**MANUAL TÉCNICO**

**LA ENCOMIENDA**

**TABLA DE CONTENIDO**

[**RESUMEN EJECUTIVO 3**](#_jez7plkgun64)

[**ESTRUCTURACIÓN API 3**](#_93uaeg5n6940)

[Decisional 3](#_7wiesbqyzbjd)

[Gastos 3](#_1m6shten2h9n)

[Paquete 3](#_bd19zz1dxzf3)

[Sucursal 3](#_9d8dfcha0j2e)

[Usuario 4](#_dcocbzkltr3b)

[Predictivo 4](#_i2507et640c8)

[Router 4](#_la3qdqx94bza)

[Controller 4](#_r59odnezmtl8)

[Service 4](#_at1vwy6h57fy)

[Repository 4](#_ade87tdvw1ss)

[**MODELO DE DATOS 5**](#_oi5m6pxadqnr)

[Diagrama ER 5](#_q8w8ztkkscs8)

[Flujograma 6](#_okq120wbx7m)

[**ARQUITECTURA DE LA SOLUCIÓN 8**](#_cpmcm165dnh5)

[**SECCIONES CLAVE 9**](#_raulg485qhf6)

[Autenticación JWT 9](#_d27uqmch0t6y)

[Modelado Prisma 10](#_oqbq8uwsjj3t)

[Paquete Router 11](#_labtw3gshrpm)

[Paquete Controller 11](#_mjq1cy9wzzhc)

[Paquete Service 12](#_83vv27leknau)

[Paquete Repository 12](#_f1435puehbry)

[**GLOSARIO 13**](#_dte0egqfbciw)

# RESUMEN EJECUTIVO

El sistema propuesto para la organización "La Encomienda" busca optimizar y centralizar sus operaciones de paquetería y administrativas, facilitando la toma de decisiones. Con un enfoque en el comercio justo, el sistema gestionará tarifas basadas en la cantidad de envíos, incentivando una mayor participación. Además, se abordará la gestión de empleados y gastos de sucursales, brindando transparencia a través de un módulo accesible a todos los empleados. Con funcionalidades como seguimiento de envíos, gestión de honorarios y opciones de gastos especiales, el sistema proporcionará un dashboard integral con metas mensuales, reportes detallados y un módulo específico para la toma de decisiones estratégicas, como la apertura de nuevas sucursales o la adquisición de vehículos adicionales. Este enfoque integral permitirá a la organización "La Encomienda" no solo mejorar sus operaciones actuales, sino también planificar eficientemente su crecimiento futuro.

# ESTRUCTURACIÓN API

La estructuración de la API se basa en un enfoque modular y organizado, utilizando Postgres como base de datos, FastAPI con Python como framework de desarrollo y Prisma como ORM. La autenticación de usuarios se realiza mediante tokens JWT. La API se ha dividido en los siguientes módulos

## Decisional

Contiene los endpoints (EP) que alimentan los dashboards en el frontend.

Incluye funciones para realizar pruebas en los módulos, ingresando información ficticia y observando cómo afecta a los ingresos o gastos en el sistema.

## 

## Gastos

Contiene los endpoints relacionados con gastos, concepto de gasto y tipo de gasto.

## 

## Paquete

Incluye los endpoints para manejar paquetes, estado del paquete, estado del seguimiento y tracking.

## 

## Sucursal

Contiene todos los EP relacionados con la gestión de sucursales, como ciudad, salidas programadas, ingresos económicos, segmentos de ruta, tarifas, tipos de salida, tipos de sucursal, vehículos y tipos de vehículos.

## Usuario

Contiene los EP para gestionar usuarios, roles y puestos.

## Predictivo

Contiene los endpoints relacionados con el análisis predictivo y la toma de decisiones estratégicas.

Incluye funciones para prever la necesidad de abrir nuevas sucursales, la cantidad de personal requerido y la adquisición de nuevos vehículos o rutas.

Utiliza datos acumulados a lo largo de los dos años de funciones para ofrecer pronósticos precisos y apoyar la planificación a largo plazo de la organización.

Este módulo es esencial para la toma de decisiones basada en datos históricos y tendencias, permitiendo a la organización anticipar y abordar proactivamente los desafíos y oportunidades en su operación.

La arquitectura de la API sigue un patrón de diseño con las siguientes capas:

## Router

Define las rutas de la API y conecta las solicitudes HTTP con los controladores correspondientes.

## Controller

Gestiona la entrada y salida de datos, dirigiendo las solicitudes del router a las funciones del servicio.

## Service

Contiene la lógica de negocio de la aplicación y se comunica con el repositorio para acceder a la base de datos.

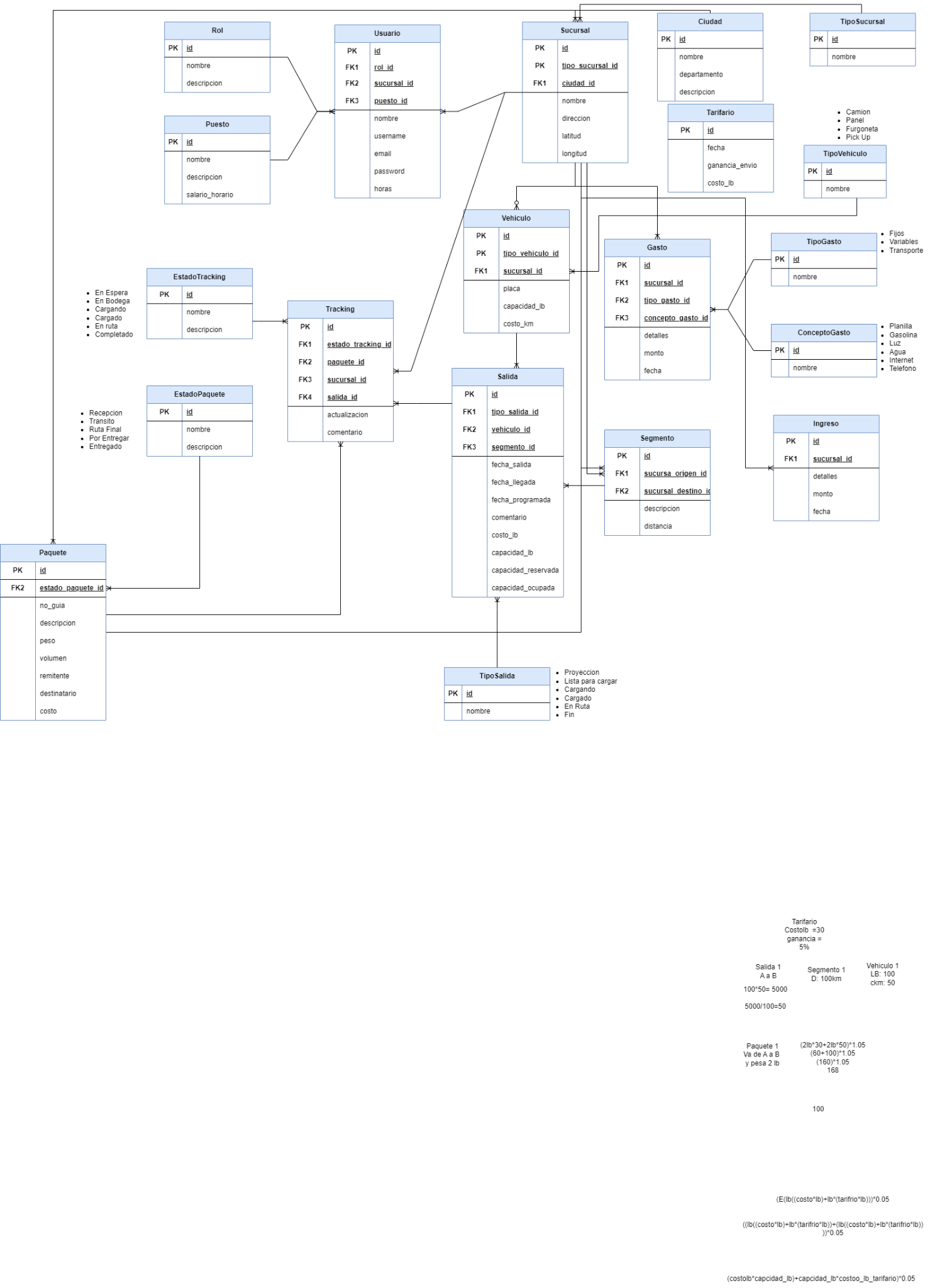
## Repository

Encargado de realizar las operaciones directas en la base de datos, encapsulando el acceso y manipulación de los datos.

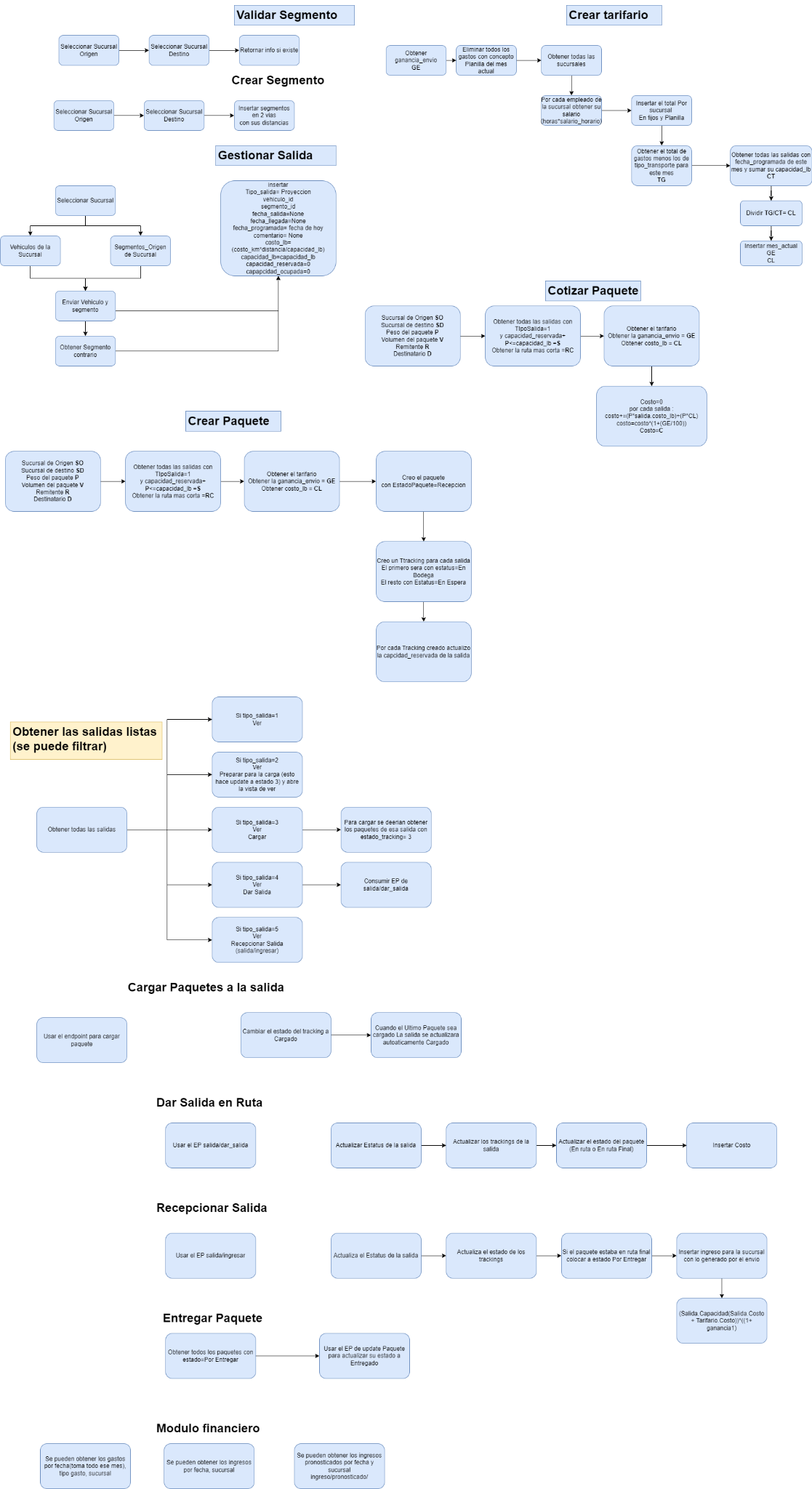
Esta estructura modular facilita el mantenimiento y escalabilidad del sistema, separando claramente las responsabilidades de cada componente y siguiendo las mejores prácticas de desarrollo de software.

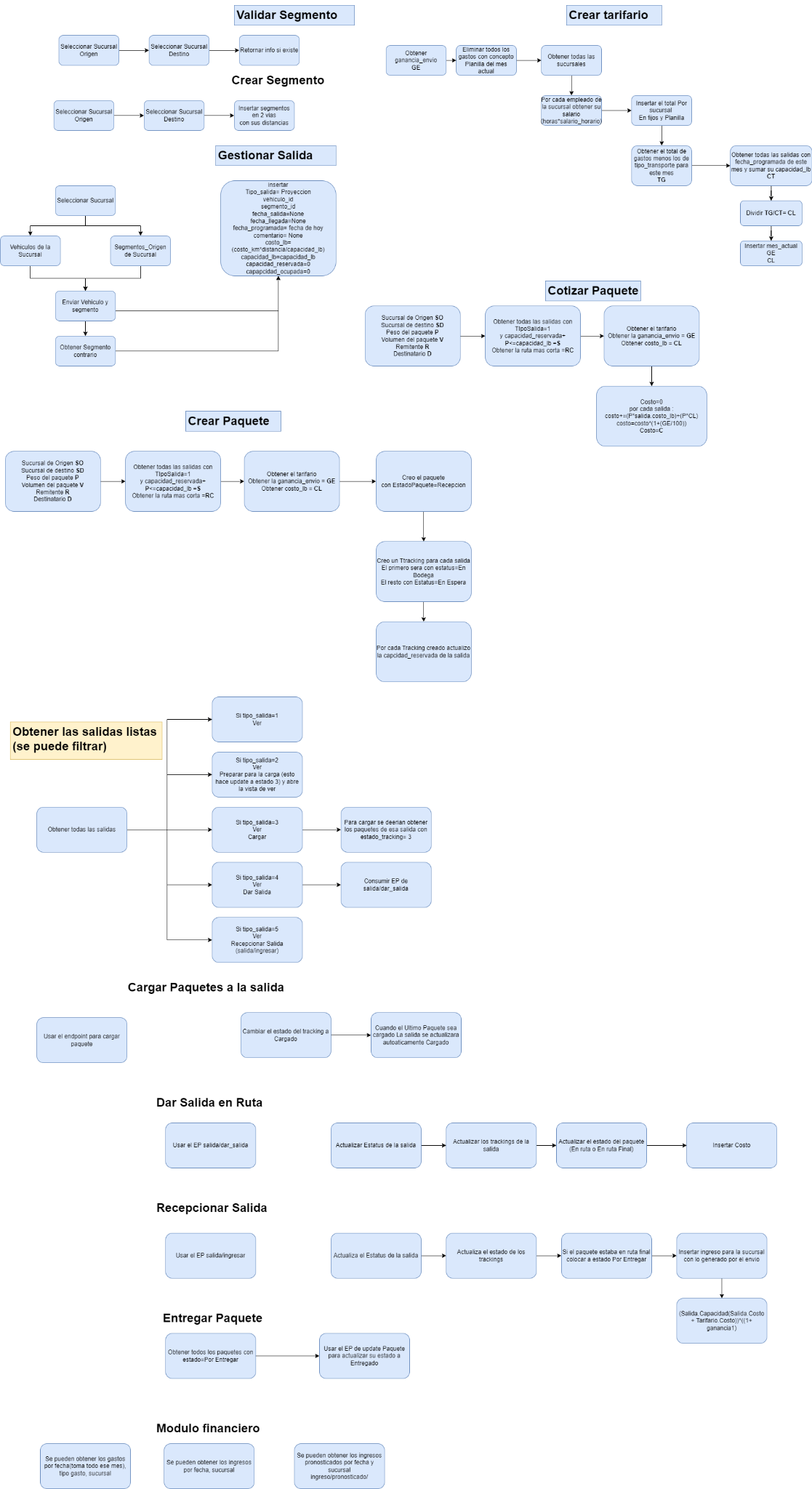
# MODELO DE DATOS

## Diagrama ER



## Flujograma





# 

# ARQUITECTURA DE LA SOLUCIÓN

# 

# 

# 

# 

# 

# 

# 

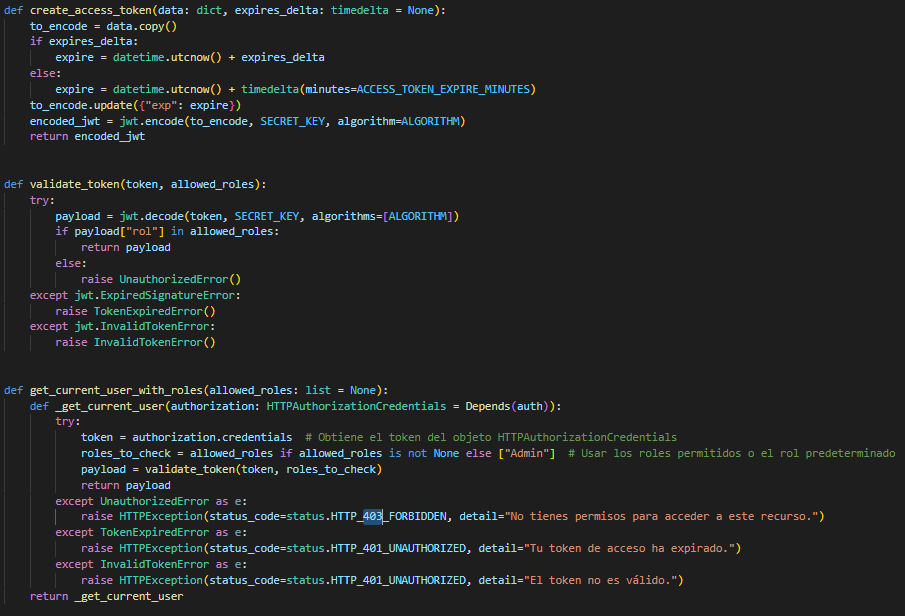
# 

# 

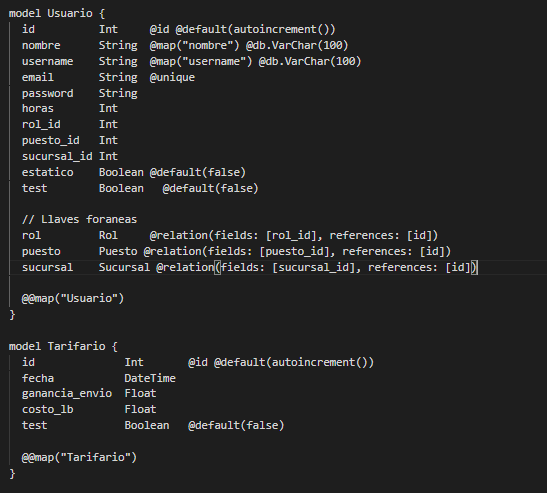
# 

# SECCIONES CLAVE

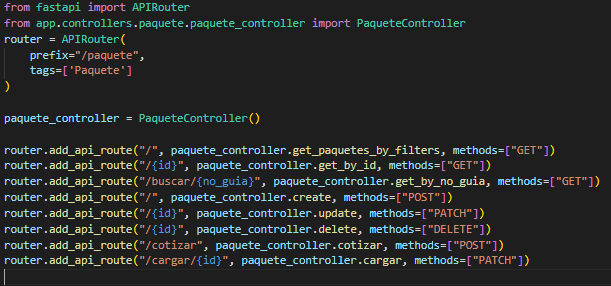
## Autenticación JWT



## Modelado Prisma



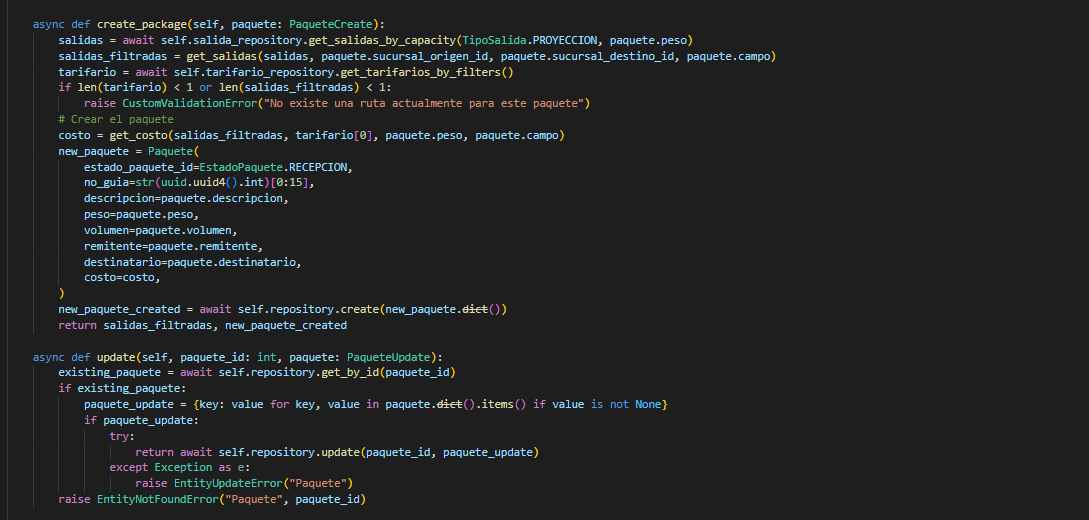
## Paquete Router



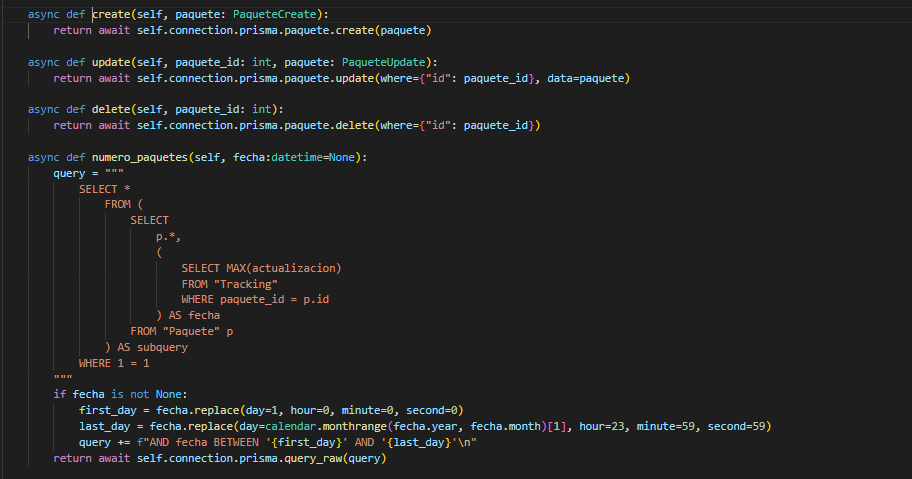
## Paquete Controller



## Paquete Service



## Paquete Repository



# GLOSARIO

**EncomiendaPostgres:** Nombre de la base de datos que almacena la información fundamental para la aplicación de paquetería.

ORM Prisma:

**EncomiendaPrisma:** Mapeador objeto-relacional que facilita la interacción entre la aplicación y la base de datos PostgreSQL.

API FastAPI:

**EncomiendaAP:** Interfaz de programación de aplicaciones construida con FastAPI que proporciona endpoints para la comunicación entre el frontend y el backend.

Autenticación JWT:

**EncomiendaAuth:** Módulo encargado de la autenticación de usuarios mediante tokens JWT para garantizar la seguridad en las interacciones.

FrontEnd Angular:

**EncomiendaFront:** Interfaz de usuario construida con Angular, que permite a los usuarios interactuar con la aplicación de manera visual.

**Capa de Datos:**

Capa responsable de la gestión y persistencia de los datos en la base de datos.

**Capa de Servicios/API:**

Capa que contiene la lógica de negocio y proporciona una interfaz de programación para la interacción entre el frontend y el backend.

**Capa de Interfaz de Usuario (UI):**

Capa donde interactúa directamente el usuario final, encargada de presentar la información visualmente y recibir las interacciones del usuario.

**Router, Controller, Service, Repository:**

Elementos de la arquitectura de la aplicación que siguen el patrón de diseño MVC (Modelo-Vista-Controlador), donde el Router maneja las rutas, el Controller gestiona la entrada y salida, el Service contiene la lógica de negocio, y el Repository maneja el acceso a la base de datos.

**Módulo Predictivo:**

Módulo de la aplicación que utiliza datos históricos para realizar análisis predictivos y apoyar la toma de decisiones estratégicas.