Memoria Técnica del Proyecto de Paquetería

Proyecto iweb - Grupo 9

January 16, 2025

1. Introducción

El presente documento describe la implementación de un sistema de envío y gestión de paquetes (paquetería) desarrollado con una arquitectura basada en **frontend** (Vue 3 + Vuetify, usando también Pinia para la gestión de estado) y **backend** (Spring Boot con persistencia en base de datos relacional).

El objetivo principal de la aplicación es permitir que distintos perfiles de usuarios (Empresas/Clientes, Repartidores y el Webmaster/Administrador del sistema) gestionen pedidos (envíos) de forma fácil e intuitiva. El sistema provee funcionalidades como:

- Creación de envíos con cálculo de tarifas.
- Seguimiento del estado del envío.
- Asignación de repartidores por parte de un administrador.
- Módulos de autenticación y registros de usuarios, con validación vía JWT.

A continuación, se presenta de manera detallada la estructura, módulos y componentes principales del proyecto.

2. Descripción General de Funcionalidades

2.1. Funcionalidades por Perfil

1. Empresas (Clientes)

- *Creación de envíos*: Permite registrar un nuevo envío indicando origen, destino, número de bultos, peso y precio estimado.
- *Consulta de tarifas*: Se conecta a un endpoint que, dada la distancia, códigos postales y peso, devuelve un costo estimado.
- *Seguimiento del envío*: A través de un número de seguimiento, se puede consultar el estado actual (PENDIENTE, EN_RUTA, ENTREGADO, etc.).

2. Repartidores

- *Gestión de rutas*: Visualización de envíos asignados para la fecha actual y el día siguiente, con detalles de dirección de destino, contenido de bultos/peso y observaciones.
- *Visualización de tareas pendientes*: Al iniciar sesión, pueden ver directamente todos los envíos en curso.

3. Webmaster (Administrador del sistema)

- *Gestión de envíos*: Visualizar todos los envíos, asignar envíos pendientes a repartidores, o modificar asignaciones.
- *Control y supervisión*: Generar reportes de envíos (pendientes, entregados, rechazados, ausentes, etc.).

2.2. Endpoints Principales

La aplicación backend, bajo la ruta base `/api/v1/envios`, ofrece varios métodos REST:

- **GET /envios**: Obtiene todos los pedidos (accesible con rol adecuado).
- **POST /envios**: Añade un pedido nuevo al sistema.
- **GET /envios/{seguimiento}**: Obtiene el detalle de un pedido por su número de seguimiento.
- **GET /envios/tarifas**: Calcula el precio estimado de un envío (según peso, volumen, etc.).
- **GET /envios/owned**: Lista todos los pedidos que pertenecen al usuario autenticado.
- **GET /envios/estado/{seguimiento}**: Devuelve el estado actual del envío.

La autenticación se basa en JWT; al iniciar sesión (`/auth/login`), el servidor devuelve un token que se envía en la cabecera `Authorization` con cada petición.

3. Arquitectura de la Solución

La solución sigue una arquitectura de **tres capas** (presentación, lógica de negocio y acceso a datos) y está dividida en dos grandes proyectos:

3.1. Frontend (Carpeta `iweb-frontend`)

- **Framework principal**: Vue 3 con Vuetify 3.
- **Router automático**: unplugin-vue-router.
- **Gestión de estado**: Pinia.
- **Estructura de carpetas**:

  - `api`: Configuración de Axios (interceptores, token JWT).
  - `assets`: Imágenes (logo, etc.) y ficheros auxiliares.
  - `components`: Componentes Vue (formularios, tablas, listados).
  - `layouts`: Layouts globales (menú, header, footer).
  - `pages`: Vistas principales mapeadas a rutas.
  - `stores`: Almacenes Pinia (auth, pedidos, etc.).
  - `plugins`: Configuraciones (Vuetify, router, etc.).

3.2. Backend (Carpeta `iweb-backend`)

- **Framework principal**: Spring Boot.
- **Módulos**:
  - `controllers`: Exponen endpoints REST (ej: `PedidoController`, `UserController`).
  - `services`: Lógica de negocio (`PedidoService`, `PagosService`, etc.).
  - `entities`: Entidades JPA (UserEntity, PedidoEntity, etc.).
  - `repository`: Extienden de `JpaRepository` para operaciones CRUD.
  - `jwt`: Manejo del token (`JwtAuthenticationFilter`, `JwtService`).
  - `config`: Configuraciones de seguridad (`SecurityConfiguration`, etc.).
- **Persistencia**: Entidades con anotaciones `@Entity`, repositorios con JPA/Hibernate.
- **Autenticación y seguridad**: Spring Security + JWT + filtro que comprueba tokens y roles.

4. Detalle de Componentes (Frontend)

- **`EnviosList.vue`**: Lista de envíos, consume `axios.get('/envios')`, con botón para ver detalles.
- **`DetallesEnvio.vue`**: Información detallada de un envío (origen, destino, bultos, estado de pago).
- **`NuevoPedido.vue`**: Formulario con pasos (ORIGEN, DESTINO, BULTOS). Llama a `POST /envios`.
- **`LoginForm.vue y RegistroForm.vue`**: Formularios de autenticación y registro. Usan la store `useAuthStore`.
- **Stores (Pinia)**:
  - `useAuthStore`: Maneja credenciales (*token*, *user*, login, registro, logout).
  - `useEnvioStore`: Creación y consulta de envíos.
  - `useApiKeyStore`: Generación de claves de API (JWT alternativo).

5. Detalle de Componentes (Backend)

- `PedidoController`: Endpoints para crear/listar pedidos (`/envios`), obtener detalles, etc.
- `PedidoService`: Lógica de creación de pedidos, cálculo de tarifa, generación de pago, número de seguimiento.
- `PagosService` y `PagosClient`:
  - Implementan la comunicación con la pasarela de pagos externa.
  - Crean la *url* de pasarela y manejan la confirmación.
- `TarifaService`:
  - Contiene la lógica para calcular el coste base, recargos por peso o mercancía peligrosa, etc.
- `RutaService` y `RutaController`:
  - Maneja la asignación de pedidos a rutas diarias de repartidores.
- **Autenticación con JWT**:
  - `JwtService`: Genera y valida tokens, extrae informaciones.
  - `SessionRepository`: Guarda sesiones activas para cada usuario.
  - `AuthService`: Registros, login, logout, mails de verificación y más.

6. Flujo Principal de Creación y Gestión de Envíos

1. **Autenticación**: El usuario se registra o inicia sesión (recibe JWT).
2. **Creación de un nuevo envío**: Envía datos a `POST /envios`, el backend calcula la tarifa y crea el pedido (estado PENDIENTE). Se crea un pago con estado PENDIENTE.
3. **Pago del pedido**: El usuario recibe *url* de pasarela. Tras confirmarse en la pasarela, el endpoint `/api/v1/pagos/confirm/{code}` cambia el estado a ACEPTADO.
4. **Asignación de Repartidor (Administrador)**: El admin ve los pedidos pendientes y asigna un repartidor vía `POST /envios/{seguimiento}/asignar?emailRepartidor=...`
5. **Seguimiento**: Con `GET /envios/{seguimiento}` el cliente ve el estado actual (PENDIENTE, EN_RUTA, etc.).

7. Base de Datos y Entidades Principales

- **`users`**: Guarda datos de usuarios (nif, nombre, email, *password* hasheado, roles, etc.).
- **`pedidos`**: Representa los envíos (origen, destino, estado, fecha de actualización, precio).
- **`bultos`**: Lista de bultos asociados a cada pedido (peso, dimensiones, mercancía peligrosa).
- **`pagos`**: Registro de pagos asociados a cada pedido (estado, URL de pasarela, código de confirmación).
- **`tarifas`**: Tabla con parámetros base, recargos y precio mínimo.

8. Conclusiones

Este sistema de paquetería ofrece:

- **Arquitectura modular**: Separación clara entre frontend y backend.
- **Escalabilidad**: Uso de servicios externos (pasarela de pagos) e integración sencilla con más módulos.
- **Seguridad**: Autenticación y autorización con JWT, roles y filtros de acceso.
- **Flexibilidad**: Cálculo de tarifas configurable y gestión de envíos adaptable.

Con ello se dispone de un **sistema de gestión de envíos** robusto y extensible, cubriendo los principales casos de negocio en logística de paquetería.