Entrega Inicial - TPI - Backend de Aplicaciones

Proyecto: Logística de traslado de contenedores
Stack Base: Java 17 + Spring Boot 3, Spring Cloud Gateway, OpenAPI/Swagger, Keycloak (OIDC), PostgreSQL 16, Docker Compose, Mapas/Distancias vía Google Maps Directions API.

Integrantes:

- Gigena, Melina 400089
- Martin Valentino, Camilo 400312
- Zapata, Catalina 400247

Diseño de la Arquitectura del Sistema de Logística

Para dar solución a los requerimientos del sistema de logística de contenedores, se implementará una arquitectura de software moderna basada en **microservicios**. Este enfoque nos permite desacoplar las responsabilidades del sistema en servicios independientes, lo que mejora la escalabilidad, el mantenimiento y la resiliencia del backend. La arquitectura se fundamenta en dos microservicios principales: ms-recursos y ms-logistica, orquestados por un API Gateway y securizados con Keycloak.

La separación de responsabilidades es clave en nuestro diseño. Para ello, definimos los siguientes microservicios:

Microservicio de Recursos

Este servicio actúa como el **gestor de los activos y entidades maestras** de la empresa. Su principal responsabilidad es manejar la información que, si bien puede cambiar, es la base sobre la que opera la logística. Gestiona las operaciones de alta, baja, modificación y consulta (CRUD) de:

- **Camiones**: La flota de camiones, incluyendo sus capacidades, costos y datos del transportista.
- **Depósitos**: Los puntos de almacenamiento intermedio, con su ubicación y costos de estadía.
- **Clientes**: La información de contacto y datos personales de los clientes.
- **Tarifas**: Los parámetros y rangos de precios que definen la base para los cálculos de costos.

Microservicio de Logística

Este es el **núcleo operacional del sistema**. Se encarga de toda la lógica de negocio relacionada con el proceso de transporte de un contenedor desde su inicio hasta su fin. Sus responsabilidades incluyen:

- **Gestión de Solicitudes**: Crear y administrar los pedidos de traslado de los clientes.
- **Cálculo de Rutas y Tramos**: Determinar las rutas tentativas y gestionar los tramos que las componen.
- **Estados**: Actualizar y consultar el estado de un contenedor a lo largo de su viaje.
- **Cálculo de Costos**: Orquestar el cálculo de los costos estimados y finales, interactuando con ms-recursos para obtener datos de tarifas, camiones y depósitos.

Tecnologías y Componentes de Soporte

Para que los microservicios funcionen de manera coordinada y segura, nos apoyaremos en un conjunto de herramientas y tecnologías estándar en la industria.

- **API Gateway**: Actuará como la **única puerta de entrada** para todas las peticiones de los clientes (Cliente, Operador, Transportista). Se encargará de enrutar cada petición al microservicio correspondiente (ms-recursos o ms-logistica), simplificando la comunicación y centralizando la aplicación de políticas de seguridad.
- **Keycloak (Seguridad)**: Será nuestro **proveedor de identidad centralizado**. Gestionará la autenticación de los usuarios y la asignación de roles. Cada petición que llegue al API Gateway deberá incluir un **token JWT** emitido por Keycloak, el cual será validado para asegurar que el usuario tiene los permisos necesarios para acceder al recurso solicitado.
- **API Externa (Google Maps Directions)**: Para cumplir con los requisitos de cálculo de distancia, el microservicio ms-logistica se integrará con la API de Google Maps. Esta herramienta externa nos proveerá la distancia precisa entre las coordenadas de origen, destino y los depósitos, dato fundamental para estimar tiempos y costos.

- **Base de Datos (PostgreSQL en Supabase)**: Toda la información del sistema persistirá en una base de datos relacional con tecnología **PostgreSQL**. Utilizaremos la plataforma **Supabase.com** como proveedor de servicios de base de datos. Ambos microservicios se conectarán a esta instancia para gestionar sus datos, asegurando la consistencia e integridad de la información a través de un esquema bien definido.
- **Framework (Java y Spring Boot)**: El desarrollo de ambos microservicios se realizará en **Java**, utilizando el ecosistema de **Spring Boot**. Este framework nos facilita la creación de APIs REST robustas, la integración con la base de datos, la implementación de la seguridad y la construcción de aplicaciones listas para ser contenedorizadas.

DIAGRAMA ENTIDAD - RELACIÓN

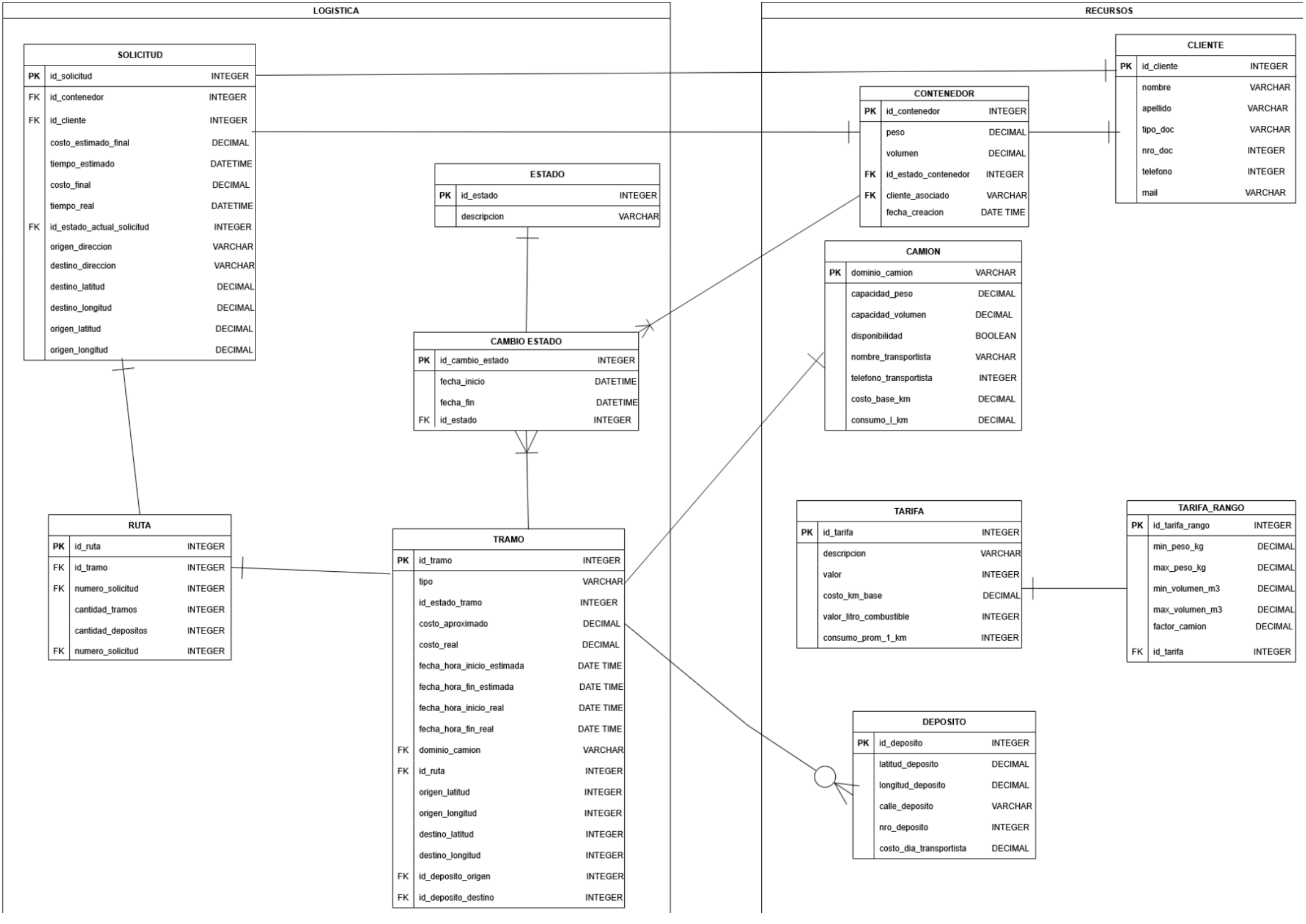

DIAGRAMA DE CONTENEDORES

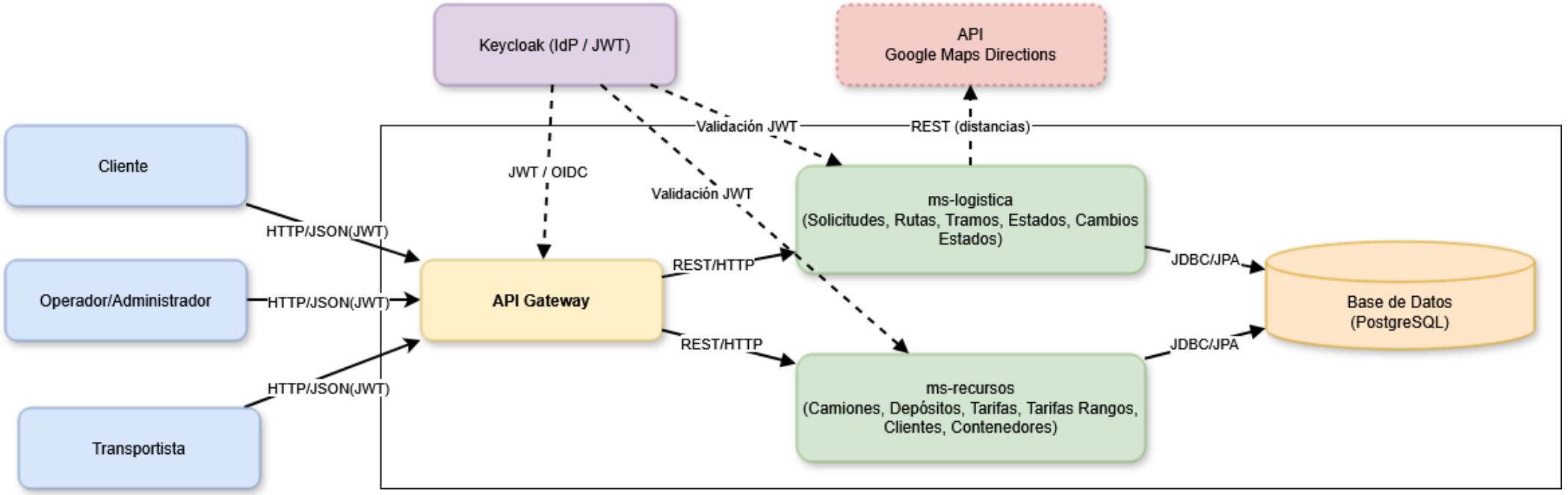

Diagrama de Contenedores – C4 Model (Nivel 2)

El siguiente diagrama representa la arquitectura del sistema a nivel de contenedores, siguiendo el C4 Model propuesto por Simon Brown (Nivel 2).
En este nivel se identifican los principales contenedores de software que conforman el sistema, las personas y sistemas externos que interactúan con él, y las relaciones entre estos elementos.

El sistema de **Logística de Contenedores** se compone de dos microservicios principales:

- **ms-logística**: gestiona las Solicitudes, Rutas, Tramos y el seguimiento de las entregas a través de los Estados y Cambios de Estados. Además, consulta la API externa de Google Maps Directions para obtener distancias y tiempos estimados de recorrido.

- **ms-recursos**: administra los recursos del sistema, incluyendo Camiones, Depósitos, Tarifas, Clientes y Contenedores.

Ambos microservicios exponen sus funcionalidades mediante **API REST** y acceden a su respectiva base de datos **PostgreSQL** a través de **JPA (Java Persistence API)**, que internamente utiliza **JDBC** para la comunicación con la base de datos.

El acceso al sistema por parte de los distintos actores (**Cliente**, **Operador/Administrador** y **Transportista**) se realiza a través de un **API Gateway**, que actúa como **punto de entrada único**, gestionando el enrutamiento, la validación de tokens y la seguridad de las peticiones.

Para la autenticación y autorización se utiliza **Keycloak**, configurado como **IdP (Identity Provider)** bajo los protocolos **OIDC (OpenID Connect)** y **OAuth2**, emitiendo **tokens JWT (JSON Web Tokens)** que son validados por el Gateway y los microservicios internos.

Asimismo, el sistema integra un servicio externo —**Google Maps Directions API**— utilizado para calcular las distancias y tiempos reales de las entregas, insumo necesario para los cálculos de costos y tiempos finales.

El diagrama muestra de forma clara las interacciones entre los distintos contenedores:

- Las líneas continuas representan **comunicaciones de datos** (REST/HTTP, JDBC/JPA).

- Las líneas punteadas representan **procesos de autenticación** (validación de JWT u OIDC).

En conjunto, la arquitectura propuesta garantiza **bajo acoplamiento, alta cohesión** y una estructura **escalable y segura**, permitiendo el despliegue independiente de cada microservicio dentro de una infraestructura basada en **contenedores Docker**.

Diseño a nivel de contenedor:

- **api-gateway (Spring Cloud Gateway):** Expone /api/v1/**, enruta a los MS, centraliza CORS, rate-limit opcional.

- **ms-recursos (Spring Boot):** Cliente, Camion, Deposito, Tarifa, Tarifa Rango,Contenedor.
- **ms-logistica (Spring Boot):** Solicitud, Ruta, Tramo, Cambio Estado, Estado.

- **PostgreSQL(única DB):** Esquema recursos (tablas maestras) y esquema operaciones (proceso).

- **Keycloak:** Realm “logistica”; roles: cliente, transportista, operador/administrador

ENDPOINTS

Microservicio Logística
Solicitudes

Método	URI	Rol	Posible Respuesta (Resultado)
POST	/api/v1/solicitudes	Cliente	**201 Created**: { "id_solicitud": 101, "estado": "borrador", ... } (Header: Location: /solicitudes/101) **422 Unprocessable Entity**: { "error": "Datos inválidos", "detalles": ["El campo 'origen_latitud' es obligatorio"] }. **Requerimiento Funcional 1**
GET	/api/v1/solicitudes	Operador	**200 OK**: [{ "id_solicitud": 101, "estado": "borrador", ... }, { "id_solicitud": 102, "estado": "en_transito", ... }]
GET	/api/v1/solicitudes/{id}	Cliente / Operador	**200 OK**: { "id_solicitud": 101, "estado": "borrador", "cliente": {...}, "tramos": [...] } **404 Not Found**: { "error": "Solicitud no encontrada" }
PATCH	/api/v1/solicitudes/{id}	Operador/Administrador	**200 OK**: { "id_solicitud": 101, "estado": "programada", ... } **404 Not Found**: { "error": "Solicitud no encontrada" }
POST	/api/v1/solicitudes/{id}/ruta	Operador / Administrador	**Body de la Petición (Ejemplo):** json { "tramos": [{ "secuencia": 1, "tipo": "origen-deposito", "id_deposito_destino": 2 }, { "secuencia": 2, "tipo": "deposito-destino", "id_deposito_origen": 2 }] } **201 Created**: json { "id_ruta": 7, "id_solicitud": 101, "cantidad_tramos": 2, "tramos": [{ "id_tramo": 15, "secuencia": 1, "estado": "estimado", ... }, { "id_tramo": 16, "secuencia": 2, "estado": "estimado", ... }] } **404 Not Found**: { "error": "Solicitud no encontrada" } **409 Conflict**: { "error": "La solicitud ya tiene una ruta asignada" }

			Requerimiento Funcional 4

Rutas

Método	**URI**	**Rol**	**Posible Respuesta (Resultado)**
GET	/api/v1/rutas/tentativas	Operador / Administrador	**200 OK**: json [{ "nombre_ruta": "Ruta Recomendada (vía 1 depósito)", "costo_estimado_total": 950000.50, "tiempo_estimado_total_horas": 22.5, "distancia_total_km": 850, "tramos_sugeridos": [{ "secuencia": 1, "origen": "Origen Cliente", "destino": "Depósito Rosario", "distancia_km": 400, "tiempo_estimado_horas": 9 }, { "secuencia": 2, "origen": "Depósito Rosario", "destino": "Destino Cliente", "distancia_km": 450, "tiempo_estimado_horas": 10.5 }] }] **400 Bad Request**: { "error": "Parámetros de consulta insuficientes. Se requieren origen, destino y detalles del contenedor." } **Requerimiento Funcional 3**

Tramos

Método	**URI**	**Rol**	**Posible Respuesta (Resultado)**
PUT	/api/v1/tramos/{id}/asignar-camion	Operador/Administrador	**200 OK**: Devuelve el tramo actualizado para confirmar que la acción se completó exitosamente. json { "id_tramo": 15, "estado": "asignado", "dominio_camion": "AD123BC", "costo_aproximado": 450000.00, ... } **404 Not Found**: { "error": "Tramo o Camión no

			encontrado" } **409 Conflict**: { "error": "El camión seleccionado no está disponible" } **422 Unprocessable Entity**: { "error": "El camión no cumple con la capacidad de peso/volumen requerida" } **Requerimiento Funcional 6**
PATCH	/api/v1/tramos/{id}/iniciar	Transportista	**200 OK**: { "id_tramo": 5, "estado": "iniciado", "fecha_hora_inicio_real": "2025-10-20T09:00:00Z", ... } **Requerimiento Funcional 7**
PATCH	/api/v1/tramos/{id}/finalizar	Transportista	**200 OK**: { "id_tramo": 5, "estado": "finalizado", "fecha_hora_fin_real": "2025-10-20T18:30:00Z", ... } **Requerimiento Funcional 7**

Microservicio Recursos
Contenedores

Método	**URI**	**Rol**	**Posible Respuesta (Resultado)**
GET	/api/v1/contenedores	Operador / Administrador	**200 OK**: Devuelve una lista de contenedores que coinciden con los filtros. Si no se aplican filtros, devuelve todos. json [{ "id_contenedor": 55, "estado": "en_deposito", "ubicacion_actual": "Depósito Rosario", "id_solicitud_asociada": 101 }, { "id_contenedor": 56, "estado": "en_transito", "ubicacion_actual": "En viaje hacia Destino Final (Camión AE456FG)", "id_solicitud_asociada": 102 }] **400 Bad Request**: { "error": "El filtro 'estado' no es válido" }

GET	/api/v1/contenedores/{id}	Operador	**200 OK**: { "id_contenedor": 55, "estado": "en_deposito", "peso": 4500, "volumen": 33, ... } **404 Not Found**: { "error": "Contenedor no encontrado" }
GET	/api/v1/contenedores/{id}/estado	Cliente/ Operador	**200 OK:** json { "id_contenedor": 55, "estado": "en_transito", "descripcion": "En viaje desde Depósito Rosario hacia Destino Final.","fecha_actualizacion": "2025-10-17T11:30:00Z" } **404 Not Found**: { "error": "Contenedor no encontrado" } **403 Forbidden**: { "error": "Acceso denegado" } **Requerimiento Funcional 2**
GET	/api/v1/contenedores?estado=pendiente	Operador / Administrador	**200 OK**: Devuelve una lista de todos los contenedores cuyo estado **no es 'entregado' ni 'cancelado'**. El backend interpreta "pendiente" como un alias para múltiples estados (programada, en_transito, en_deposito, etc.). json [{ "id_contenedor": 55, "estado": "en_deposito", "ubicacion_actual": "Depósito Rosario", "id_solicitud_asociada": 101 }, { "id_contenedor": 56, "estado": "en_transito", "ubicacion_actual": "En viaje hacia Destino Final (Camión AE456FG)", "id_solicitud_asociada": 102 }, { "id_contenedor": 58, "estado": "programada", "ubicacion_actual": "Origen Cliente (Córdoba)", "id_solicitud_asociada": 105 }] **400 Bad Request**: { "error": "El valor del filtro 'estado' no es válido" } **Requerimiento Funcional 5**

Camiones

Método	**URI**	**Rol**	**Posible Respuesta (Resultado)**
POST	/api/v1/camiones	Operador/ Administrador	**201 Created**: { "dominio": "AD123BC", "disponibilidad": "disponible", ... } (Header: Location: /camiones/AD123BC) **Requerimiento Funcional 10**
GET	/api/v1/camiones	Operador/ Administrador	**200 OK**: [{ "dominio": "AD123BC", ... }, { "dominio": "AE456FG", ... }]
GET	/api/v1/camiones/{dominio}	Operador/ Administrador	**200 OK**: { "dominio": "AD123BC", "capacidad_peso": 25000, ... } **404 Not Found**: { "error": "Camión no encontrado" }
PUT	/api/v1/camiones/{dominio}	Operador/ Administrador	**200 OK**: { "dominio": "AD123BC", "nombre_transportista": "Nuevo Nombre", ... } **404 Not Found**: { "error": "Camión no encontrado" } **Requerimiento Funcional 10**
DELETE	/api/v1/camiones/{dominio}	Operador/ Administrador	**204 No Content**: (Sin cuerpo de respuesta) **404 Not Found**: { "error": "Camión no encontrado" }

Cliente

Método	URI	Rol	Posible Respuesta (Resultado)
POST	/api/v1/clientes	Operador / Cliente	**201 Created**: { "id_cliente": 12, "nombre": "Empresa SRL", ... } (Header: Location: /clientes/12) **422 Unprocessable Entity**: { "error": "El email ya se encuentra registrado" }
GET	/api/v1/clientes	Operador	**200 OK**: [{ "id_cliente": 12, "nombre": "Empresa SRL", ... }, { "id_cliente": 13, "nombre": "Constructora XYZ", ... }]
GET	/api/v1/clientes/{id}	Operador / Cliente	**200 OK**: { "id_cliente": 12, "nombre": "Empresa SRL", "telefono": "351-1234567", ... } **404 Not Found**: { "error": "Cliente no encontrado" }
PUT	/api/v1/clientes/{id}	Operador / Cliente	**200 OK**: { "id_cliente": 12, "nombre": "Empresa S.A.", "telefono": "351-7654321", ... } **404 Not Found**: { "error": "Cliente no encontrado" }

Depósitos

Método	URI	Rol	Posible Respuesta (Resultado)
POST	/api/v1/depositos	Operador/ Administrador	**201 Created**: { "id_deposito": 3, "nombre": "Depósito Rosario", ... } (Header: Location: /depositos/3) **Requerimiento Funcional 10**
GET	/api/v1/depositos	Operador/ Administrador	**200 OK**: [{ "id_deposito": 1, ... }, { "id_deposito": 2, ... }]

PUT	/api/v1/depositos/{id}	Operador/ Administrador	**200 OK**: { "id_deposito": 3, "nombre": "Depósito Rosario Central", "costo_dia_estadia": 15000, ... } **404 Not Found**: { "error": "Depósito no encontrado" } **Requerimiento Funcional 10**
DELETE	/api/v1/depositos/{id}	Operador/ Administrador	**204 No Content**: (Sin cuerpo de respuesta) **404 Not Found**: { "error": "Depósito no encontrado" }

Tarifas

Método	**URI**	**Rol**	**Posible Respuesta (Resultado)**
POST	/api/v1/tarifas	Operador/ Administrador	**201 Created**: { "id_tarifa": 7, "nombre": "Tarifa base", ... } (Header: Location: /tarifas/3) **Requerimiento Funcional 10**
GET	/api/v1/tarifas	Operador/ Administrador	**200 OK**: { "id_tarifa": 1, "descripcion": "...", ... }, { "id_tarifa": 2, ... }
PUT	/api/v1/tarifas/{id}	Operador/ Administrador	**200 OK**: { "id_tarifa": 7, "descripcion": "Tarifa actualizada", "rangos": [...] } **404 Not Found**: { "error": "Depósito no encontrado" } **Requerimiento Funcional 10**
DELETE	/api/v1/tarifas/{id}	Operador/ Administrador	**204 No Content**: (Sin cuerpo de respuesta) **404 Not Found**: { "error": "Tarifa no encontrado" }

VIDEO

Link del Video: https://youtu.be/WUga4jw0QZ4