FOOD DELIVERY

API

Grupo inf25dw1g13

Marta Vieira a046756

Felipe Castilho a047152

Juliana Moreira a047188

Introdução
Food Delivery API

Este projeto consiste numa API REST para simular um sistema de gestão de entrega de comida, designada por Food Delivery API.

Através de métodos HTTP, temos como objetivo demonstrar o funcionamento da API integrada com replicação de bases de dados e determinação de rotas relativamente a operações de escrita ou de leitura através do MaxScale.

Visão geral do Projeto

DB-master
Servidor principal responsável pelas operações de escrita.

DB-réplicas
Servidores secundários configurados como réplicas, para operações de leitura.

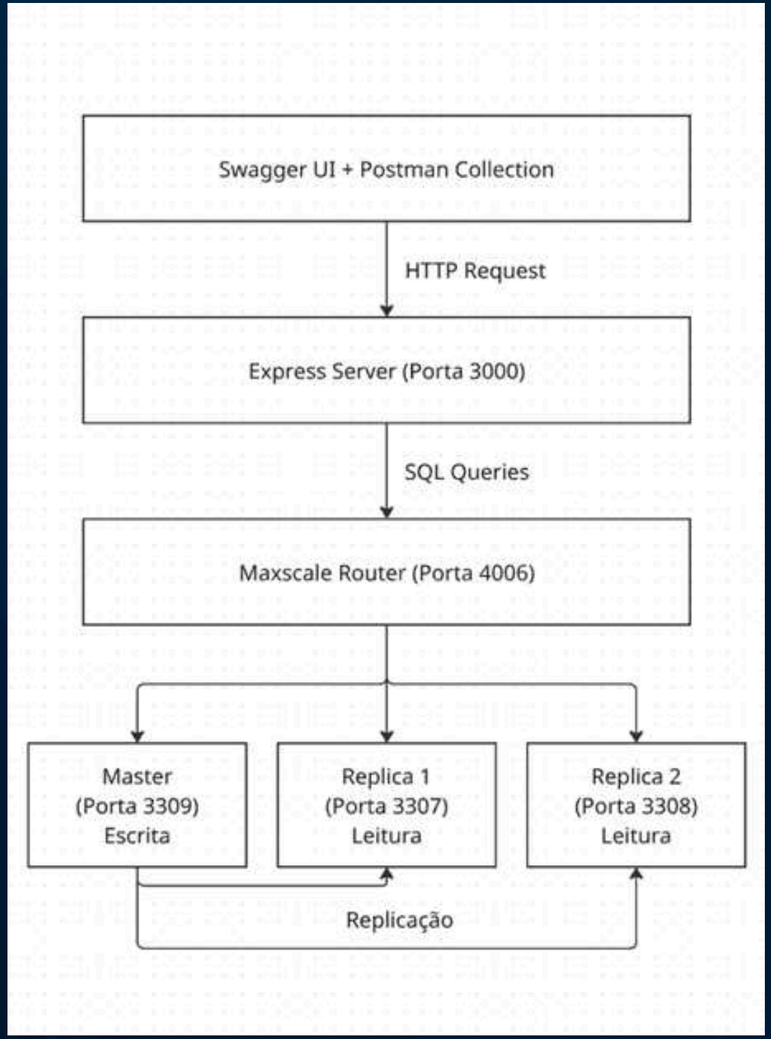
MaxScale
Proxy e roteador que implementa o r*ead/write splitting*, e encaminha automaticamente os pedidos de escrita e leitura para as respetivas bases de dados.

Api
Serviço Node.js/Express que implementa os endpoints da API e comunica com a base de dados exclusivamente através do MaxScale.

Fluxo de um Pedido HTTP

Postman→ API → MaxScale → Master / Réplicas

1. Através do Postman são enviados pedidos à API REST.
2. A API recebe a requisição e encaminha a query para o MaxScale.
3. O MaxScale analisa o tipo de operação:
4. Operações insert/update/delete - aplicadas na database master
5. Operações select - distribuídas pelas réplicas
6. A resposta é retornada para a API.



Docker compose

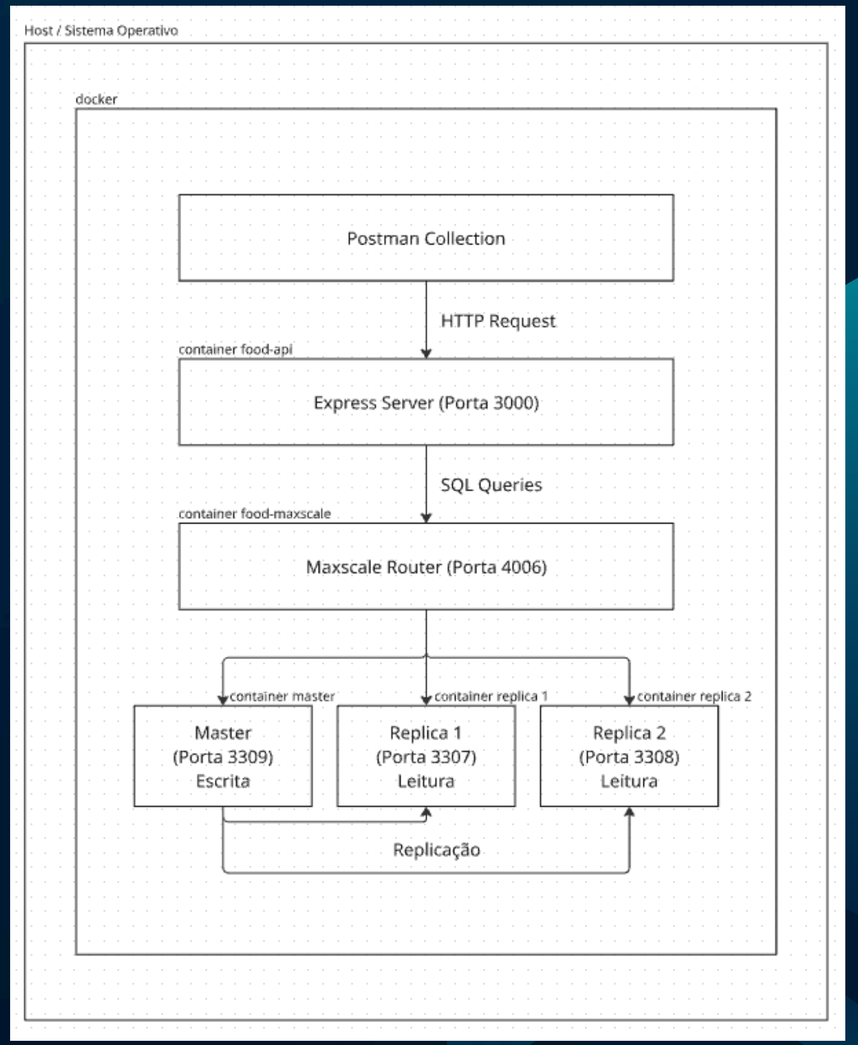
1. **Rede -** food_network
2. **Volumes -** master_data, replica1_data, replica2_data

3. **Master - porta 3309**

- Iniciada com os scripts de criação da base de dados e inserção dos dados iniciais.
- Executa scripts na ordem:
  Cria utilizador para MaxScale *(maxscale_user.sh)*
  Cria base de dados e tabelas *(schema.sql)/init*
  Insere dados iniciais *(seed.sql)*
  Cria *triggers* de *audit (audit.sql)*

4. **Réplicas 1 e 2 - portas 3307 e 3308**

- Cada réplica espera Master estar ligado para se conectar e iniciar o processo de replicação.



Docker compose

5. **Inicialização do MaxScale - porta 4006**

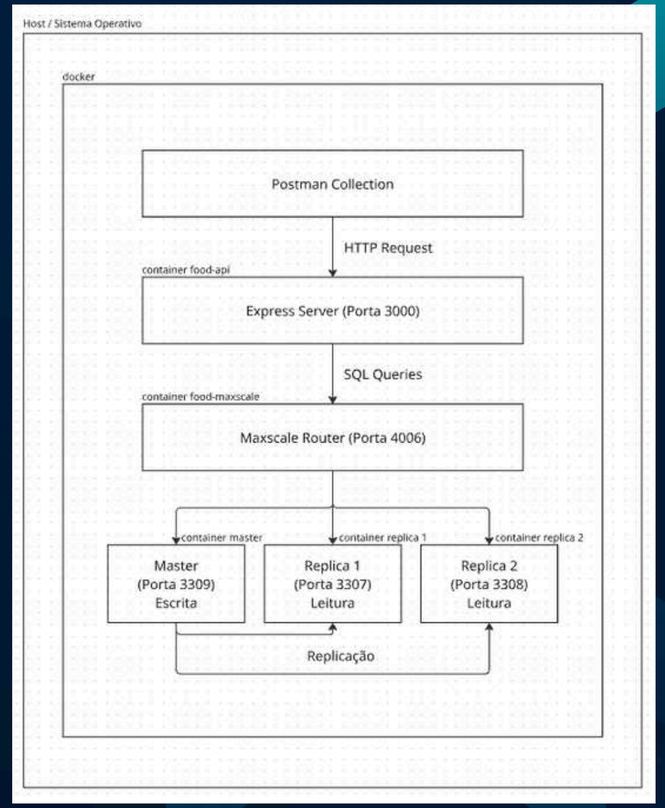
- Lê as configurações definidas no ficheiro maxscale.cnf, que cria o monitor.
- Estabelece ligação às três bases de dados: *Master*, Replica1 e Replica2.

6. **MariaDB Monitor**

- Verifica o estado do Master e das Réplicas a cada 2 segundos com *pings*.
- Promove réplica a Master se Master cair - ***auto-failover***
- Reconecta servidores que voltam online - ***auto-rejoin***
- *Router*: *ReadWriteSplit* - ***read-only enforcement***

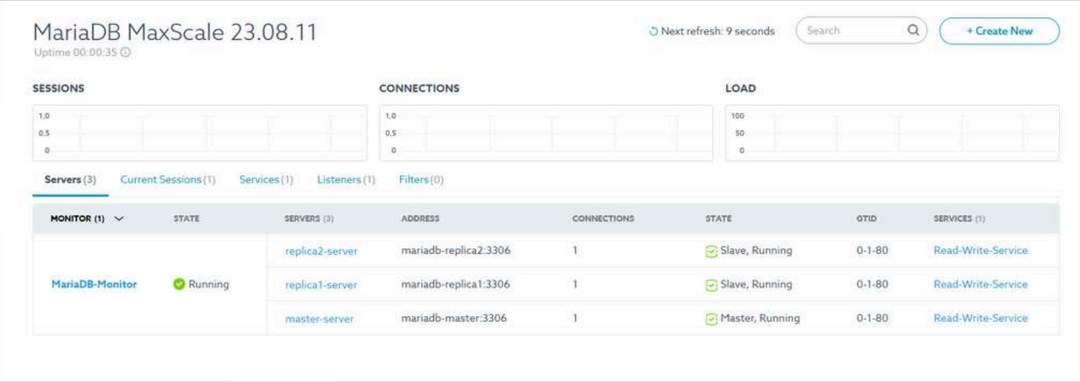
  SELECTs são encaminhados para as réplicas.

  INSERTs / *UPDATEs* / *DELETEs* são enviados para o Master.
- **Interface - porta 8989**
- Credenciais: admin / mariadb
- Permite monitorizar o estado do Master e das Réplicas.



Serviços ativos

- localhost:3000 – Express API
- localhost:3309 – MariaDB Master
- localhost:3307 – MariaDB Replica 1
- localhost:3308 – MariaDB Replica 2
- localhost:4006 – MaxScale (router)
- localhost:8989 – MaxScale Dashboard



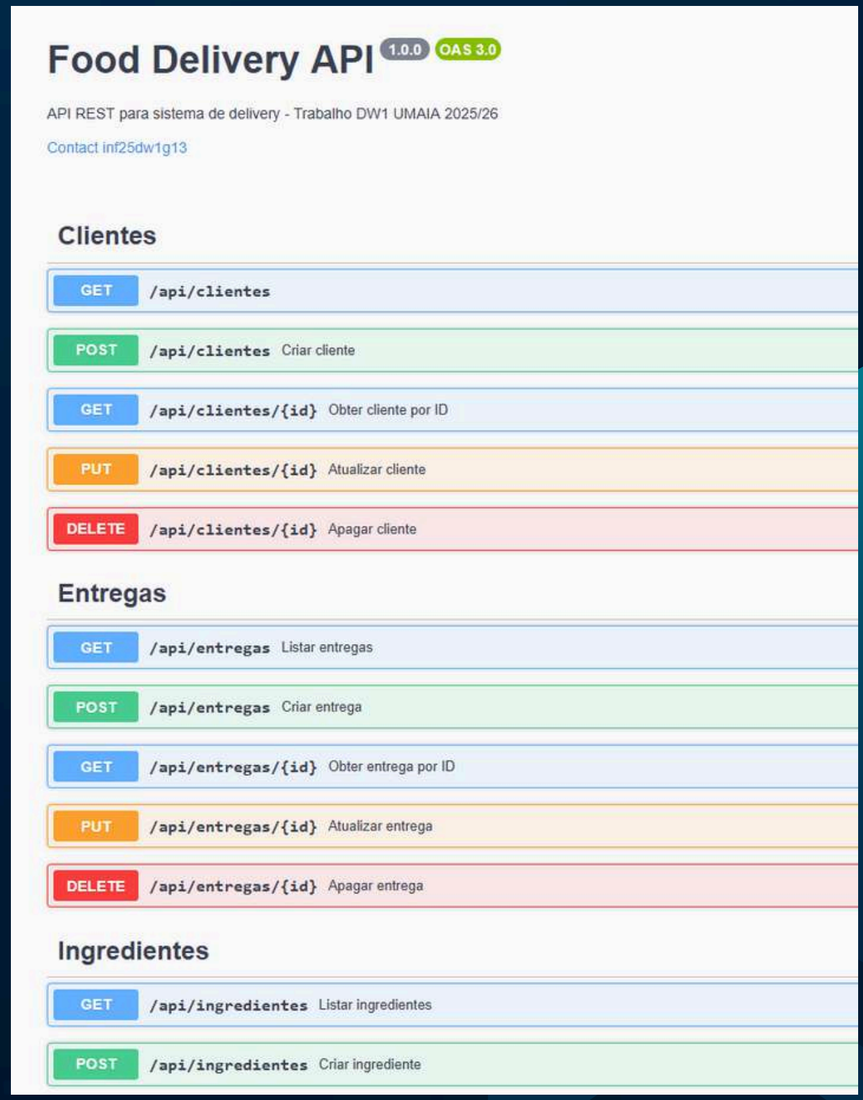


Imagem food-api

/express-server/Dockerfile

FROM node:20-alpine	Faz *download* da imagem base.
WORKDIR /app	Define o diretório de trabalho.
COPY package*.json	Copia ambos os packa
RUN npm install	Instala as dependências.
COPY . .	Copia o resto do código.
EXPOSE 3000	Indica a porta usada pelo Express.
CMD ["node", "index.js"]	Inicia o servidor.

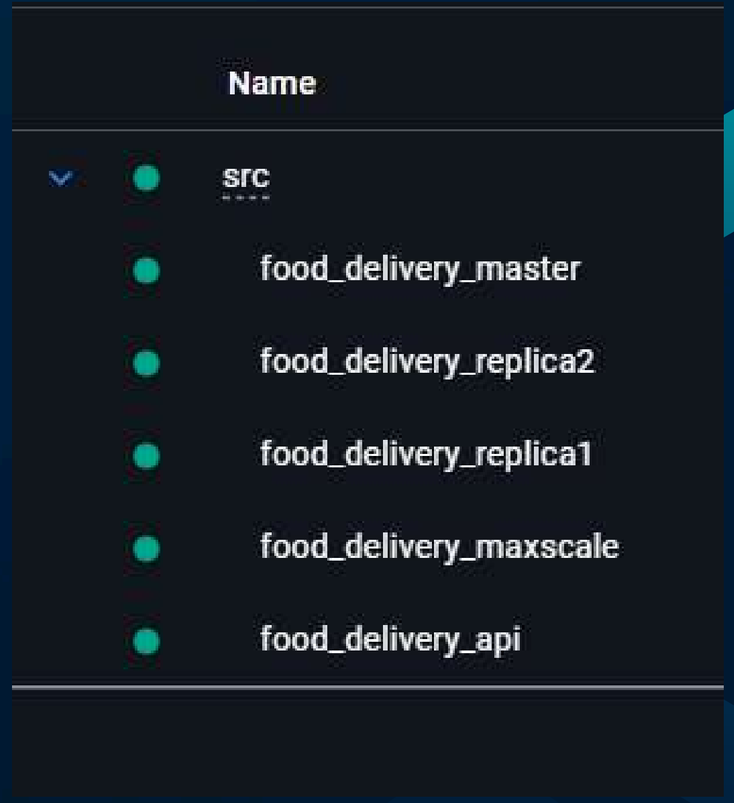


Imagem mariadb-master

/mariadb-master/Dockerfile

```
FROM mariadb:11.2
COPY my.cnf /etc/mysql/...
COPY scripts/ /docker-entrypoint-initdb.d/
```

Faz download da imagem base MariaDB.
Aplica configurações do servidor (IDs e logs).
Copia os scripts de inicialização.

Scripts Executados

- 00_maxscale_user.sh
- 01_schema.sql
- 02_seed.sql
- 03_audit.sql
- 04_setup_replication.sql

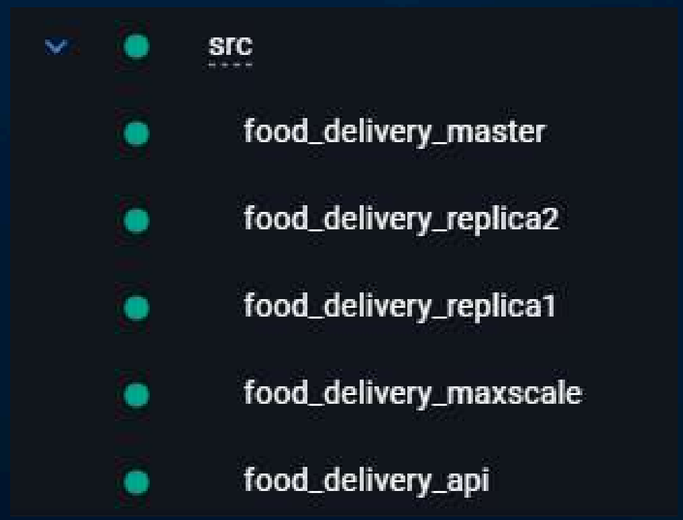


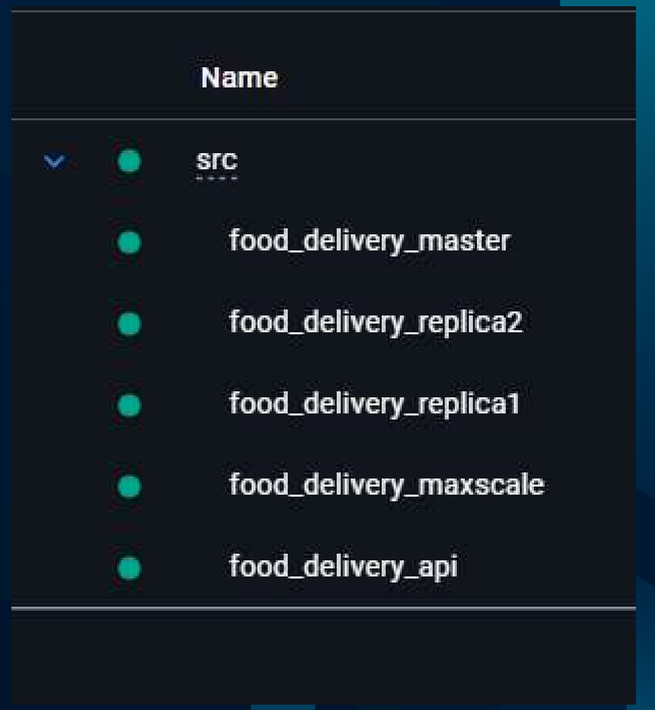
Imagem mariadb-replicas

/replicas/Dockerfile

- FROM mariadb:11.2
- COPY my.cnf /etc/mysql/...
- COPY scripts/ /docker-entrypoint-initdb.d/

Faz download da imagem base MariaDB.
Aplica configurações do servidor (IDs, logs, leitura)
Copia os scripts de inicialização.

- RUN chmod +x /docker-entrypoint-initdb.d/*.sh 2>/dev/null || true
  Dá permissão de execução aos ficheiros .sh dentro do container.



Scripts Executados

- 01_setup_replication.sql

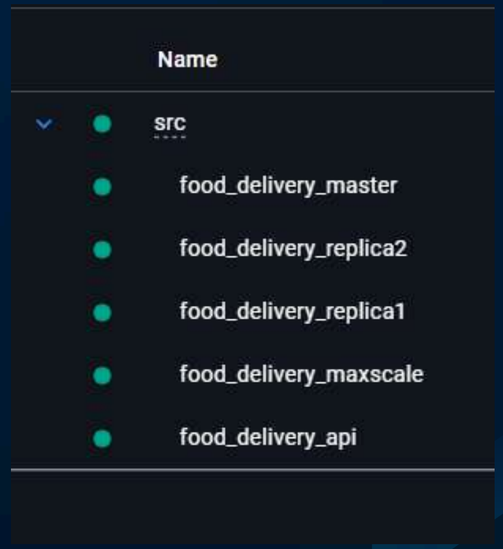
Imagem maxscale

/maxscale/Dockerfile

- FROM mariadb/maxscale:23.08
- COPY maxscale.cnf.ini /etc/maxscale.cnf

Descarrega a imagem do maxscale
Copia a configuração do servidor, monitor e routing

Assim, o MaxScale liga-se ao master, replica1 e replica2.
Depois inicia o MariaDB monitor e ativa o Read/WriteSlipt Router.
Por fim cria o listener da API e o painel de administração.



Base de Dados – Estrutura

restaurantes

- id, nome, morada, telefone, especialidade,
- tempo_medio_preparacao, taxa_entrega, pedido_minimo, ativo, created_at, updated_at.

pratos

- *FK:* restaurante_id
- id, nome, descricao, preco, fvegetariano/vegan/sem_gluten.

ingredientes

- id, nome, unidade, alergeno.

pratos_ingredientes

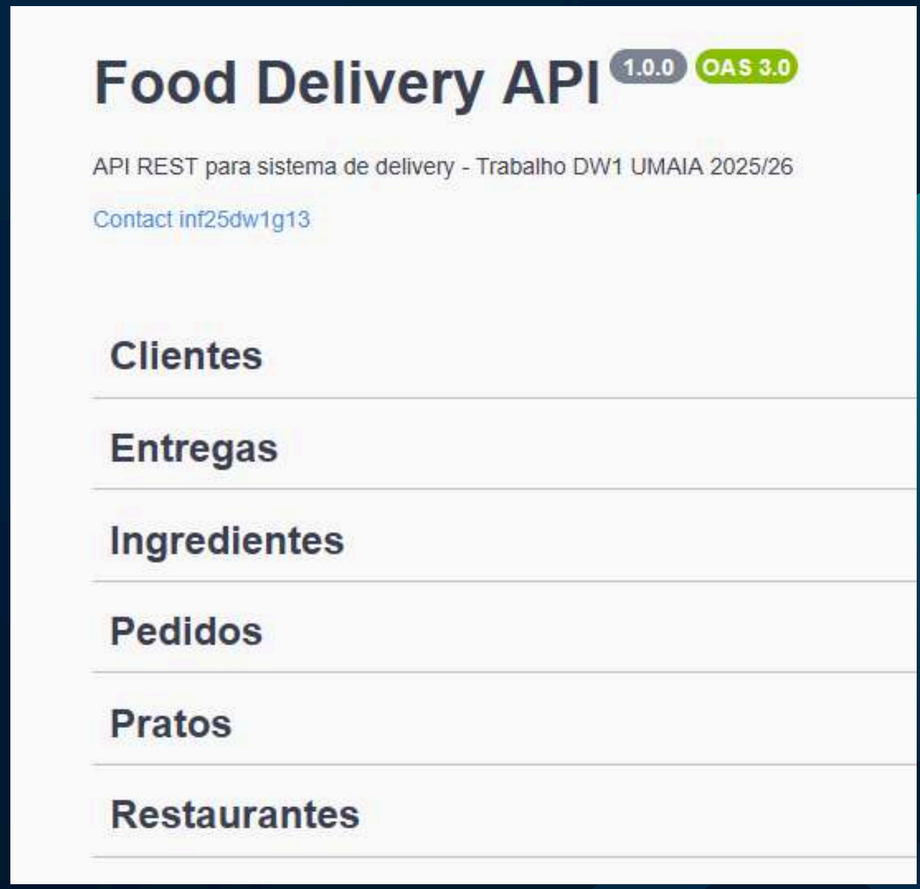
- *PK:* prato_id, ingrediente_id.
- quantidade, obrigatório.

Base de Dados – Estrutura

categorias_pratos
- id, nome , descricao

clientes
- nome, email, telefone, ativo.

moradas_entrega
- *FK:* cliente_id
- tipo, morada, codpostal, cidade



Base de Dados – Estrutura

pedidos

- FKs: cliente_id, restaurante_id, morada_entrega_id
- Campos: subtotal, total, estado ENUM, metodo_pagamento.

entregas

- FKs: cliente_id, restaurante_id, morada_entrega_id
- estado, created_at, updated_at

entregadores

- nome, email, telefone, veículo, matricula, GPS, rating

Base de Dados – Estrutura

pedidos_pratos

- quantidade, preco_unitario, subtotal_item.

audit_log

Os Triggers preenchem as alterações nas tabelas: restaurantes, pratos, clientes, pedidos, entregas.

Endpoints – Restaurantes

- **GET** /api/restaurantes - Lista restaurantes ativos.
  Filtros: especialidade (opcional)
  Ex: http://localhost:3000/api/restaurantes?especialidade=Italiana
- **POST** /api/restaurantes - Cria um restaurante.
- **GET** /api/restaurantes/{id} - Devolve um restaurante correspondente ao ID.
- **PUT** /api/restaurantes/{id} - Atualiza outros campos do restaurante.
- **DELETE** /api/restaurantes/{id} - Apaga um restaurante.
- **GET** /api/restaurantes/{id}/pratos - Lista pratos de um restaurante (relação 1-N).

Endpoints – Pratos

- **GET** /api/pratos - Lista os pratos diponíveis.
  Filtros: restaurante_id e vegetariano
  Ex: http://localhost:3000/api/pratos?restaurante_id=1&vegetariano=true
- **POST** /api/pratos- Cria um prato.
- **GET** /api/pratos/{id} - Devolve um prato correspondente ao id.
- **PUT** /api/restaurantes/{id} - Atualiza outros campos do prato.
- **DELETE** /api/restaurantes/{id} - Apaga um prato.
- **GET** /api/pratos/{id}/ingredientes - Lista ingredientes associados ao prato (M-N).
- **POST** /api/pratos/{id}/ingredientes - Adiciona um ingrediente ao prato.
- **DELETE** /api/restaurantes/{id} - Remove um ingrediente ao prato.

Endpoints – Clientes

- **GET** /api/clientes - Lista os clientes
- **POST** /api/clientes - Cria um cliente.
- **GET** /api/clientes/{id} - Devolve um cliente correspondente ao ID.
- **PUT** /api/clientes/{id} - Atualiza outros campos do cliente.
- **DELETE** /api/clientes/{id} - Apaga um cliente.

O delete só vai eliminar um cliente que não tenha entregas associadas ao id do mesmo, por causa das suas respetivas chaves estrangeiras.

Endpoints – Ingredientes

- **GET** /api/ingredientes - Lista os ingredientes
  Filtros: alergeno
  Ex: http://localhost:3000/api/ingredientes?alergeno=true
- **POST** /api/ingredientes - Cria um ingrediente
- **GET** /api/ingredientes/{id} - Devolve um ingrediente correspondente ao ID.
- **PUT** /api/ingredientes/{id} - Atualiza o ingrediente
- **DELETE** /api/ingredientes/{id} - Apaga um ingrediente.

Endpoints – Pedidos

- **GET** /api/pedidos - Lista os pedidos.
  Filtro: estado
  Ex:http://localhost:3000/api/pedidos?estado=pendente
- **POST** /api/pedidos - Cria um pedido.
- **GET** /api/pedidos/{id} - Devolve um pedido correspondente ao ID.
- **PUT** /api/pedidos/{id} - Atualiza o pedido.
- **DELETE** /api/pedidos/{id} - Apaga um pedido.

Endpoints – Entregas

- **GET** /api/entregas - Lista as entregas

  Filtros: estado, cliente_id, restaurante_id, morada_entrega_id

  Ex: http://localhost:3000/api/entregas?estado=pendente&cliente_id=46&restaurante_id=4&morada_entrega_id=1
- **POST** /api/entregas - Cria um entrega.
- **GET** /api/entregas/{id} - Devolve uma entrega correspondente ao ID.
- **PUT** /api/entregas/{id} - Atualiza outros campos da entrega.
- **DELETE** /api/entregas/{id} - Apaga uma entrega.

healthcheck

- **GET** /health — Health check.

Postman

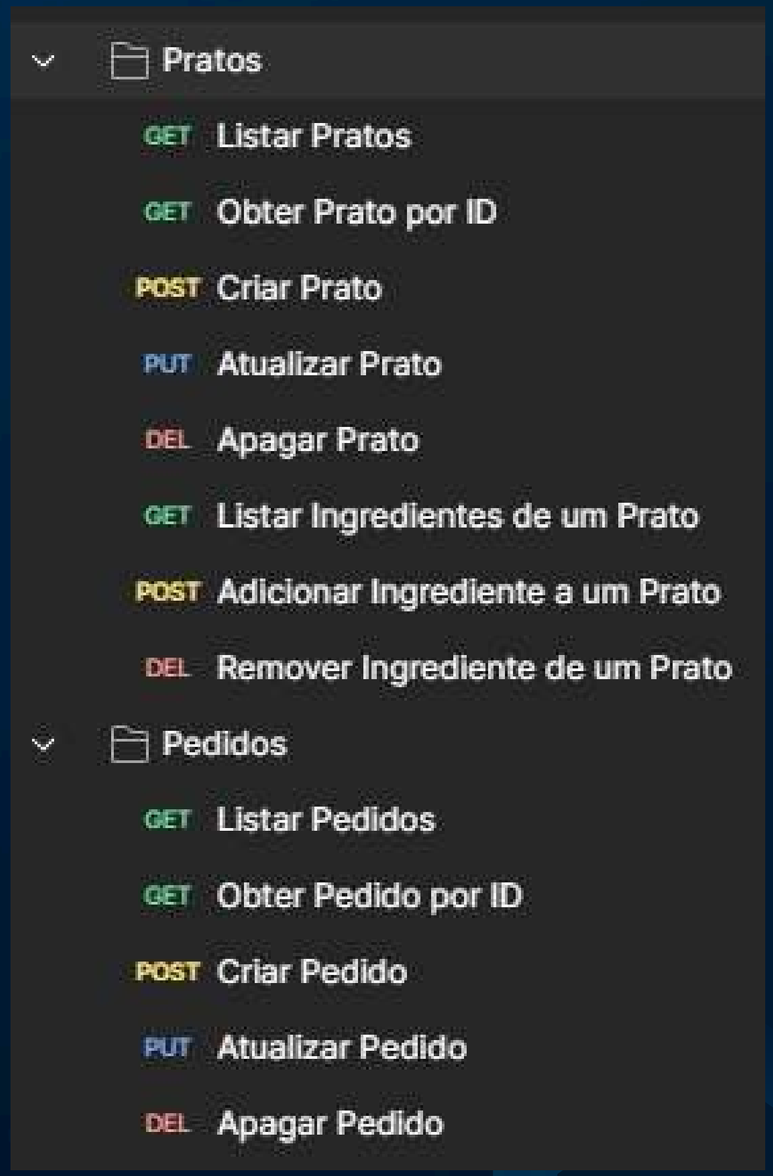
Pratos ↔ Ingredientes (M:N)

1. Criar 2 ingredientes (M:N)
2. Criar 1 prato
3. Ver pratos do restaurante a que pertence
4. Adicionar 2 ingredientes ao prato
5. Listar ingredientes do prato (M:N)
6. Remover um ingrediente do prato (M:N)
7. Atualizar o preço do prato
8. Listar prato pelo id e Eliminá-lo

Postman

Pedidos ↔ Pratos (M:N)

1. Verificar pratos disponíveis num restaurante
2. Criar pedido com múltiplos pratos (M:N)
3. Obter pedido pelo id
4. Atualizar pedido
5. Verificar alteração
6. Eliminar pedido



Postman

Restaurante ↔ Pratos (1:N)

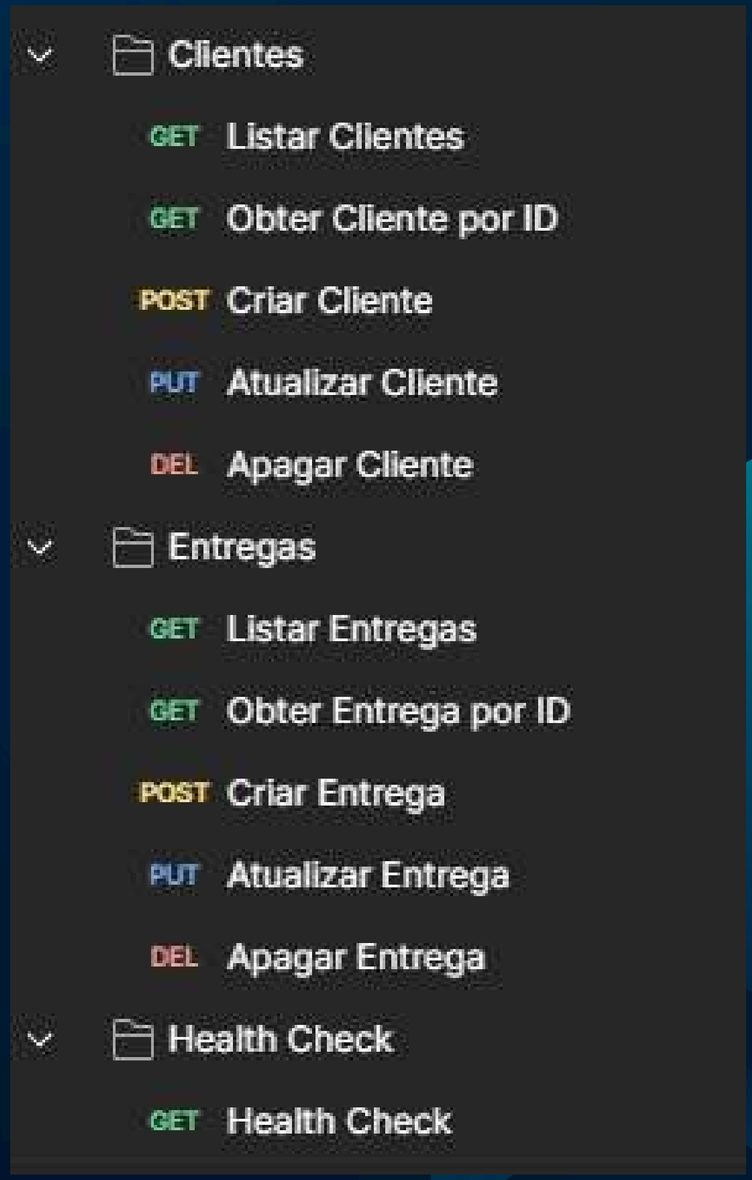
1. Criar restaurante
2. Criar prato para este restaurante (1:N)
3. Criar outro prato para o mesmo restaurante (1:N)
4. Listar todos os pratos do restaurante (1:N)
5. Atualizar Restaurante
6. Verificar Alterações
7. Eliminar Restaurante

Postman

Cliente ↔ Entregas (1:N)

1. Mostrar clientes
2. Criar e atualizar cliente
3. Listar pratos de um restaurante
4. Criar pedido com 2 pratos para esse cliente
5. Criar a entrega
6. Listar todas as entregas
7. Atualizar e Eliminar a entrega

Extra: Verificar o estado do servidor.



FIM