Obligatorio 1

Desarrollo Full Stack

Módulo backend

Octubre 2025

Santiago Neira Estudiante 334109
Cesar Martinez Estudiante 330903

Fecha de entrega 9/10/2025

Descripción de la aplicación

API Cadetería sirve para gestionar envíos de paquetes de punta a punta: permite registrar clientes, crear pedidos con datos de origen y destino, consultar y filtrar envíos, seguir su estado (pendiente, en ruta, entregado, cancelado), reprogramar o actualizar información según permisos, y administrar categorías y usuarios. Está pensada para que empresas o particulares organicen y controlen sus envíos de forma simple y ordenada, con niveles de servicio diferenciados y un historial claro de cada pedido.

Entidades implementadas

[Envio, User, Category]

Detalle de las entidades:

```
User: {
  username: "string",
  email: "string",
  password: "string",
  nombre: "string",
  apellido: "string",
  role: "admin | cliente",
  plan: "plus | premium"
}

Envio: {
  user: "ObjectId (ref: Usuario)",
  origen: "Direccion",
  destino: "Direccion",
  fechaRetiro: "date (>= hoy)",
  horaRetiroAprox: "string (HH:mm)",
  tamanoPaquete: "chico | mediano | grande",
  notas: "string (max 500)",
  category: "ObjectId (ref: Categoria)",
  estado: "pendiente | en_ruta | entregado | cancelado",
  codigoSeguimiento: "string (único)"
}

Category: {
  name: "string"
}
```

URL productiva:

Despliegue Cesar: *apicadeteria-g62jum3rq-cesars-projects-2539e6a6.vercel.app*

Despliegue Santiago: *https://apicadeteria-pmjgfv8nm-sanei1509s-projects.vercel.app*

Documentación de los endpoints

Cesar: *https://apicadeteria-g62jum3rq-cesars-projects-2539e6a6.vercel.app/docs*

Santiago: *https://apicadeteria-pmjgfv8nm-sanei1509s-projects.vercel.app/docs*

Reglas de negocio implementadas

Colección de Postman:

Usuarios de prueba

Uso de IAG

Uso en general

La IA generativa se utilizó como **asistente técnico,** Interpretación de requisitos, modelado, solución de errores no controlados en las respuestas de los requests, refactorización de código en búsqueda de utilizar buenas prácticas y el orden/prolijidad esperado en la materia.

Apoyo en la creación de endpoints (públicos/privados), definir reglas de negocio (plan plus) y esquemas de datos.

Nos guío en la estructura de carpetas y el orden correcto de llamadas de middlewares (Swagger antes de rutas en Vercel, CORS y express.json()).

Diseño de validaciones

Guía en la construcción de los schemas Joi con mensajes específicos (fecha ≥ hoy, formato HH:mm, límites de longitud, Robustez en búsqueda de categorías, etc.

Creación de los DTOs para controlar la seguridad y la consistencia en las respuestas de nuestra api.

Debugging guiado por logs

Agrego logs en los lugares convenientes (ruta del controller, ruta del schema, body crudo y normalizado) que permitieron ubicar algunos problemas, chequear que los outputs estuvieran correctos según lo esperados y entender otros errores desconocidos por nosotros como el cors.

Casos de prueba reproducibles

Proporcionó un set de cURL positivos/negativos par debug y un seed script opcional para poblar categorías, acelerando la verificación funcional.

Claridad documental (Swagger), apoyo con la redacción de los endpoints, parámetros y debug para el acceso a la documentación desde link del despliegue.

Extras

Documentación*:* creamos la documentación en swagger y adicionalmente en postman.

Health check (GET /): endpoint simple para verificar disponibilidad y monitoreo sin tocar rutas de negocio.

Reglas adicionales por rol*:* el cliente no puede cambiar estado, sólo reprograma con ≥1 día de anticipación y no puede eliminar en fecha del retiro.

Filtros avanzados en listados: por estado, tamaño, fecha puntual, rangos y “última semana/mes” para consultas más potentes.