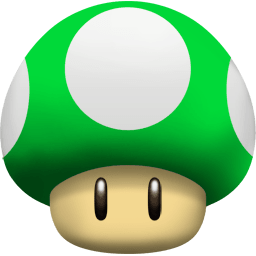


I.E.S. “ÁGORA”

Dpto.: Informática y Comunicaciones

Ciclo Formativo de G. Superior: Desarrollo de Aplicaciones Multiplataformas

LevelUp Shop



Autor: Juan Cuello Gutiérrez

Tutor: Antonio Álvarez Gómez

Cáceres, a 6 de junio de 2025

Resumen

LevelUp Shop es una aplicación de eCommerce enfocada en la venta de videojuegos en formato digital.

El sistema implementa funcionalidades clave como gestión de productos, autenticación de usuarios, pasarelas de pago e integración con servicios externos, con el objetivo de construir una experiencia fluida y completa para el usuario final.

Desde el punto de vista técnico, el proyecto se ha desarrollado con una arquitectura escalable y modular, orientada a facilitar el mantenimiento y futuras ampliaciones. Se ha priorizado la calidad del código, la seguridad de los datos y la eficiencia del rendimiento en diferentes dispositivos.

A lo largo del desarrollo, se han aplicado metodologías ágiles y buenas prácticas de diseño, simulando las condiciones reales de un entorno profesional. Este trabajo supone un ejercicio práctico de consolidación de conocimientos y una propuesta viable para un producto real en el sector digital de videojuegos.

Introducción

LevelUp Shop: Pasión por los videojuegos y el desarrollo, combinados

Contexto:

Desde mis primeros recuerdos, los videojuegos han sido una constante en mi vida: fuente de entretenimiento, inspiración y ahora, el eje central de mi proyecto de fin de grado. El entorno actual del gaming digital es altamente competitivo y exige soluciones de eCommerce especializadas que ofrezcan entregas instantáneas, gestión segura de bibliotecas de usuarios y compatibilidad multiplataforma.

La problemática radica en que muchas tiendas online no ofrecen una experiencia suficientemente optimizada para este tipo de contenido digital. Algunas se centran en la distribución, pero descuidan la usabilidad, otras presentan riesgos en seguridad o limitaciones técnicas. En muchos casos, estas soluciones existentes no cubren las necesidades específicas del usuario *gamer* moderno o están desactualizadas frente a las tendencias tecnológicas actuales.

Ideas principales:

Con esta situación como punto de partida, LevelUp Shop surge como una propuesta innovadora: una aplicación de eCommerce multiplataforma centrada en videojuegos digitales, que cumpla con altos estándares de usabilidad, rendimiento y seguridad.

El principal objetivo del proyecto es desarrollar una solución técnica funcional y profesional, simulando un entorno real de desarrollo de producto. A través del uso de tecnologías multiplataforma, se busca maximizar la compatibilidad entre dispositivos sin sacrificar rendimiento, algo esencial en el sector *gaming*.

La solución que propongo integra funcionalidades críticas como autenticación de usuarios, pasarelas de pago y consumo de APIs de videojuegos. Además, he alineado el trabajo con el módulo Empresa e Iniciativa Emprendedora para explorar la viabilidad de esta tienda como emprendimiento real, abordando desde la gestión de licencias hasta los requisitos legales.

En este contexto, el proyecto invita también a reflexionar sobre cómo los conocimientos adquiridos a lo largo del ciclo pueden materializarse en un producto escalable, con potencial comercial y alineado con una industria en constante evolución.

1. Propuesta de Proyecto

DESCRIPCIÓN GENERAL:

Proyecto consistente en una aplicación multiplataformas de una tienda online de venta de videojuegos digitales. Posibilidad de visualización de videojuegos, carrito, realizar pedidos, darse de alta e iniciar sesión... y demás funcionalidades esperadas en un e-commerce.

2. OBJETIVO:

Desarrollar una aplicación multiplataforma que permita la compra de videojuegos digitales. La plataforma debe permitir la visualización de los videojuegos, la gestión del carrito de compras, realizar pedidos, y la creación e inicio de sesión de usuarios. Además, se debe desarrollar un panel de administración que permita a los usuarios con permisos gestionar productos, pedidos y usuarios.

3. JUSTIFICACIÓN:

En el contexto de mi formación dual en ViewNext, encontrándome en el área de e-commerce, considero que este proyecto me permitirá aplicar los conocimientos adquiridos tanto en el IES como en el entorno laboral. Este proyecto, además de ser interesante y alineado con las tendencias del mercado, plantea retos técnicos significativos que me permitirán desarrollar mis habilidades profesionales en programación, diseño de interfaces y seguridad en ecommerce.

4. ASPECTOS PRINCIPALES QUE SE PRETENDEN ABORDAR:

Desarrollo de la tienda online: Crear una plataforma web robusta y funcional que permita a los usuarios navegar por la tienda, visualizar videojuegos, añadirlos al carrito y proceder con la compra.

Autenticación y gestión de usuarios: Implementar un sistema de registro e

inicio de sesión de usuarios utilizando autenticación segura (por ejemplo,

autenticación JWT).

Interfaz de usuario intuitiva: Desarrollar una UI que sea accesible y fácil de

usar, tanto en dispositivos móviles como de escritorio, garantizando la experiencia

de usuario (UX).

Panel de administración: Crear un panel para administradores que permita

gestionar productos, realizar cambios en la tienda y gestionar pedidos y usuarios de manera eficiente.

Métodos de pago: Integrar métodos de pago comunes y seguros como

tarjetas de crédito/débito y plataformas como PayPal o Stripe.

Seguridad de la información: Asegurar la protección de datos mediante el

uso de encriptación y otras medidas de seguridad como protección de contraseñas y más medidas de seguridad.

Optimización multiplataforma: Asegurar que la aplicación sea responsiva,

es decir, que funcione correctamente en diferentes dispositivos y navegadores.

5. MEDIOS QUE SE UTILIZAN:

Para este proyecto se emplearán diversas herramientas y tecnologías que

garantizan el desarrollo eficaz de la aplicación:

Lenguajes de programación:

- Backend en Java con Spring Boot, para gestionar la lógica de la

aplicación y la base de datos.

- Frontend en Flutter, ya que permite desarrollar aplicaciones

multiplataformas nativas (Android/IOS/Web/Desktop) con una sola

implementación.

Base de datos:

- Utilización de MongoDB (base de datos NoSQL) para el almacenamiento de información, como catálogos de productos y detalles de usuarios.

Servicio de alojamiento:

- Heroku, AWS o DigitalOcean para alojar la

aplicación web. MongoDB Atlas se utilizará para la base de datos,

garantizando escalabilidad y facilidad de acceso.

Herramientas de diseño:

- Figma para el diseño de la interfaz gráfica, creando prototipos

interactivos y atractivos.

Control de versiones:

- Git y GitHub para llevar el control del desarrollo y

garantizar la correcta gestión de las versiones del código.

6. ÁREAS DE TRABAJO Y OTROS ELEMENTOS ESTABLECIDOS POR EL EQUIPO DOCENTE DEL CICLO FORMATIVO A EFECTOS DE VALORACIÓN DE LA PROPUESTA:

Desarrollo y calidad del código:

Se evaluará la estructura del código, su modularidad, el uso de buenas prácticas de programación y el cumplimiento de los estándares de calidad en desarrollo web.

Funcionalidad de la aplicación:

Se valorará si la tienda online permite realizar correctamente las operaciones de visualización de productos, añadir al carrito, hacer el pedido, gestionar el registro e inicio de sesión, entre otros.

Diseño de la interfaz:

Evaluación de la usabilidad de la interfaz, asegurando que sea intuitiva y accesible para los usuarios finales, y que la experiencia de usuario sea óptima.

Implementación de seguridad:

Se comprobará la seguridad de los datos del usuario, la correcta implementación de medidas de protección para las transacciones y la autenticación.

Uso de tecnologías:

Se evaluará la integración y uso adecuado de las tecnologías y herramientas seleccionadas, y si están alineadas con las mejores prácticas del sector.

Innovación:

Se valorará la posible integración de funcionalidades innovadoras, como un chatbot IA para asistir a los usuarios en sus compras (dependiendo del tiempo disponible).

Documentación:

Calidad y exhaustividad de la documentación técnica y de usuario, que describa el funcionamiento de la plataforma y facilite su mantenimiento.

1. Estado del arte

Para fundamentar el desarrollo de LevelUp Shop, se ha llevado a cabo un análisis del estado del arte, investigando productos y servicios similares en el sector del eCommerce de videojuegos digitales. Esta revisión permite:

* Obtener ideas sobre interfaces, procesos de compra, métodos de autenticación y sistemas de administración usados actualmente.
* Identificar oportunidades de mejora y aspectos innovadores que no están siendo suficientemente explotados por las plataformas actuales.

Productos/servicios analizados:

1. Steam

Fortalezas: Plataforma robusta, experiencia de usuario muy pulida, comunidad integrada.

Debilidades: No es estrictamente una tienda independiente (requiere software propio), y no es multiplataforma web-móvil de forma nativa.

1. GOG (Good Old Games)

Fortalezas: DRM-Free, buena experiencia web.

Debilidades: Catálogo limitado, funcionalidades más simples comparadas con competidores.

1. Epic Games Store

Fortalezas: Grandes lanzamientos, interfaz moderna, promociones frecuentes.

Debilidades: Interfaz pesada, dependiente de su launcher, experiencia web limitada.

1. Instant Gaming / Eneba / CDKeys

Fortalezas: Especializadas en claves digitales, buena usabilidad web.

Debilidades: Poca transparencia en la procedencia de claves, experiencia limitada a compra (sin gestión de biblioteca ni comunidad).

Aportes y novedades de LevelUp Shop:

Plataforma multiplataforma real (móvil/web/desktop) desde el inicio, usando Flutter.

Interfaz completamente adaptada a UX moderna, priorizando velocidad, legibilidad y navegación simple.

Panel de administración personalizado, funcionalidad muchas veces ausente o limitada en tiendas centradas solo en ventas.

Enfoque en seguridad desde el diseño, con autenticación JWT, protección de datos personales y cifrado.

Arquitectura escalable y modular, pensada para mantenimiento y ampliación futura.

Potencial integración con APIs externas de videojuegos, ofreciendo una experiencia enriquecida (trailers, valoraciones, etc.).

Aunque algunas funcionalidades puedan coincidir con plataformas existentes, LevelUp Shop aporta valor al combinar usabilidad moderna, tecnologías actuales y un enfoque técnico riguroso que muchas de las soluciones actuales no ofrecen de forma conjunta.

1. Desarrollo del proyecto
   1. Manual Técnico

La arquitectura que se ha adoptado es Cliente-Servidor.

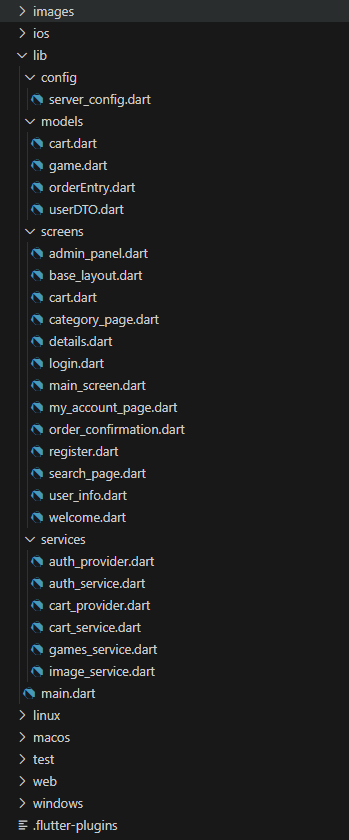
* + 1. Frontend

La tecnología seleccionada para construir la UI o interfaz gráfica, ha sido Flutter.

Este es un framework que hace uso del lenguaje de programación Dart.

Fue lanzado en 2017 por Google, y su propósito principal es el desarrollo de interfaces multiplataformas con un solo código base, ofreciendo despliegues Web, Android, iOS, Windows… entre otros.

La estructura de proyecto Flutter escogida para este proyecto, ha sido la separación por paquetes dependiendo de la funcionalidad interna, puesto que todos los ficheros que componen la aplicación Flutter obtienen extensión “.dart”.



Esta es la estructura base de un proyecto Flutter, en este caso también la del Frontend del proyecto. Dentro de la carpeta *lib,* se encuentra el código base de la aplicación, siguiendo la organización comentada en subcarpetas:

* Config: Configuración básica

del servidor

* Models: Clases de definición

de los distintos tipos de datos.

* Screens: Las propias pantallas

de la interfaz de usuario.

* Services: Funcionalidades necesarias para la lógica de Frontend.

El resto de carpetas, son para alojar el código propio de las distintas plataformas, que en el caso de uso del proyecto no han sido necesarias, y por tanto no se han modificado.

**Screens**

1. Home
   1. Header
   2. Juego destacado
   3. Productos populares
   4. Juegos en oferta
   5. Chatbot IA
   6. Footer
2. Detalle
3. Categorías
4. Buscar
5. Carrito
6. Login
7. Registro
8. Mi cuenta

8.1 Mi perfil

8.1.1 Cambio de avatar

8.1.2 Consola favorita

8.1.3 Cerrar Sesión

8.2 Mis pedidos

8.3 Cambiar contraseña

8.4 Admin Panel

8.4.1 Usuarios

8.4.2 Productos

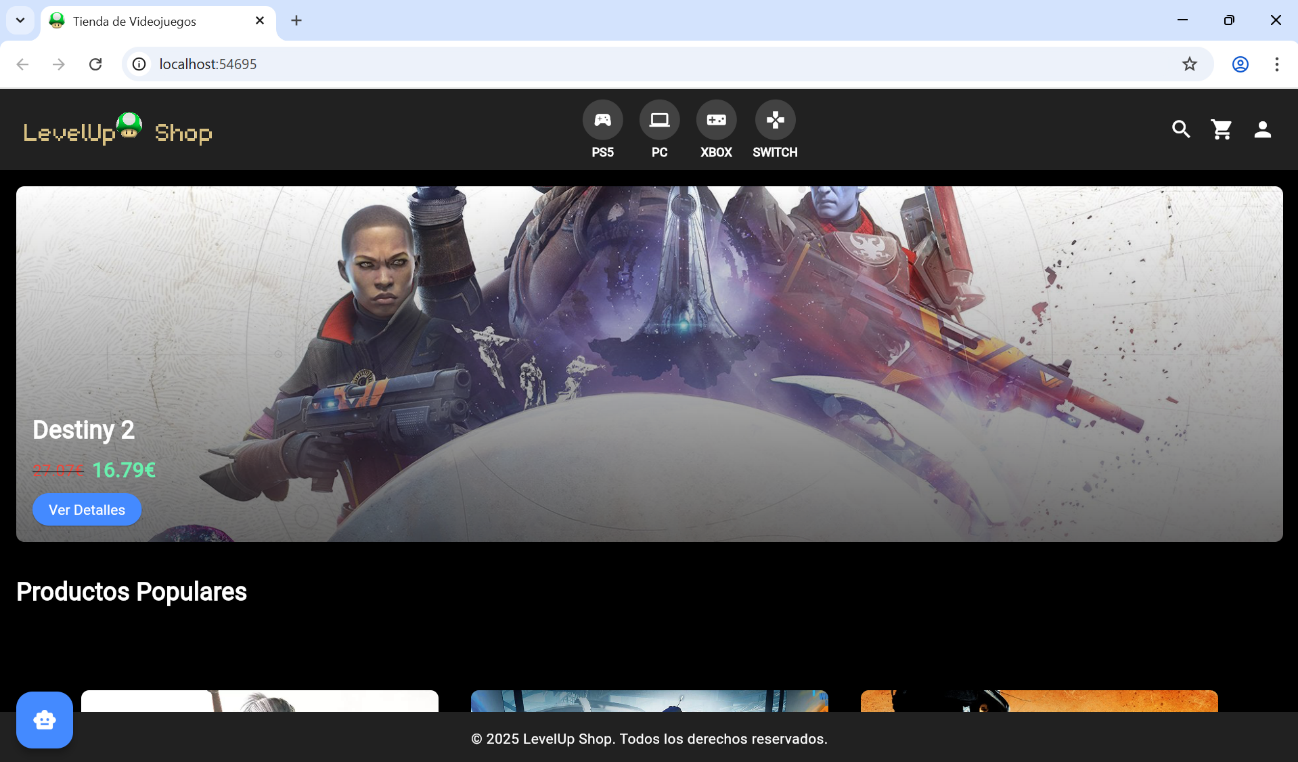
8.4.3 Pedidos

8.4.2 Volver a Mi cuenta

1. Home

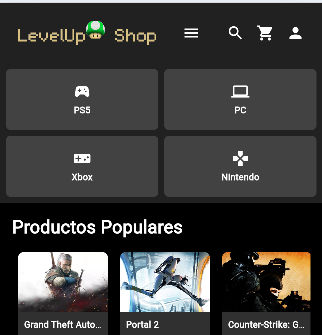
1.1 Header

Nada más iniciar la aplicación, nos encontramos con una pantalla de bienvenida, que muestra un header con el logo de la aplicación, en el centro del mismo 4 botones con las distintas categorías de los videojuegos para poder filtrar por las mismas, y a la derecha el botón de búsqueda, carrito, y mi cuenta, respectivamente:





En dispositivos de pantalla reducida, como por ejemplo un iPhone 14 Pro Max, la interfaz tomaría un diseño ajustado ligeramente diferente, ya que las categorías se encontrarían ocultas y en su lugar se muestra un botón de tipo “hamburguesa”, que lanza un desplegable al interactuar con el mismo:



1.2 Juego destacado

Debajo del header, nos encontramos con una sección de Juego destacado, que mostrará uno de los juegos de los que se encuentran en oferta, al azar con cada recarga de la página.

Esta vista, nos permite ver el nombre del juego, su precio antes de rebajar y después de aplicar el descuento del 15%.

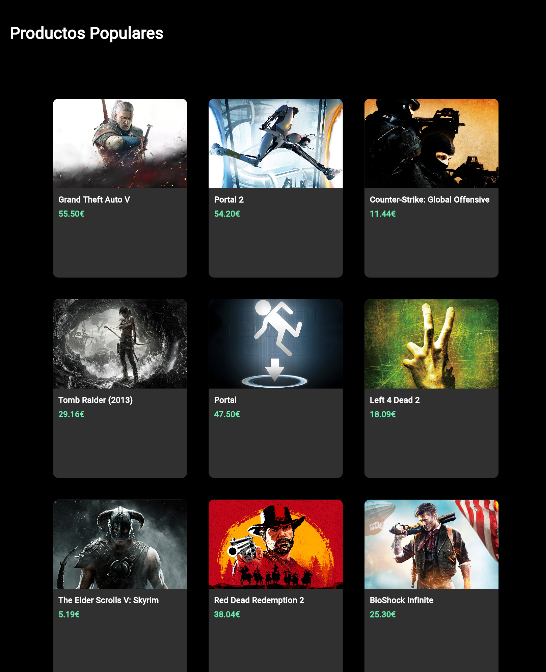
Además, hay un botón con el texto “Ver detalles”, que independientemente de si pulsamos sobre este, o sobre cualquiera de los juegos en toda la aplicación, nos llevará a una vista detalle del juego correspondiente:



1.3 Productos populares

Debajo, encontraremos la sección Juegos populares, que nos muestra un Grid de 3x3, o un carrusel, con una selección de juegos que he decidido incluir puesto que son juegos que personalmente me gustan, y para dar un “guiño” a los juegos de mi infancia.

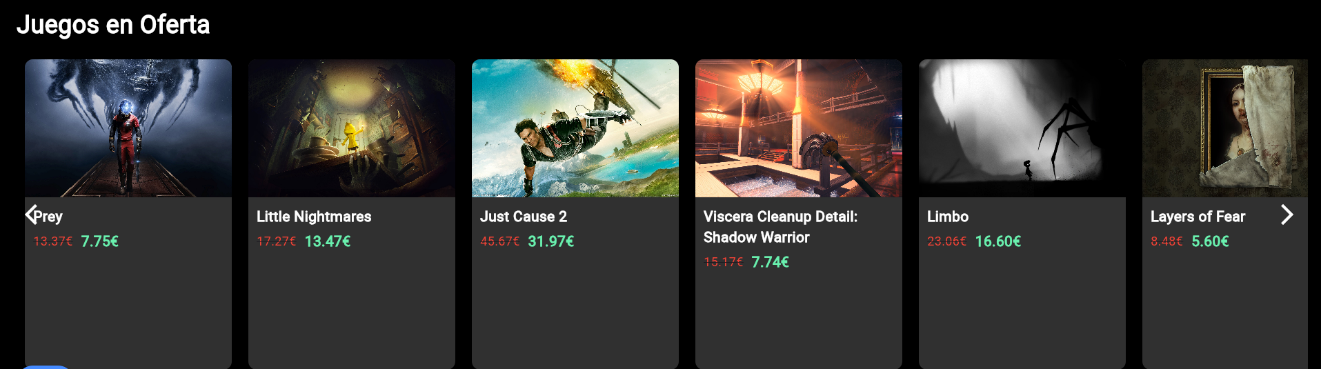
En teléfono móvil, al ser un carrusel, se puede deslizar la lista de juegos horizontalmente, mientras que en pantallas más grandes, será un grid fijo.





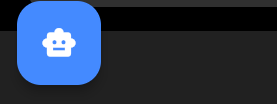
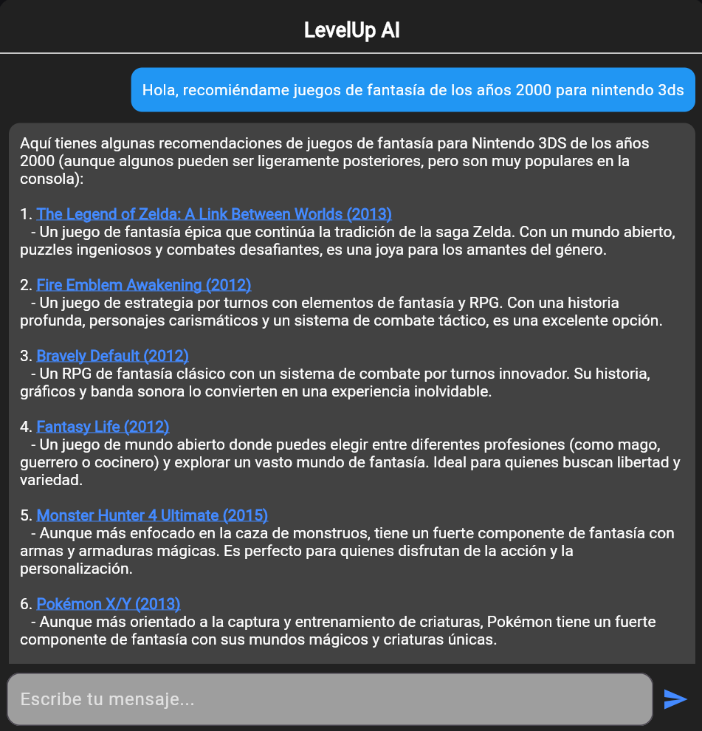
1.4 Juegos en oferta

A continuación, se muestra una lista tipo carrusel con una selección de juegos en oferta, al 15% de descuento, a la cual he agregado flechas a derecha e izquierda para poder ir pasando por los distintos juegos de manera cómoda y fácil visualmente, y para darle diferenciación con la sección anterior de Productos populares. Las tarjetas de los juegos, mostrarán el nombre del mismo, y precio antes, y después de aplicarse el descuento.

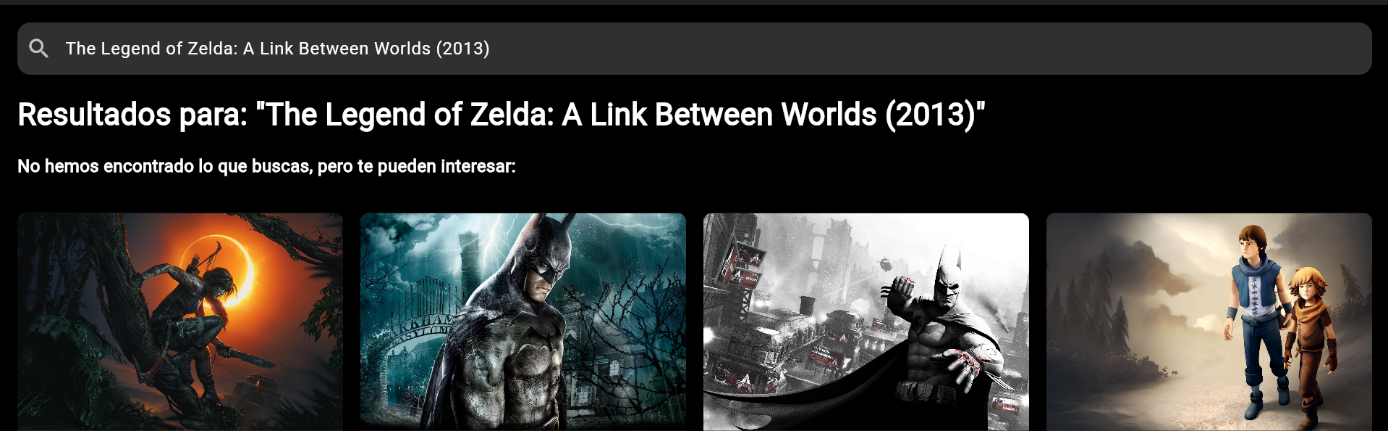


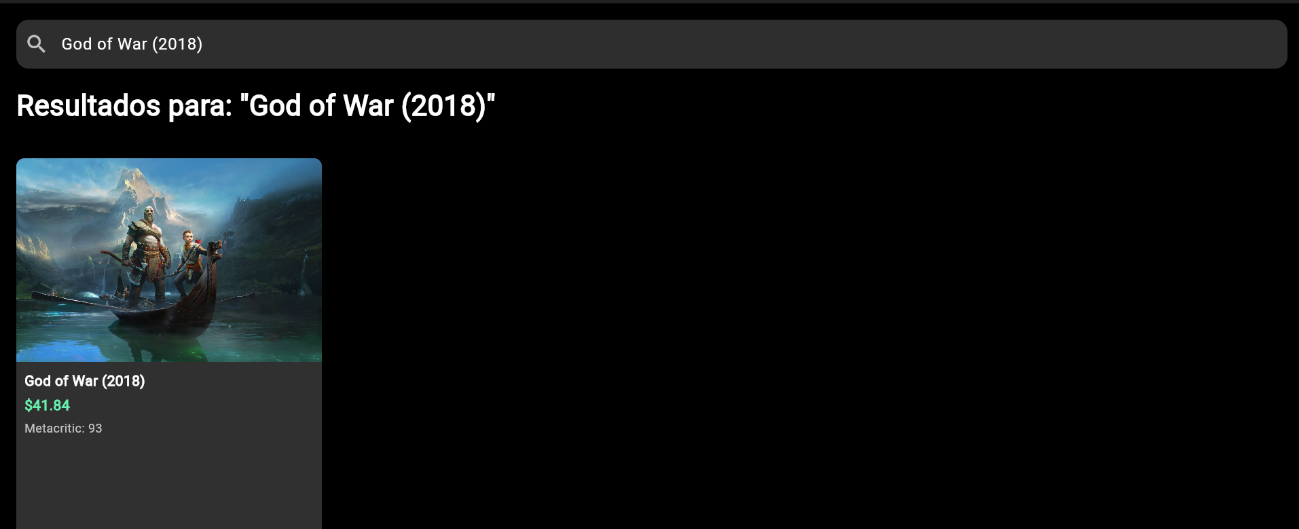
1.5 Chatbot IA

Justo debajo de la sección anterior, y compartiendo espacio con el footer, se encuentra este botón azul con icono de robot, que al interactuar con el mismo, nos abrirá una pantalla adicional que nos permite hablar con un bot de IA configurado como un LLM similar a chatgpt que nos permitirá conversar con el mismo para obtener recomendaciones de videojuegos, o cualquier otro tipo de conversación:



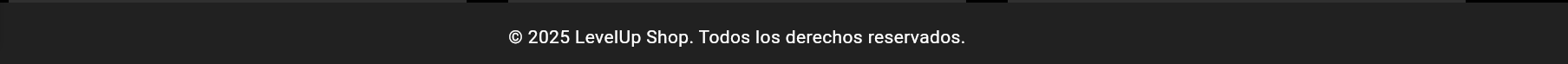
Interactuar con cualquiera de las sugerencias de videojuego nos llevará directamente a la página de resultados de la tienda, con dicho juego buscado, y los diferentes resultados (si existe en BBDD).





1.6 Footer

La sección Footer es un simple texto informativo sobre el Copyright (simulado) de la página web:



1. Detalle
   * 1. Backend

Por la parte del Servidor, además, tenemos como arquitectura los microservicios, basados en un único punto de entrada (Gateway), que hace de filtro y distribuidor para todas las solicitudes recibidas.

Todos los servicios exponen una API REST con endpoints CRUD (Create Read Update Delete) para así ofrecer todas las operaciones posibles y coherentes.

A continuación, se enumeran y posteriormente desarrollan las funcionalidades de la aplicación, y por tanto los microservicios, siendo estos:

1. Servidor de Registro Eureka

[2. Mo. Autenticación](#Auth)

[3. Mo. Usuarios](#Usuarios)

[4. Mo. Juegos](#Juegos)

[5. Mo. Pedidos](#Pedidos)

[6. Mo. Mail Sender](#MailSender)

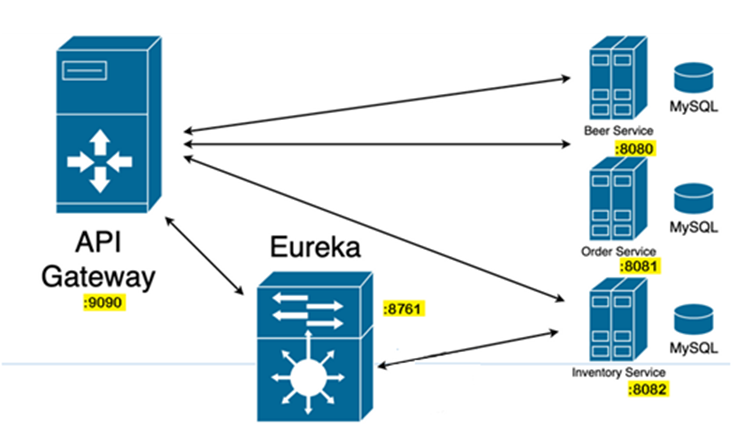
[7. Mo. Gateway](#gateway)

Mo = Microservicio

3.1.1 Servidor de Registro Eureka:

Es un servidor cuya funcionalidad es facilitar la comunicación entre los distintos servicios.

Está basado en un endpoint al cual los servicios se subscriben, y a la hora de realizar comunicaciones, sólo se necesita conocer el nombre del otro microservicio, no su dirección IP + puerto.











[1. Arquitectura Ejemplo Eureka Server](https://keepcoding.io/blog/que-es-eureka-server-en-la-arq-de-microservicio/)

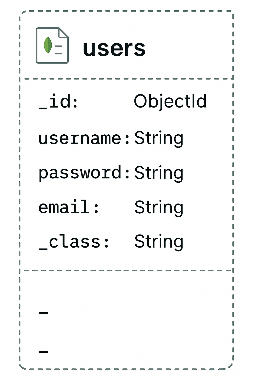
Esta imagen (de ejemplo) ilustra a la perfección la arquitectura de LevelUp Shop a nivel de Backend, ya que existe un servidor Eureka entre la Puerta de Entrada (Gateway) y los microservicios, que facilita la comunicación entre ellos, ya que, al no necesitar la dirección completa de cada instancia de microservicio, esta configuración es la más actual y escalable posible. Puede portarse el sistema entero a otra máquina, y no habrá que cambiar nada en la configuración interna de cada componente.

También, podrá escalarse horizontalmente, ya que no habría posibles conflictos de ocupación de puertos en la misma red.

3.1.2 Microservicio Autenticación

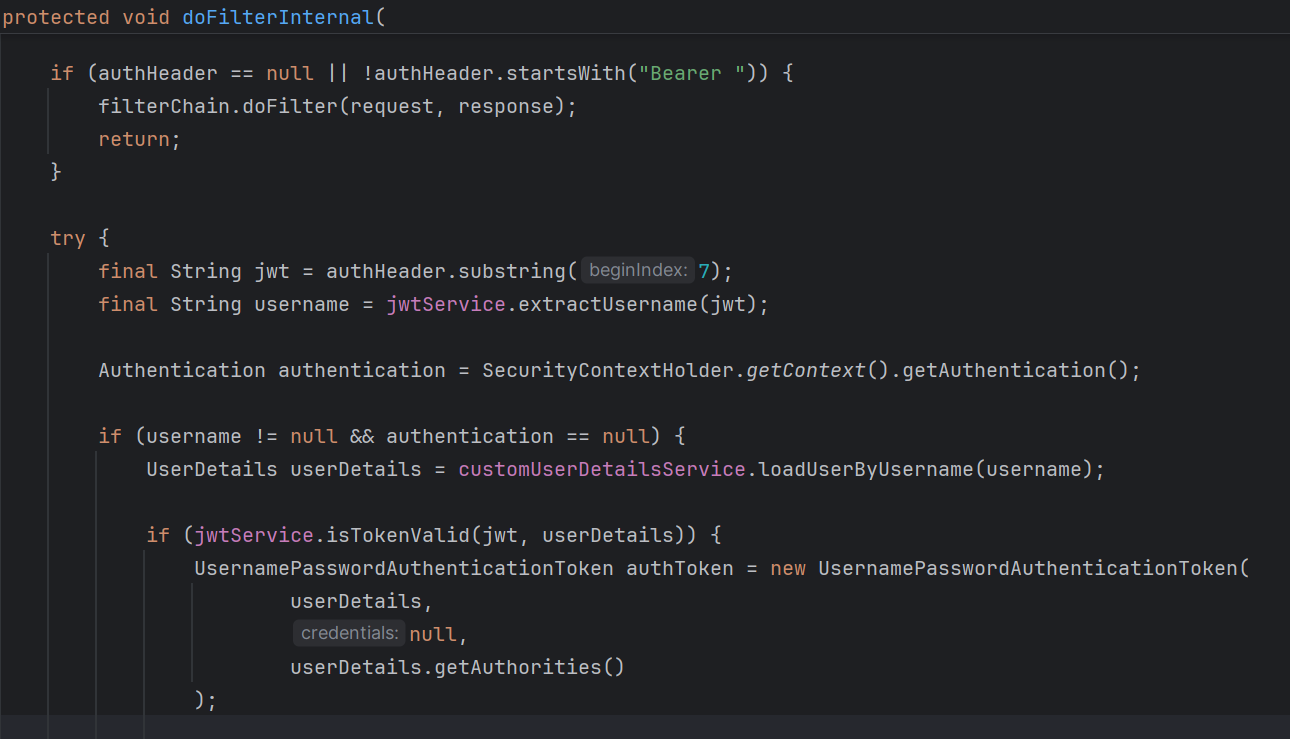
Maneja la lógica de registro, inicio de sesión, autenticación y autorización.

En su Base de Datos se guarda información sensible de los usuarios relacionada con la seguridad:



Implementa una barrera de seguridad por token JWT que permite a los usuarios realizar operaciones recurrentemente sin necesidad de iniciar sesión a menudo. Se puede fijar un tiempo de expiración, y el mensaje del token está cifrado por una clave simétrica.

2. Modelo Autenticación



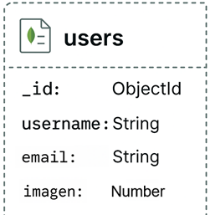
3. Microservicio Autenticación.

En este fragmento de código, se puede observar cómo se procesa el token de autenticación JWT, para su posterior validación y obtención de las credenciales del usuario para proceder a permitirle el acceso al recurso protegido, o en su defecto devolverle un error de autenticación que se traducirá en una redirección al login por parte del Frontend.

El servicio auxiliar JwtService se encarga de obtener todas las credenciales, así como verificar que idénticas a las almacenadas en Base de Datos de manera cifrada.

3.1.3 Microservicio Usuarios

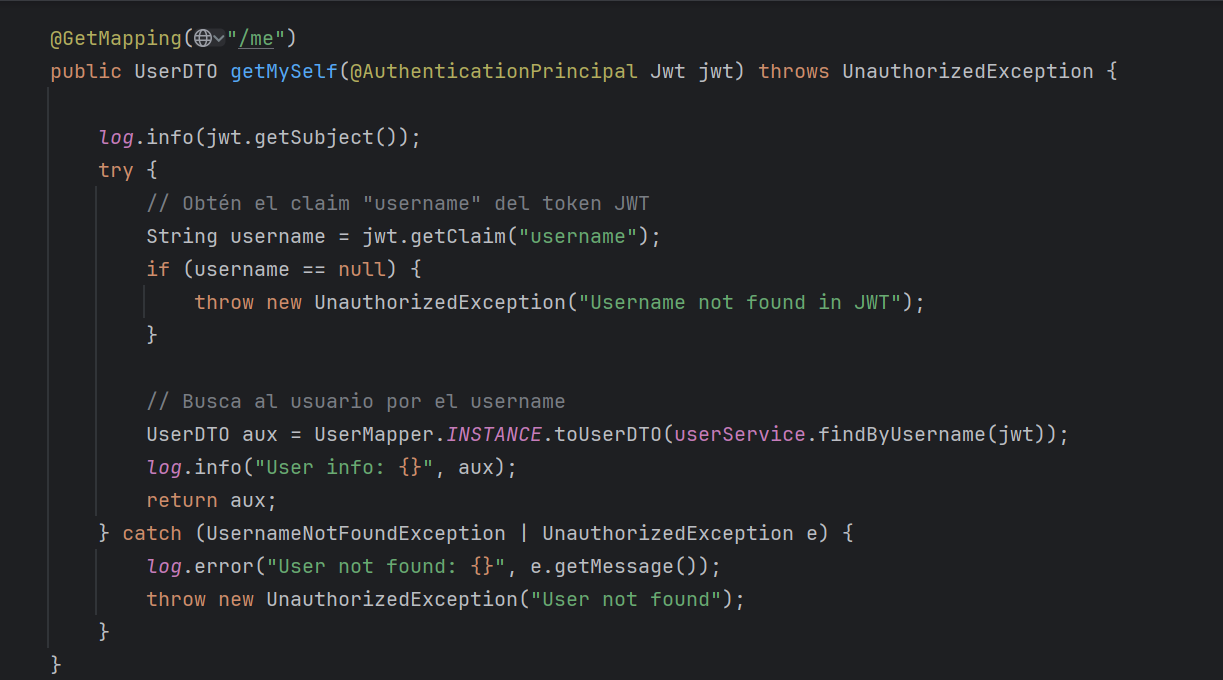
Íntimo cooperador del Microservicio Autenticación, pero sin embargo se encarga de almacenar y procesar la información menos sensible del usuario, pero de igual manera importante, como son:



Estos atributos tienen posible ampliación según necesidades más concretas del ecommerce. No almaceno dirección del usuario puesto que la tienda está basada en la premisa de la compra 100% digital.



4. Modelo Usuario



5. Controlador Microservicio Usuarios

En este método del controlador, podemos observar cómo a partir del token de autenticación JWT, sin necesidad de volver a iniciar sesión de nuevo, este servicio nos devuelve la información disponible del usuario en tiempo real realizando consulta a BBDD.

Primero se obtiene el nombre de usuario aprovechando los métodos de la propia clase JWT, para más tarde realizar una búsqueda en BBDD a partir de este dato, para devolver toda la información del usuario, o una excepción personalizada UsernameNotFoundException en caso de no existir el usuario, o una UnauthorizedException en caso de que el token no fuera válido.

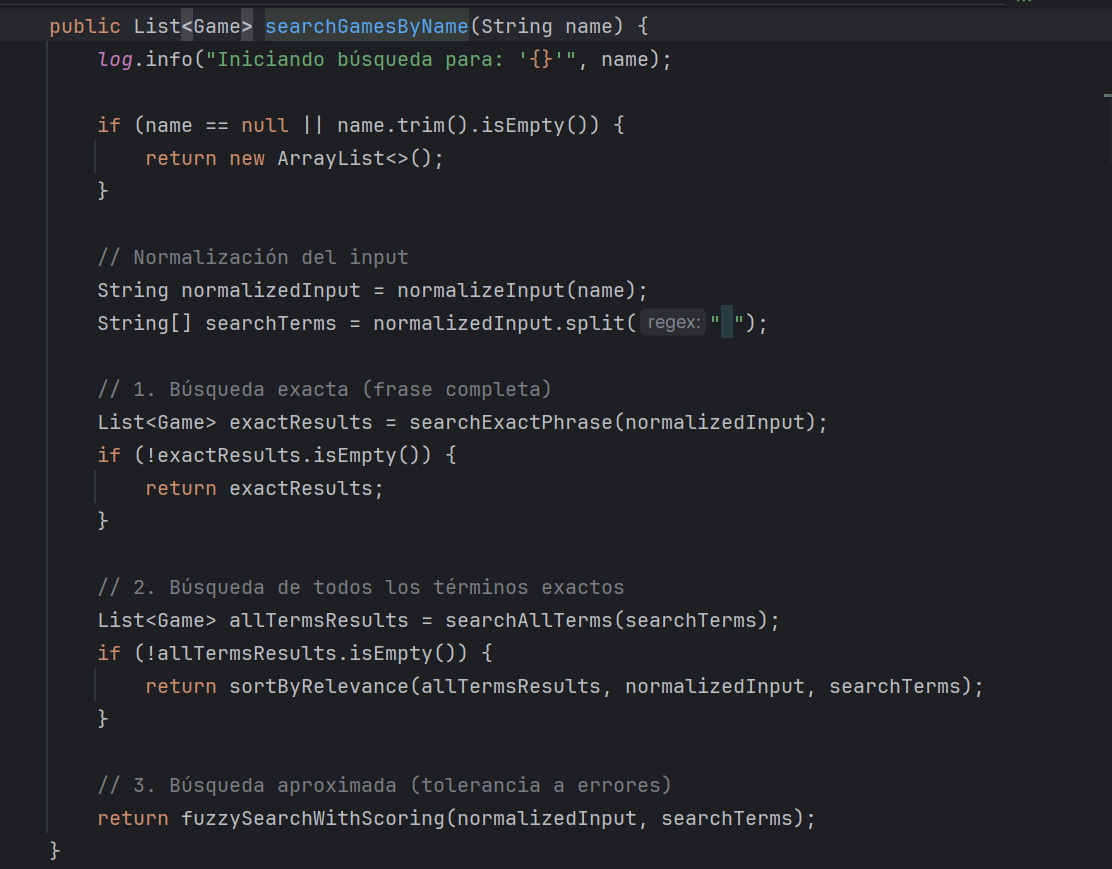
3.1.4 Microservicio Juegos

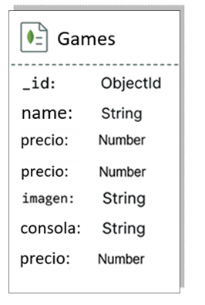
Servicio encargado de la lógica relativa a los juegos.

Posee todo tipo de operaciones necesarias para el tipo de actividad de la empresa.

Debido a que los juegos son obtenidos a partir de la consumición de una API externa, me pareció ético tratar dichos datos, y antes de almacenarlos, variar sus cualidades, descartar ciertos atributos, y agregar otros personalizados para aumentar la complejidad de este servicio para no ser un simple cliente de una API que almacena juegos en BBDD y después los muestra.

Es por esto, que, aunque inicialmente los juegos consumidos sólo ofrecían nombre, valoración, imagen, finalmente la colección Games queda de esta forma:





6. Modelo Juego



7. Microservicio Games.

En este fragmento de código, se puede observar el método de búsqueda por nombre de juego.

Este, realiza una búsqueda exhaustiva, otorgando puntuaciones a los distintos posibles resultados, para después poder ordenarlos en la lista que devuelve. Es por esto, que *The Witcher 3* se devolverá en primer lugar si la búsqueda es “the witcher”, pero también se devolverán en sucesión todos los juegos con “The” o “Witcher” en su nombre, como podrían ser *The Crew*.

Cabe decir que esta operación tiene métodos auxiliares como serían searchAllTerms, o searchExactPhrase.

Este servicio no lo he considerado uno de los más complejos, pero la operación de búsqueda sí que me llevó bastante tiempo, porque quise ofrecer una experiencia lo más cómoda y real al usuario.

Aunque el propio repositorio ofrece un método de búsqueda exacta findByName, pero no me pareció suficiente.

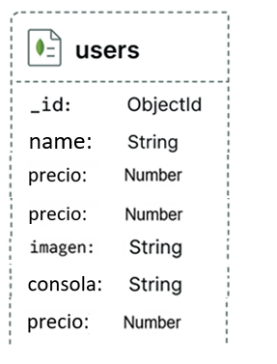
3.1.5 Microservicio Pedidos

Servicio encargado de la lógica de los pedidos.

Microservicio simple que sólo se encarga de almacenar los pedidos. CRUD limitado a Create y Read.

Los Read en su mayoría se reservan a operaciones de administrador, salvo el findAllByUserId, que permite a cada usuario visualizar en su perfil los pedidos realizados por sí mismo.

La colección orders quedaría definida de la siguiente forma:











8. Modelo Pedido

3.1.6 Microservicio Mail Sender

Servicio totalmente auxiliar, encargado de la comunicación a nivel de correo electrónico de la empresa.

Se ha optado por mensajes codificados en HTML para poder ofrecer al usuario una experiencia visual más atractiva.



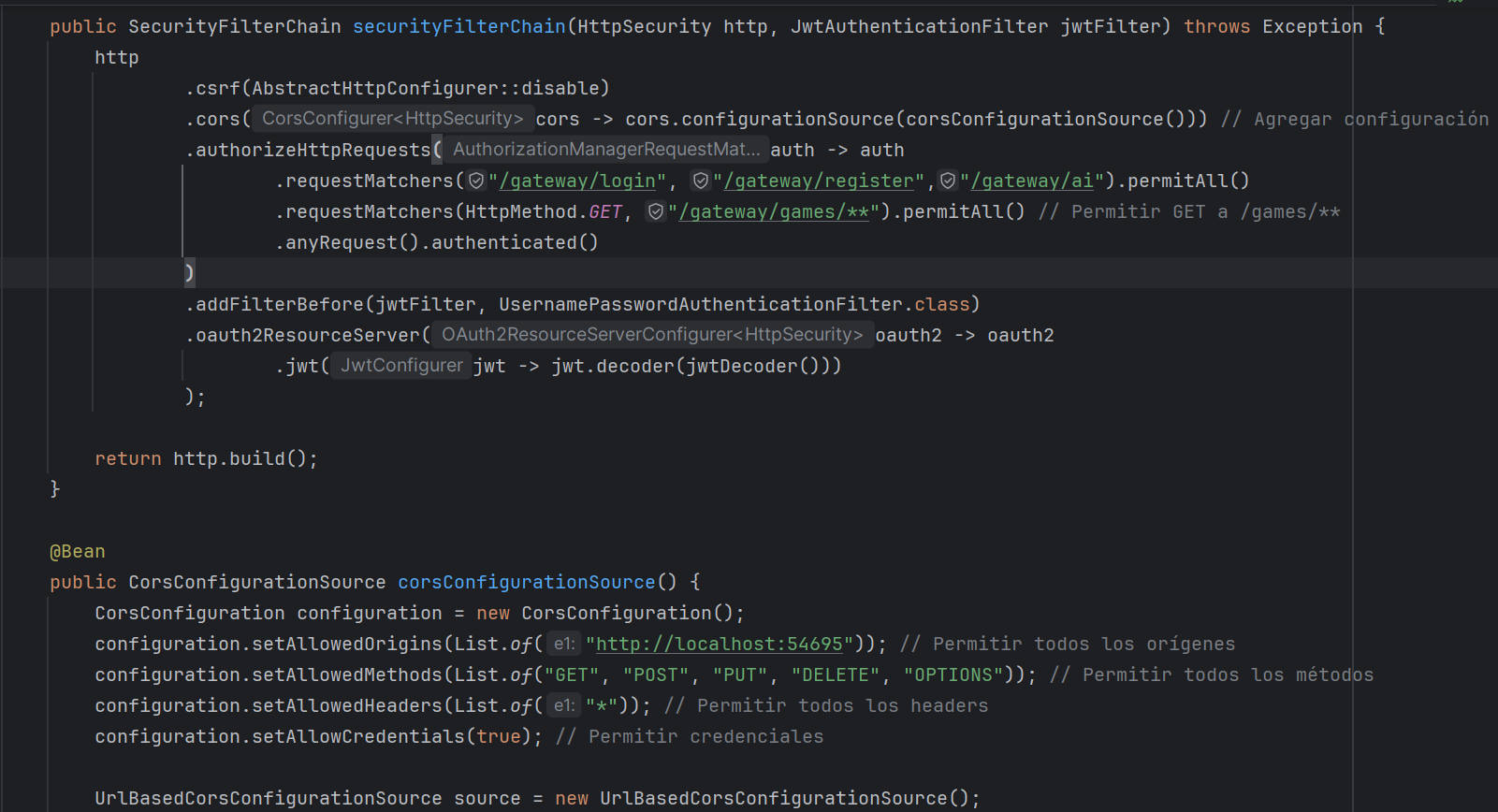
9. HTML correo compra.

3.1.7 Microservicio Gateway

Es el servicio más complejo de la aplicación, y el que le da sentido y soporte a la estructura al completo.

Es por esto, que se ha desarrollado siguiendo las mejores prácticas posibles, posibilitando que exista un gran flujo de peticiones y de distintas procedencias recibiéndose al mismo tiempo sin bloquearse. Lo he logrado implementando Load Balancer en el mecanismo de comunicación entre servicios.

Ni el [Microservicio Mail Sender](#MailSender) ni la Gateway poseen BBDD, ya que su propia naturaleza no lo requiere. No hay dato alguno que sea imperativo almacenar relativo a estas dos funcionalidades.



10. Configuración Gateway

En este pantallazo, se muestra la configuración de seguridad de la Gateway, la cual como se puede observar, expone varios endpoints de manera pública, además de todas las peticiones GET al [microservicio de Juegos](#Juegos) para poder mostrar los juegos al usuario sin necesidad de registro.

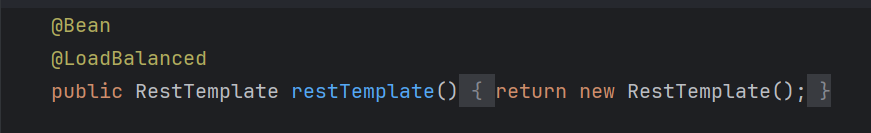
Sin embargo, el resto de peticiones requerirán de autenticación.

Dado que la Gateway es una especie de filtro, esta no protege los propios métodos, sólo protege la entrada a los microservicios a través de los endpoints.

La seguridad de métodos específicos, como por ejemplo intentar un DELETE a un usuario, estará configurada dentro del propio microservicio concreto.

Por otro lado, también está configurada la seguridad frente a CORS (Cross-Origin Resource Sharing), medida obligatoria en navegadores actuales, para que no se puedan realizar peticiones a la API desde dominios externos a la propia máquina y el puerto designado al Frontend.

En el siguiente pantallazo de código, se puede observar cómo en la dependencia de comunicación con los microservicios (RestTemplate), se implementa la anotación *@LoadBalanced*,para así hacer uso del balanceo de carga y mejorar el rendimiento cuando la exigencia es alta.



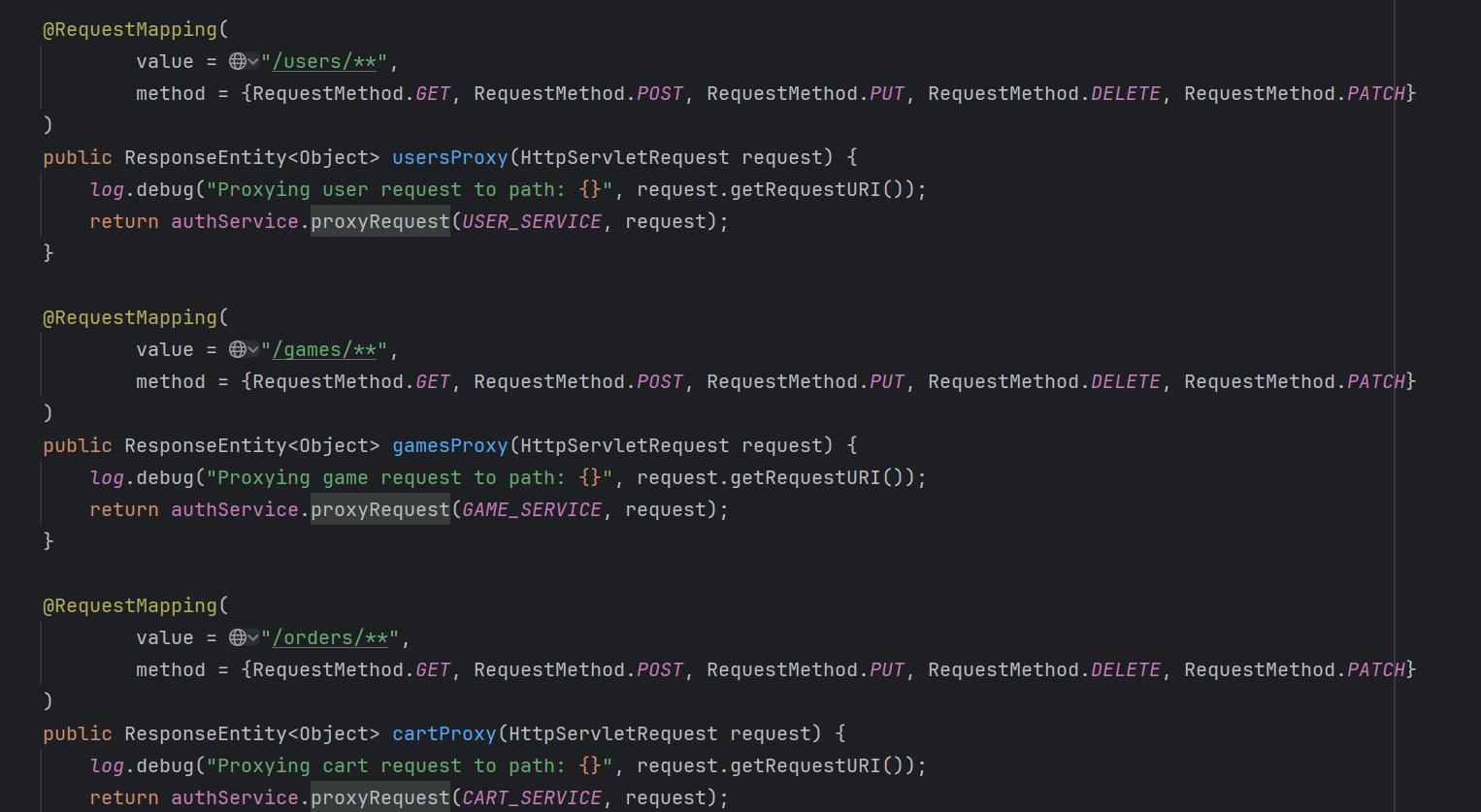
11. Configuración Balanceo Carga en Gateway



12. Métodos públicos Gateway

Por la parte del controlador, en la imagen superior se muestran el endpoint *login* y el de *registro*, mapeados para aceptar sólo parámetros previamente definidos como serían las clases *LoginRequest* y *RegisterUsersRequest*. Son Operaciones públicas en cuanto a autenticación puesto que son las que permiten al usuario iniciar sesión y registrarse.

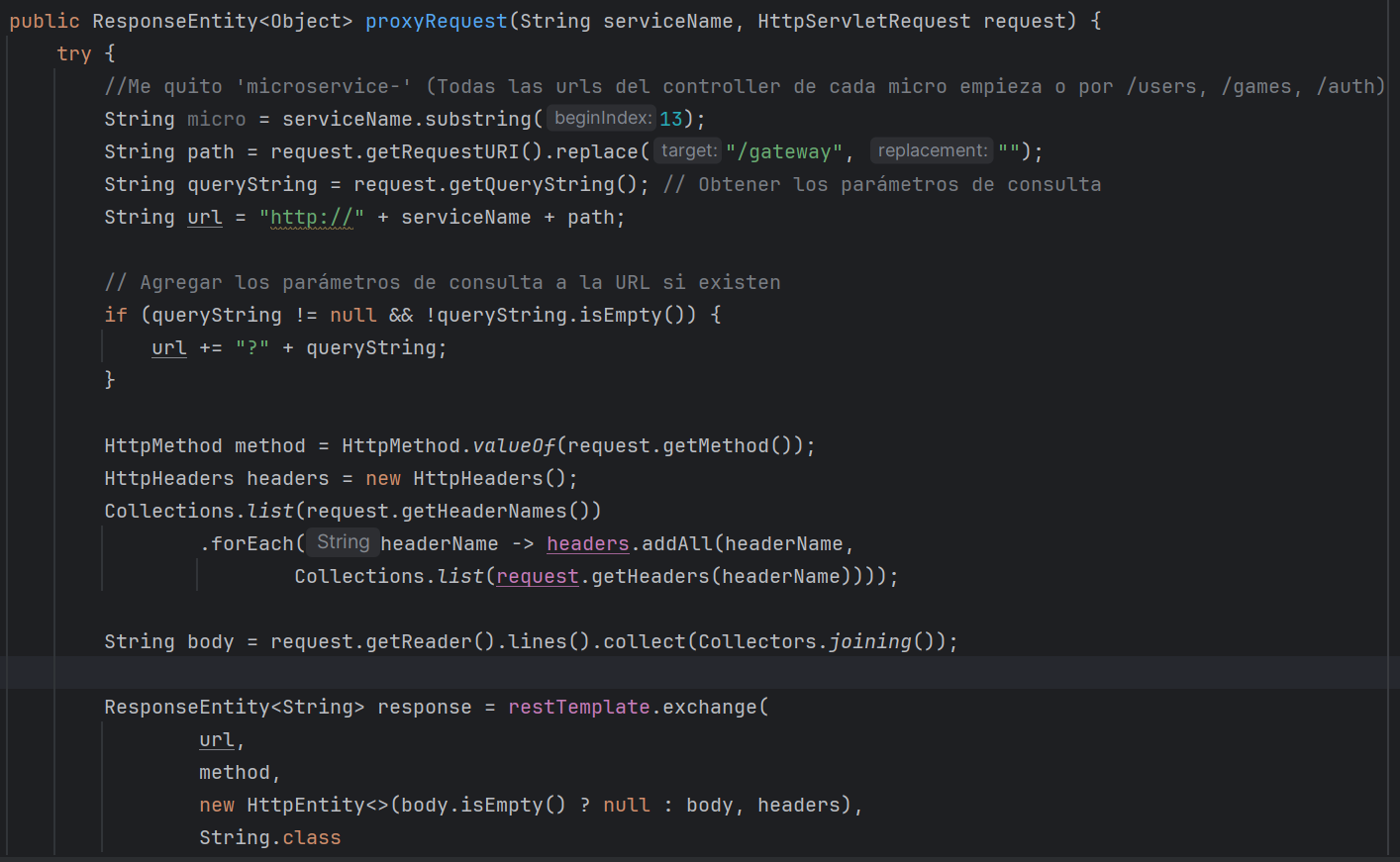
La respuesta del login será un ResponseEntity (Código de respuesta 200 OK o 401 Unauthorized) + cadena de texto que contiene el token JWT que se usará para seguir realizando peticiones de forma segura y disponible.



Los mostrados en la imagen anterior, son los métodos que redireccionan a los microservicios.

He definido los distintos endpoints para poder filtrar y no recibir peticiones al azar que en un caso dado pudieran llegar de rebote a algún micro.

Usan un método común de la clase *AuthService*, que se encarga de hacer el redireccionamiento al servicio correspondiente:



* 1. Manual de Usuario

Todos estos servicios necesitan colaborar estrechamente entre sí para lograr las funcionalidades de la aplicación.

1. Alternativas

Conclusión

1. Referencias Bibliográficas

Anexos