usuario, juego, etc.)
displayText	String	Texto mostrado en la búsqueda
timestamp	Long	Fecha/hora de la búsqueda (epoch millis)
tab	SearchTab	Pestaña o tipo de búsqueda (enum)
imageUrl	String	URL de la imagen asociada

En resumen, la arquitectura de datos combina la flexibilidad y escalabilidad de Firestore para la información principal de la aplicación, con la eficiencia de Room para la gestión local de búsquedas recientes. Esta separación permite optimizar tanto la experiencia online como offline, asegurando que los datos críticos estén siempre disponibles y actualizados.

Capítulo 6

Desarrollo

En este apartado se describen las herramientas, lenguajes, estructura del proyecto y decisiones técnicas relevantes adoptadas en el desarrollo de la aplicación.

- **Entorno y build:** El desarrollo se realiza en Android Studio Otter | 2025.2.1 Patch 1 sobre Windows, utilizando Gradle como sistema de construcción y gestión de dependencias. El proyecto está configurado para aprovechar las últimas características del ecosistema Android.

- **Lenguaje:** Kotlin es el único lenguaje utilizado en todo el proyecto, aprovechando sus ventajas en concisión, seguridad y compatibilidad con las bibliotecas modernas de Android.

- **Dependencias y tecnologías principales:**

- **Jetpack Compose:** Framework declarativo para la construcción de interfaces de usuario reactivas y modernas.
- **Firebase:** Utilizado para autenticación de usuarios (Firebase Auth) y persistencia de datos en la nube (Cloud Firestore).
- **Room:** Base de datos local para almacenar búsquedas recientes y mejorar la experiencia offline.
- **Material3:** Conjunto de componentes y estilos para una interfaz visual coherente y actual.
- **JUnit:** Framework de pruebas unitarias para la lógica de negocio y repositorios.

- **Estructura del proyecto:** El código se organiza en los siguientes módulos y paquetes principales:

- **model:** Definición de entidades de dominio como Videogame, User, Comment y RecentSearch.
- **repository:** Orquestación y acceso a datos, implementando el patrón repositorio para desacoplar la lógica de acceso a Firestore y Room.

- **ui**: Pantallas, navegación y lógica de presentación, estructuradas siguiendo el patrón MVVM.
- **components**: Elementos de interfaz reutilizables y personalizados.
- **data.local** y **data.remote**: Capas de persistencia local (Room) y remota (Firebase).

- **Decisiones técnicas:**

- Adopción del patrón MVVM y repositorio para separar claramente la presentación, lógica y acceso a datos, facilitando la mantenibilidad y testabilidad.
- Uso de Jetpack Compose para reducir el boilerplate y permitir interfaces reactivas y adaptables.
- Integración de Firebase para simplificar la autenticación y la sincronización de datos en tiempo real.
- Implementación de Room exclusivamente para búsquedas recientes, evitando redundancia de datos y optimizando la experiencia offline.
- Organización modular y uso de buenas prácticas de arquitectura para facilitar la escalabilidad y futuras ampliaciones.

Esta configuración y las decisiones técnicas adoptadas buscan maximizar la mantenibilidad, testabilidad y velocidad de desarrollo, permitiendo que el proyecto escale de forma eficiente y sostenible.

Capítulo 7

Interfaces

Interfaces principales

1. Pantalla de Inicio

Presenta el logotipo y las acciones principales: registro e inicio de sesión. Incluye llamadas a la acción claras para acceder a la exploración y al contenido destacado.

2. Pantalla de Exploración

Interfaz para navegar por categorías y filtros, con carruseles de títulos destacados y opciones de búsqueda para facilitar el descubrimiento.

3. Detalle de Videojuego

Ficha completa con sinopsis, capturas, especificaciones, calificaciones agregadas y sección de comentarios del usuario.

4. Pantalla de Valoración

Formulario para puntuar y publicar reseñas, con validación de entrada, límites de caracteres y retroalimentación inmediata.

5. Perfil de Usuario

Visualización de información del usuario, historial de valoraciones, opciones de edición y acceso a configuración.

Flujo de navegación

La navegación se gestiona mediante `NavController`, definiendo rutas claras y transiciones coherentes entre pantallas. Ejemplo de flujo típico:

```
inicio → exploración → detalle → valoración → perfil
```

El sistema permite rutas profundas, retornos consistentes y manejo de parámetros para abrir fichas concretas de videojuego.

Accesibilidad

1. Contraste y legibilidad

Selección de paleta y combinaciones de color que aseguren contraste suficiente entre

texto y fondo; tipografías con jerarquía clara.

2. Tamaños y escalabilidad de fuente

Soporte a las preferencias del sistema para tamaño de texto y diseños que se adaptan a distintas escalas sin perder legibilidad.

3. Compatibilidad con lectores de pantalla

Etiquetado semántico de componentes en Jetpack Compose (por ejemplo, `contentDescription`), orden lógico de foco y roles accesibles para todos los elementos interactivos.

4. Áreas táctiles y espaciamiento

Controles con dimensión mínima recomendable y separación suficiente para evitar pulsaciones accidentales; retroalimentación táctil y visual.

5. Internacionalización y localización

Uso de recursos de cadenas (`strings.xml`) para facilitar traducción; formatos locales para fechas, horarios y números.

6. Pruebas y validación de accesibilidad

Estrategia que combina pruebas automáticas (inspección semántica) y pruebas manuales con herramientas como TalkBack, además de revisión por usuarios cuando sea posible.

7. Buenas prácticas adoptadas

Mensajería clara, estados accesibles, manejo de errores comprensible y soporte para configuraciones de alta visibilidad y entradas alternativas.

Capítulo 8

Conclusiones

El desarrollo de GameCritic ha resultado en una plataforma funcional y modular para la valoración y exploración de videojuegos, integrando autenticación, persistencia en la nube y una interfaz moderna basada en Jetpack Compose. El proyecto se ha implementado íntegramente en Kotlin, aprovechando las ventajas de este lenguaje para el desarrollo Android.

Arquitectura y MVVM

La adopción del patrón Model-View-ViewModel (MVVM) ha favorecido la separación de responsabilidades, facilitando el mantenimiento y la escalabilidad. El uso de `ViewModel` para la gestión del estado y el ciclo de vida, junto con corutinas y `StateFlow`, ha permitido un manejo reactivo y predecible de operaciones asíncronas y de la interfaz de usuario.

Calidad y mantenibilidad

La estructura modular del proyecto, con capas bien definidas (model, repository, ui, components, data.local, data.remote), facilita la incorporación de nuevas funcionalidades y el trabajo colaborativo. La arquitectura adoptada permite la automatización de pruebas unitarias y el análisis estático del código, mejorando la calidad y mantenibilidad.

Integración continua (GitHub Actions)

Se ha configurado un pipeline de CI/CD mediante GitHub Actions, que ejecuta builds automáticos, pruebas unitarias y análisis de calidad (lint). Esto reduce riesgos en las entregas y mejora la estabilidad del proyecto.

Limitaciones

El proyecto depende de servicios externos como Firebase para autenticación y persistencia, lo que puede afectar la disponibilidad en caso de incidencias externas. La cobertura de pruebas instrumentadas es limitada y aún no se han implementado funcionalidades avanzadas como recomendaciones personalizadas, filtros complejos o una gestión offline completa. La sincronización de datos en escenarios concurrentes y la mejora de la experiencia offline requieren desarrollo adicional.