# TRABAJO DE FIN DE GRADO DE INGENIERÍA INFORMÁTICA – JAVA EE

## “CODESFORGAMERS”

**Eduardo Puga Hernández**

Grado de Ingeniería Informática – TFG JAVA EE

**Antoni Oller Arcas**

19 de enero de 2024



**B) GNU Free Documentation License (GNU FDL)**



**C) Copyright**

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | CodesForGamers |
| **Nombre del autor:** | Eduardo Puga Hernández |
| **Nombre del consultor/a:** | Antoni Oller Arcas |
| **Fecha de entrega (mm/aaaa):** | 19/01/2024 |
| **Titulación:** | Grado de Ingeniería Informática |
| **Área del Trabajo Final:** | Java EE |
| **Idioma del trabajo:** | Castellano |
| **Palabras clave** | J2EE, Spring Boot, Thymeleaf, Aplicación Web |


**Resumen del Trabajo (máximo 250 palabras):**

Este proyecto de fin de grado en Ingeniería Informática tiene como objetivo principal la planificación y ejecución de la creación de una tienda en línea especializada en la venta de códigos de videojuegos. Para lograr este propósito, se emplearán herramientas y tecnologías avanzadas, aprovechando los conocimientos adquiridos a lo largo del programa académico.

Las etapas de planificación, análisis y desarrollo han sido fundamentales, aplicando competencias obtenidas en diversas asignaturas del grado. La gestión de proyectos, ingeniería de requisitos, programación orientada a objetos, análisis y diseño con patrones, así como ingeniería de componentes distribuidos, son solo algunas de las disciplinas que se han integrado en el proceso.

La elección de este proyecto se sustenta en la combinación de la pasión del alumno por el mundo de los videojuegos y la oportunidad de desarrollar una aplicación web para la venta online de estos. Esta sinergia entre interés personal y aplicación práctica de conocimientos representa un enfoque enriquecedor, no solo desde el punto de vista académico, sino también como una experiencia de inmersión en el ámbito profesional.

El proyecto busca no solo alcanzar los objetivos establecidos, sino también proporcionar al estudiante una oportunidad para aplicar de manera integral los conocimientos teóricos adquiridos, consolidando así su formación en Ingeniería Informática. Con esta iniciativa, se pretende no solo satisfacer las expectativas académicas, sino también contribuir al desarrollo de habilidades y competencias esenciales en el ámbito laboral relacionado con la tecnología y el comercio electrónico.

**Abstract (in English, 250 words or less):**

This final year project in Computer Engineering aims to plan and execute the creation of an online store specialized in selling video game codes. To achieve this goal, advanced tools and technologies will be employed, leveraging the knowledge acquired throughout the academic program.

The stages of planning, analysis, and development have been crucial, applying competencies gained in various subjects of the degree. Project management, requirements engineering, object-oriented programming, analysis, and design with patterns, as well as distributed component engineering, are just a few of the disciplines integrated into the process.

The choice of this project is based on combining the student's passion for the world of video games and the opportunity to develop a web application for their online sale. This synergy between personal interest and the practical application of knowledge represents an enriching approach, not only from an academic perspective but also as an immersive experience in the professional field.

The project aims not only to achieve the established objectives but also to provide the student with an opportunity to comprehensively apply the theoretical knowledge acquired, thereby consolidating their education in Computer Engineering. With this initiative, the goal is not only to meet academic expectations but also to contribute to the development of essential skills and competencies in the workplace related to technology and e-commerce.

# Índice

## Lista de ilustraciones

# 1 Introducción

En el desarrollo de este trabajo, se presenta una estructura que guía la exploración y comprensión detallada de la creación de una tienda en línea especializada en la venta de códigos de videojuegos.

En el primer capítulo, se aborda la motivación y los antecedentes que dieron origen a este proyecto, destacando la fusión de la pasión por los videojuegos con la aplicación práctica de conocimientos adquiridos en el ámbito académico.

El segundo capítulo se centra en la descripción del proyecto, delineando sus objetivos principales y características fundamentales. A continuación, se presenta un perfil tecnológico exhaustivo en el tercer capítulo, detallando las herramientas, lenguajes y frameworks utilizados en el desarrollo de la aplicación, así como también las decisiones tomadas durante el diseño de la aplicación.

El cuarto capítulo, se detalla la implementación del proyecto, abordando las APIs externas y las APIs desarrolladas. También, se desarrolla la seguridad de la plataforma, describiendo las medidas implementadas para salvaguardar la información y garantizar la integridad de la aplicación. Por último, se explora el programa de versionado y control de código adoptado para garantizar la eficiencia y consistencia en el desarrollo de la aplicación.

En la quinta sección, se describe un manual de uso en el cual se explican todos los pasos necesarios para poner en marcha el producto final del trabajo desarrollado. Posteriormente, en el sexto capitulo, se lista las posibles mejoras e implementaciones futuras a desarrollar.

Para concluir, se presentan las conclusiones derivadas de la ejecución del proyecto de fin de grado. A lo largo de esta experiencia, el estudiante ha alcanzado diversas percepciones y reflexiones que resultan fundamentales para comprender el alcance y las implicaciones de su trabajo.

## 1.1 Contexto y justificación

La iniciativa para desarrollar este TFG, trabajo de final de grado, surge de la ambición por fusionar la fascinación por los videojuegos con la aplicación práctica de los conocimientos adquiridos durante el programa académico.

La esencia de este proyecto reside en la creación de una tienda en línea especializada en la venta de códigos de videojuegos. Esta elección se fundamenta en la observación de la creciente cultura de compras online y la evolución de las preferencias de consumo, especialmente en el ámbito de los videojuegos.

A lo largo de las etapas de planificación, análisis y desarrollo, se han aplicado habilidades y conocimientos provenientes de diversas asignaturas del grado, abarcando desde la gestión de proyectos hasta la programación orientada a objetos. Este enfoque interdisciplinario tiene como objetivo no solo alcanzar los objetivos académicos, sino también crear una aplicación web que responda a las demandas de la comunidad *Gamer*.

La visión estratégica de este proyecto se centra en la creación de una plataforma de compra de videojuegos que no solo sea funcional, sino también estéticamente atractiva para las nuevas generaciones. Se busca ofrecer una experiencia de usuario intuitiva y moderna, capitalizando la convergencia de la pasión personal por los videojuegos con la oportunidad de contribuir al mundo del comercio electrónico.

En resumen, este TFG representa un esfuerzo por aplicar conocimientos teóricos a la creación de una solución práctica y relevante en el contexto actual, proporcionando una experiencia de compra en línea especializada y adaptada a la dinámica del mercado de videojuegos.

## 1.2 Descripción del proyecto

El objetivo principal del proyecto se basa en realizar una aplicación web, en la cual se puedan comprar, por parte del usuario básico, y vender, por parte del administrador o propietario de la aplicación, códigos digitales de videojuegos.

Con esta aplicación, nos haremos un hueco en el mercado emergente de la venta de códigos de videojuegos y podremos acercar las distribuidoras al comprador final haciendo de intermediario.

Según las características de la aplicación y tratándose de venta online, el proyecto se realizará mediante una página web ya que se considera el método primordial donde se realizan compras por internet.

## 1.3 Lógica de negocio

La lógica de negocio de vender códigos para canjear en plataformas de videojuegos se basa en ofrecer a los usuarios acceso inmediato a contenido digital. La operación implica adquirir códigos de videojuegos a través de proveedores autorizados y luego venderlos a los usuarios finales. Los clientes compran estos códigos en una plataforma en línea especializada, seleccionando los títulos deseados.

Una vez completada la compra, los clientes reciben los códigos de canje, que son claves únicas asociadas a los videojuegos seleccionados. Estos códigos permiten a los usuarios desbloquear y descargar los juegos directamente desde plataformas reconocidas, como *Steam* o *Epic Games*. La entrega de los códigos puede ser instantánea, lo que brinda a los usuarios la capacidad de disfrutar rápidamente de su compra.

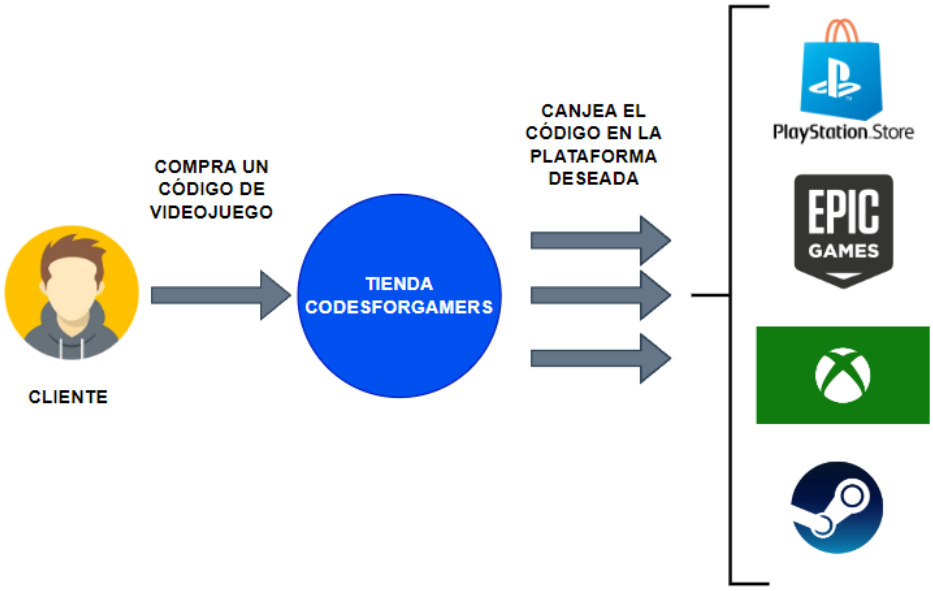


*Ilustración 1 - Lógica de Negocio*

Como se detalla en la *Ilustración 1*, la lógica de negocio se centra en la experiencia de compra del cliente. En este proceso, el cliente adquiere un juego en nuestra tienda en línea especializada y, posteriormente, realiza el canje del código asociado en la plataforma deseada para desbloquear y descargar el juego seleccionado.

Esta lógica de negocio se beneficia de la tendencia hacia el contenido digital, proporcionando a los consumidores una forma conveniente de adquirir y disfrutar de videojuegos sin esperas. Además, la flexibilidad y accesibilidad de los códigos de canje contribuyen a una experiencia de usuario ágil y satisfactoria.

## 1.4 Planificación en el tiempo

Para realizar una buena planificación del tiempo dedicado hacia el proyecto hay que tener en cuenta varias variables.

Una de ellas, si es la más importante, es el marco temporal en el cual podremos desarrollar el proyecto, es decir, desde la fecha inicial hasta la fecha final de entrega. Debido a que estos tiempos son establecidos firmemente, hará que el proyecto se tenga que ajustar e integrar al calendario.

**Calendario**

| Actividad | Fecha inicio | Fecha entrega |
|---|---|---|
| PEC 1 | 27/09/23 | 11/10/23 |
| PEC 2 | 12/10/23 | 11/11/23 |
| PEC 3 | 12/11/23 | 02/01/24 |
| Memoria y Presentación | 03/01/24 | 19/01/24 |
| Defensa | 24/01/24 | 31/01/24 |

*Ilustración 2 - Calendario entregas*

Otra variable fundamental en cuanto al uso del tiempo será la curva de aprendizaje de las tecnologías a implementar. Dependiendo del esfuerzo que se deba realizar, esto también se verá reflejado en las horas previstas y marcadas.

A continuación, se muestra una planificación del tiempo inicial mediante un diagrama de Gantt.


Ilustración 3 - Planificación temporal

# 2 Exploración

La fase de exploración es crucial en el proceso de desarrollo de un proyecto, ya que establece las bases y los parámetros clave que guiarán la creación del Producto Mínimo Viable (MVP). En este contexto, se abordarán los requerimientos principales, identificación de actores y casos de uso, elementos esenciales que definirán la funcionalidad y la interacción dentro de la aplicación. La exploración meticulosa de estos aspectos garantiza una comprensión clara de los objetivos y facilita la toma de decisiones estratégicas en la etapa inicial del proyecto. A continuación, se detallarán los requerimientos principales, los actores involucrados y los casos de uso para orientar el desarrollo del MVP.

## 2.1 Requerimientos principales (MVP)

La identificación y enumeración de estos requerimientos principales se basa en la construcción de un Producto Mínimo Viable (MVP) que satisfaga las necesidades esenciales de los usuarios y garantice la viabilidad del proyecto. Cada uno de estos elementos cumple un papel fundamental en la creación de una aplicación funcional y atractiva para la venta de códigos de videojuegos. Aquí se presentan las justificaciones para incluir estos requisitos en el desarrollo del MVP:

- **Gestión de usuarios:** La creación y modificación de usuarios son esenciales para establecer cuentas individuales, lo que permite una experiencia personalizada y facilita la gestión de compras y preferencias. La administración por parte de un administrador garantiza el control y la seguridad del sistema.
- **Gestión de productos:** La capacidad de añadir, modificar y eliminar productos, así como gestionar el stock, es vital para mantener actualizado el catálogo y ofrecer una variedad constante de opciones a los usuarios. Esto asegura la dinámica y competitividad de la plataforma.

- **Buscador de productos:** Un buscador eficiente facilita a los usuarios encontrar rápidamente los productos deseados, mejorando la experiencia de navegación. La búsqueda por categorías y etiquetas proporciona una organización eficaz del catálogo.
- **Sistema de carrito de compra:** La funcionalidad de compra es esencial para la operatividad de la tienda en línea. Permite a los usuarios seleccionar y adquirir productos de manera conveniente, mientras que la capacidad de modificar el carrito proporciona flexibilidad durante el proceso de compra.
- **Sugerencias de compras:** Ofrecer sugerencias basadas en el historial de compras de los usuarios mejora la experiencia personalizada y fomenta compras adicionales. Esto contribuye a la retención de clientes y aumenta las oportunidades de venta.
- **Notificación por nuevo stock:** La notificación por correo electrónico sobre el restablecimiento de stock satisface las expectativas de los usuarios que desean adquirir un producto específico. Esta funcionalidad mejora la comunicación y la satisfacción del cliente.
- **Gestión de alertas:** Permitir a los usuarios crear y eliminar alertas de stock garantiza una mayor interactividad y personalización. Los clientes pueden recibir notificaciones sobre productos específicos, mejorando la eficiencia de compra.

En resumen, la inclusión de estos requisitos principales en el MVP refleja una estrategia centrada en el usuario, asegurando una experiencia completa y funcional desde la creación de cuentas hasta la gestión de compras y la interacción con el catálogo de productos. Estos elementos fundamentales son cruciales para el éxito inicial del proyecto y sientan las bases para futuras mejoras y expansiones del producto.

## 2.2 Storyboard o guion gráfico

Para hacer una mejor ilustración de las funcionabilidades y/o requerimientos anteriores, a continuación, se describirán en forma de historieta o guion gráfico para así ejemplificar lo descrito anteriormente.

*"Javi quiere comprarse el nuevo juego de moda y decide que la mejor manera es haciendo uso de nuestra aplicación. Una vez en la página principal, escribe el nombre del juego que quiere comprar y le da a buscar. Una vez mostrado los resultados observa que ha encontrado el juego que quería y le hace clic. A continuación, se le muestra una pantalla con la foto del juego con una breve descripción, un desplegable para seleccionar la plataforma en la que desea canjear el código y un botón de "comprar". Después de seleccionar la plataforma y hacer clic en "comprar" la página detecta que no es un usuario registrado y le hace registrarse si desea continuar con su compra. Después de registrarse, se le muestra el carrito de su orden de compra de forma detallada y un botón para finalizar la compra y otro para seguir comprando. Juan, justo cuando está a punto de finalizar su compra, recuerda que un amigo le dijo que se comprara un videojuego en concreto para jugar juntos. Otra vez más hace una búsqueda encontrando el producto que él desea, pero, esta vez, el producto en cuestión permanece sin stock. Entonces Juan decide crear una alerta para que se le notifique por mail de nuevo stock. Para finalizar, vuelve a mostrar el carrito y decide acabar su compra, donde se le muestra una pasarela de pago. Una vez el pago se ha realizado correctamente, se le muestra el código digital del videojuego para que lo pueda canjear en la plataforma escogida".*

## 2.3 Actores

En la planificación y desarrollo de nuestra tienda de videojuegos en línea, es esencial comprender a fondo la interacción entre los elementos que conforman el sistema y las partes interesadas involucradas en el proceso. El "Modelo de Cosas de Uso y Actores" es una parte fundamental de este análisis, ya que nos proporcionará una representación clara de las entidades clave, los procesos y las interacciones que darán vida a nuestra plataforma.

En este apartado, nos centraremos en identificar y definir a los actores involucrados en el sistema, es decir, las personas, roles u otros sistemas que interactúan con nuestra tienda en línea. También identificaremos las cosas de uso que representan los principales objetos y procesos en el entorno de nuestra tienda de videojuegos. A través de este modelo, lograremos una comprensión más profunda de cómo se llevarán a cabo las operaciones, cómo se gestionará el inventario, cómo se realizarán las transacciones y, en última instancia, cómo brindaremos una experiencia excepcional a nuestros clientes.

En nuestro caso, hemos identificado tres actores clave que interactuarán con nuestro sistema: **el Administrador, el Usuario Registrado y el Usuario No Registrado**. Cada uno de estos actores tiene funcionabilidades similares y otras especificas a su naturaleza de perfil. A continuación, se explican los diferentes perfiles y su rol en nuestra aplicación:

- **Administrador**: El administrador es el encargado de gestionar y supervisar la tienda en línea. Sus funciones incluyen la gestión de productos, la administración de usuarios y la revisión de transacciones. El administrador tiene acceso a herramientas de administración avanzadas para mantener la tienda y garantizar su correcto funcionamiento.

- **Usuario Registrado**: Los usuarios registrados son clientes que han creado una cuenta en nuestra tienda. Tienen la capacidad de buscar, explorar y comprar videojuegos. Además, pueden gestionar sus perfiles, crear alarmas para nuevo stock y revisar su historial de compras. Los usuarios registrados disfrutan de una experiencia personalizada mediante la opción de productos sugeridos según su compras anteriores.

- **Usuario No Registrado**: Los usuarios no registrados son visitantes de nuestra tienda que aún no han creado una cuenta. A pesar de no tener acceso a funciones avanzadas, como la creación de alarmas o el seguimiento de pedidos, pueden explorar nuestro catálogo y ver detalles de productos.

Estos actores representan una parte fundamental de nuestro modelo de interacción. A través del análisis y el entendimiento de sus roles y necesidades, podemos diseñar una experiencia de usuario que atienda a las expectativas y requerimientos de cada uno de ellos. Además, este modelo servirá como base para la creación de casos de uso y flujos de trabajo que garantizarán que nuestra tienda de videojuegos en línea sea accesible, efectiva y atractiva para todos los involucrados.

## 2.4 Casos de uso

Los casos de uso son una parte esencial en la definición y planificación de nuestro proyecto de tienda de videojuegos en línea. Representan situaciones específicas en las que interactúan nuestros actores (Administrador, Usuario Registrado y Usuario No Registrado) con la aplicación. Los casos de uso se centran en cómo los usuarios y el sistema trabajan juntos para lograr ciertos objetivos.

En esta sección, detallaremos una serie de casos de uso que describen las principales funcionalidades y flujos de trabajo que nuestros usuarios experimentarán al utilizar la plataforma.

A continuación, se detalla en forma de listado los casos de uso principales de nuestro proyecto.

**Administrador:**

- **Gestionar Catálogo:** Modificación de datos de los productos (videojuegos).
- **Gestionar Inventario:** Modificación de unidades de los productos (stock).
- **Ver Ventas:** Listado general de ventas realizadas.
- **Gestión de Usuarios:** Control de los usuarios que hayan sido creados


*Ilustración 4 - Casos de uso Admin*

**Usuario No Registrado:**

- **Consultar Catálogo:**

- Buscar Videojuegos por Título.
- Explorar Categorías de Videojuegos.
- Filtrar Resultados de Búsqueda.
- Ver Detalles del Producto.

- **Registrarse en la aplicación:** Registro de un nuevo usuario a partir de un formulario


*Ilustración 5 - Casos de uso Usuario no registrado*

**Usuario Registrado**

- **Consultar Catálogo:**
  - Buscar Videojuegos por Título.
  - Explorar Categorías de Videojuegos.
  - Filtrar Resultados de Búsqueda.
  - Ver Detalles del Producto.
- **Gestionar Carrito de Compras:** Cuando un usuario decide comprar un videojuego específico y añade, elimina o modifica su carrito con productos.

- **Ver Videojuegos Recomendados:** Mostrar videojuegos similares según su historial de compras.

- **Gestionar Alertas para Productos sin Stock:** Cuando un producto no tiene stock, el usuario podrá crear alertas para ser notificado por correo electrónico. Lo mismo si quisiera remover la alerta creada.

- **Gestionar Información del Usuario:** Modificar los datos personales del propio usuario.


*Ilustración 6 - Casos de Uso Usuario Registrado*

**FICHAS DE CASOS DE USO**

En esta sección, se presentan fichas de casos de uso mediante tablas que describen distintos escenarios y acciones dentro del sistema. Estas fichas proporcionan una visión detallada de cómo los diferentes actores interactúan con el sistema y cómo se espera que se comporte en respuesta a estas interacciones. Cada ficha incluye información esencial, como el nombre del caso de uso, los actores involucrados, la descripción del escenario y las acciones específicas que se llevan a cabo. Estas fichas sirven como herramientas esenciales para comprender y documentar los aspectos funcionales clave del sistema durante la fase de exploración y planificación del desarrollo.

| **Caso de Uso** | Gestionar Catálogo | **Actores Principales** | Administrador |
|---|---|---|---|
| **Descripción** | El administrador modifica los datos de los productos (videojuegos) en el catálogo de la tienda en línea. | | |
| **Precondiciones** | El administrador ha iniciado sesión en la plataforma de administración y accede a la sección de gestión del catálogo. | | |
| **Postcondiciones** | Los datos de los productos se actualizan en el catálogo. | | |
| **Flujo Básico** | 1. El administrador selecciona la opción "Gestionar Catálogo" en el panel de administración.<br>2. El sistema muestra una lista de productos disponibles.<br>3. El administrador puede realizar una o más de las siguientes acciones:<br>• Modificar datos del producto (título, descripción, precio, etc.).<br>• Agregar nuevos productos al catálogo.<br>• Eliminar productos del catálogo.<br>4. El sistema registra los cambios realizados por el administrador. | | |

| | |
|---|---|
| **Flujos Alternativos** | • Si se intenta eliminar un producto, el sistema muestra una confirmación para evitar eliminaciones accidentales |

| | | | |
|---|---|---|---|
| **Caso de Uso** | Gestionar Inventario | **Actores Principales** | Administrador |
| **Descripción** | El administrador modifica las unidades de los productos (stock) en el catálogo de la tienda en línea. | | |
| **Precondiciones** | El administrador ha iniciado sesión en la plataforma y accede a la sección de gestión del inventario. | | |
| **Postcondiciones** | Las unidades de los productos se actualizan en el catálogo. | | |
| **Flujo Básico** | 1. El administrador selecciona la opción "Gestionar Inventario" en el panel de administración.<br>2. El sistema muestra una lista de códigos disponibles con información.<br>3. El administrador puede eliminar códigos y crear nuevos.<br>4. El administrador realizar la acción que desea<br>5. El sistema registra los cambios realizados por el administrador. | | |
| **Flujos Alternativos** | • | | |

| | | | |
|---|---|---|---|
| **Caso de Uso** | Ver Ventas | **Actores Principales** | Administrador |
| **Descripción** | El administrador ve un listado de las compras de un usuario en concreto | | |
| **Precondiciones** | El administrador ha iniciado sesión en la plataforma de administración.<br>El usuario sobre el que busca las compras existe y ha realizado mínimo una compra. | | |
| **Postcondiciones** | El administrador visualiza un listado de ventas realizadas. | | |

| | |
|---|---|
| **Flujo Básico** | 1. El administrador selecciona la opción "Ver Ventas" en el panel de administración.<br>2. El sistema pide que se introduzca el email del usuario el cual se pretende buscar sus compras.<br>3. El sistema retorna las compras del usuario en cuestión. |
| **Flujos Alternativos** | • |

| | | | |
|---|---|---|---|
| **Caso de Uso** | Gestión de Usuarios | **Actores Principales** | Administrador |
| **Descripción** | El administrador tiene el control sobre los usuarios creados en la plataforma | | |
| **Precondiciones** | El administrador ha iniciado sesión en la plataforma de administración. | | |
| **Postcondiciones** | El administrador visualiza un listado de usuarios y puede eliminar usuarios | | |
| **Flujo Básico** | 1. El administrador seleccionar la opción “Gestionar Usuarios” en el panel de administración.<br>2. El sistema muestra los usuarios creados en la plataforma y da la opción de eliminar uno en especifico | | |
| **Flujos Alternativos** | • Si se intenta eliminar un usuario, el sistema muestra una confirmación para evitar eliminaciones accidentales. | | |

| | | | |
|---|---|---|---|
| **Caso de Uso** | Consultar Catálogo | **Actores Principales** | Usuarios no registrado y usuario registrado |
| **Descripción** | Los usuarios no registrados y registrados tienen acceso a todo el catálogo para ser visualizado de las formas siguientes:<br>• Buscar Videojuegos por Título.<br>• Explorar Categorías de Videojuegos.<br>• Filtrar Resultados de Búsqueda.<br>• Ver Detalles del Producto. | | |
| **Precondiciones** | El usuario accede a nuestra plataforma | | |

| | |
|---|---|
| **Postcondiciones** | El usuario ha visualizado el catálogo con la forma o formas que le ha sido preferida. |
| **Flujo Básico** | 1. El usuario accede a nuestra plataforma y se encuentra en la página de inicio.<br>2. El usuario escoge una de las siguientes opciones para encontrar o ver el producto o productos que le interesan:<br>• Buscar Videojuegos por Título.<br>• Explorar Categorías de Videojuegos.<br>• Filtrar Resultados de Búsqueda.<br>• Ver Detalles del Producto.<br>3. El sistema muestra los resultados de la búsqueda para ser visualizados por el usuario. |
| **Flujos Alternativos** | • Si se intenta encontrar un producto, pero no existe en nuestra base de datos. |

| | | | |
|---|---|---|---|
| **Caso de Uso** | Registrarse en la aplicación | **Actores Principales** | Usuario no registrado |
| **Descripción** | Un usuario no registrado crea una cuenta en la plataforma. | | |
| **Precondiciones** | El usuario no registrado accede a la página de registro de la tienda en línea. | | |
| **Postcondiciones** | El usuario se convierte en un usuario registrado con una cuenta válida en el sistema. | | |
| **Flujo Básico** | 1. El usuario selecciona la opción "Registrarse" en la página de inicio de sesión o registro.<br>2. El usuario completa un formulario de registro, proporcionando información personal como nombre, dirección de correo electrónico y contraseña.<br>3. El sistema verifica que la dirección de correo electrónico no esté asociada con otra cuenta existente.<br>4. El sistema crea una cuenta para el usuario con la información proporcionada. | | |
| **Flujos Alternativos** | • | | |

| **Caso de Uso** | Gestionar Carrito de Compras | **Actores Principales** | Usuario registrado |
|---|---|---|---|
| **Descripción** | El usuario registrado gestiona su carrito de compras, lo que incluye agregar, eliminar o modificar productos en el carrito. | | |
| **Precondiciones** | El usuario ha iniciado sesión en su cuenta de usuario y tiene o no, productos en su carrito de compras. | | |
| **Postcondiciones** | Los productos en el carrito se actualizan según las acciones del usuario. | | |
| **Flujo Básico** | 1. El usuario inicia sesión en su cuenta de usuario.<br>2. El usuario selecciona la opción "Gestionar Carrito de Compras" en su panel de control o perfil.<br>3. El sistema muestra una lista de productos en el carrito con detalles como título, precio y cantidad.<br>4. El usuario puede realizar una o más de las siguientes acciones:<br>• Agregar más unidades de un producto al carrito.<br>• Eliminar productos del carrito.<br>• Modificar la cantidad de unidades de productos en el carrito.<br>5. El sistema registra los cambios realizados por el usuario y actualiza el contenido del carrito. | | |
| **Flujos Alternativos** | • Si el usuario intenta eliminar un producto, el sistema muestra una confirmación para evitar eliminaciones accidentales. | | |

| **Caso de Uso** | Ver Videojuegos Recomendados | **Actores Principales** | Usuario registrado |
|---|---|---|---|
| **Descripción** | El usuario registrado ve una lista de videojuegos recomendados en función de su historial de compras. | | |
| **Precondiciones** | El usuario ha iniciado sesión en su cuenta de usuario y ha realizado compras previas. | | |
| **Postcondiciones** | El usuario visualiza una lista de videojuegos recomendados. | | |

| | |
|---|---|
| **Flujo Básico** | 1. El usuario inicia sesión en su cuenta de usuario.<br>2. El usuario selecciona la opción "Ver Videojuegos Recomendados" en su panel de control o perfil.<br>3. El sistema analiza el historial de compras del usuario y genera una lista de videojuegos recomendados.<br>4. El sistema muestra los videojuegos recomendados al usuario. |
| **Flujos Alternativos** | • Si el usuario intenta ver recomendaciones sin haber realizado ninguna compra, se le muestra un mensaje explicando que ha de realizar una compra para hacer uso de la funcionabilidad |

| | | | |
|---|---|---|---|
| **Caso de Uso** | Gestionar Alertas para Productos sin Stock | **Actores Principales** | Usuario registrado |
| **Descripción** | El usuario registrado crea alertas para ser notificado por correo electrónico cuando un producto está agotado (sin stock). También permite al usuario eliminar alertas creadas previamente. | | |
| **Precondiciones** | El usuario ha iniciado sesión en su cuenta de usuario y está explorando productos en el catálogo. | | |
| **Postcondiciones** | El usuario recibe notificaciones por correo electrónico cuando un producto sin stock vuelve a estar disponible. Si se elimina una alerta, el usuario ya no recibe notificaciones para ese producto. | | |
| **Flujo Básico** | 1. El usuario inicia sesión en su cuenta de usuario.<br>2. El usuario explora el catálogo y ve un producto que está agotado (sin stock).<br>3. El usuario selecciona la opción "Crear Alerta para Producto sin Stock".<br>4. El sistema registra la alerta del usuario y la asocia al producto agotado.<br>5. Cuando el producto vuelve a estar disponible, el sistema envía una notificación por correo electrónico al usuario. | | |
| **Flujos Alternativos** | • Si el usuario decide eliminar una alerta, selecciona la opción "Eliminar Alerta para Producto sin Stock". El sistema retira la alerta asociada al producto. | | |

| **Caso de Uso** | Gestionar Información del Usuario | **Actores Principales** | Usuario registrado |
|---|---|---|---|
| **Descripción** | El usuario registrado modifica sus datos personales en la tienda en línea. | | |
| **Precondiciones** | El usuario ha iniciado sesión en su cuenta de usuario. | | |
| **Postcondiciones** | La información del usuario se actualiza en el sistema. | | |
| **Flujo Básico** | 1. El usuario inicia sesión en su cuenta de usuario.<br>2. El usuario selecciona la opción "Gestionar Información del Usuario" en su panel de control o perfil.<br>3. El usuario puede realizar una o más de las siguientes acciones:<br>• Actualizar información de contacto (nombre, dirección, número de teléfono, etc.).<br>• Cambiar la dirección de correo electrónico o la contraseña.<br>4. El sistema registra los cambios realizados por el usuario y actualiza la información en su cuenta. | | |
| **Flujos Alternativos** | • Si el usuario intenta cambiar su dirección de correo electrónico a una ya utilizada por otro usuario, el sistema muestra un mensaje de error y solicita una dirección de correo electrónico única. | | |

| **Caso de Uso** | Ver historial de Compras | **Actores Principales** | Usuario Registrado |
|---|---|---|---|
| **Descripción** | El usuario registrado ve un listado de sus compras | | |
| **Precondiciones** | El usuario ha iniciado sesión en la plataforma. | | |
| **Postcondiciones** | El usuario visualiza un listado de sus compras realizadas. | | |
| **Flujo Básico** | 1. El usuario selecciona la opción "Ver Compras" en el desplegable.<br>2. El sistema retorna las compras del usuario en cuestión. | | |
| **Flujos Alternativos** | Si no ha realizado ninguna compra, se retorna un texto informando de la situación. | | |

| **Caso de Uso** | Comprar el carrito | **Actores Principales** | Usuario Registrado |
|---|---|---|---|
| **Descripción** | El usuario registrado desea comprar los artículos añadidos a su carrito. | | |
| **Precondiciones** | El usuario ha iniciado sesión en la plataforma y tiene artículos en su carrito | | |
| **Postcondiciones** | El usuario visualiza un listado de los ítems comprados. | | |
| **Flujo Básico** | 1. El usuario se dirige al carrito en el desplegable de la aplicación<br>2. El usuario verifica los ítems que desea comprar y pulsa en continuar a PayPal.<br>3. El sistema le redirige a la pasarela de pago de PayPal.<br>4. El usuario inicia sesión de PayPal y realiza el pago con el método que prefiera<br>5. Si el pago es correcto, el sistema muestra el listado de los ítems comprados con los códigos correspondiente. | | |
| **Flujos Alternativos** | Si el pago se cancela o es erróneo, el sistema muestra una pagina informando al usuario de la situación | | |

| **Caso de Uso** | Creación de usuario Administrador | **Actores Principales** | Administrador |
|---|---|---|---|
| **Descripción** | El administrador crea otro usuario administrador | | |
| **Precondiciones** | No existe un usuario creado con el correo que se dispone a crear el administrador | | |
| **Postcondiciones** | Se ha creado un segundo usuario administrador | | |
| **Flujo Básico** | 1. El administrador ha iniciado sesión y se dirige al dashboard de la aplicación<br>2. El adminstrado elige la opción “Crear Admin user”. | | |

| | |
|---|---|
| | 3. El administrador rellena el formulario con los campos indicados y pulsa en registrar administrador |
| **Flujos Alternativos** | Si ya existe un usuario con ese mail se informa de la situación y se pide que se use otro correo electrónico. |

| | | | |
|---|---|---|---|
| **Caso de Uso** | Envío de la notificación de la Alerta | **Actores Principales** | Sistema |
| **Descripción** | El sistema envía un email notificando a los usuarios indicados que hay nuevo stock disponible de un producto. | | |
| **Precondiciones** | Usuarios han creado una alerta sobre un producto | | |
| **Postcondiciones** | Se envía correctamente la notificación mail al correo electrónico de los usuarios. | | |
| **Flujo Básico** | 1. El estado del stock de un producto pasa de estar no disponible a disponible<br>2. El sistema escucha la acción y este llama al sistema de mail para enviar la notificación<br>3. El sistema recoge los usuarios los cuales deben ser notificados<br>4. El sistema envía el mail a los usuarios correspondientes. | | |
| **Flujos Alternativos** | Si no existen alertas creadas, el sistema no enviara ninguna notificación. | | |

# 3 Diseño

En el presente apartado, exploraremos detalladamente el diseño integral que sustenta nuestra aplicación web. Comprender la arquitectura, el modelo de datos y las tecnologías empleadas resulta esencial para construir una plataforma robusta y eficiente que cumpla con los objetivos propuestos.

## 3.1 Arquitectura

El diseño arquitectónico de nuestro proyecto es un componente crítico que guía la organización y la interacción de los diversos módulos y tecnologías que conforman nuestra aplicación. Esta sección presenta el Diagrama de Arquitectura de nuestra tienda, que se basa en una arquitectura en tres capas y utiliza tecnologías clave, como Spring Boot JPA, Docker, PostgreSQL, HTML y Bootstrap con Thymeleaf.

### 3.1.1 Arquitectura en Tres Capas (MVC)

Dentro del diseño integral de nuestra aplicación web, adoptamos un enfoque arquitectónico que se alinea estrechamente con el Modelo Vista Controlador (MVC). Este modelo proporciona una estructura organizativa que separa las responsabilidades de manera clara y eficiente, contribuyendo a la modularidad y mantenibilidad de nuestro sistema.

**Capa de Presentación (Vista):**

- En el corazón de la experiencia del usuario, la Capa de Presentación, equivalente a la "Vista" en MVC, se encarga de la interfaz de usuario y la presentación visual de nuestra tienda en línea. Hacemos uso de tecnologías web como HTML, Bootstrap y Thymeleaf para construir páginas web atractivas y responsivas. Estas tecnologías son las artífices de la interacción visual que permite a nuestros usuarios explorar, buscar y adquirir videojuegos de manera eficiente.

**Capa de Lógica de Negocio (Controlador):**

- La Capa de Lógica de Negocio, nuestra "Controlador" en el contexto MVC, es la encargada de la lógica central de nuestra aplicación. Aquí es donde reside la gestión de operaciones relacionadas con usuarios, el catálogo de videojuegos, carritos de compras y alertas. Para implementar esta lógica, confiamos en la potencia de Spring Boot JPA. Esta tecnología simplifica el acceso a nuestra base de datos PostgreSQL, permitiendo la creación de servicios y repositorios eficientes, actuando como el intermediario crucial entre la vista y el modelo.

**Capa de Datos (Modelo):**

- La Capa de Datos, que se corresponde con el "Modelo" en MVC, constituye la base de nuestra aplicación. Aquí es donde almacenamos información crítica sobre videojuegos, categorías, usuarios y más. Para gestionar eficientemente esta capa, adoptamos PostgreSQL como nuestra base de datos principal. Docker desempeña un papel esencial al contenerizar nuestra base de datos, simplificando así el despliegue y la administración en diversos entornos.

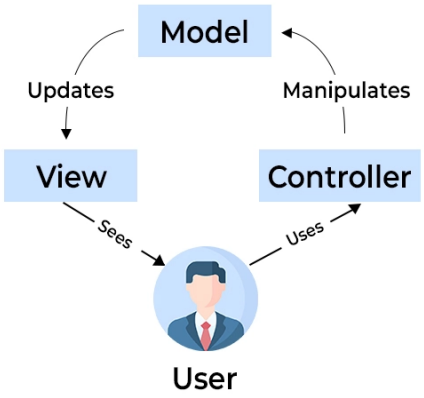


*Ilustración 7 - Modelo MVC*

Este enfoque de tres capas, inspirado en el MVC, proporciona una estructura clara y modular para nuestra aplicación web. La separación de responsabilidades entre la presentación, la lógica de negocio y la gestión de datos no solo mejora la mantenibilidad del sistema, sino que también facilita la evolución y expansión de nuestra plataforma en línea.

### 3.1.2 Tecnologías y Frameworks

En la construcción de nuestra arquitectura de microservicios, hemos seleccionado un conjunto de tecnologías y frameworks que potencian distintos aspectos de nuestro sistema:

**Java 17**

- Adoptamos Java 17 como nuestra versión principal del lenguaje de programación. Las características y mejoras en esta versión nos permiten aprovechar la última tecnología y mantener nuestro código actualizado.


*Ilustración 8 - Java 17*

**Maven**

- Maven se emplea como herramienta de gestión de proyectos, simplificando la construcción y gestión de dependencias. Facilita la estructuración y mantenimiento de nuestro código.


*Ilustración 9 - Maven*

**Spring Boot JPA:**

- Elegimos Spring Boot JPA para la capa de lógica de negocio. Su capacidad para simplificar el acceso a la base de datos y gestionar eficazmente operaciones CRUD (Crear, Leer, Actualizar, Eliminar) nos permite construir de manera ágil y robusta.


*Ilustración 10 - Spring Boot JPA*

**PostgreSQL**

- Como sistema de gestión de bases de datos relacional, PostgreSQL desempeña un papel crucial en el almacenamiento y recuperación de datos. Su robustez y capacidad para manejar grandes cantidades de información son fundamentales para el funcionamiento eficiente de nuestro sistema.


*Ilustración 11 - PostgreSQL*

**Docker**

- La contenerización de nuestra base de datos PostgreSQL con Docker proporciona una solución versátil y escalable. Esto garantiza una administración consistente de la base de datos en diversos entornos de desarrollo y producción, ofreciendo portabilidad y eficiencia.

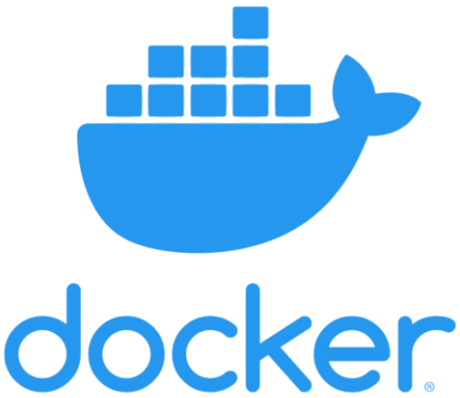


*Ilustración 12 – Docker*

**HTML, Bootstrap, CSS y Thymeleaf**

- Estas tecnologías se entrelazan para crear una interfaz de usuario atractiva y funcional. HTML establece la estructura, Bootstrap agrega estilo y responsividad, mientras que Thymeleaf facilita la integración de datos dinámicos. El resultado es una experiencia de usuario efectiva en nuestra tienda en línea de videojuegos.


*Ilustración 13 - HTML, Bootstrap, CSS y Thymeleaf*

**Apache Kafka y Zookeeper**

- Utilizamos Apache Kafka y Zookeeper para la gestión de eventos y la comunicación entre microservicios. Kafka proporciona una plataforma robusta de streaming, mientras que Zookeeper actúa como coordinador para mantener la coherencia en un entorno distribuido.


*Ilustración 14 - Apache Kafka y Zookeeper*

Estas elecciones tecnológicas no solo respaldan la eficiencia en el desarrollo, sino que también fortalecen la robustez y la escalabilidad de nuestro sistema. Cada tecnología se integra de manera sinérgica, contribuyendo al éxito global de nuestra arquitectura de microservicios.

### 3.1.3 Decisiones Arquitectónicas

El Diagrama de Arquitectura proporcionará una visión general de cómo estas capas y tecnologías se interconectan para brindar una experiencia de usuario sólida y permitir la gestión eficiente de la tienda en línea. Además, destacará las decisiones arquitectónicas clave que respaldan la funcionalidad y el rendimiento de nuestra aplicación.

**Modelo Arquitectónico Inicial**

En el proceso de desarrollo de nuestra aplicación web, nos embarcamos en la tarea de diseñar un modelo arquitectónico que reflejara de manera precisa las necesidades y requisitos iniciales de la tienda en línea. Este diseño inicial, representado en la Ilustración 8 - Modelo Arquitectónico Inicial, proporcionó la base para la implementación inicial de nuestra arquitectura.

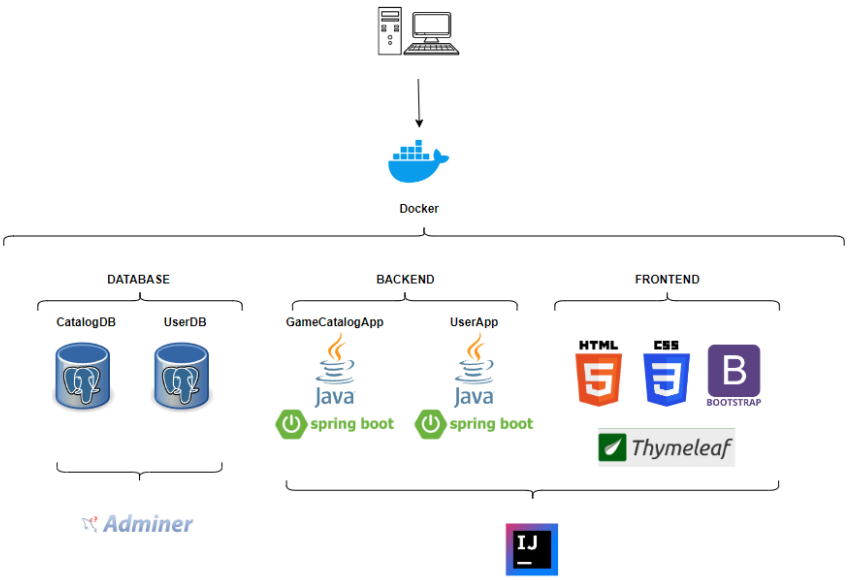


*Ilustración 15 - Modelo Arquitectónico Inicial*

En el modelo arquitectónico inicial, Docker abarcaba toda la aplicación, proporcionando un entorno consistente que encapsulaba tanto las bases de datos como el backend y frontend de nuestra tienda en línea.

**Modelo Arquitectónico Final**

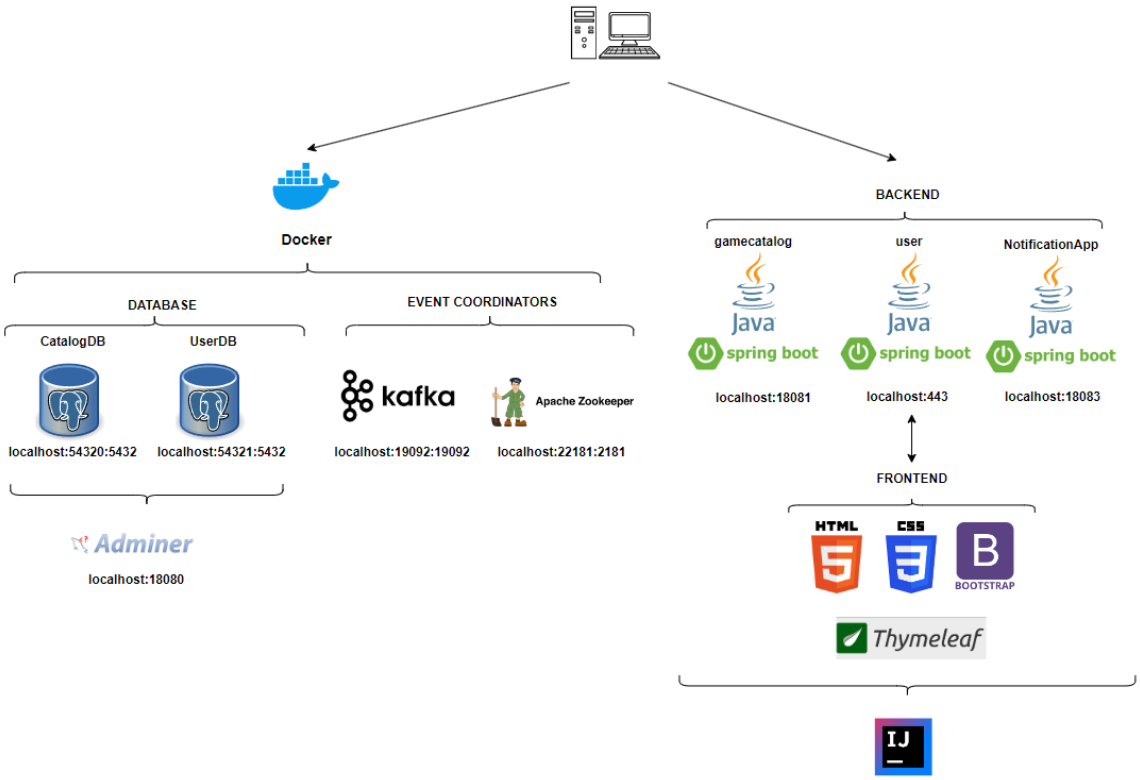


*Ilustración 16 - Modelo Arquitectónico Final*

En la versión final, como se representa en la Ilustración 9, hemos tomado la decisión estratégica de separar las APIs Backend de Docker. Este cambio tiene como objetivo principal mejorar la flexibilidad y la gestión de cada componente, permitiendo adaptaciones ágiles y específicas a las necesidades cambiantes del sistema.

Esta separación permite que las APIs Backend operen de manera más autónoma, facilitando actualizaciones y modificaciones sin afectar otros componentes.

Además, se abre la posibilidad de contenerizar las APIs Backend como una mejora potencial en el futuro.

### 3.1.4 Microservicios y APIs

La arquitectura de microservicios está diseñada de manera modular, dividiendo las responsabilidades en distintos servicios independientes. Cada microservicio tiene un propósito específico y se comunica con otros servicios para realizar operaciones más complejas. Aquí hay una breve descripción de cada uno de los microservicios que se han visto en el modelo anterior:

**gamecatalog API:**

- Responsable de la lógica de negocio relacionada con el catálogo e inventario de productos, específicamente códigos de juegos.
- Contiene la funcionalidad necesaria para gestionar la información sobre los juegos y su disponibilidad.

**user API:**

- Encargado de la lógica de negocio relacionada con los usuarios, carritos de compra, compras y alertas.
- Actúa como controlador sobre el gamecatalog API, llamando a sus métodos para realizar operaciones relacionadas con los juegos.
- Integra el Frontend acoplado gracias a Thymeleaf, lo que hace que las páginas web se generen en el servidor y se envíen al cliente.

**notificationApp API:**

- Se ocupa de la lógica para escuchar los cambios en el inventario.
- Envía alertas a los usuarios utilizando un servicio de correo electrónico cuando se detectan cambios significativos en el inventario.
- Puede actuar como un servicio de fondo que se ejecuta de forma independiente para manejar notificaciones de forma asíncrona.

## 3.2 Modelo de Datos

En esta sección, presentaremos un diagrama de clases orientado a objetos que modela las entidades clave y sus relaciones en nuestro proyecto de tienda en línea de videojuegos. Este modelo de clases proporciona una representación visual de cómo organizamos y gestionamos la información relacionada con nuestra tienda.

El diagrama de clases también destaca las relaciones entre estas entidades, lo que nos permite comprender cómo interactúan en el contexto de nuestra tienda en línea.

Este modelo de clases es esencial para comprender la arquitectura subyacente de nuestra tienda y sirve como base para el diseño y la implementación del sistema. A medida que avancemos en la descripción detallada de cada entidad y sus relaciones en el diagrama, podremos apreciar cómo se traduce la estructura y la lógica de nuestro sistema en una representación visual que facilita la comunicación y la planificación del proyecto.

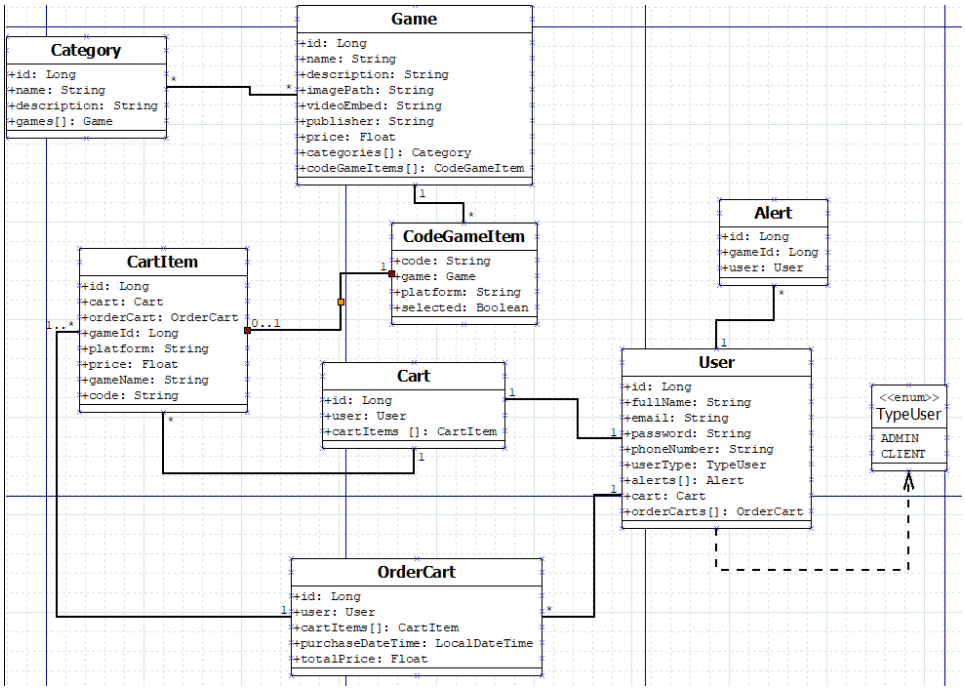


*Ilustración 17 - Modelo UML*

Las clases representan las entidades esenciales de nuestro sistema, como "Game" (videojuego), "CodeGameItem" (código de videojuego), "Category" (categoría), "User" (usuario), "Alert" (alerta), "Cart" (carrito de compras), "CartItem" (ítem de carrito) y "OrderCart" (historial de compras).

A continuación, se desarrollan las relaciones vistas anteriormente en el diagrama UML:

**Game (Videojuego):**

- Relación con "Category" (Categoría): Un videojuego puede pertenecer a múltiples categorías, y una categoría puede contener varios videojuegos. Esto se logra mediante una tabla intermedia en la base de datos que asocia videojuegos con categorías.
- Relación con "CodeGameItem" (Código de Videojuego): Un videojuego puede tener múltiples códigos asociados. Esto implica una relación de uno a muchos, ya que un videojuego puede tener varios códigos relacionados con él.
- Relación con "Alert" (Alerta): Un videojuego puede tener múltiples alertas asociadas a él. Esto implica que varios usuarios pueden seguir un videojuego y recibir notificaciones sobre él.

**Category (Categoría):**

- Relación con "Game" (Videojuego): Una categoría puede contener varios videojuegos y un videojuego puede pertenecer a múltiples categorías. Esta relación se logra mediante una tabla intermedia en la base de datos que asocia categorías con videojuegos.

**User (Usuario):**

- Relación con "Alert" (Alerta): Un usuario puede crear múltiples alertas para productos sin stock, lo que implica una relación de uno a muchos.
- Relación con "Cart" (Carrito de Compras): Cada usuario tiene su propio carrito de compras. Esta es una relación de uno a uno, ya que un usuario tiene un solo carrito de compras.

**Alert (Alerta):**

- Relación con "User" (Usuario): Cada alerta está asociada a un usuario específico. Esto implica una relación de muchos a uno, ya que varios usuarios pueden tener alertas, pero una alerta pertenece a un solo usuario.
- Relación con "Game" (Videojuego): Cada alerta está asociada a un videojuego específico. Esto significa que una alerta se refiere a un solo videojuego y su estado, como el stock o la disponibilidad.

**Cart (Carrito de Compras):**

- Relación con "User" (Usuario): Cada carrito de compras pertenece a un usuario específico. Esta es una relación de uno a uno, ya que un carrito de compras está asociado a un solo usuario.
- Relación con "CartItem" (Ítem de Carrito): Un carrito de compras puede contener múltiples ítems de carrito. Esto permite a los usuarios agregar varios productos a su carrito de compras.

**CodeGameItem (Código de Videojuego):**

- Relación con "Game" (Videojuego): Cada código de videojuego está asociado a un videojuego específico. Esto significa que varios códigos de videojuegos pueden estar relacionados con el mismo videojuego.

**CartItem (Ítem de Carrito):**

- Relación con "CodeGameItem" (Código de Videojuego): Un ítem de carrito está asociado a un código de videojuego específico. Esto permite a los usuarios agregar múltiples códigos de videojuegos a su carrito de compras.
- Relación con "Cart" (Carrito de Compras): Cada ítem de carrito pertenece a un carrito de compras específico. Esto establece una relación de muchos a uno, ya que un carrito de compras puede contener varios ítems.

**OrderCart (Historial de Compras):**

- Relación con "CartItem" (Ítem de Carrito): Un historial de compras (OrderCart) puede contener múltiples ítems de carrito. Esto permite almacenar los productos comprados en un historial específico.

- Relación con "User" (Usuario): Cada historial de compras está asociado a un usuario específico. Esto establece una relación de muchos a uno, ya que varios historiales de compras pueden pertenecer a un solo usuario.

Además, hay que tener en cuenta que, en nuestro sistema, la entidad “User” desempeña un papel crucial al representar a los usuarios de la tienda en línea de videojuegos. Para diferenciar entre dos roles distintos, es decir, los Usuarios Registrados (CLIENT) y los Administradores (ADMIN), hemos implementado una clase enumeración llamada “TypeUser”.

La clase “TypeUser” permite asignar a cada usuario uno de estos dos tipos, lo que facilita la distinción de las funcionalidades y privilegios disponibles para cada categoría de usuario. Los Usuarios Registrados (CLIENT) tienen acceso a ciertas funcionalidades, como explorar y comprar videojuegos, mientras que los Administradores (ADMIN) tienen la capacidad de gestionar la tienda y supervisar las transacciones.

La distribución de las entidades entre las APIs es la siguiente:

- gamecatalog: Category, Game, CodeGameItem
- user: CartItem, Cart, OrderCart, User, Alert, TypeUser

Debido a la separación de las entidades entre las APIs de gamecatalog y user, se crea la clase CartItem, la cual hace de puente para CodeGameItem. Esta nueva clase será usada como objeto en el carrito (Cart) como en las órdenes realizadas (OrderCart). Por lo tanto, una vez compremos el carrito, en vez de eliminarlo, lo vaciaremos y crearemos un OrderCart el cual contendrá los ítems comprados. Una vez vaciado el carrito, este se volverá a utilizar para futuras compras y los OrderCarts serán nuestro historial de compras.

## 3.3 Diseño Relacional de la Base de Datos con SpringBoot y JPA

En nuestro proyecto de tienda en línea de videojuegos, hemos optado por utilizar Spring Boot en combinación con Java Persistence API (JPA) para gestionar la base de datos de manera eficiente y escalable. Una de las ventajas clave de esta tecnología es la creación automática de la base de datos a partir de nuestras entidades y atributos definidos en el código Java.

### 3.3.1 División de la Base de Datos

Hemos tomado la decisión estratégica de dividir nuestra base de datos en dos esquemas separados para alinearla con nuestra arquitectura de microservicios. Esta elección se basa en la idea de descomponer nuestra aplicación en módulos independientes, cada uno responsable de una funcionalidad específica. En nuestro caso, los microservicios son el "Catálogo" y la "Gestión de Usuarios".

Esta división en microservicios nos proporciona ventajas como la escalabilidad independiente, el despliegue modular y la facilidad de mantenimiento. Cada microservicio puede evolucionar y escalarse de forma independiente, lo que facilita la administración de nuestra aplicación a medida que crece.

**catalogDB (Base de Datos del Catálogo):**

- Este esquema se encarga de almacenar información relacionada con los videojuegos, sus categorías, códigos de videojuegos y detalles del catálogo.
- Contiene tablas como "Game," "Category," "CodeGameItem," y otras que representan los elementos del catálogo y sus propiedades.

**userDB (Base de Datos de Usuarios):**

- Este esquema se enfoca en la gestión de usuarios, sus datos de registro, carritos de compras, alertas y otras funcionalidades relacionadas con la interacción de los usuarios con la tienda.

- Incluye tablas como "User," "Cart," "Alert," y otras que almacenan información relevante para la autenticación, el seguimiento de pedidos y las preferencias de los usuarios.

### 3.3.2 Creación Automática de la Base de Datos

Una de las ventajas clave de utilizar Spring Boot y JPA es la creación automática de la base de datos y las tablas correspondientes a partir de nuestras entidades Java. Esto significa que no necesitamos crear manualmente la estructura de la base de datos ni las tablas.

Simplemente definimos nuestras entidades Java, anotadas con anotaciones JPA, y Spring Boot se encarga del resto. Al iniciar la aplicación, las tablas se generarán automáticamente en las bases de datos "catalogDB" y "userDB" de acuerdo con nuestras definiciones de clases.
Este enfoque simplifica enormemente el proceso de desarrollo y mantenimiento de la base de datos, ya que cualquier cambio en las entidades se reflejará automáticamente en la base de datos durante la inicialización de la aplicación.

En resumen, gracias a Spring Boot y JPA, nuestra base de datos se construirá automáticamente de acuerdo con nuestras definiciones de entidades y se dividirá en dos esquemas, lo que nos permite gestionar eficazmente los datos relacionados con el catálogo y los usuarios en nuestra tienda en línea de videojuegos.

## 3.4 Modelo de pantallas: Prototipado

En el proceso de desarrollo de nuestra tienda de videojuegos en línea, es fundamental visualizar y planificar la interfaz de usuario que nuestros clientes experimentarán. Para lograrlo, hemos optado por utilizar la herramienta de diseño Figma, que nos permite crear prototipos de las pantallas de nuestra aplicación. En esta sección, presentaremos dos de las pantallas clave que hemos diseñado: la página de inicio y la vista de detalles del producto.

### 3.4.1 Prototipos de Pantallas

- Página de Inicio

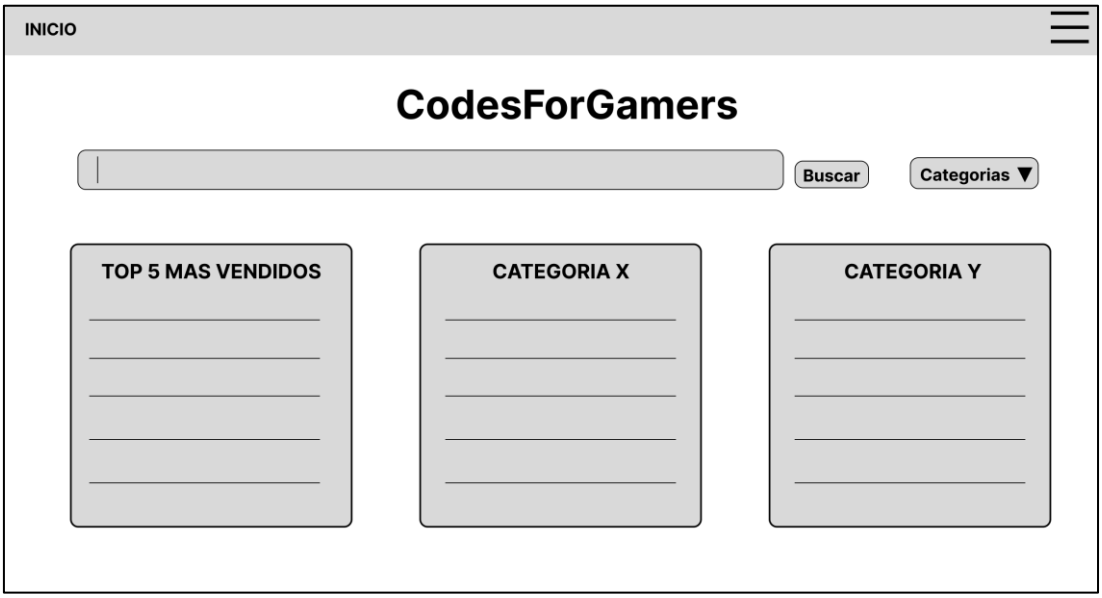


*Ilustración 18 - Prototipo: Página Inicio*

La página de inicio es la puerta de entrada a nuestra tienda de videojuegos en línea. Hemos diseñado esta pantalla de manera que los visitantes se sientan bienvenidos y se les brinde una vista previa de la experiencia que ofrecemos. Aquí, los elementos visuales y las llamadas a la acción se han dispuesto estratégicamente para guiar a los usuarios hacia la exploración de nuestro catálogo de juegos. La simplicidad y la claridad son nuestros principales objetivos en esta pantalla, y hemos evitado incluir detalles más específicos, como formularios de inicio de sesión o registro, ya que estos serán implementados en versiones posteriores del diseño.

- Vista de Detalles del Producto:

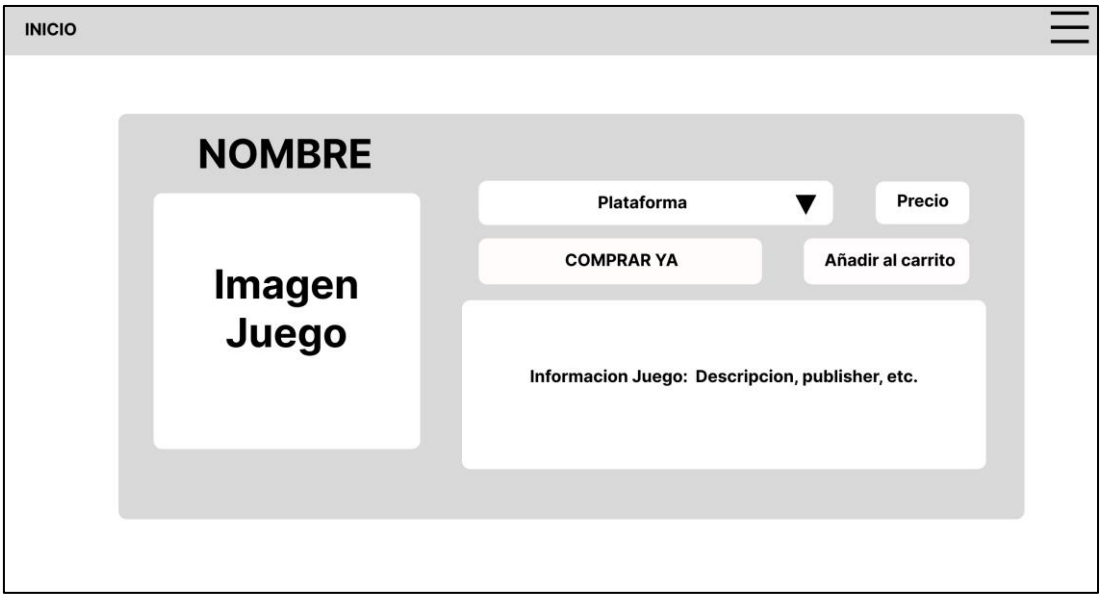


*Ilustración 19 - Prototipo: Detalles Producto*

La vista de detalles del producto es una de las páginas más importantes de nuestra aplicación, ya que es el lugar donde los usuarios pueden obtener información completa sobre un videojuego específico y tomar decisiones de compra. En esta pantalla, hemos enfocado la atención en la presentación de los detalles del juego, como su título, imagen, precio y descripción. Además, hemos incorporado un llamado claro a la acción que permite a los usuarios agregar el juego a su carrito de compras. Dada la naturaleza específica de esta pantalla, hemos decidido omitir otros elementos de la interfaz, como formularios de contacto o perfiles de usuario, que serán diseñados en fases posteriores del desarrollo.

Estos dos prototipos iniciales sientan las bases de nuestra interfaz de usuario. A medida que avancemos en el desarrollo de la tienda de videojuegos, continuaremos diseñando y mejorando otras pantallas y elementos de la interfaz para proporcionar a nuestros usuarios una experiencia fluida y atractiva.

# 4 Implementación

Después de definir los objetivos del sistema y las especificaciones de diseño, que abarcan tanto la parte técnica y arquitectónica como el enfoque en la experiencia del usuario, estamos listos para iniciar la implementación de la aplicación. En este proceso, adoptaremos principios fundamentales de diseño, como maximizar el desacoplamiento y aprovechar al máximo la reutilización de componentes. Asimismo, nos adheriremos a varios estándares en el desarrollo de la aplicación y en el uso de herramientas y APIs, ya sean internas o externas. A continuación, se proporciona información detallada sobre las herramientas empleadas y los aspectos técnicos relacionados con el desarrollo de la aplicación y sus funcionalidades.

## 4.1 Recursos y APIs externas

Este apartado tiene como objetivo proporcionar una visión detallada de los elementos tecnológicos y las interfaces de programación de aplicaciones (APIs) que han sido incorporados en el desarrollo del sistema. En este análisis, destacaremos las herramientas y servicios externos que desempeñan un papel crucial en la funcionalidad y operatividad de la aplicación, subrayando su contribución al éxito global del proyecto. A través de esta exposición, obtendremos una comprensión más profunda de cómo estas fuentes externas enriquecen y potencian las capacidades de nuestra aplicación.

### 4.1.1 Recursos

Estos recursos representan las herramientas tecnológicas que han sido empleadas en diversas etapas del proceso de implementación. A continuación, se detallan los elementos clave:

- **IntelliJ:** IntelliJ, un entorno de desarrollo integrado (IDE), ha sido utilizado como plataforma principal para la codificación y gestión del proyecto. Su conjunto de características avanzadas proporciona un entorno eficiente y efectivo para el desarrollo de software.

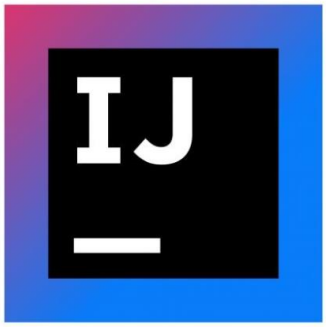


*Ilustración 20 - IntelliJ*

- **GitHub:** GitHub ha desempeñado un papel crucial como plataforma de control de versiones. Facilita el seguimiento del código fuente, permitiendo un desarrollo continuo y una gestión eficiente de las modificaciones en el código.


*Ilustración 21 - GitHub*

- **Navegador basado en Chromium:** Se ha empleado un navegador basado en Chromium como herramienta de prueba y visualización. Este tipo de navegador proporciona un entorno de prueba representativo para asegurar la compatibilidad y funcionalidad óptima de la aplicación en diversos entornos de navegación. Ejemplos de este tipo de navegador son Google Chrome, Opera, Microsoft Edge.

*Ilustración 22 - Navegador Chromium*

Estos recursos, cuidadosamente seleccionados, han contribuido significativamente a la eficacia y coherencia del proceso de desarrollo, permitiendo un trabajo eficiente y una implementación exitosa de la aplicación.

### 4.1.2 APIs Externas

En el desarrollo de las funcionalidades relacionadas con la compra, búsqueda de video-tráileres de juegos en YouTube y el envío de notificaciones por correo electrónico sobre nuevo stock, se han integrado conexiones a servicios externos clave. Estos servicios son los siguientes:

**PayPal:**

- Se ha implementado la API de PayPal para proporcionar un sistema de pasarela de pago en la tienda. La configuración inicial se ha realizado en modo Sandbox, permitiendo transacciones de prueba para evaluar la funcionalidad de compra. En el futuro, para llevar a cabo transacciones reales, solo será necesario cambiar el estado de la API a modo Live.

  *La documentación detallada se encuentra en el archivo README.md del repositorio.*


*Ilustración 23 - PayPal*

**YouTube Data API v3:**

- Para evitar la tarea manual de buscar el tráiler de cada juego, se ha incorporado la API "YouTube Data API v3". Esta API permite realizar búsquedas de videos utilizando el nombre del juego. Es importante destacar que la API tiene un límite de 100 búsquedas diarias. Por lo tanto, se ha implementado la funcionalidad de almacenar el videoId en la base de datos para optimizar el uso diario de la API. La aplicación realizará la búsqueda y guardará el videoId al ingresar un nuevo juego. Si el juego ya tiene un videoId guardado, la búsqueda no se llevará a cabo, y se mostrará el tráiler asociado al videoId almacenado.

  *Se ha documentado la configuración de la API en el archivo README.md del repositorio.*

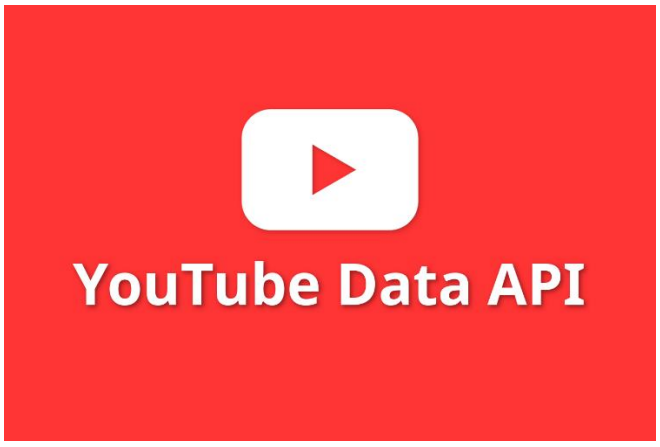


*Ilustración 24 - YouTube Data API*

**Cuenta de Gmail:**

- Para notificar a los usuarios sobre el nuevo stock de un juego para el cual hayan creado alertas previamente, se ha seleccionado una cuenta de Gmail. Esta cuenta enviará correos electrónicos cuando el estado del stock cambie de 0 a 1, indicando la disponibilidad del producto.

  *La configuración detallada de la cuenta de Gmail se encuentra documentada en el archivo README.md del repositorio.*


*Ilustración 25 - Gmail*

Estas integraciones con servicios externos amplían las capacidades de la aplicación, proporcionando una experiencia completa y funcional para los usuarios, así como una gestión eficiente de transacciones y comunicaciones importantes.

## 4.2 Estructura de las APIs desarrolladas

A continuación, se proporciona una visión integral de la organización y disposición de los elementos fundamentales que componen el proyecto en su conjunto. La aplicación, como descrita en apartados anteriores, se ha desarrollado utilizando la arquitectura Spring y consta de tres APIs principales: "gamecatalog", "user" y "notificationApp".

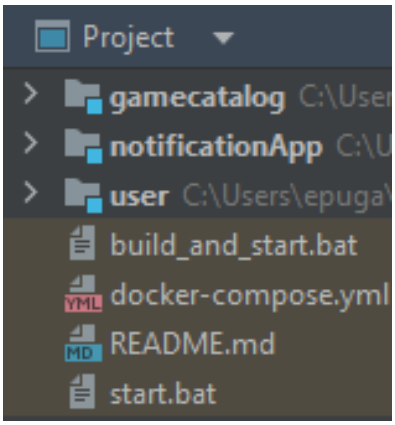


*Ilustración 26 - Estructura General*

A través de esta revisión, se profundizará en la disposición de los directorios, módulos y archivos clave que componen cada uno de los proyectos, permitiendo una comprensión completa de la arquitectura general de la aplicación. La estructura del proyecto no solo facilita la colaboración entre equipos de desarrollo, sino que también establece un marco organizativo que optimiza la mantenibilidad y escalabilidad del sistema.

A continuación, se detallarán los componentes esenciales de cada proyecto, subrayando la interconexión y colaboración entre ellos para lograr una aplicación coherente y eficiente. Esta exploración de la estructura del proyecto brindará una visión holística que servirá como guía para desarrolladores, administradores y cualquier persona involucrada en el proyecto.

### 4.2.1 Estructura API gamecatalog

Esta estructura proporciona una visión general de cómo se organiza y distribuye la lógica de la API "gamecatalog". Cada componente juega un papel crucial en la implementación de las funcionalidades específicas de esta parte del sistema.

**Estructura de Paquetes:**

- **com.example.gamecatalog:**

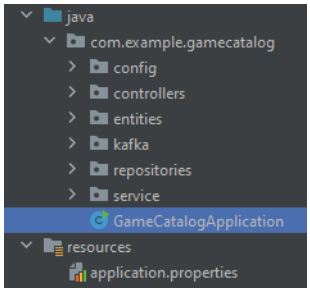


*Ilustración 27 - Estructura gamecatalog*

- **config:** Incluye una clase de configuración denominada "DataLoader" la cual añade información predeterminada a la BBDD.

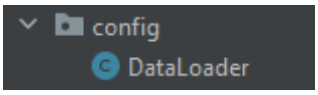


*Ilustración 28 - Paquete config gamecatalog*

- **controllers:** Incluye clases controladoras para manejar las solicitudes HTTP.
    - **dtos:** Contiene objetos de transferencia de datos utilizados para la comunicación entre APIs internas.

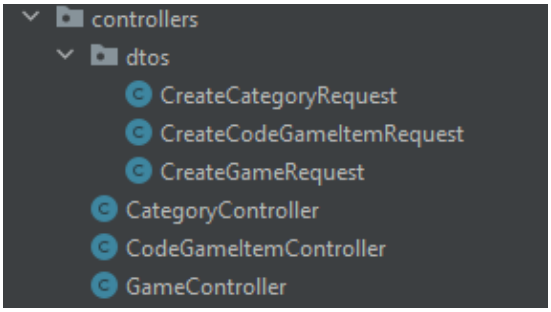


*Ilustración 29 - Paquete controllers gamecatalog*

- **service:** Contiene clases de servicio que implementan la lógica de negocio.

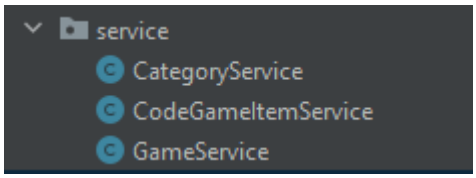


*Ilustración 30 - Paquete service gamecatalog*

- **repositories:** Almacena interfaces de repositorio para interactuar con la base de datos.

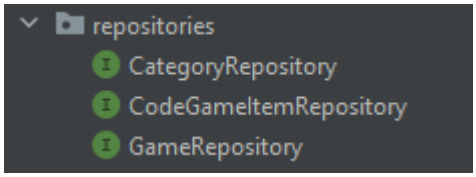


*Ilustración 31 - Paquete repositories gamecatalog*

- **entities:** Define las entidades y objetos de datos.
    - **dto:** Contiene objetos basados en las entities los cuales ayudan a tratar llamadas recursivas.

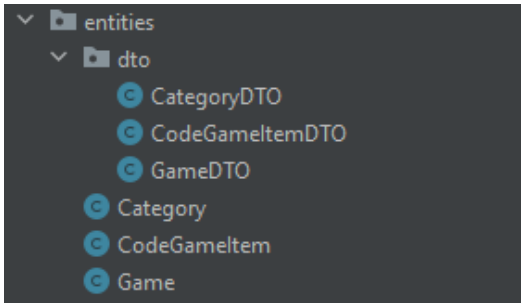


*Ilustración 32 - Paquete entities gamecatalog*

- **kafka:** Define la cola Kafka para ser enviada a la gestión de eventos sobre el stock

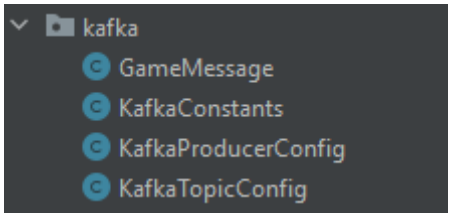


*Ilustración 33 - Paquete kafka gamecatalog*

**Configuración:**

- **application.properties:** Archivo de configuración con propiedades específicas de la aplicación.
- **pom.xml:** Archivo de configuración Maven que define las dependencias y configuraciones del proyecto.

**Funcionalidades Principales:**

- **Gestión Catálogo e Inventario:** Implementa operaciones CRUD para gestionar la información del catalog y del inventario. Estas operaciones se llamarán a través de la API interna user, la cual hace de controlador sobre gamecatalog.
- **Notificación de Stock:** Implementa la lógica para notificar a los usuarios sobre cambios en el stock. Cada vez que se interactúa con el stock y este recibe un cambio de estado, se envía una cola Kafka, la cual es recibida por la API notificationApp

### 4.2.2 Estructura API user

La siguiente disposición ofrece una panorámica sobre la organización y distribución de la lógica de la API "user". Cada elemento desempeña un papel fundamental en la ejecución de funciones específicas dentro de esta sección del sistema.

**Estructura de Paquetes:**

- **com.example.user:**

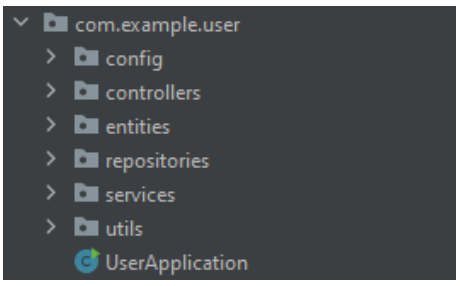


*Ilustración 34 - Estructura user*

- **config:** Incluye clases que ayudan a configurar varias funcionalidades de la aplicación

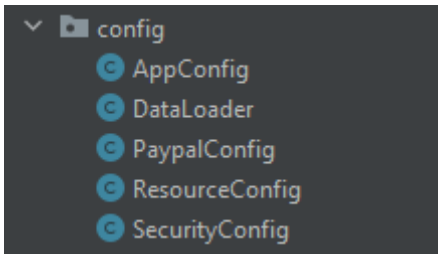


*Ilustración 35 - Paquete config user*

- **controllers:** Incluye clases controladoras para manejar las solicitudes del frontend.
    - **dtos:** Contiene objetos de transferencia de datos utilizados para la comunicación entre APIs internas.

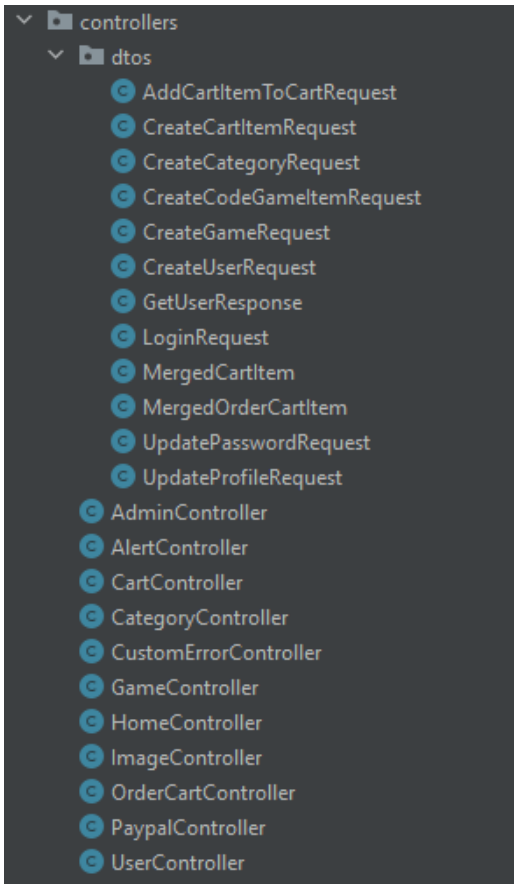


*Ilustración 36 - Paquete controllers user*

- **services:** Contiene clases de servicio que implementan la lógica de negocio.

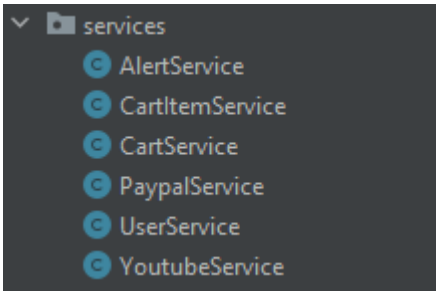


*Ilustración 37 - Paquete services user*

- **repositories:** Almacena interfaces de repositorio para interactuar con la base de datos.

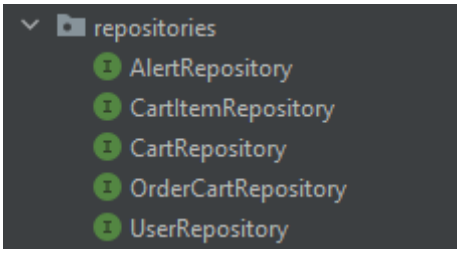


*Ilustración 38 - Paquete repositories user*

- **entities:** Define las entidades y objetos de datos.

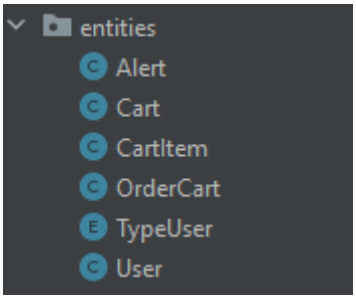


*Ilustración 39 - Paquete entities user*

- **utils:** Define dos clases de apoyo para gestionar sesiones y respuestas JSON

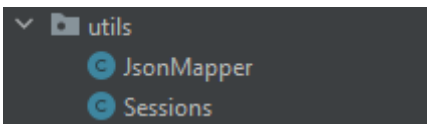


*Ilustración 40 - Paquete utils user*

**Estructura de plantillas de Frontend:**

- **resources.templates:** Estructura de carpetas donde se encuentran todos los archivos HTML

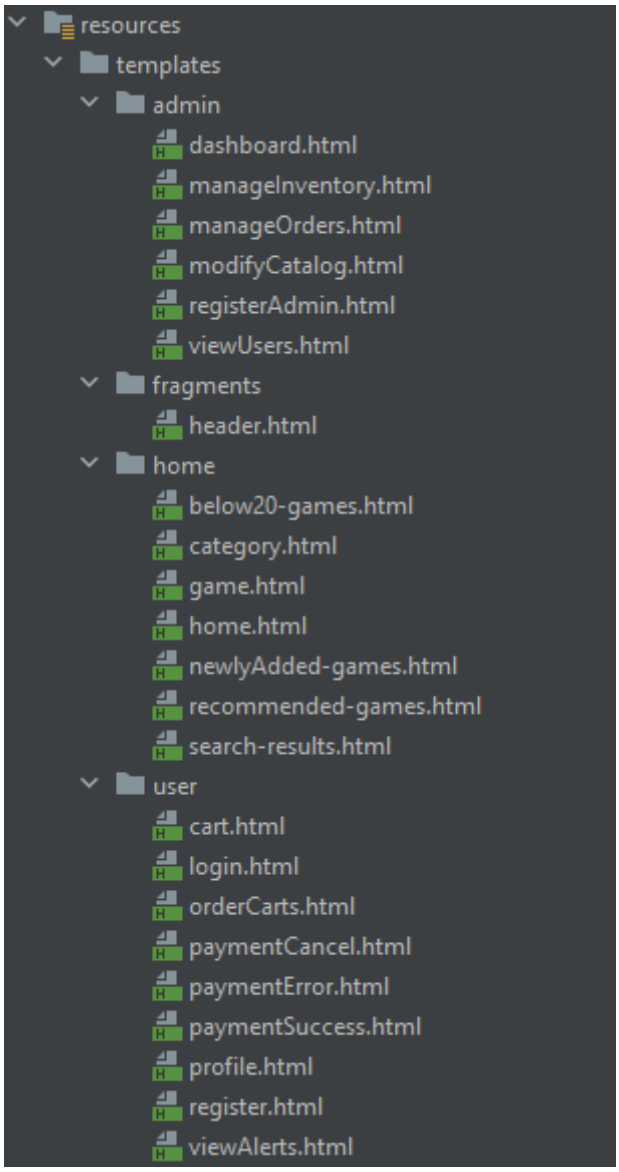


*Ilustración 41 - Plantillas HTML user*

**Configuración:**

- **application.properties:** Archivo de configuración con propiedades específicas de la aplicación.
- **pom.xml:** Archivo de configuración Maven que define las dependencias y configuraciones del proyecto.

- **keystore.p12:** Archivo que contiene la clave SSL para usar HTTPS en nuestra aplicación.

**Funcionalidades Principales:**

- **Gestión de Carrito:** Proporciona operaciones para agregar, eliminar y gestionar productos en el carrito de compras del usuario.
- **Gestión de Compras:** Facilita operaciones para realizar y rastrear compras de productos, integrándose con la lógica interna de "user".
- **Gestión de Alertas:** Permite a los usuarios configurar y recibir alertas sobre cambios relevantes, como la disponibilidad de productos.
- **Controlador sobre "gamecatalog":** Actúa como controlador principal, coordinando operaciones con la API "gamecatalog" para gestionar el catálogo e inventario de juegos.
- **Integración con Frontend (Thymeleaf):** Desarrolla y acopla el frontend utilizando Thymeleaf para una experiencia de usuario integrada y coherente

### 4.2.3 Estructura notificationApp

Esta estructura proporciona una visión global de la organización y distribución de la lógica en la API de notificaciones ("notificationApp"). Cada componente desempeña un papel fundamental en la implementación de las funcionalidades específicas que definen esta parte del sistema.

**Estructura de Paquetes:**

- **com.example.notificactionApp:**

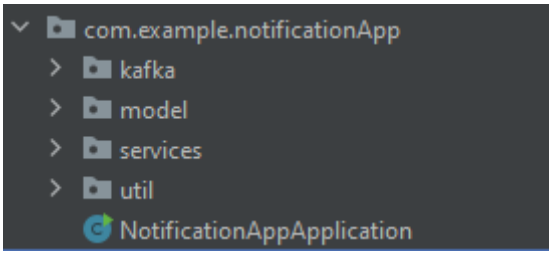


*Ilustración 42 - Estructura notificationApp*

- **kafka:** Define la cola Kafka y la escucha de eventos para ser recibido de gamecatalog

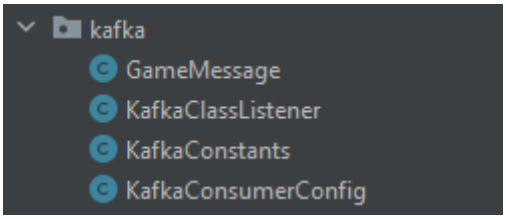


*Ilustración 43 - Paquete kafka notificationApp*

- **model:** Define el DTO de user para ser tratado en la lógica

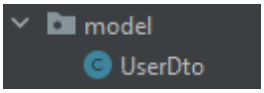


*Ilustración 44 - Paquete model notificationApp*

- **services:** Contiene clases de servicio que implementan la lógica de negocio.

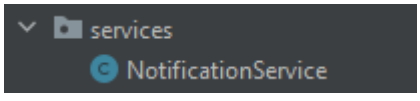


*Ilustración 45 - Paquete services notificationApp*

- **util:** Define dos clases de apoyo para gestionar las solicitudes HTTPS que se hacen a user

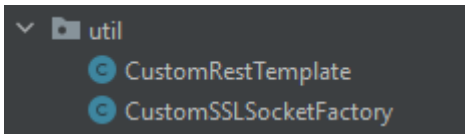


*Ilustración 46 – Paquete util notificationApp*

**Configuración:**

- **application.properties:** Archivo de configuración con propiedades específicas de la aplicación.
- **pom.xml:** Archivo de configuración Maven que define las dependencias y configuraciones del proyecto.

**Funcionalidades Principales:**

- **Envío de Notificaciones**: A partir de la escucha de eventos por colas Kafka, esta API gestiona la información de los eventos del inventario y determina el envío de correos electrónicos a los usuarios correspondientes.

## 4.3 Seguridad

En la sección 4.3, abordamos el aspecto crítico de la "Seguridad" en nuestro proyecto, reconociendo la importancia de salvaguardar la aplicación dada su naturaleza. Se han implementado medidas clave para fortalecer la seguridad de la plataforma.

**HTTPS en la API user:**

Dada la necesidad de asegurar las comunicaciones entre la aplicación y los usuarios, se ha optado por la implementación de HTTPS en la API "user". Esta capa de seguridad garantiza la encriptación de los datos durante la transmisión, proporcionando una capa adicional de protección contra posibles ataques de interceptación y garantizando la integridad de la información transmitida.

```
protected void configure(HttpSecurity http) throws Exception {
    http.requiresChannel(channelConfigurer -> channelConfigurer.
                    anyRequest().requiresSecure()) HttpSecurity
            .csrf().disable()
            .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
            .and() HttpSecurity
            .formLogin() FormLoginConfigurer<HttpSecurity>
            .loginPage("/login")
            .permitAll()
            .and() HttpSecurity
            .logout() LogoutConfigurer<HttpSecurity>
            .logoutUrl("/user/logout") // specify your logout URL
            .logoutSuccessUrl("/home") // redirect to home after logout
            .invalidateHttpSession(true)
            .deleteCookies( ...cookieNamesToClear: "JSESSIONID") // if using cookies
            .permitAll();

}
```

*Ilustración 47 - Configuración de HTTPS en user (SecurityConfig.java)*

Además, se ha de mencionar que se ha autogenerado una clave SSL debido a que el proyecto aún no tiene un dominio asignado. Este certificado fue creado gracias al JDK de JAVA de forma local, y, por lo tanto, no está respaldados por una autoridad de certificación externa. Aunque ofrece encriptación, puede generar advertencias en los navegadores, ya que no hay una entidad externa que haya verificado la autenticidad del certificado.

```
server.port=443
server.ssl.key-store=classpath:keystore.p12
server.ssl.key-store-password=123456
server.ssl.key-password=123456
security.require-ssl=true
server.error.include-message=always
```

*Ilustración 48 - Configuración SSL en user (application.properties)*

**Encriptación de Contraseñas con BCrypt:**

Para salvaguardar la información sensible almacenada en la base de datos, se ha tomado la decisión de encriptar las contraseñas mediante el algoritmo de hash BCrypt. Este enfoque garantiza que las contraseñas no estén almacenadas en texto plano, sino que se almacenen en una forma irreversible y segura. BCrypt es conocido por ser un algoritmo de hash robusto y resistente a ataques de fuerza bruta.


*Ilustración 49 - Creación del Bean para encriptar contraseñas (SecurityConfig.java)*

Estas medidas se han implementado con la intención de crear un entorno seguro para los usuarios y proteger los datos críticos manejados por la aplicación. La combinación de HTTPS en la API expuesta y la encriptación de contraseñas en la base de datos refuerza la seguridad global del sistema.

Por último, es importante destacar que, si bien se ha implementado una serie de medidas de seguridad significativas en el proyecto, no se ha abarcado exhaustivamente todos los posibles problemas de seguridad.

## 4.4 Repositorio GitHub

En el Repositorio GitHub, se alberga el código fuente completo de la aplicación. Este repositorio sirve como un depósito centralizado donde se puede acceder, gestionar y colaborar en el desarrollo del software. Aquí es donde reside la totalidad de las instrucciones y archivos que conforman la aplicación, permitiendo a los desarrolladores, colaboradores y cualquier interesado examinar, modificar y contribuir al proyecto.

**Repositorio GitHub público:**

**https://github.com/mrepuga/CodesForGamers**

# 5 Manual de uso

En este apartado, detallaremos los pasos necesarios para poner en funcionamiento la aplicación de manera efectiva. A continuación, encontrarás instrucciones precisas que te guiarán desde la instalación inicial hasta el uso práctico de las funciones clave. Sigue cada paso cuidadosamente para asegurarte de aprovechar al máximo la aplicación desde el primer momento.

*Toda la siguiente información está disponible en el archivo README.md del repositorio.*

## 5.1 Requisitos Previos

Antes de poner en marcha el proyecto de CodesForGamers se han de obtener los programas siguientes:

- Java JDK 17
- Maven
- Docker Desktop

## 5.2 Credenciales APIs Externas

Una vez tenemos todos los softwares necesarios para ejecutar la aplicación, tendremos que configurar las claves de las APIs externas siguientes:

**user API:**

En el mismo application.properties se deben configurar las Keys para las distintas APIs externas:

- **YouTube data Api v3:**
    - **youtube.api.key**:

Como obtener la clave de YouTube Data Api v3 en el siguiente tutorial: https://www.youtube.com/watch?v=qWUobN0xtcE

- **PayPal:**
    - **paypal.client.id**
    - **paypal.client.secret**
    - **paypal.client.mode**

Como obtener las claves de PayPal en el siguiente tutorial: https://www.upwork.com/resources/paypal-client-id-secret-key

**notificationApp API:**

En el archivo de configuración application.properties se deben configurar las credenciales para la notificación mail:

- Gmail:
    - **spring.mail.username**
    - **spring.mail.password**

Como obtener la clave de Gmail en el siguiente tutorial: https://www.sysinfotools.com/how-to/generate-app-password-in-gmail.html

## 5.3 Despliegue

El despliegue se ha realizado en el S.O de Windows 10 y se han detallado dos formas para conseguirlo:

**Windows 10 - Scripts:**

Si se dispone de este S.O, se han creado dos ficheros para el lanzamiento de la aplicación:

- **build_and_start**: Se compila el código y se ejecuta la aplicación. **(Primera vez que se lanza la aplicación y/o se ha cambiado el código de alguna de las APIs)**

- **start.bat**: Se ejecuta la aplicación.

**Windows 10 - Comandos:**

Si se desea ejecutar las APIs de con comandos a partir de CMD/BASH, se ha de seguir el siguiente orden:

**1. Levantamiento de los contendores**

Nos dirigimos al PATH del Proyecto y lanzamos los contenedores Docker

```
cd PROYECTO
docker compose up -d
```

**2. Compilación de código** (Saltar si ya se disponen de los .jars)

Ahora compilaremos las 3 APIs con los siguiente comandos

```
cd PROYECTO
cd gamecatalog
mvn clean install -DskipTest
cd ..

cd user
mvn clean install -DskipTest
cd ..

cd notificationApp
mvn clean install -DskipTest
cd ..
```

**3. Ejecución de las APIs**

Para finalizar lanzaremos las 3 APIs con los siguientes comandos.

```
cd PROYECTO
start java -jar gamecatalog/target/gamecatalog-0.0.1-SNAPSHOT.jar
start java -jar user/target/user-0.0.1-SNAPSHOT.jar
start java -jar notificationApp/target/notificationApp-SNAPSHOT.jar
```

## 5.4 Funcionamiento

Una vez levantado los contenedores correctamente como así los archivos .jar, la aplicación se encontrará en funcionamiento.

Para acceder a la web, vaya **https://localhost/home** donde se encuentra la aplicación en marcha.

También puede acceder a **http://localhost:18081/swagger-ui/index.html** para acceder al Swagger de la API de **gamecatalog** donde encontrará todos los métodos implementados de esta API

## 5.5 Configuración

De forma predeterminada, la aplicación tiene información ya agregada:

- 20 Juegos y 10 Categorías.
- Un usuario registrado para las funciones de Administrador:
    - Acceda a application.properties para obtener la credenciales:
        - **web.admin.email**
        - **web.admin.password**

# 6 Mejoras futuras y puntos pendientes

A continuación, se presentan algunas sugerencias para mejorar y expandir la aplicación en el futuro:

- **Contenerización de las APIs:** Explorar la opción de contenerizar las APIs mediante tecnologías como Docker para facilitar la implementación, escalabilidad y gestión del entorno.
- **Segurización de Todas las APIs:** Extender la seguridad a todas las APIs, asegurándose de que utilicen HTTPS para garantizar la transmisión segura de datos.
- **Implantación de Test Unitarios:** Incorporar más pruebas unitarias en el proceso de desarrollo para garantizar la estabilidad y el rendimiento del código, facilitando la identificación de posibles problemas.
- **Ampliación del Uso de Notificaciones por Correo Electrónico:** Maximizar la utilidad de las notificaciones por correo electrónico, considerando su implementación en diversas áreas como la activación de cuentas, comunicaciones publicitarias, entre otras.
- **Mejora del Diseño Frontend:** Realizar actualizaciones en el diseño del frontend para mejorar la experiencia visual y la usabilidad de los usuarios.
- **Desacoplamiento del Frontend y el Backend:** Considerar la separación del frontend y el backend, abandonando tecnologías como Thymeleaf y optando por frameworks frontend modernos como Angular. Esto permite una arquitectura más modular y escalable.

# 7 Conclusión

En conclusión, este proyecto de fin de grado en Ingeniería Informática ha representado una oportunidad valiosa para fusionar la pasión personal por los videojuegos con la aplicación práctica de conocimientos adquiridos a lo largo del programa académico. La planificación, análisis y desarrollo de una tienda en línea especializada en la venta de códigos de videojuegos ha sido un proceso enriquecedor, integrando competencias clave como la gestión de proyectos, ingeniería de requisitos, programación orientada a objetos y diseño con patrones, entre otras disciplinas fundamentales.

La sinergia entre el interés personal y la aplicación práctica de conocimientos no solo ha permitido alcanzar los objetivos establecidos para la creación de la tienda en línea, sino que también ha proporcionado una experiencia de inmersión significativa en el ámbito profesional. Este proyecto no solo se buscaba satisfacer las expectativas académicas, sino también la contribución al desarrollo integral de habilidades y competencias esenciales en el campo de la tecnología y el comercio electrónico.

El proyecto no concluye con la implementación de la tienda en línea, sino que establece las bases para un mantenimiento continuo, la evaluación constante del rendimiento y la posibilidad de futuras actualizaciones. Este proceso no solo ha consolidado mi formación en Ingeniería Informática, sino que también ha fomentado un enfoque proactivo hacia la resolución de problemas y la adaptabilidad en un entorno tecnológico en constante evolución.

En resumen, este proyecto representa no solo el cierre de una etapa académica, sino el inicio de una trayectoria profesional marcada por la aplicación práctica de conocimientos, la pasión por la tecnología y el compromiso con la excelencia en el ámbito del desarrollo de aplicaciones web.

# 8 Glosario

- Gamer: Persona apasionada por los videojuegos; aquel que participa activamente en la cultura del juego.
- Steam: Plataforma de distribución digital de videojuegos y software, desarrollada por *Valve Corporation*.
- Epic Games: Empresa de desarrollo de videojuegos y tecnología, conocida por crear el motor de juego *Unreal Engine* y el popular juego *Fortnite.*
- MVC (Modelo-Vista-Controlador): Patrón de arquitectura de software que separa la aplicación en tres componentes principales: Modelo (datos y lógica), Vista (interfaz de usuario) y Controlador (manejo de eventos y coordinación).
- API (Interfaz de Programación de Aplicaciones): Conjunto de reglas y herramientas que permite la comunicación entre diferentes software.
- Microservicio: Enfoque arquitectónico que estructura una aplicación como un conjunto de servicios pequeños e independientes, cada uno ejecutando un proceso único.
- UML (Lenguaje de Modelado Unificado): Conjunto de estándares utilizados para visualizar, especificar, construir y documentar los artefactos de un sistema de software.
- IDE (Entorno de Desarrollo Integrado): Software que proporciona herramientas y funciones para facilitar el desarrollo de software.
- Modo SandBox: Entorno de prueba aislado del sistema principal, utilizado para ejecutar y evaluar software de forma segura.
- Modo Live: Entorno real del sistema
- DTO (Objeto de Transferencia de Datos): Patrón de diseño que representa un objeto que transporta datos entre capas de una aplicación.
- Stock: Cantidad de productos o bienes disponibles para la venta en un momento dado.

- JSON (Notación de Objetos JavaScript): Formato ligero de intercambio de datos basado en texto, comúnmente utilizado para transmitir datos entre un servidor y una aplicación web.
- Cola Kafka: Sistema de mensajería distribuida que utiliza un modelo de publicación/suscripción para gestionar flujos de datos.
- HTTPS (Protocolo de Transferencia de Hipertexto Seguro): Protocolo de comunicaciones seguro utilizado en internet para la transferencia segura de datos.
- Repositorio: Almacén de datos o código fuente, utilizado para gestionar versiones y facilitar la colaboración en el desarrollo de software.
- Software: Conjunto de programas, instrucciones y datos que permiten el funcionamiento de un sistema informático.
- Keys: Claves o códigos utilizados para acceder a funciones específicas, como claves de licencia para software o claves de activación.
- Streaming: Procesamiento de datos en tiempo real.
- Script: Serie de comandos o instrucciones escritas para realizar tareas automatizadas.

# 9 Bibliografía


[1] Stack Overflow. (n.d.). Recuperado de https://stackoverflow.com/

[2] YouTube. (n.d.). Recuperado de https://www.youtube.com/

[3] Spring Initializer. (n.d.). Recuperado de https://start.spring.io/

[4] draw.io. (n.d.). Recuperado de https://app.diagrams.net/

[5] PayPal. (n.d.). Recuperado de https://www.paypal.com/es/home

[6] Gmail. (n.d.). Recuperado de https://gmail.com/

[7] Maven Central Repository. (n.d.). Recuperado de https://repo.maven.apache.org/maven2/

[8] Universitat Oberta de Catalunya (UOC). (n.d.). Recuperado de https://www.uoc.edu/portal/en/index.html

[9] JetBrains IntelliJ IDEA. (n.d.). Recuperado de https://www.jetbrains.com/idea/

[10] GitHub. (n.d.). Recuperado de https://github.com/

[11] Baeldung. (n.d.). Thymeleaf in Spring MVC. Recuperado de https://www.baeldung.com/thymeleaf-in-spring-mvc

[12] Google Developers. (n.d.). YouTube API. Recuperado de https://developers.google.com/youtube/v3?hl=es-419

[13] SysInfoTools. (n.d.). How to Generate App Password in Gmail. Recuperado de https://www.sysinfotools.com/how-to/generate-app-password-in-gmail.html

[14] Upwork. (n.d.). PayPal Client ID and Secret Key. Recuperado de https://www.upwork.com/resources/paypal-client-id-secret-key