Manual Técnico

Proyecto 2

e-commerce-GT

Curso: Manejo e Implementación de Archivos
Estudiante: Herberth Julian Reyes Pacajoj
Registro académico: 202230236

Índice

Generalidades

Herramientas utilizadas

Sistema operativo: Ubuntu - versión 24.04.1 LTS

Ubuntu es un sistema operativo de código abierto basado en Linux, diseñado para ser accesible, seguro y fácil de usar. Es una de las distribuciones de Linux más populares en el mundo.

Angular - versión 20.3.0

Angular es un framework de desarrollo de código abierto, mantenido por Google, que se utiliza para crear aplicaciones web modernas, rápidas y escalables, especialmente aplicaciones de una sola página (SPA). Está basado en TypeScript y utiliza una arquitectura modular de componentes, ofreciendo un conjunto completo de herramientas para simplificar el flujo de trabajo de los desarrolladores.

Angular Material - versión 20.2.8

Angular Material es una biblioteca de componentes de interfaz de usuario (UI) para aplicaciones web construidas con el framework Angular que proporciona un conjunto de componentes prediseñados.

Spring Boot - versión 3.5.6

Spring Boot es un framework de código abierto de Java que simplifica la creación de aplicaciones web y microservicios independientes. Se basa en el Spring Framework y ofrece características como autoconfiguración, servidores integrados y dependencias preconfiguradas para reducir el tiempo de desarrollo y la configuración manual

Java - versión 17

Java es un lenguaje de programación y una plataforma informática, creada por Sun Microsystems en 1995 y ahora propiedad de Oracle. Es conocido por ser orientado a objetos, multiplataforma y seguro.

Generalidades

Herramientas utilizadas

Maven - versión 3.8.7

Maven es una herramienta de código abierto de la Fundación Apache que se utiliza para la gestión y automatización de la construcción de proyectos de software, principalmente en Java.

PostgreSQL - versión 16.10

PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto muy avanzado, conocido por su fiabilidad, robustez y flexibilidad. Es compatible tanto con consultas relacionales (SQL) como no relacionales (JSON) y soporta propiedades ACID para garantizar la integridad de los datos.

Hibernate - versión 6.6.29

Hibernate es un framework de mapeo objeto-relacional (ORM) para Java que simplifica la interacción entre aplicaciones y bases de datos relacionales.

MapStruct - versión 1.6.3

MapStruct es una librería de código abierto para Java que automatiza la generación de código para mapear objetos (bean) durante la compilación, lo que evita escribir manualmente el código repetitivo de conversión y hace el código más legible y eficiente.

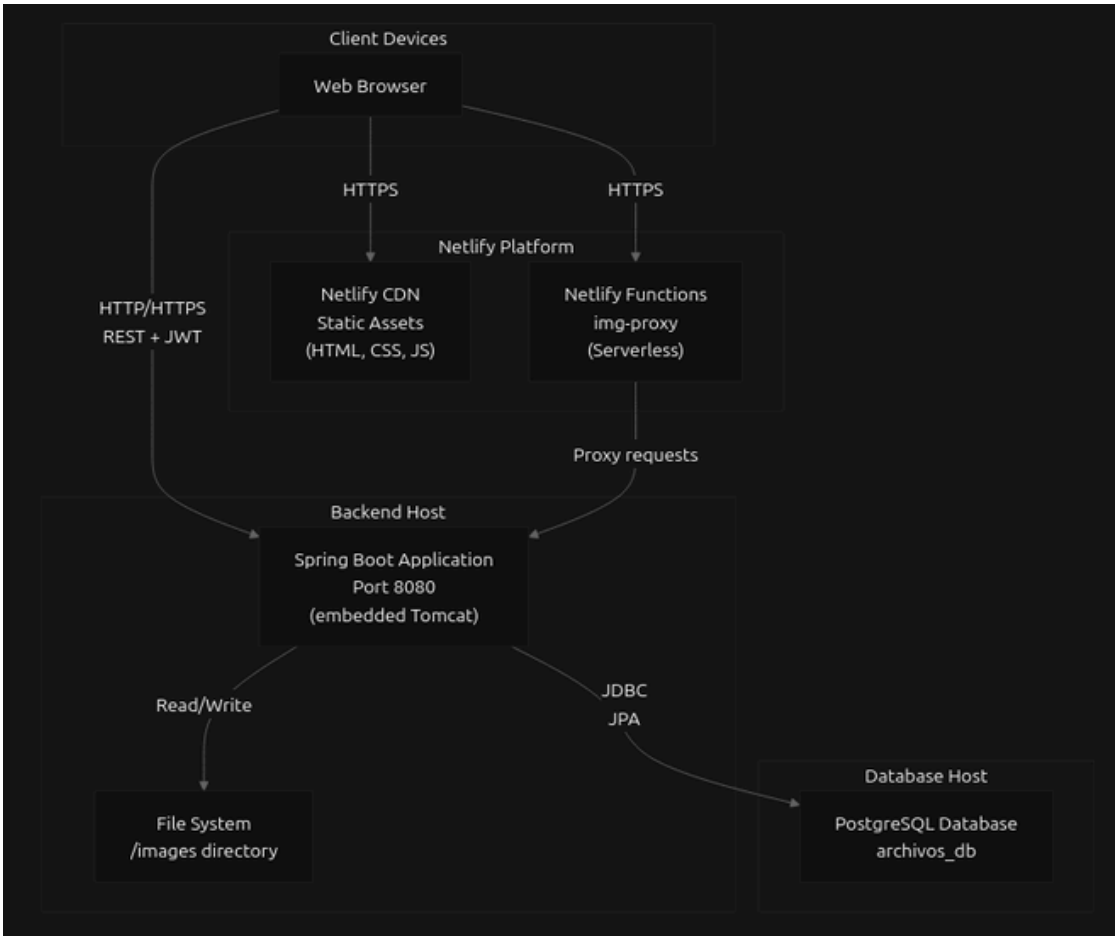
Lombok - versión 1.18.34

Lombok es una biblioteca de Java que ayuda a los desarrolladores a reducir el código repetitivo y de baja relevancia ("boilerplate") mediante el uso de anotaciones. Genera automáticamente métodos comunes como getters, setters, constructores, equals(), hashCode(), y toString() en tiempo de compilación, haciendo el código más conciso, legible y rápido de escribir.

Arquitectura

Arquitectura general

El sistema sigue una arquitectura cliente-servidor con el frontend y el backend desplegados de forma independiente.

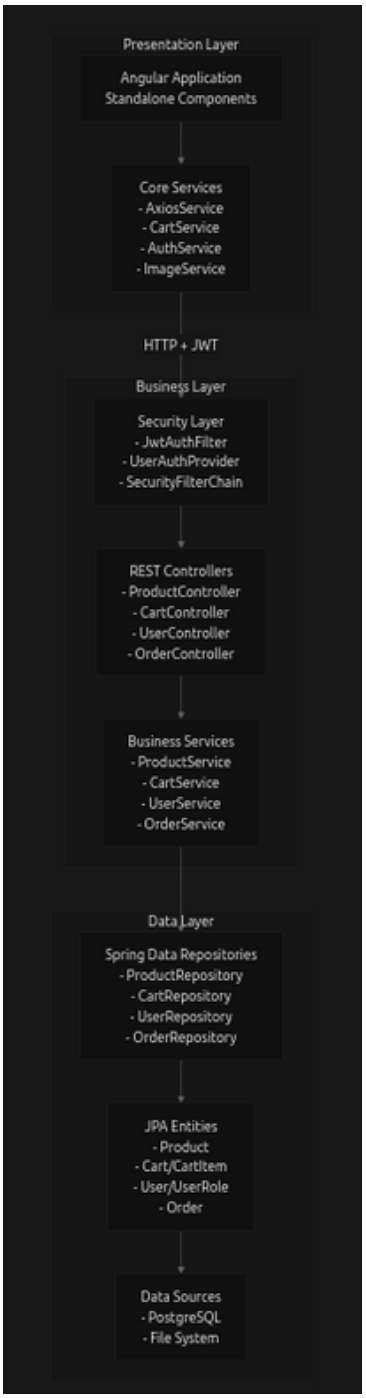


El frontend se implementa como una aplicación estática de una sola página (SPA) en la CDN de Netlify. Netlify también aloja funciones sin servidor, específicamente img-proxy, que representa las solicitudes de imágenes del backend para evitar advertencias CORS de ngrok durante el desarrollo. El backend se ejecuta como una aplicación independiente Spring Boot con un servidor Tomcat integrado, que se conecta a PostgreSQL a través de JDBC y almacena imágenes de productos en el sistema de archivos local.

Arquitectura

Arquitectura lógica

El sistema implementa una arquitectura de tres niveles con una clara separación entre la presentación, la lógica de negocio y las capas de acceso a los datos.



Cada nivel tiene distintas responsabilidades:

Presentación

- Componentes: Angular components, services, routing
- Responsabilidades: Renderización de la interfaz de usuario, interacción del usuario, gestión de estado del lado del cliente, comunicación HTTP

Negocios

- Componentes: Spring controllers, services, security filters
- Responsabilidades: Enrutamiento de solicitudes, autenticación/autorización, lógica de negocio, transformación de datos

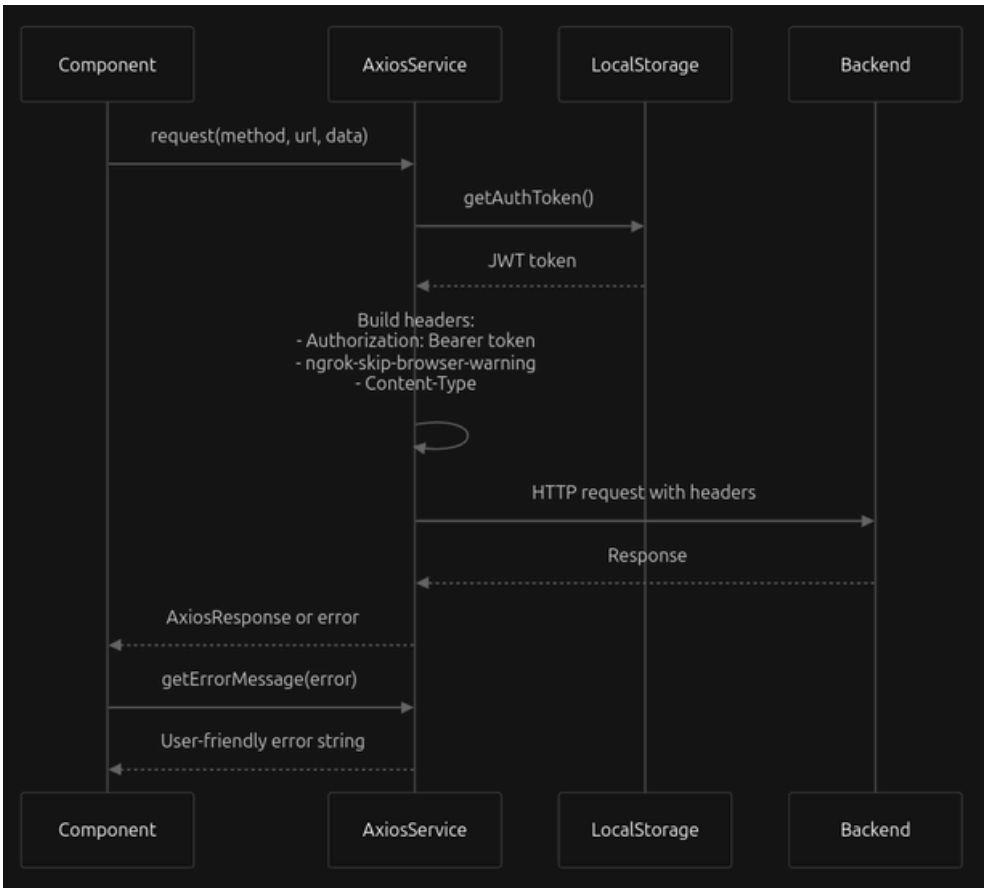
Datos

- Componentes: JPA repositories, entities, database
- Responsabilidades: Persistencia de datos, ejecución de consultas, gestión de transacciones

Patrón de comunicación HTTP

Axios Service

AxiosService centraliza toda la comunicación HTTP con el backend:



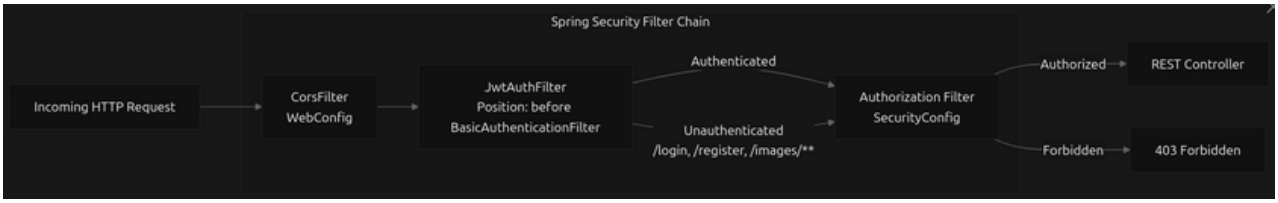
El método AxiosService.request() automáticamente:

- Recupera el token JWT de localStorage con la clave "auth_token"
- Adjunta Authorization: Bearer <token> al header
- Añade el header ngrok-skip-browser-warning: true para evitar la advertencia del navegador de ngrok
- Agrega Content-Type: application/json para solicitudes que no sean de FormData
- Maneja correctamente FormData permitiendo que Axios establezca el límite

Seguridad

Arquitectura de seguridad

La seguridad se aplica a través de un mecanismo de autenticación stateless basado en JWT integrado con Spring Security.



El método **SecurityConfig.securityFilterChain()** configura:

- CSRF desactivado - Apropiado para APIs basadas en JWT stateless
- Gestión de sesiones - SessionCreationPolicy.STATELESS para evitar la creación de sesiones
- Filtrar registro - JwtAuthFilter añadido antes de BasicAuthenticationFilter
- Endpoints públicos - POST /login, POST /register, y GET /images/**

JwtAuthFilter:

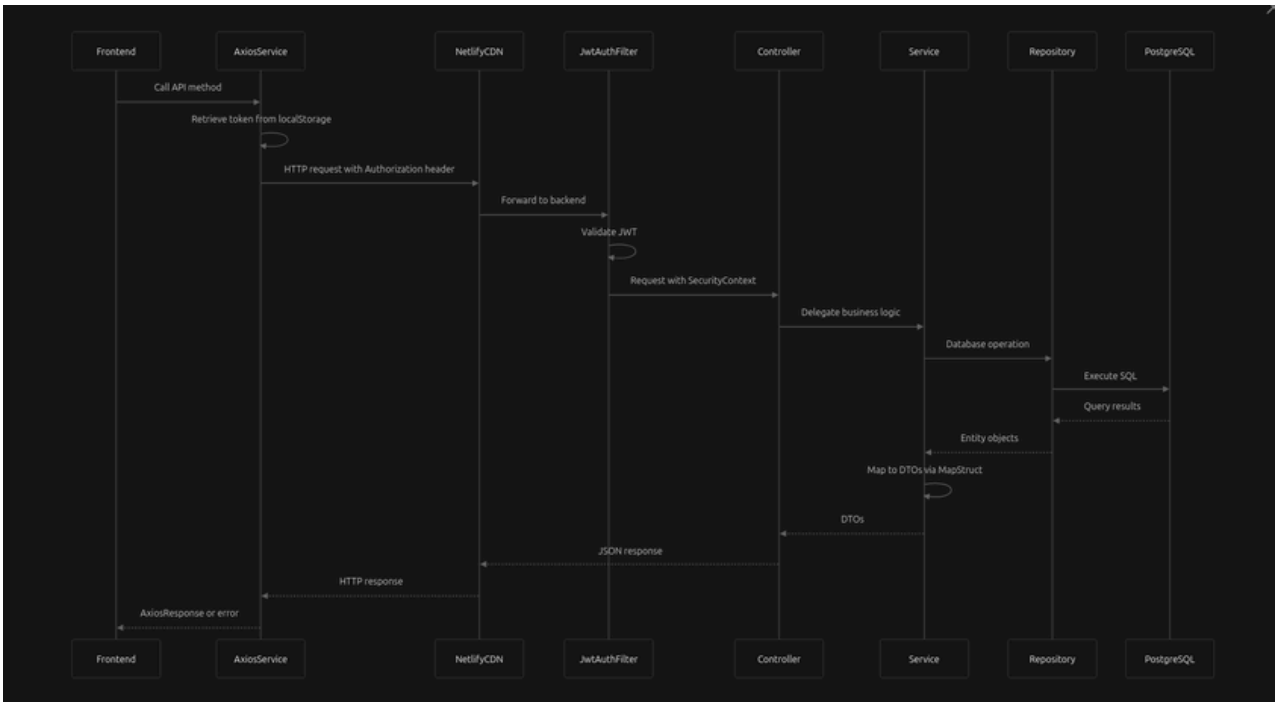
- Extiende de OncePerRequestFilter para ejecutar una vez por solicitud
- Extrae JWT de Authorization: Bearer <token> header
- Llama a userAuthProvider.validateTokenStrongly() para validar el token y recuperar los detalles del usuario
- Añade un SecurityContextHolder con la información del usuario autenticado

WebConfig.corsFilter(): Permite solicitudes de origen cruzado de:

Origin Pattern	Purpose
`http://localhost:4200`	Local Angular development server
`https://*.netlify.app`	Production Netlify deployments
`https://*.ngrok-free.app`	Backend development tunnel

Comunicación REST API

Toda la comunicación frontend-backend utiliza REST sobre HTTP/HTTPS con JSON payloads (excepto las cargas de imágenes que utilizan multipart/form-data).



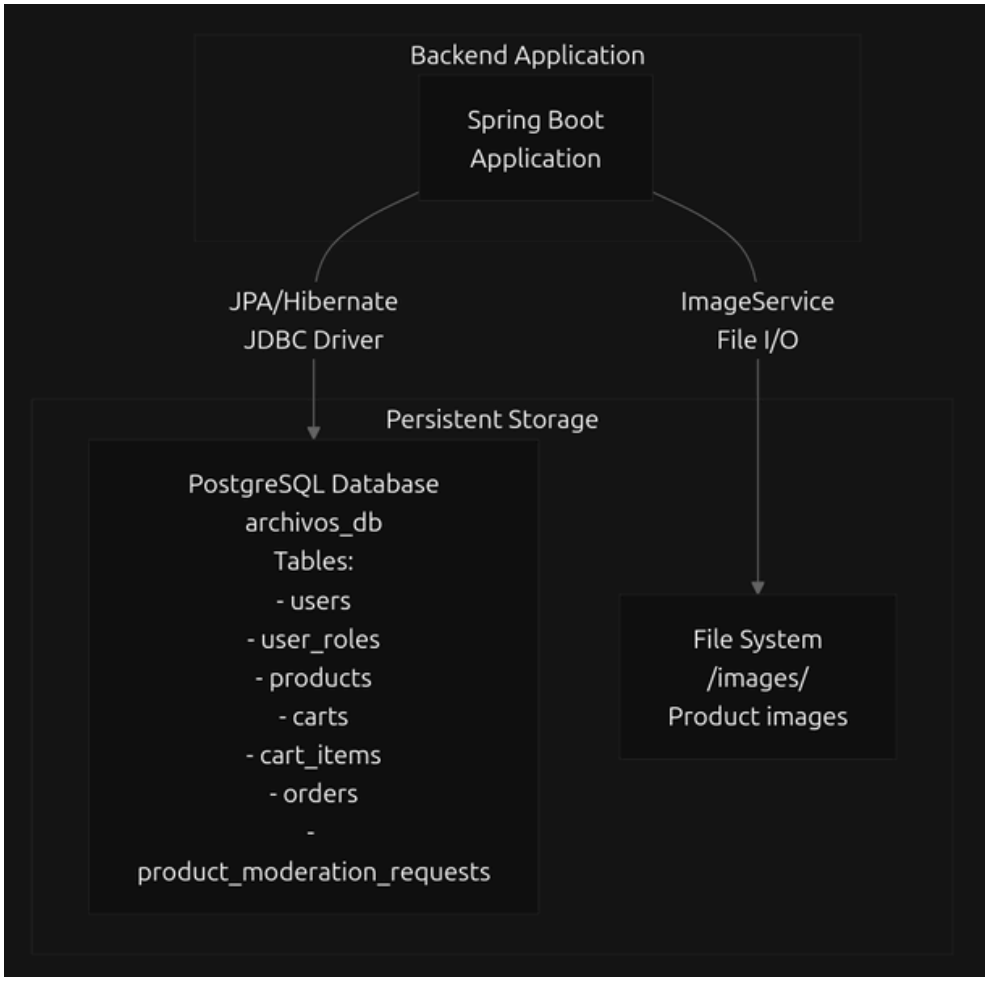
REST API Controllers

La aplicación implementa tres controladores principales de REST que forman la capa de API HTTP:

Controlador	Responsabilidad	Ruta Base
`ProductController`	Producto CRUD, catálogo y detalles	`/products`
`CartController`	Gestión de artículos de carrito y carrito	`/cart`
`UserController`	Autenticación y gestión de usuarios	`/login`, `/register`, `/users`

Capa de Datos

Componentes de almacenamiento de datos

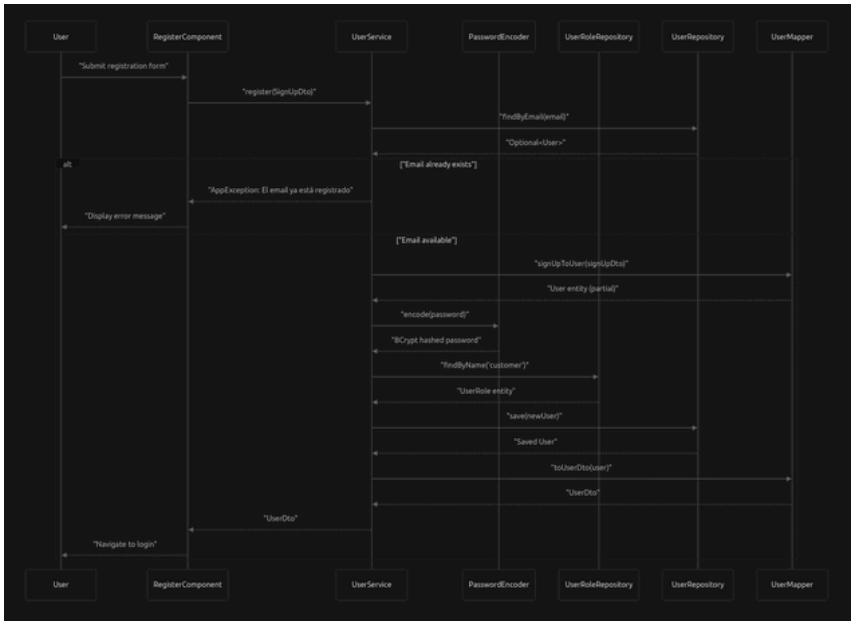


La base de datos PostgreSQL (configurada a través de Spring Data JPA) almacena todos los datos estructurados, incluidos los usuarios, productos, carritos y pedidos. Las imágenes del los productos se almacenan en el sistema de archivos en el /images/, con metadatos (nombre de archivo, ruta) almacenados en la base de datos. Este enfoque híbrido optimiza el rendimiento de la base de datos al tiempo que proporciona un servicio de imagen eficiente. El controlador PostgreSQL JDBC permite a Hibernate ORM de Spring Boot gestionar conexiones de base de datos y ejecutar consultas a través de repositorios de Spring Data JPA.

Características clave

Registro de usuario e inicio de sesión

El workflow de autenticación maneja el registro de usuarios, la validación de credenciales, la generación de tokens JWT y la autenticación basada en tokens para solicitudes posteriores. El sistema admite dos rutas de registro: autoinscripción del cliente a través de SignUpDto y registro de empleados administrados a través de EmployeeRegisterDto.

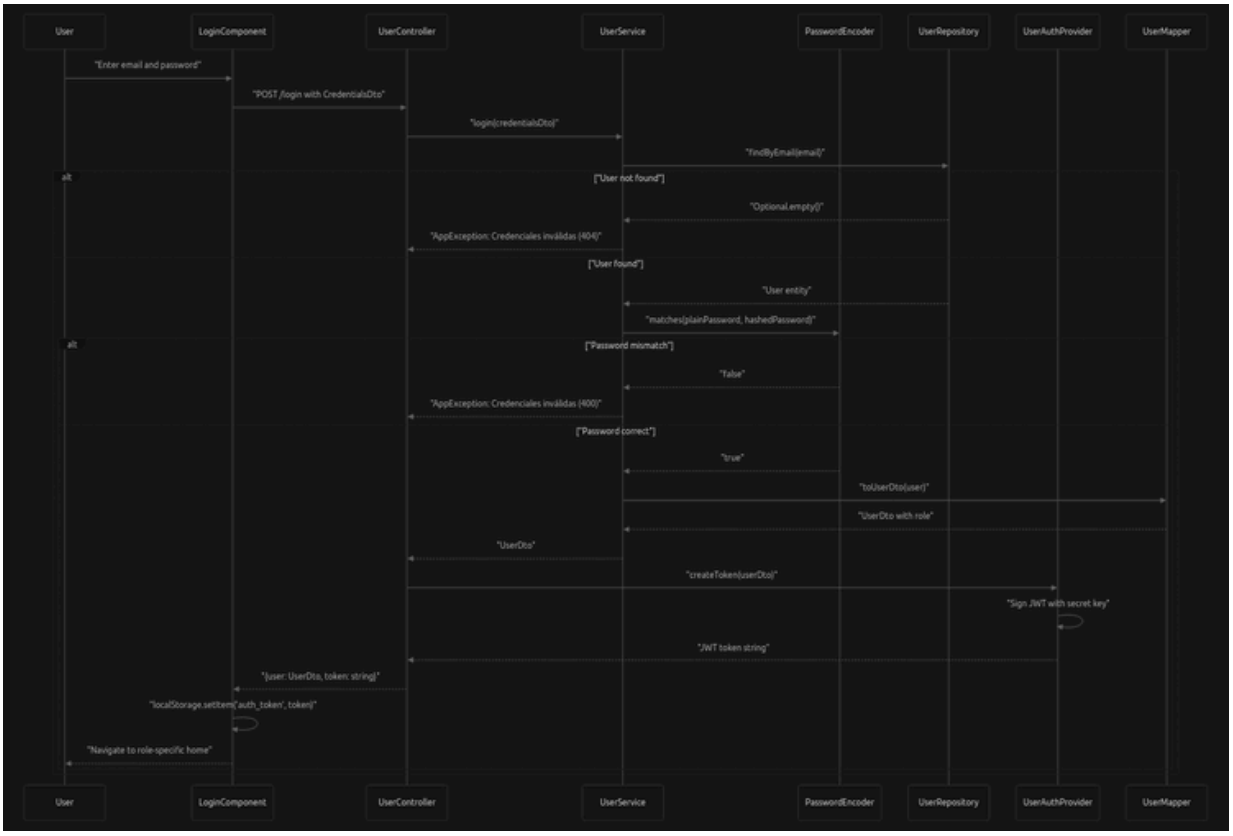


Detalles clave de la implementación:

- Seguridad de contraseñas: En el método PasswordEncoder.encode(CharBuffer.wrap(password)) se aplica BCrypt hash para encriptar las contraseñas
- Asignación de rol: Los nuevos usuarios reciben automáticamente el rol customer vía userRoleRepository.findByName("customer")
- Email único: Se verifica en userRepository.findByEmail() para evitar cuentas duplicadas
- Estrategia de Mapeo: UserMapper.signUpToUser() Convierte DTO a la entidad, ignorando los campos innecesarios

Características clave

Iniciar sesión y flujo de creación de tokens



El token JWT creado por UserAuthProvider contiene los claims:

- sub: Correo electrónico del usuario
- firstName: Nombre del usuario
- lastName: El apellido del usuario
- role: Nombre de rol del usuario (cliente, moderador, administrador, logística)
- exp: Tiempo de caducidad del token

Después de un inicio de sesión exitoso, el frontend almacena el JWT en localStorage con la llave auth_token. El AuthService recupera este token para solicitudes posteriores. Todas las rutas protegidas utilizan authGuard para verificar la presencia de tokens y el acceso basado en roles.