Documentación Técnica

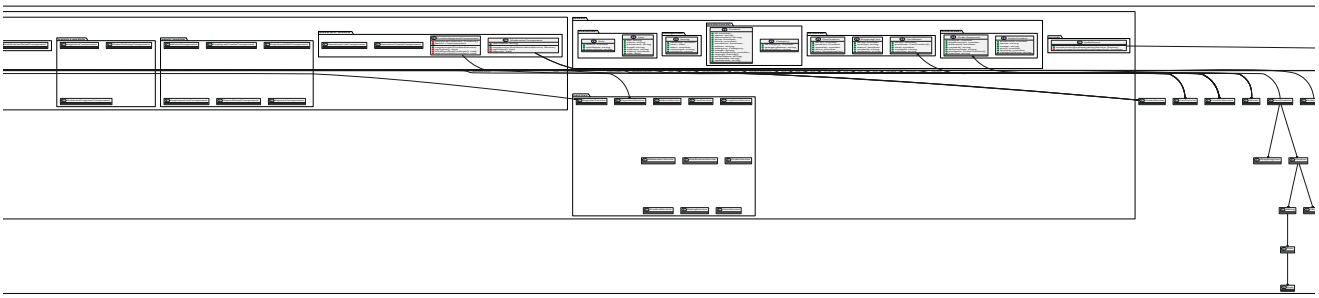
A continuación se presenta la documentación del frontend y backend, estos permiten la funcionalidad de la página web de eCommerce-GT y la api que consumirá. Para ello se utilizaron diferentes tecnologías que se describen a continuación.

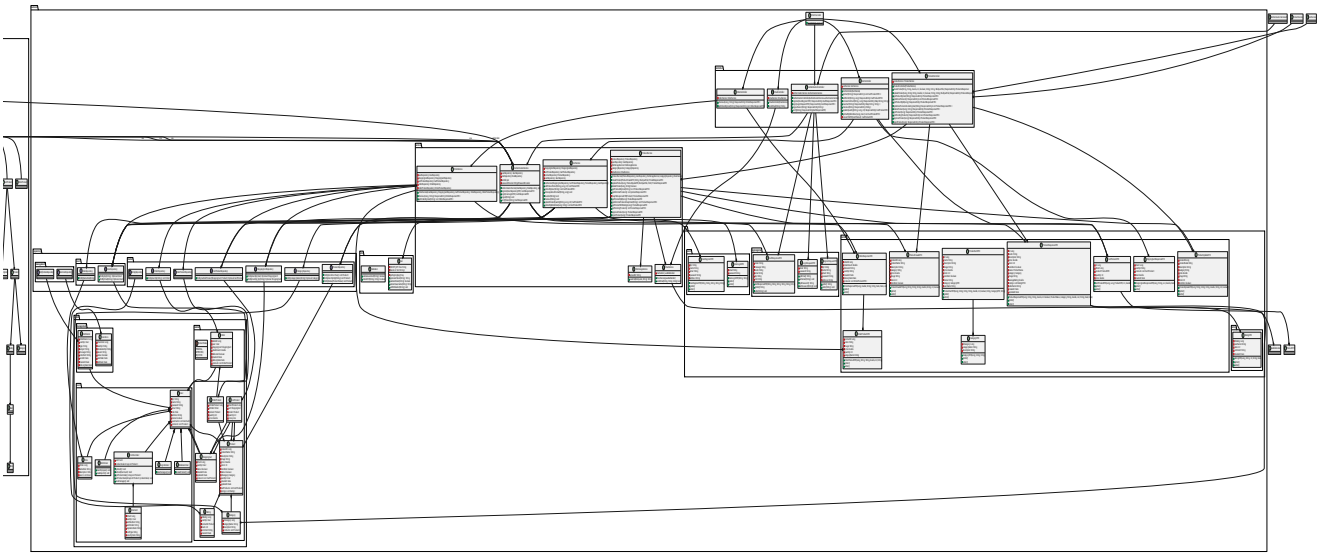
Tecnologías utilizadas

- PostgreSQL para la base de datos.
- Spring Boot para el desarrollo de la api.
- Angular para el diseño de la vista de usuario.
- Ngrok para desplegar el backend.
- Netlify para desplegar el frontend.

Diagrama de clases

A continuación se presenta el diagrama de clases, contiene las relaciones de cada clase y entidad en la aplicación. Adjunto a este documento puede encontrar la imagen para que pueda verla con mayor detalle.





Distribución de paquetes

Se distribuye en dos partes, una que es la encargada de gestionar todo el backend y otra el frontend, a continuación se describen ambas

Backend

Contiene toda la lógica del lado del servidor, organizada en controladores, servicios, modelos, repositorios y utilidades.

Paquete controller

Son controladores REST que exponen endpoints HTTP al frontend.

AuthenticationController

Maneja el registro, inicio y cierre de sesión y verificación de usuarios, contiene un atributo de tipo AuthenticationService que maneja la autenticación, sus métodos son:

register(userRegisterDTO)
registra un nuevo usuario y devuelve un token JWT

login(LoginRequestDTO)
Autentica un usuario existente

logoutSesion(String)
Invalida un token JWT y cierra la sesión

verify(String)

Verifica la validez de un token y devuelve datos del usuario autenticado.

CartController

Gestiona las operaciones del carrito de compras, tiene un atributo que implementa la lógica del carrito, sus métodos son:

getCart(String)
Obtiene los productos del carrito de compras del usuario.

addToCart(String, Long)
Agrega un producto al carrito de compras del usuarios.

removeFromCart(String, Long)
Elimina un producto del carrito de compras.

clearCart(String)
Vacía todos los productos del carrito de compras.

checkout(String)
Genera una orden a partir del carrito de compras.

updateQuantity(String, Long, int)
Cambia la cantidad de un producto en el carrito de compras, añade o elimina en una cantidad.

getCartByIdAndUser(Long, User)
Obtiene el carrito de compras específico de un usuario.

convertToDTO(CartProduct)
Transforma una entidad CartProduct en DTO asegurando la integridad de los datos de la base de datos.

EmailController

Envía correos electrónicos a usuarios, estos se envían cuando se rechazan o aprueban productos y para notificaciones de las ordenes de productos, contiene un único atributo de tipo EmailService y su método es:

sendEmail(String)
Envía un correo con el contenido indicado por parámetro.

OrderController

Gestiona las órdenes realizadas por los usuarios, tiene un atributo de tipo OrderService y sus métodos son:

checkout(Long, String)
Procesa el pago del carrito y genera la orden.

getOrdersByUser(String)
Obtiene las órdenes de un usuario en específico.

ProductController

Maneja el CRUD de productos y acciones necesarias para cumplir con las funcionalidades de la aplicación, tiene un atributo de tipo ProductService y sus métodos son:

createproduct()
Crea un nuevo producto, recibe todos parámetros que un producto nuevo debe contener.

updateProduct()
Actualiza un producto, recibe todos los parámetros que se permiten editar para actualizar el producto.

getProductByUser(String)
Obtiene todos los productos de un usuario en específico, se utiliza para que en el frontend el usuario pueda ver todos los productos que actualmente está vendiendo.

getActiveProductsExcludingUser(String)
Obtiene todos los productos excepto los que coincidan con el usuario actual, esto es porque un usuario no puede comprar un producto que él mismo vende.

deleteProduct(Long, String)
Elimina un producto del listado de productos que está vendiendo un usuario.

getPendingProducts()
Devuelve todos los productos en estado “pendiente” para que pueda verlos el usuario con rol moderador.

approveProduct(Long)
Cambia el estado del producto a “aprobado” y notifica enviando un correo electrónico.

rejectProduct(Long)
Cambia el estado del producto a “rechazado” y notifica enviando un correo electrónico.

Paquete dto

Contiene los Data Transfer Objects usados para enviar y recibir datos sin exponer directamente las entidades JPA. Este paquete contiene subpaquetes los cuales contienen los dto de cada entidad.

Paquete loginregister

Este se encarga gestionar los DTOs para los proceso de inicio de sesión y registro de nuevos usuarios.

AuthResponseDTO

Contiene la respuesta de autenticación que se enviará al frontend, sus atributos son: token, mensaje, nombre, email. rol, dpi.

LoginRequestDTO

Contiene los datos de inicio de sesión, email y password.

RegisterRequestDTO

Contiene los datos de registro de nuevo usuario: dpi, nombre, email, password, dirección.

Paquete market

Contiene los DTOs relacionados con productos, carritos y órdenes.

CartProductDTO

Representa un producto dentro del carrito de compras: id, product, cantidad, precio.

CategoryDTO

Representa la categoría de un producto: id, nombre, descripción.

OrderResponseDTO

Representa el detalle completo de una orden.

ProductCartDTO

Representa la información del producto cuando se ve desde el carrito.

ProductCreateDTO

Contiene los datos necesarios para crear o actualizar productos.

ProductUpdateDTO

Contiene los datos necesarios para crear o actualizar productos.

ProductResponseDTO

Contiene una respuesta completa con toda la información del producto: categoría, calificaciones, fechas, vendedor.

ShoppingCartResponseDTO

Contiene la respuesta del carrito con total, fecha y lista de productos.

Paquete rating

Contiene una clase con comentarios y estrellas dados por los usuarios.

Paquete users

Contiene las clases para el inicio de sesión y registro.

UserLoginDTO

Contiene los datos para el login: email, password.

UserRegisterDTO

Contiene los datos del registro: dpi, nombre, email, dirección, pasword.

Paquete service

Contiene todos los servicios que son usados por los controladores.

AuthenticationService

Maneja registro, login, logout y verificación y JWT.

CartService

Se encarga de gestionar el carrito: agrega, quitar, actualizar cantidades, limpiar, generar compra.

OrderService

Se encarga de la gestión de de la creación y consulta de órdenes.

ProductService

Implementa el CRUD de productos, aprobación, rechazo y carga de imágenes, tiene atributos cómo ProductRepository, CategoryRepository, FileStorageService, EmailService.

EmailService

Se encarga de enviar correos usando JavaMailSender.

FileStorageService

Se encarga de guardar las imágenes en el servidor.

Paquete utils

Contiene todas las utilidaes que el sistema pueda necesitar.

Jwt

Se encarga de generar y revocar tokesn jwt.

Validation

Se encarga de validar datos de las tarjetas que registraran los usuarios.

Frontend

El frontend de la aplicación está desarrollado con Angular y se encarga de manejar la interfaz de usuario, mostrando los datos recibidos desde el backend y permitiendo la interacción con la aplicación. Se organiza en módulos y componentes según el tipo de usuario y funcionalidad.

Módulos y Componentes

`components.admin-module`

Está dedicado a las vistas y funcionalidades para administradores. Funcionalidades principales:

- Administración de empleados: crear, editar, listar empleados.
- Gestión de reportes: ver detalles de reportes y listados generales.

Componentes destacados:

- `admin.component`
- `employee-create.component`
- `employee-edit.component`
- `employee-list.component`
- `report-detail.component`
- `reports.component`

`components.moderator-module`

Diseñado para moderadores, encargados de supervisar productos y usuarios. Funcionalidades principales:

- Aprobar o rechazar productos.
- Crear y gestionar sanciones a usuarios.

Componentes destacados:

- `moderator.component`
- `product-approval.component`
- `sanction-create.component`
- `sanctions-list.component`

`components.logistics-module`

Orientado a la gestión de órdenes y entregas. Funcionalidades principales:

- Seguimiento de órdenes en progreso.
- Gestión de entregas.

Componentes destacados:
- `logistics.component`
- `order-delivery.component`
- `orders-in-progress.component`

`components.user-module`

Contiene las vistas para usuarios comunes, enfocadas en la interacción con productos y pedidos.
Funcionalidades principales:
- Gestión de carrito de compras.
- Visualización y edición de productos propios.
- Seguimiento de pedidos y detalles de los mismos.
- Calificación y comentarios de productos.

Componentes destacados:
- `cart-item.component`
- `checkout.component`
- `common-user.component`
- `my-products.component`
- `order-detail.component`
- `order-list.component`
- `product-create.component`
- `product-detail.component`
- `product-edit.component`
- `product-list.component`
- `rating-comment.component`
-

Módulos adicionales

Login

`login.component`: permite iniciar sesión en la aplicación.

Register

`register.component`: permite crear una cuenta de usuario.

Notifications-module

Maneja las notificaciones dentro de la aplicación.
Componentes: `notification-detail.component` y `notifications-list.component`.

Servicios

Los servicios se encargan de la comunicación con el backend, la gestión de datos y lógica de negocio.
Principales servicios:

- `admin.service.ts`, `moderator.service.ts`, `user.service.ts`: gestión de usuarios.
- `product.service.ts`, `order.service.ts`, `cart.service.ts`, `payment.service.ts:` gestión de productos, pedidos y pagos.
- `notification.service.ts`, `rating.service.ts`, `sanctions.service.ts:` notificaciones, calificaciones y sanciones.
- `authentication.service.ts`, `register.service.ts:` manejo de autenticación y registro.

Guards e Interceptors

Auth.guard: protege rutas que requieren autenticación.
Ngrok.interceptor: modifica peticiones HTTP para pruebas o redirección temporal.

Modelos

Representan las estructuras de datos que maneja el frontend:

- `user.model.ts`: define la estructura de un usuario.
- `product.model.ts`: define un producto.
- `order.model.ts`: define un pedido.
- `cart.model.ts`: define un carrito de compras.
- `rating.model.ts`: define calificaciones y comentarios.
- `sanction.model.ts`: define sanciones aplicadas a usuario