Índice

[Contexto de la aplicación 2](#_Toc197697802)

[Mundo real del problema 2](#_Toc197697803)

[Qué aplicaciones existen 2](#_Toc197697804)

[Mejoras de la aplicación respecto a las existentes 2](#_Toc197697805)

[Casos de Uso - F 2](#_Toc197697806)

[Requisitos funcionales y no funcionales 2](#_Toc197697807)

[Funcionales 2](#_Toc197697808)

[No funcionales 3](#_Toc197697809)

[Diseño 3](#_Toc197697810)

[GUI 3](#_Toc197697811)

[UI (vistas) 3](#_Toc197697812)

[UX (usabilidad) 3](#_Toc197697813)

[Diagrama navegación 3](#_Toc197697814)

[Reutilización (fragmentos futuros) 3](#_Toc197697815)

[Arquitectura 3](#_Toc197697816)

[Despliegue 3](#_Toc197697817)

[Componentes 3](#_Toc197697818)

[Base de datos 3](#_Toc197697819)

[Paquetes, Interfaces y Clases 3](#_Toc197697820)

[Plan de pruebas (cómo) 3](#_Toc197697821)

[Implementación 4](#_Toc197697822)

[Entorno de Desarrollo 4](#_Toc197697823)

[Implantación/Puesta en producción 4](#_Toc197697824)

[Capturas de la ejecución de la funcionalidad 4](#_Toc197697825)

[Información sobre la versión y software necesario 4](#_Toc197697826)

[Elementos destacables del desarrollo 4](#_Toc197697827)

[Conclusiones 4](#_Toc197697828)

[Bibliografía 4](#_Toc197697829)

[Anexos 5](#_Toc197697830)

# Contexto de la aplicación

## Mundo real del problema

Muchos jugadores acumulan videojuegos en diferentes formatos y plataformas, lo que dificulta su organización. Esta aplicación busca centralizar toda esta información en un único lugar, resolviendo problemas como la desorganización, dificultad de localización, y falta de estadísticas.

## Qué aplicaciones existen

Actualmente existen algunas aplicaciones que ayudan a gestionar colecciones de videojuegos, como GG App, HowLongToBeat o Backloggd. Estas apps permiten realizar un seguimiento, pero no ofrecen funcionalidades como el almacenamiento físico. Además, todas son aplicaciones web, esta es una alternativa a nivel de aplicación para la gente que prefiera tener la aplicación a mano, y que no sea como puede ocurrir con las otras, que un fallo en la web puede evitar que accedes a tu biblioteca durante mucho tiempo. Aquí únicamente necesitas conexión a internet y que la web donde se aloja la base de datos funcione, no dependes de las web del servidor de la empresa o posibles ataques a la misma.

## Mejoras de la aplicación respecto a las existentes

La aplicación permite localizar juegos físicos, mostrar estadísticas personalizadas (sobre nosotros, como los géneros más jugados, consola para la que más juegos tienes…), además de funcionalidades como exportar colecciones en PDF y conectar con bases de datos en la nube. Y como mencionamos anteriormente, no dependes tanto de factores externos como el servidor de una web.

## Casos de Uso - F

• Inicio de sesión  
• Añadir videojuegos a la biblioteca  
• Añadir videojuegos a wishlist  
• Filtrar juegos de tu biblioteca  
• Exportar la biblioteca

[AÑADIR DIAGRAMAS DE CASOS DE USO AQUÍ]

## Requisitos funcionales y no funcionales

### Funcionales

• Login obligatorio  
• Permitir guardar, añadir y eliminar (No hecho aún) videojuegos.  
• Gestión de wishlist  
• Estadísticas  
• Exportación a PDF  
• Cuentas con bibliotecas propias.

### No funcionales

• Conexión a APIs  
• Conexión a la BBDD en línea  
• Contraseñas hasheadas  
• Escalabilidad

# Diseño

## GUI

### UI (vistas)

Pantallas: login, registro, biblioteca, wishlist, estadísticas, detalle del juego.

### UX (usabilidad)

Diseño con JavaFX y Scene Builder, navegación intuitiva con controladores por cada pantalla.

### Diagrama navegación

[AÑADIR DIAGRAMA DE NAVEGACIÓN ENTRE PANTALLAS AQUÍ]

### Reutilización (fragmentos futuros)

Diseño modular con componentes reutilizables y vistas separadas por FXM.

## Arquitectura

### Despliegue

Aplicación de escritorio con base de datos en la nube.  
[AÑADIR DIAGRAMA DE DESPLIEGUE AQUÍ]

### Componentes

MVC + Spring Boot. Backend arranca junto con JavaFX.  
[AÑADIR DIAGRAMA DE COMPONENTES AQUÍ]

## Base de datos

Base de datos PostgreSQL alojada en Neon.  
[AÑADIR DIAGRAMA ENTIDAD-RELACIÓN AQUÍ]

## Paquetes, Interfaces y Clases

Paquetes: api, dentro de api: retrofit, configuraciones, controlador,modelo, dentro de modelo: ids,repositorio,servicio,utils, resources, vista   
Clases dentro de api: RAWGContenedorPlataforma,RAWGGenero,RAWGPlataforma,RAWGRespuesta,RAWGServicioy RAWGVideojuego.

Clases dentro de retrofit: RAWGCliente

Clases dentro de configuraciones: ConfiguracionBean y ConfiguracionRestTemplate

Clases dentro de controlador: BibliotecaControlador,ControladorBusqueda, LogInControlador y RegistroControlador (faltan clases por crear)

Clases dentro de modelo:

## Plan de pruebas (cómo)

Pruebas unitarias con JUnit, conexión con APIs, pruebas manuales.  
[AÑADIR EVIDENCIA DE PLAN DE PRUEBAS AQUÍ]

# Implementación

## Entorno de Desarrollo

Java 22, Visual Studio Code, Maven, Neon, GitHub.

## Implantación/Puesta en producción

La aplicación se ejecuta localmente mediante Maven con conexión a la nube.

# Capturas de la ejecución de la funcionalidad

[AÑADIR CAPTURAS DE EJECUCIÓN Y PRUEBAS FUNCIONALES AQUÍ]

# Información sobre la versión y software necesario

• Java 22  
• Visual Studio Code  
• Neon DB  
• Maven  
• GitHub  
• Scene Builder  
• Conexión a Internet para uso completo

# Elementos destacables del desarrollo

• Integración con APIs  
• Base de datos en la nube  
• Estadísticas y exportación  
• Uso de JavaFX moderno  
• Modularidad y escalabilidad

# Conclusiones

[AÑADIR CONCLUSIONES AL FINAL DEL PROYECTO]

# Bibliografía

• OpenAI - ChatGPT - https://chatgpt.com/  
• Google Search - https://www.google.com/  
• Gemini AI - https://gemini.google.com/  
• Stack Overflow - https://stackoverflow.com/  
• Neon Docs - https://neon.tech/docs  
• Java Documentation - https://docs.oracle.com/en/java/  
• RAWG API - https://rawg.io/apidocs

# Anexos

[AÑADIR CÓDIGO, DIAGRAMAS, DATOS EXTRA EN ESTA SECCIÓN]