TRABAJO PRÁCTICO INTEGRADOR

UTN
Córdoba

Tema: Spring Boot, Spring Security, Docker, Bases de Datos PostgreSQL - Primera entrega

Materia: Back End de Aplicaciones

Curso: 3K1

Carrera que cursa el alumno: Ingeniería en Sistemas de Información

Grupo: 14

Integrantes:

- Bruno Gerardo Giraudo — Legajo: 400236
- Luis Antonio Muñoz Segura — Legajo: 55764
- Sergio Gabriel Garzón — Legajo: 54330

Profesores:

- German Ariel Romani (Teórico)
- Gaspar Arrigo (Práctico)
- Rodrigo Tomás Díaz Mac William (Práctico)

Fecha de entrega: 22 de Octubre del 2025

Índice:

Trabajo Práctico Integrador

Backend de Aplicaciones – 2025

📌 Introducción

La presente propuesta corresponde al Trabajo Práctico Integrador (TPI) de la asignatura Backend de Aplicaciones, correspondiente al año 2025. El objetivo del trabajo es implementar una solución backend basada en microservicios que permita gestionar un sistema de logística de transporte terrestre de contenedores a ser utilizados en la construcción de viviendas, es decir el objeto de transporte es el propio contenedor y no su contenido.

Se espera una solución que se apoye en los contenidos de la asignatura, implemente buenas prácticas de diseño, seguridad, arquitectura y documentación.

Además, se evaluará el proceso de decisiones tomadas para la solución a los desafíos que se presenten durante la construcción del trabajo, esto se deja expresado para que conste que los estudiantes pueden utilizar todas las herramientas que tengan disponibles, tanto para la definición de diseño, como para la construcción de la solución pero, serán evaluados por su defensa de las decisiones e implementación realizada.

🎯 Alcance y modelo funcional

El sistema a desarrollar simula el backend de una empresa transportista que:

- Recibe solicitudes de traslado de contenedores desde un punto de origen a un terreno o ubicación de destino. De estas ubicaciones se conoce la dirección textual y siempre y en todos los casos la geolocalización en términos de latitud y longitud, para un contenedor con un cliente asociado al que se debe poder contactar.
- Los contenedores tienen dimensiones y peso que se vuelven restricciones para los camiones que los pueden trasladar.
- También gestiona los depósitos, que son puntos intermedios de almacenamiento temporal de contenedores, de estos depósitos debe conocer su dirección textual y su geolocalización.
- Determina la hoja de ruta del traslado que incluye tramos entre el origen y el destino pasando por uno o más depósitos.
- Asigna camiones a los tramos en base a los camiones disponibles es decir que no están trasladando un contenedor.
- Gestiona camiones, registrando capacidad en cuanto a volumen y peso, consumo de combustible por kilómetro para determinar costo de traslado y datos del transportista.
- Calcula el costo del traslado y el tiempo estimado de entrega.
- Permite a los transportistas registrar inicio y fin de sus viajes.
- Permite a los clientes realizar seguimiento del estado del contenedor durante el proceso de traslado, indicando si fue o no retirado, si está en viaje o en un depósito o si ya fue entregado.
- Permite observar los contenedores que actualmente están en un depósito para asignarles camiones a su próximo tramo.
- Permite determinar los camiones libres y ocupados.

👥 Roles

Se definen tres roles principales:

Cliente

- Puede registrar un pedido de traslado de contenedor.
- Puede consultar el estado actual de su contenedor (seguimiento).
- Puede ver el costo y tiempo estimado de entrega.

Operador/Administrador

- Carga y actualiza ciudades, depósitos, tarifas, camiones y contenedores.
- Asigna camiones a tramos de traslado.
- Modifica parámetros de tarifación.

Transportista (Camionero o chofer)

- Puede ver los tramos asignados que tiene.
- Puede registrar un inicio o fin de tramo.

🔧 Requerimientos funcionales mínimos

1. Registrar una nueva solicitud de transporte de contenedor. (Cliente)
    a. . La solicitud incluye la creación del contenedor con su identificación única.
    b. La solicitud incluye el registro del cliente si no existe previamente.
    c. Las solicitudes deben registrar un estado, por ejemplo: [borrador - programada - en tránsito - entregada]
2. Consultar el estado del transporte de un contenedor. (Cliente)
3. Consultar rutas tentativas con todos los tramos sugeridos y el tiempo y costo estimados. (Operador / Administrador)
4. Asignar una ruta con todos sus tramos a la solicitud. (Operador/Administrador)
5. Consultar todos los contenedores pendientes de entrega y su ubicación / estado con filtros. (Operador/Administrador)
6. Asignar camión a un tramo de traslado de un contenedor. (Operador/Administrador)
7. Determinar el inicio o fin de un tramo de traslado. (Transportista)
8. Calcular el costo total de la entrega, incluyendo:
    a. Recorrido total (distancia entre origen → depósitos y depósitos → destino)
    b. Peso y volumen del contenedor
    c. Estadía en depósitos (calculada a partir de la diferencia entre fechas reales de entrada y salida del tramo correspondiente)
9. Al finalizar registrar el cálculo de tiempo real y el cálculo de costo real en la solicitud.
10. Registrar y actualizar depósitos, camiones y tarifas.
11. Validar que un camión no supere su capacidad máxima en peso ni volumen.

[!Note] Se deja libertad a los alumnos para la configuración de las tarifas de traslado y estadía en depósito teniendo en cuenta que:

- Debe existir un valor de costo por kilómetro base para el cálculo aproximado que depende del volumen del contenedor.
- Debe existir un valor de litro de combustible configurado y el consumo de combustible aproximado general, que surge del promedio de los consumos de los camiones aptos, para el cálculo aproximado.
- Cada depósito debe mantener un costo de estadía diario
- Los camiones deben conocer su costo base de traslado por km
- Los camiones deben conocer su consumo de combustible promedio

El costo del traslado real surge de la suma de los costos de traslado por kilómetro, mas las estadías en depósito, más el costo de combustible del/los camión/es específico/s.

🧩 Modelo de datos mínimo sugerido

- **Depósito:** identificación, nombre, dirección, coordenadas.
- **Contenedor:** identificación, peso, volumen, estado, cliente asociado.
- **Solicitud:** número, contenedor, cliente, costoEstimado, tiempoEstimado, costoFinal, tiempoReal.
- **Ruta:** solicitud, cantidadTramos, cantidadDepósitos.
- **Tramo:** origen, destino, tipo(origen-deposito, deposito-deposito, deposito-destino, origen-destino), estado (estimado, asignado, iniciado, finalizado), costoAproximado, costoReal, echaHoraInicio, fechaHoraFin, camion.
- **Camión:** dominio (patente del camión o cadena a modo identificatorio), nombreTransportista, teléfono, capacidad peso, capacidad volumen, disponibilidad y costos.
- **Cliente:** datos personales y de contacto.
- **Tarifa:** de acuerdo con el diseño de cada grupo.

🧱 Microservicios esperados

Se esperan al menos **dos microservicios independientes** aunque en una solución básica probablemente sean necesarios más, desplegados en contenedores docker independientes e interconectados, además de un **API Gateway** central.

Se espera aquí que los equipos implementen todos los conceptos vertidos acerca del diseño de soluciones backend con microservicios y que cada microservicio incluya las capas internas de acuerdo con lo documentado en los materiales de cada componente.

🔐 Seguridad y autenticación

- Se utilizará **Keycloak** como proveedor de identidad federada.
- El acceso a los endpoints estará restringido por rol.
- Todos los servicios deberán validar el token JWT.

[!TIP] Con los materiales de seguridad y spring security se publicará un instructivo de instalación de uso de Keycloak paso a paso por lo que se sugiere a los alumnos postergar esta etapa hasta dicho punto. Teniendo en cuenta que tanto los roles como los datos de acceso de cada usuario del rol que corresponda deberán ser creados en esta herramienta.

🌐 API externa obligatoria

Debe integrarse con la API de **Google Maps Directions** (o similar) para consultar **la distancia entre dos puntos** expresados en latitud y longitud. Esta información será utilizada para calcular:

- El recorrido entre origen y depósito
- El recorrido entre depósito y destino
- El recorrido entre depósitos
- O el recorrido entre origen y destino

Según corresponda con la ruta finalmente asignada.

[!TIP] Luego de la revisión del material de interconexión de APIs Se publicará un instructivo con todos los detalles de la conexión a la API de Google maps, por lo que se propone postergar este paso hasta haber tenido acceso a dicho material.

📏 Reglas de negocio obligatorias

- Un camión **no puede transportar contenedores que superen su peso o volumen máximo.**
- La tarifa final del envío se calcula como:
    - Cargos de Gestión valor fijo en base a la cantidad de tramos + costo por kilómetro de cada camión + costo de combustible calculado como (consumo del camión en el tramo × valor del litro) + costo por estadía en depósito (por día)
    - Se debe contemplar costos diferenciados por camión en base a su capacidad de volumen y peso soportado
    - Se debe determinar la tarifa aproximada del envío en base a valores promedio entre los camiones elegibles por características del contenedor.
- El tiempo estimado se calcula en base a las distancias entre los puntos involucrados.
- El seguimiento debe mostrar los estados del envío en orden cronológico.
- Los tramos de ruta deben registrar fechas estimadas y reales para calcular el desempeño del servicio.

💡 El cálculo de la tarifa podrá implementarse libremente según el diseño adoptado por cada grupo. Se recomienda utilizar un esquema por rangos de peso y volumen o asociar tarifas a tipos de camión. Este aspecto formará parte de la evaluación del diseño lógico.

⚙ Requerimientos técnicos

- Proyecto backend en **Java con Spring Boot**.
- Exposición de endpoints REST con respuestas en **JSON**.
- Toda la API debe estar documentada con **Swagger / OpenAPI**.
- Uso correcto de códigos de respuesta HTTP.
- Seguridad implementada vía Keycloak y token JWT.
- Toda consulta o modificación debe estar autenticada.
- Se deben presentar logs de las operaciones importantes.

✅ Evaluación
Se solicita para la entrega inicial
Video de los integrantes del grupo compartiendo:

- DER completo de la solución incluyendo los modelos de datos de todos los microservicios e indicando si se va a utilizar una sola base de datos o bases de datos independientes y qué DBMS van a utilizar en la solución.
    - Si bien el enunciado propone una estructura de DER sugerida, esta no cumple todos los conceptos ni incluye claves y relaciones por lo que se espera aquí el diseño lógico definitivo de la base de datos.
- Diseño a nivel de contenedor donde figuren cada uno de los microservicios o contenedores involucrados y las relaciones entre ellos.
- Diseño de cada microservicio funcional donde figuren los recursos y endpoints que se tendrán en cuenta junto con los roles que tendrán acceso a ellos y una breve referencia inicial a los datos de entrada y respuesta de cada uno.

Se solicita para la entrega final

- Solución funcionando con un despliegue a partir de docker compose que levante todo el sistema en su conjunto.
- Colección de pruebas que puedan ser ejecutadas en conjunto para ejemplificar flujos completos o de a una por vez de forma tal que permitan interactuar con los endpoints para verificar el cumplimiento de las reglas de negocio y requerimientos básicos. Puede usar para esto, Postman, Bruno, Thunder Client o cualquier herramienta que permita la ejecución de colección de pruebas en conjunto.
- Documentación de los desafíos y decisiones tomadas ante las distintas alternativas en la construcción

- Todo otro material que el grupo crea pertinente para el momento de la presentación y defensa del trabajo.

Se evaluará

- Implementación correcta del modelo
- Cumplimiento de reglas de negocio
- Uso de microservicios y gateway
- Uso de Keycloak y autenticación JWT
- Consumo real de API externa
- Documentación completa con Swagger
- Buenas prácticas de diseño y separación de responsabilidades
- Validación de datos y manejo de errores
- Despliegue funcional y pruebas básicas
- Generación de Logs y posibilidad de revisión de los mismos

IMPORTANTE: Las dudas sobre este enunciado deben plantearse en el foro específico del aula virtual. Las respuestas de los docentes allí tendrán la misma validez que este documento.

✍ Versión preliminar 26 - Septiembre 2025

Resoluciones:

Sistema de Logística de Transporte de Contenedores

1) DER completo de la solución incluyendo los modelos de datos de todos los microservicios e indicando si se va a utilizar una sola base de datos o bases de datos independientes y qué DBMS van a utilizar en la solución.
    a) Si bien el enunciado propone una estructura de DER sugerida, esta no cumple todos los conceptos ni incluye claves y relaciones por lo que se espera aquí el diseño lógico definitivo de la base de datos.

Arquitectura de Base de Datos

DBMS: PostgreSQL

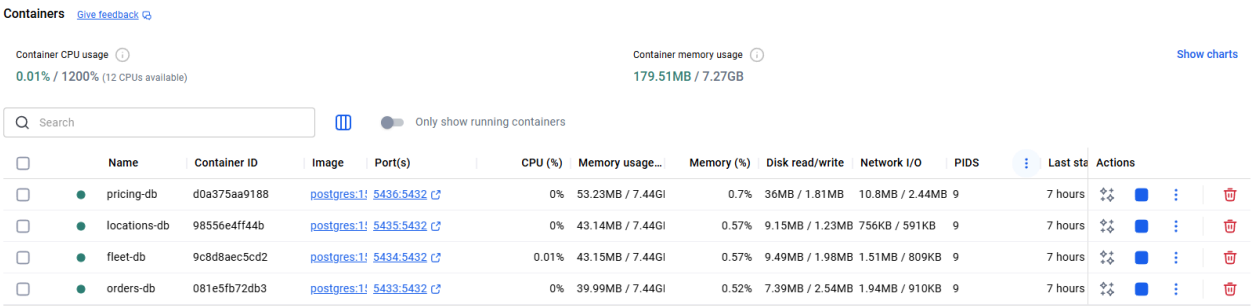
Gestión: pgAdmin

Contenerización: Docker

Vamos a utilizar una arquitectura de microservicios con bases de datos independientes. Cada microservicio tiene su propia base de datos desplegadas en contenedores docker independientes

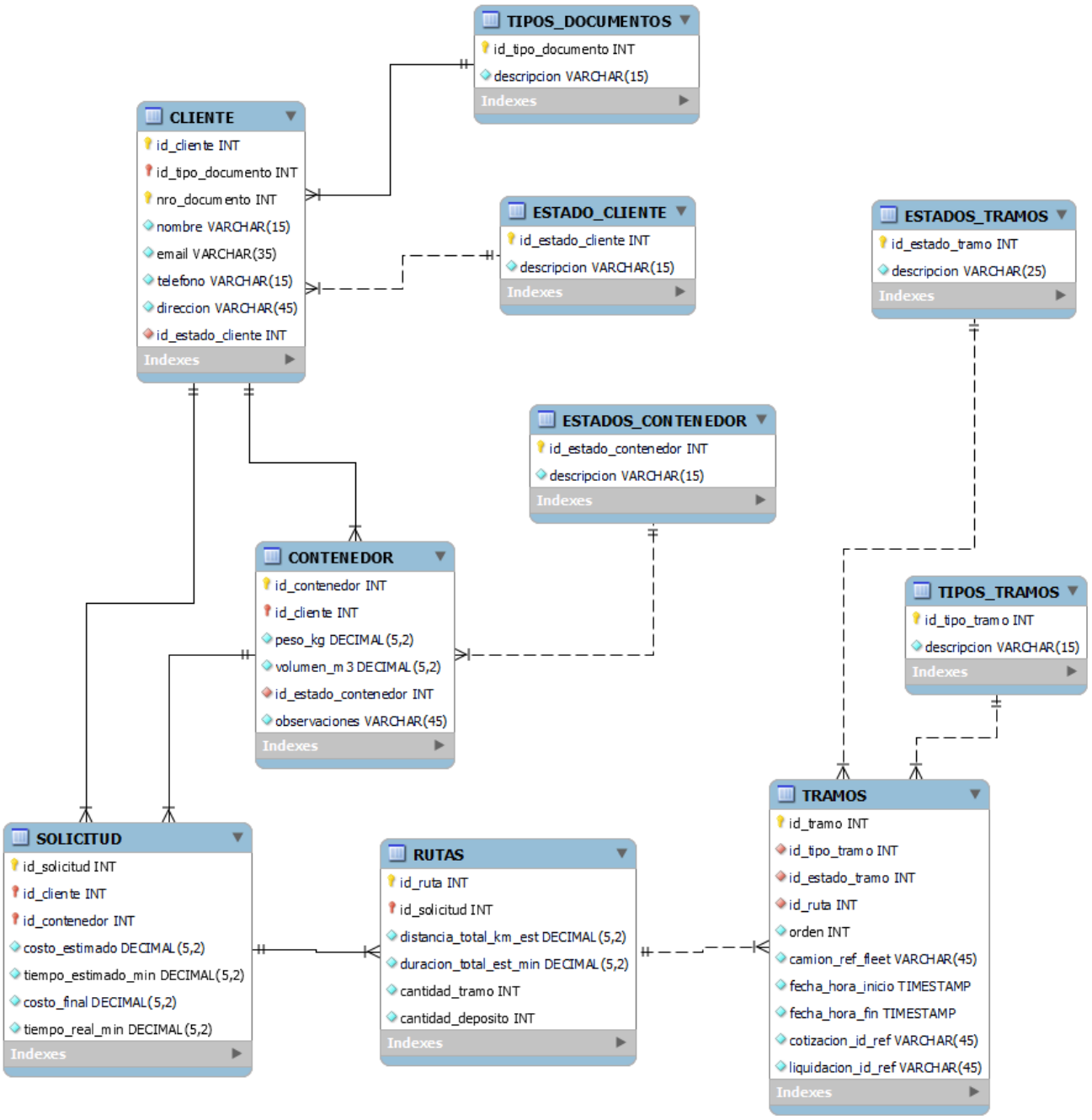
Base de Datos	Puerto	Microservicio	Responsabilidad
orders_db	5433	Orders Service	Gestión de solicitudes, rutas y tracking
fleet_db	5434	Fleet Service	Gestión de camiones y disponibilidad
locations_db	5435	Locations Service	Gestión de depósitos y cálculos de distancias.
princing_db	5436	Pricing Service	Gestión de tarifas y costos

Captura de pantalla de Docker Desktop con los contenedores de bases de datos postgreSQL ejecutándose.



Diagramas de las bases de datos

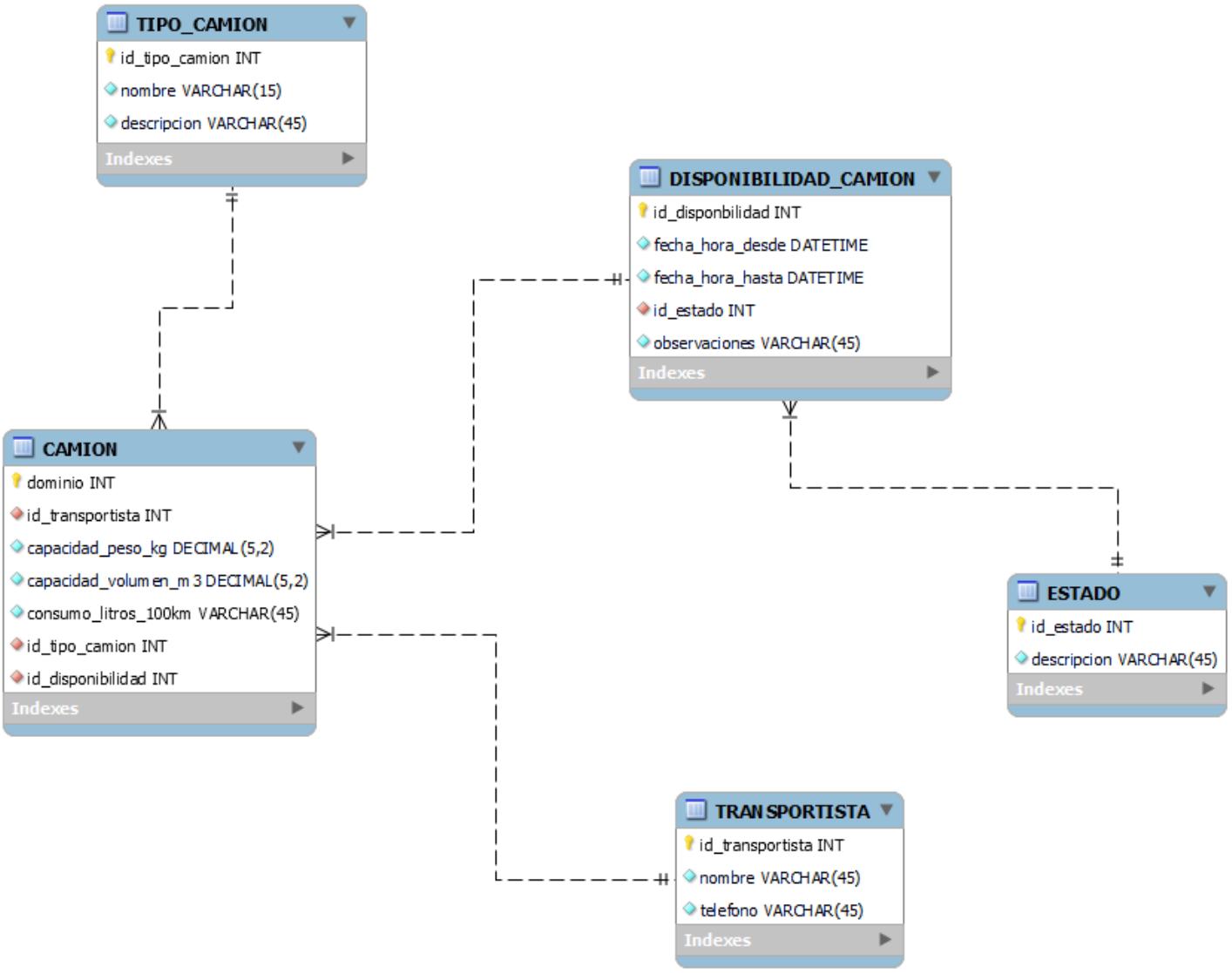
Bases de datos: orders_bd (Pedidos)



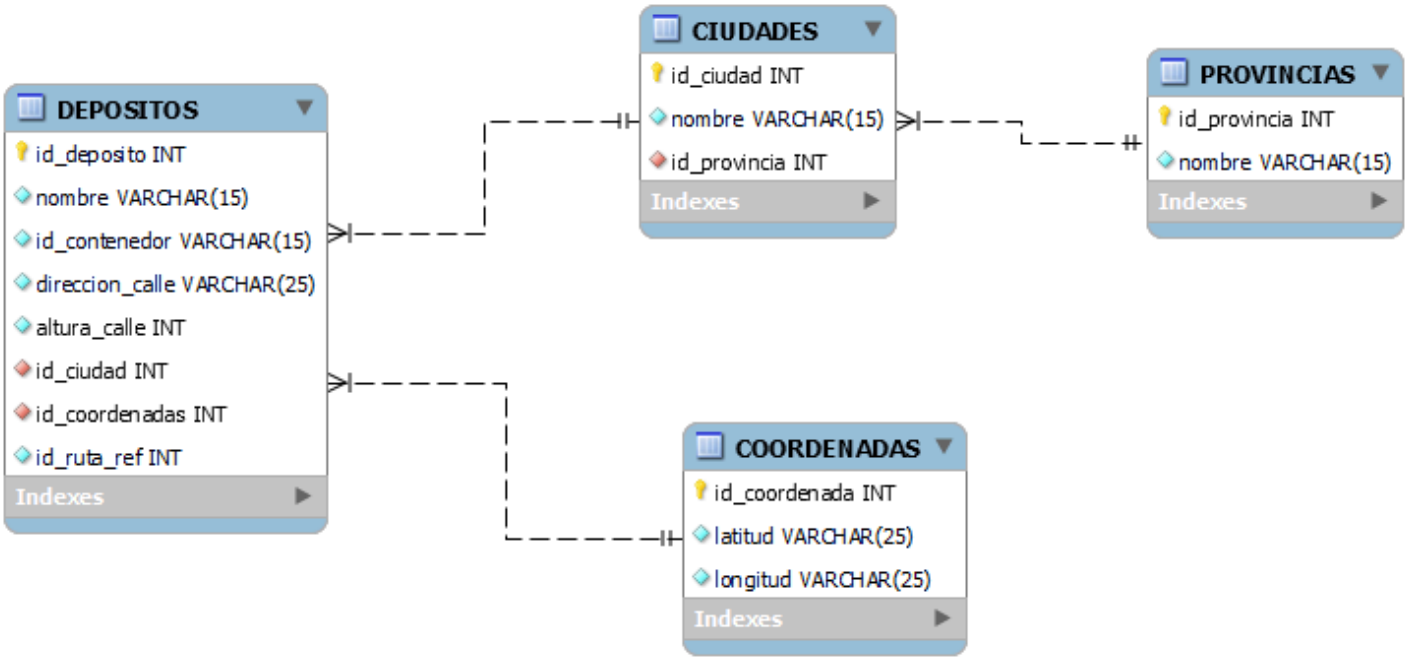
Gestiona el ciclo de vida completo de las solicitudes de transporte desde que el cliente pide transportar un contenedor hasta que es entregado.

Orders Service implementa la lógica de negocio central del sistema: gestiona el ciclo de vida completo de las solicitudes de transporte. Los otros microservicios son componentes especializados que Orders orquesta según sea necesario. Esta separación sigue el principio de responsabilidad única a nivel de microservicios: cada servicio hace una cosa, pero Orders es el que coordina cómo se usan esas capacidades para cumplir con el objetivo de negocio.

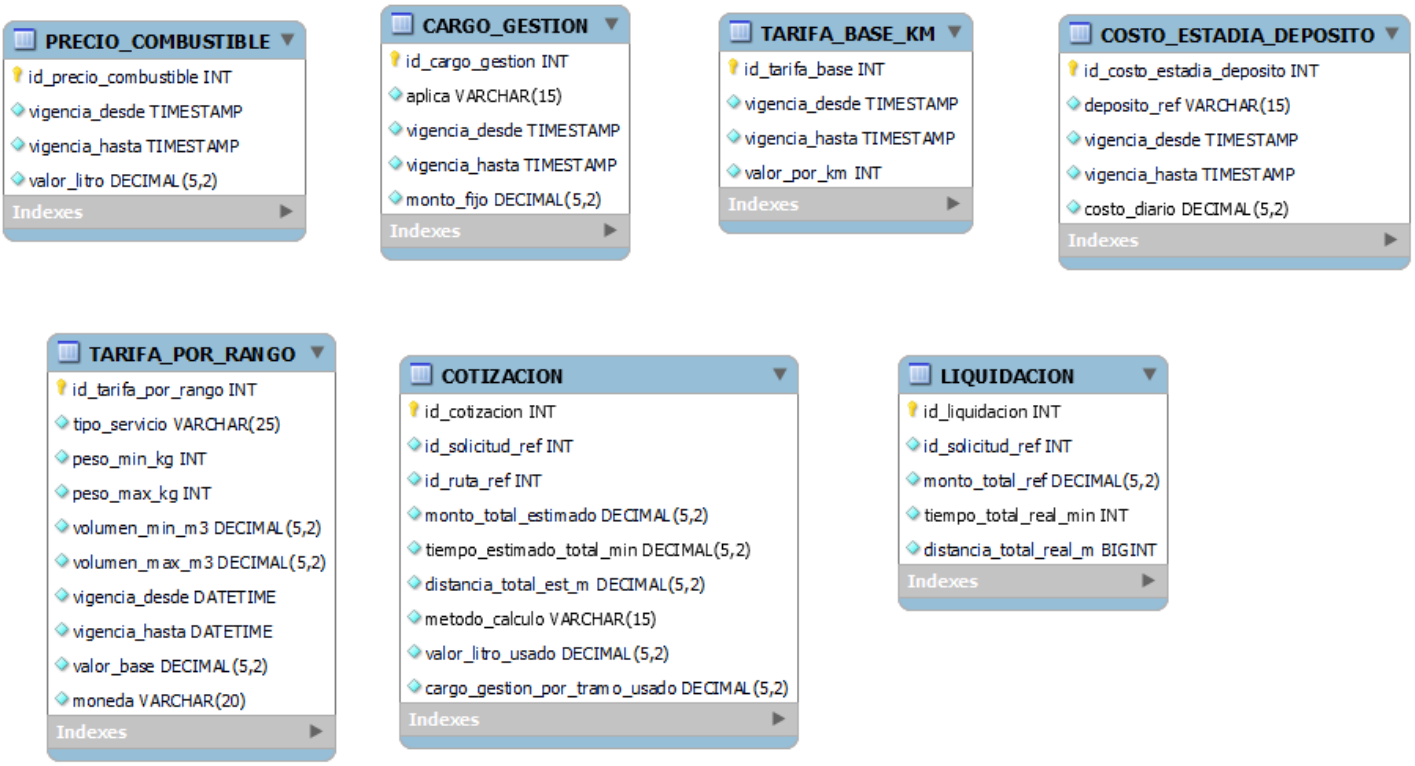
Base de datos: fleet_db (flota)



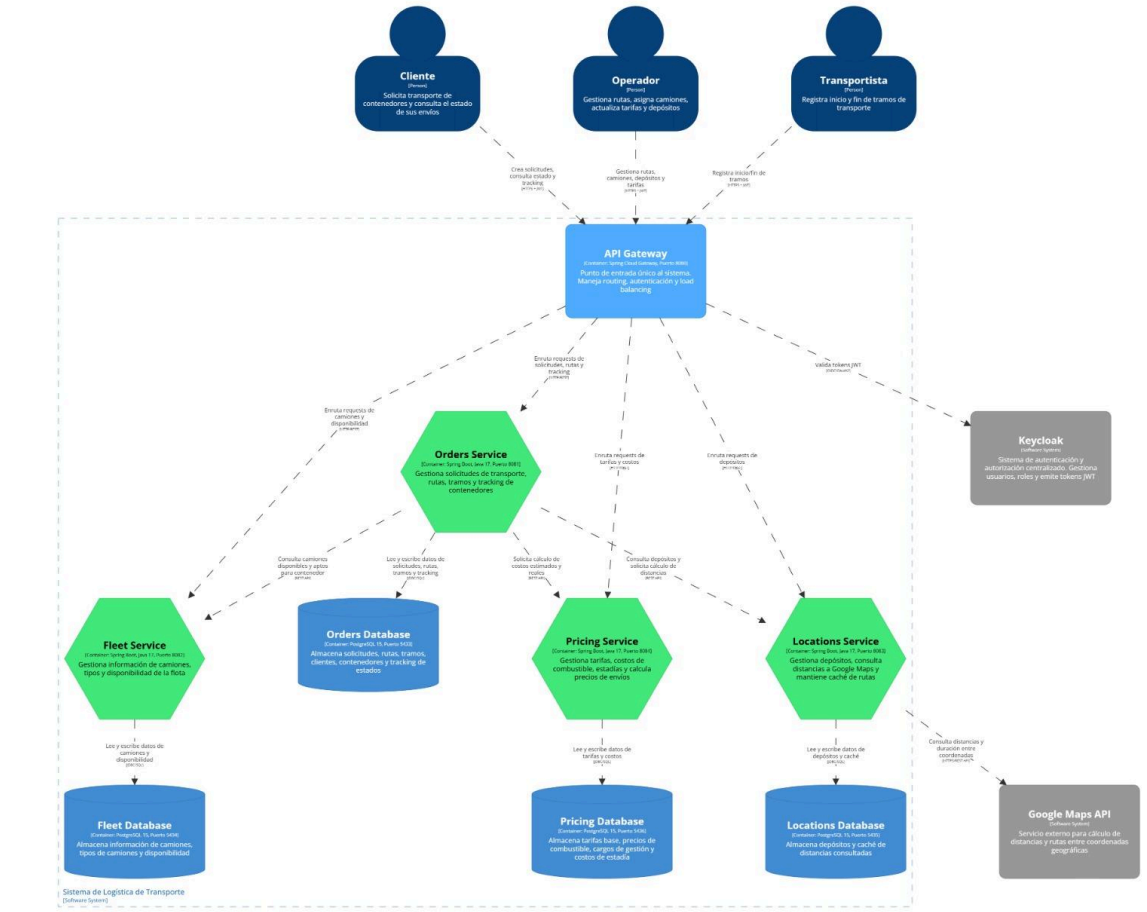
Bases de datos: locations_db (Ubicaciones)



Base de datos: pricing_db (Precios):



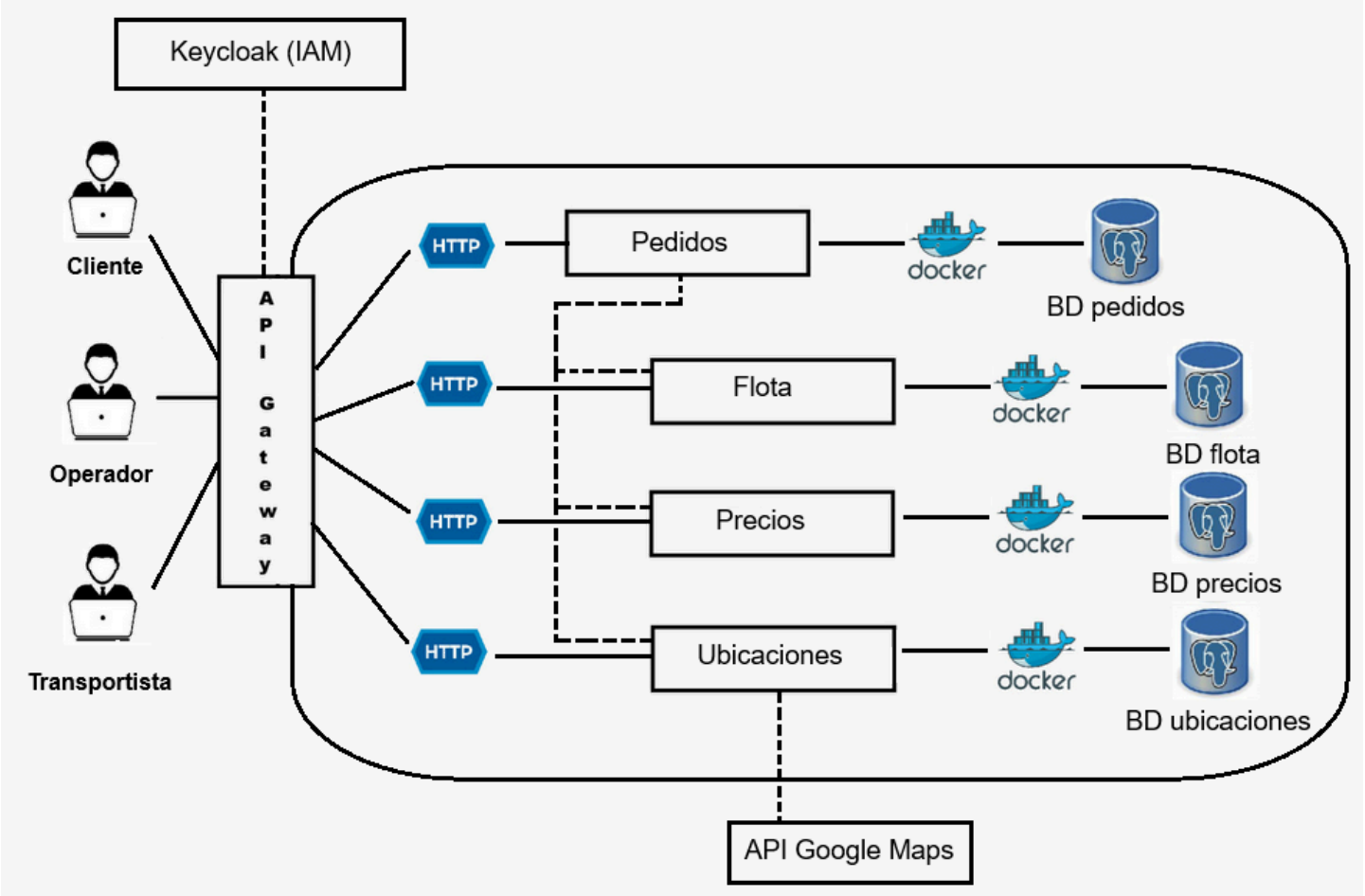
2) Diseño a nivel de contenedor donde figuren cada uno de los microservicios o contenedores involucrados y las relaciones entre ellos.



[Container] Sistema de Logística de Transporte

Diagrama de contenedores mostrando la arquitectura de microservicios y bases de datos independientes
domingo, 19 de octubre de 2025, 11:26 hora estándar de Argentina

C4 Model

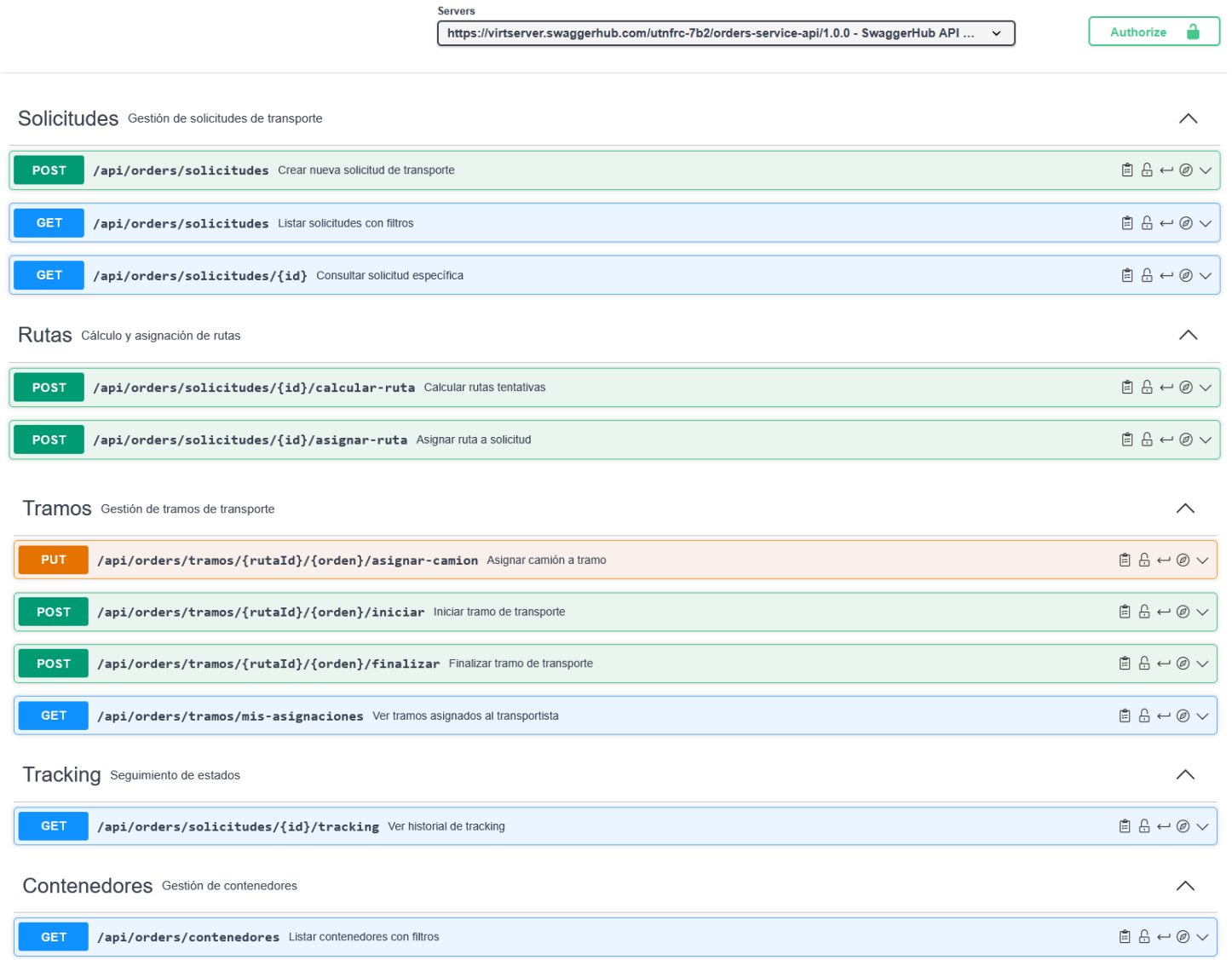

Keycloak (IAM)
Cliente
Operador
Transportista
API Gateway
HTTP
HTTP
HTTP
HTTP
Pedidos
Flota
Precios
Ubicaciones
docker
docker
docker
docker
BD pedidos
BD flota
BD precios
BD ubicaciones
API Google Maps

- Diseño de cada microservicio funcional donde figuren los recursos y endpoints que se tendrán en cuenta junto con los roles que tendrán acceso a ellos y una breve referencia inicial a los datos de entrada y respuesta de cada uno.

Para este punto, utilizamos Swagger.

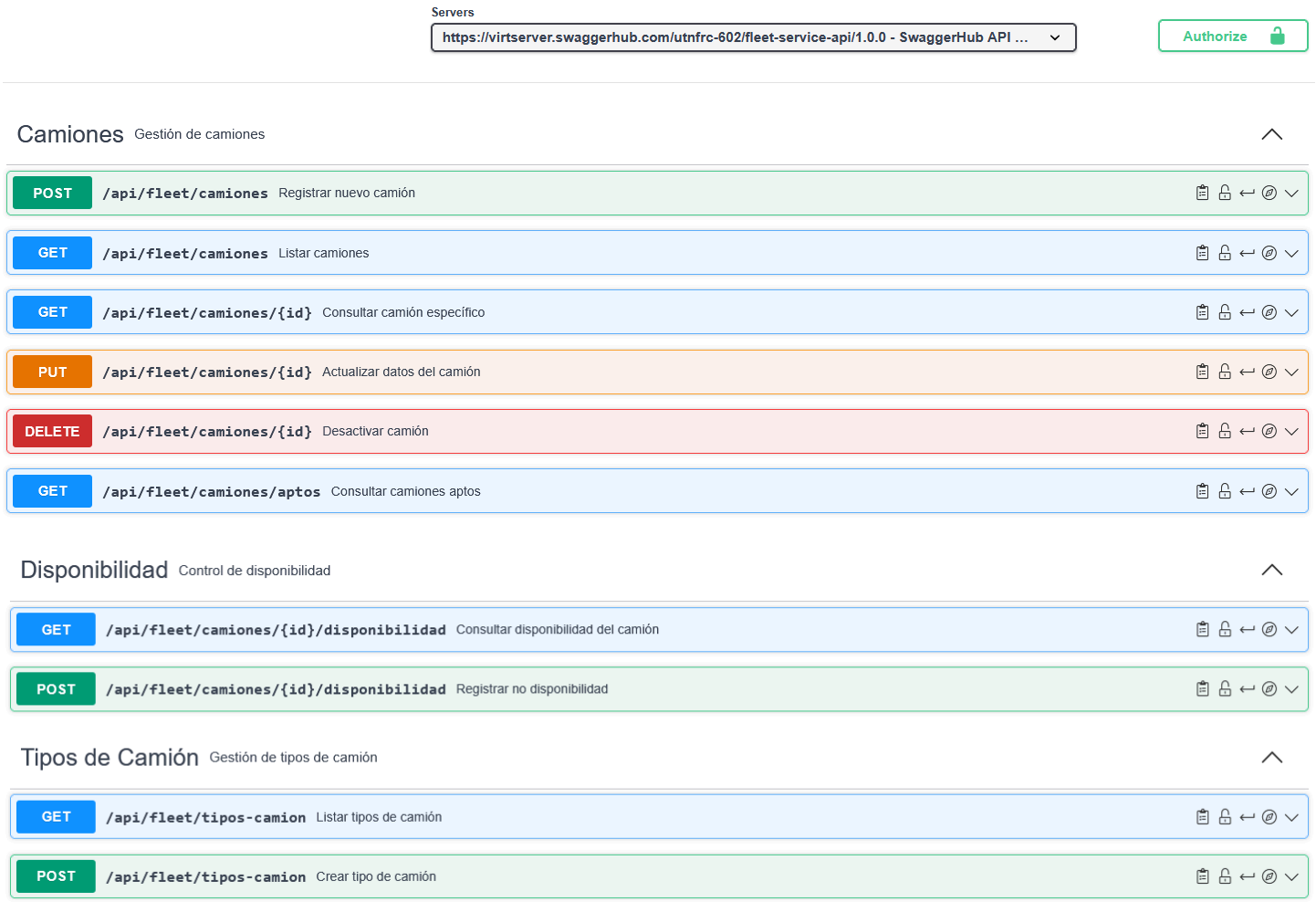
Links del microservicio orders:

https://app.swaggerhub.com/apis-docs/utnfrc-7b2/orders-service-public-api/1.0.0



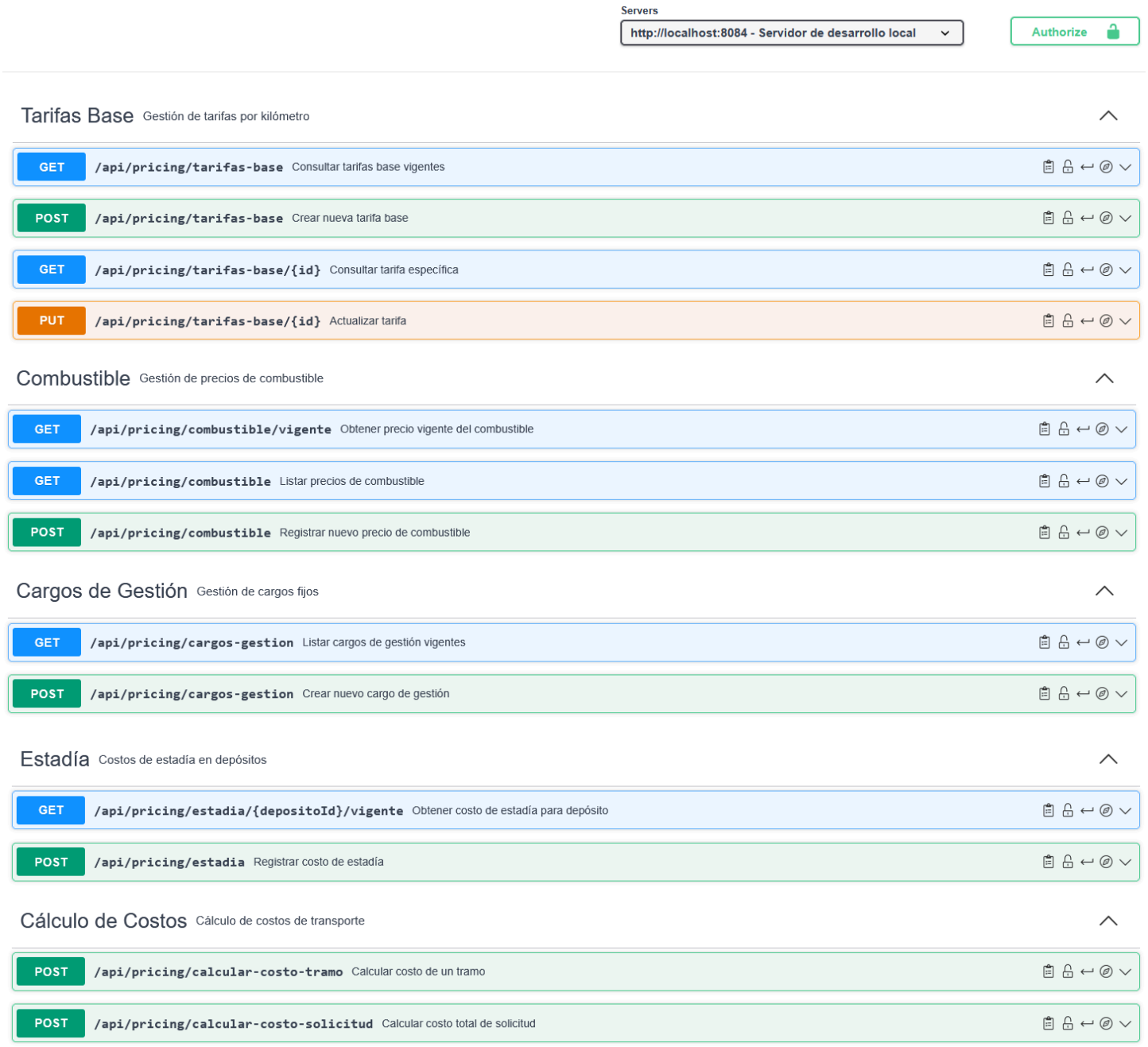
Link del microservicio Flota:

https://app.swaggerhub.com/apis-docs/utnfrc-602/fleet-service-public-api/1.0.0



Link del microservicio Precios:

https://app.swaggerhub.com/apis-docs/utnfrc-7b2/pricing-service-public-api/1.0.0



Link del microservicio Ubicaciones:

https://app.swaggerhub.com/apis-docs/utnfrc-7b2/locations-service-public-api/1.0.0