

# 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 *read/write splitting*, e encaminha automaticamente os pedidos de escrita e leitura para as respetivas bases de dados.

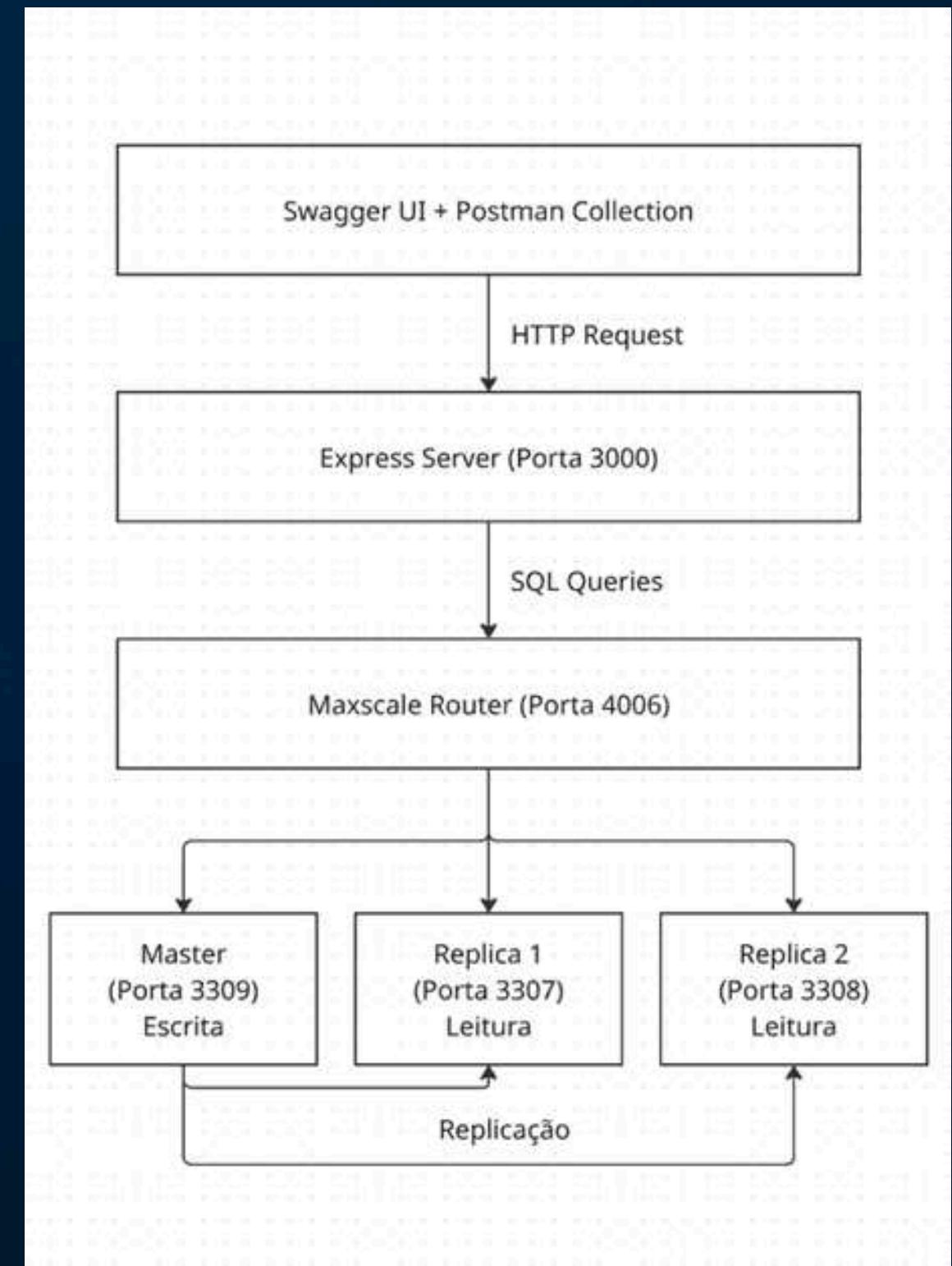
## Api

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:
  - Operações *insert/update/delete* - aplicadas na database master
  - Operações *select* - distribuídas pelas réplicas
4. 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*)

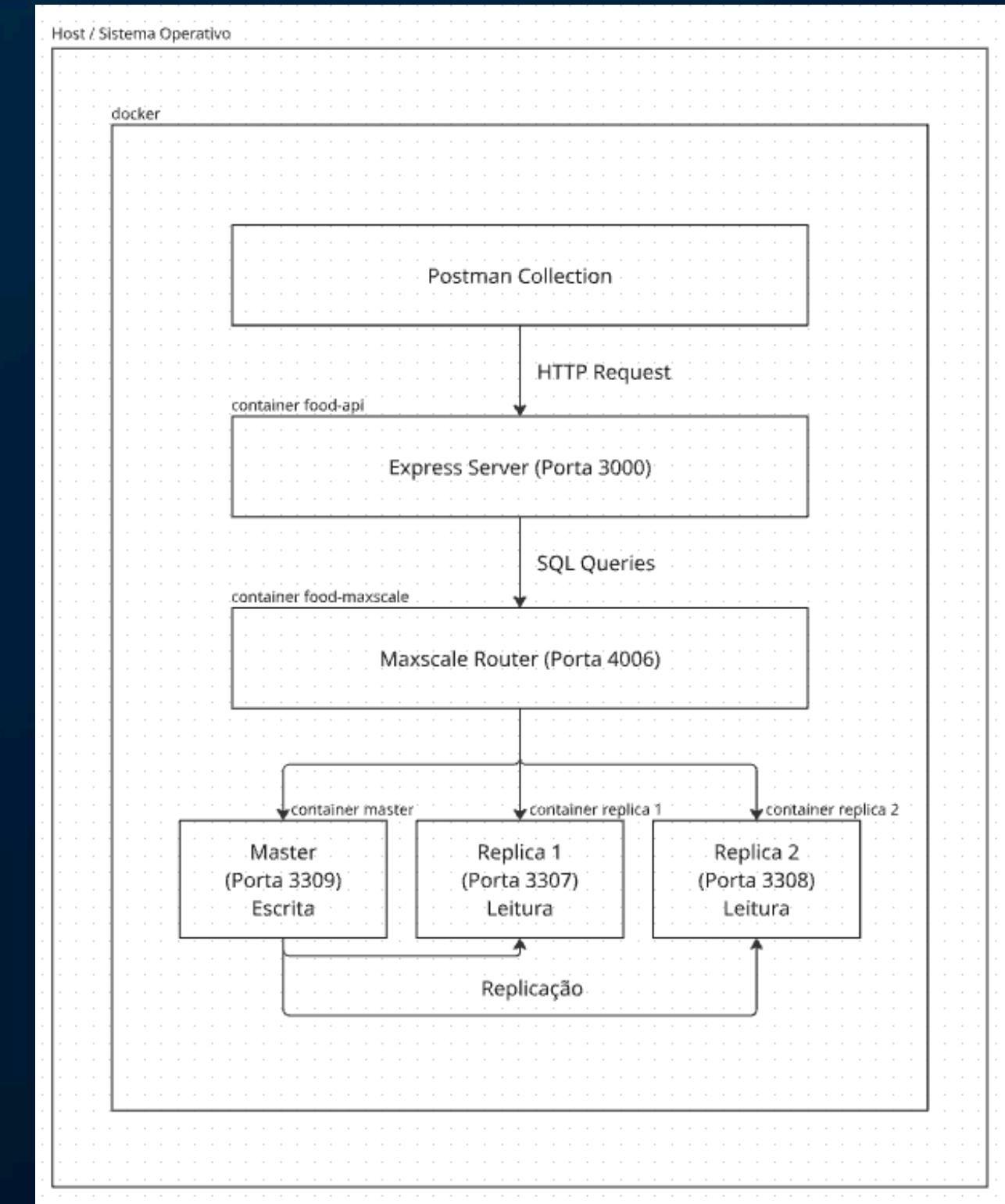
Cria base de dados e tabelas (*schema.sql*)/*init*

Insere dados iniciais (*seed.sql*)

Cria *triggers* de *audit* (*audit.sql*)

## 4. Rélicas 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***

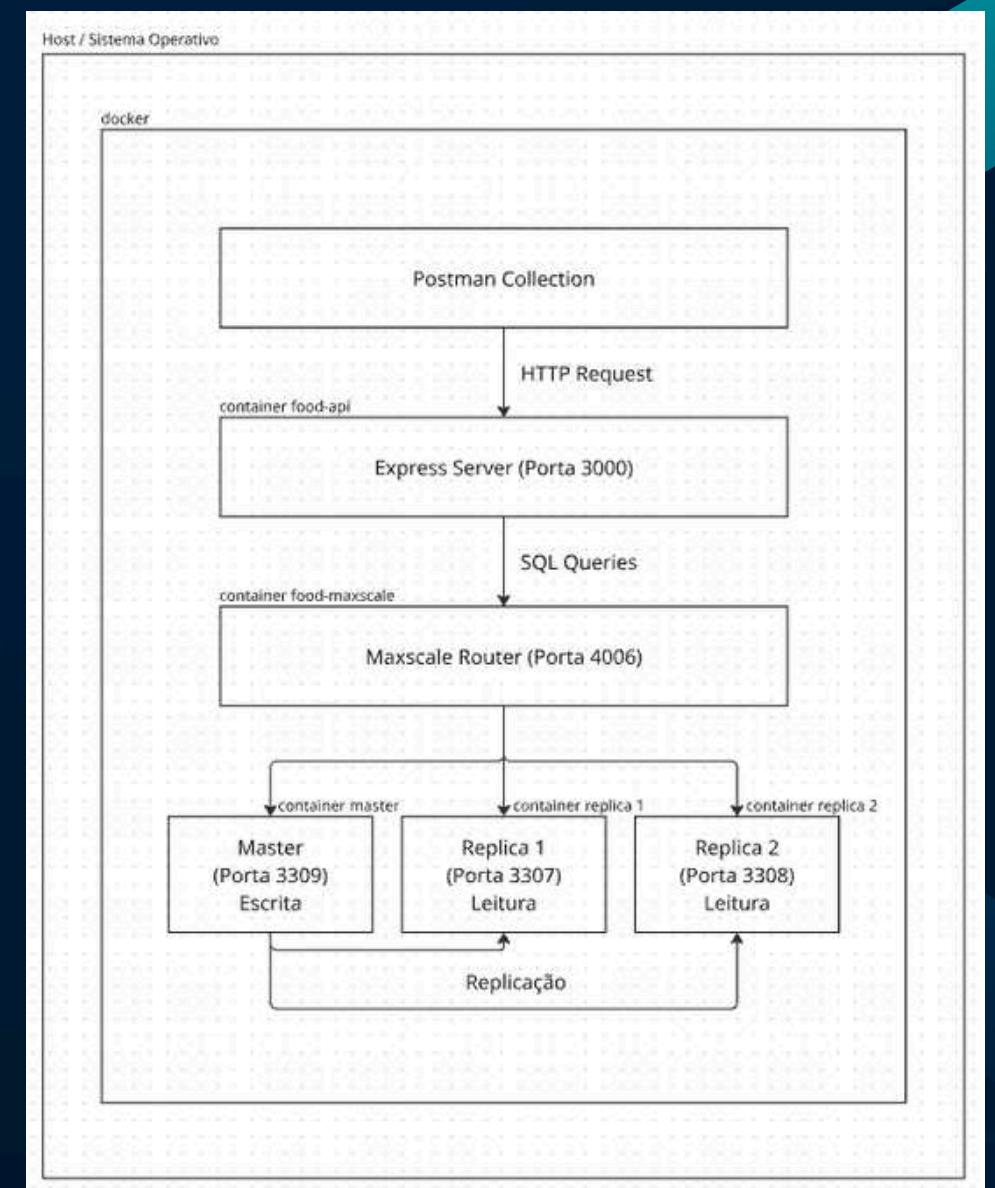
*SELECTs* são encaminhados para as réplicas.

*INSERTs / UPDATEs / DELETEs* são enviados para o Master.

- **Interface - porta 8989**

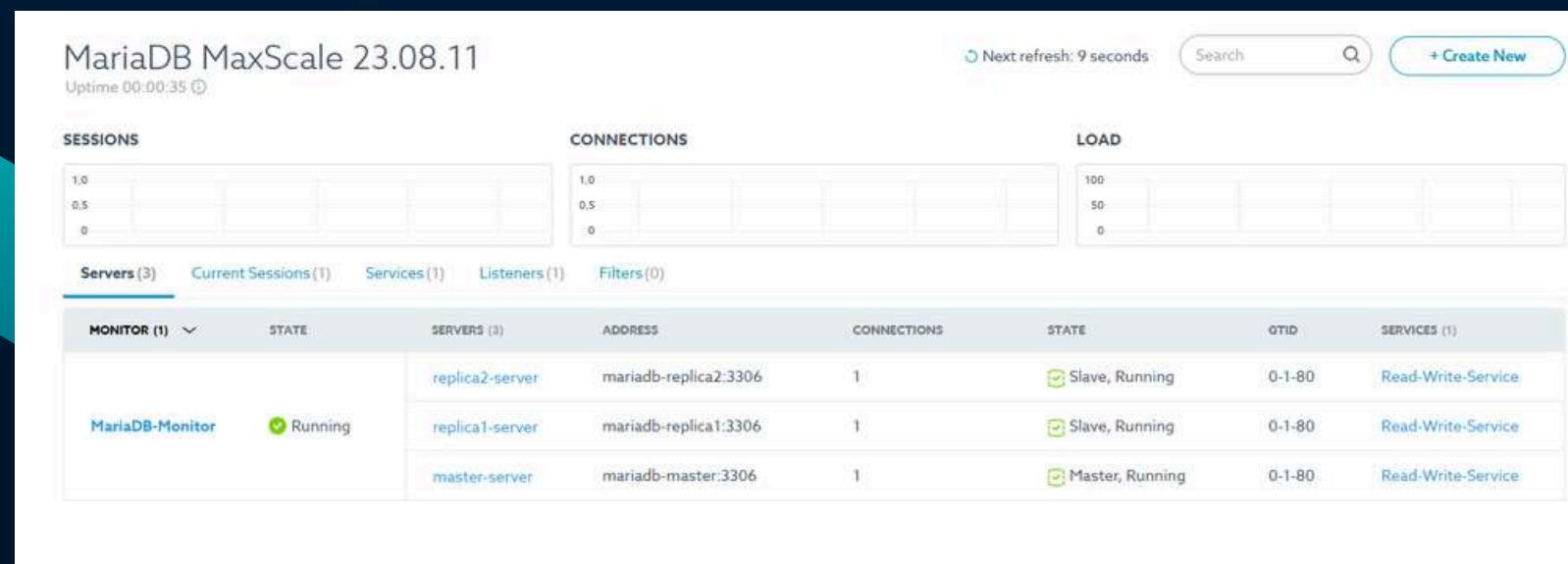
- Credenciais: admin / mariadb

- 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



## Food Delivery API 1.0.0 OAS 3.0

API REST para sistema de delivery - Trabalho DW1 UMAIA 2025/26  
Contact inf25dw1g13

### Clientes

GET	/api/clientes
POST	/api/clientes Criar cliente
GET	/api/clientes/{id} Obter cliente por ID
PUT	/api/clientes/{id} Atualizar cliente
DELETE	/api/clientes/{id} Apagar cliente

### Entregas

GET	/api/entregas Listar entregas
POST	/api/entregas Criar entrega
GET	/api/entregas/{id} Obter entrega por ID
PUT	/api/entregas/{id} Atualizar entrega
DELETE	/api/entregas/{id} Apagar entrega

### Ingredientes

GET	/api/ingredientes Listar ingredientes
POST	/api/ingredientes Criar ingrediente

# Imagen food-api

## /express-server/Dockerfile

```
FROM node:20-alpine
WORKDIR /app
COPY package*.json
RUN npm install
COPY ..
EXPOSE 3000
CMD ["node", "index.js"]
```

Faz *download* da imagem base.  
Define o diretório de trabalho.  
Copia ambos os *packages*.  
Instala as dependências.  
Copia o resto do código.  
Indica a porta usada pelo Express.  
Inicia o servidor.

Name
src
food_delivery_master
food_delivery_replica2
food_delivery_replica1
food_delivery_maxscale
food_delivery_api

# Imagen mariadb-master

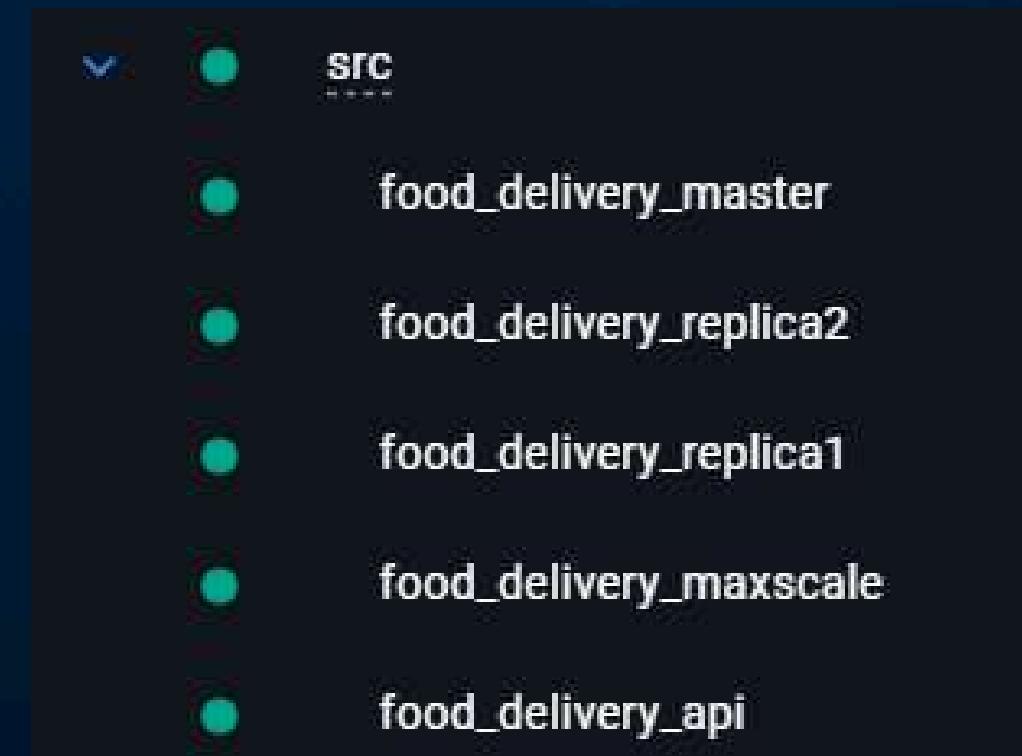
## /mariadb-master/Dockerfile

```
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



# Imagen mariadb-replicas

## /replicas/Dockerfile

- FROM mariadb:11.2
  - COPY my.cnf /etc/mysql/...
  - COPY scripts/ /docker-entrypoint-initdb.d/
  - RUN chmod +x /docker-entrypoint-initdb.d/\*.sh 2>/dev/null || true
- Faz *download* da imagem base MariaDB.  
Aplica configurações do servidor (IDs, logs, leitura)  
Copia os *scripts* de inicialização.
- Dá permissão de execução aos ficheiros .sh dentro do *container*.

## Scripts Executados

- 01\_setup\_replication.sql

Name
src
food_delivery_master
food_delivery_replica2
food_delivery_replica1
food_delivery_maxscale
food_delivery_api

# Imagen 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.

Name
src
food_delivery_master
food_delivery_replica2
food_delivery_replica1
food_delivery_maxscale
food_delivery_api

# 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, vegetariano/vegan/semp\_gluten.

## ingredientes

- id, nome, unidade, alergeno.

## pratos\_ingredientes

- PK: prato\_id, ingrediente\_id.
- quantidade, obrigatório.

The screenshot shows a documentation page for the Food Delivery API. At the top right, it says "Food Delivery API 1.0.0 OAS 3.0". Below that, it says "API REST para sistema de delivery - Trabalho DW1 UMAIA 2025/26" and "Contact inf25dw1g13". On the left, there is a vertical navigation menu with links: "Clientes", "Entregas", "Ingredientes", "Pedidos", "Pratos", and "Restaurantes".

# Base de Dados - Estrutura

## **categorias\_pratos**

- id, nome, descricao

## **clientes**

- nome, email, telefone, ativo.

## **moradas\_entrega**

- FK: cliente\_id
- tipo, morada, codpostal, cidade

## Food Delivery API 1.0.0 OAS 3.0

API REST para sistema de delivery - Trabalho DW1 UMAIA 2025/26

Contact inf25dw1g13

Cientes

Entregas

Ingredientes

Pedidos

Pratos

Restaurantes

# Base de Dados - Estrutura

## pedidos

- FKs: cliente\_id, restaurante\_id, morada\_entrega\_id
- 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

## Food Delivery API 1.0.0 OAS 3.0

API REST para sistema de delivery - Trabalho DW1 UMAIA 2025/26

Contact [inf25dw1g13](mailto:inf25dw1g13)

[Clientes](#)

[Entregas](#)

[Ingredientes](#)

[Pedidos](#)

[Pratos](#)

[Restaurantes](#)

# 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.

The screenshot shows a documentation page for the "Food Delivery API". At the top right, it says "1.0.0" and "OAS 3.0". Below that, it says "API REST para sistema de delivery - Trabalho DW1 UMAIA 2025/26" and "Contact inf25dw1g13". On the left, there is a vertical navigation menu with links: "Clientes", "Entregas", "Ingredientes", "Pedidos", "Pratos", and "Restaurantes".

# 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 disponíveis.  
Filtros: restaurante\_id e vegetariano  
Ex: [http://localhost:3000/api/pratos?restaurante\\_id=1&vegetariano=true](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/pratos/{id} - Atualiza outros campos do prato.
- **DELETE** /api/pratos/{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/prato/{id}/ingredientes - 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](http://localhost:3000/api/entregas?estado=pendente&cliente_id=46&restaurante_id=4&morada_entrega_id=1)
- **POST** /api/entregas - Cria uma 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 - Verifica a ligação à base de dados.

# Postman

## Pratos ↔ Ingredientes (M:N)

1. Criar 2 ingredientes (M:N).
2. Criar um 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. Obter o prato pelo id e eliminá-lo.



# Postman

## Pedidos ↔ Pratos (M:N)

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

The screenshot shows the Postman interface with two main sections: 'Pratos' and 'Pedidos'. The 'Pratos' section contains six requests: 'Listar Pratos' (GET), 'Obter Prato por ID' (GET), 'Criar Prato' (POST), 'Atualizar Prato' (PUT), 'Apagar Prato' (DELETE), and 'Listar Ingredientes de um Prato' (GET). The 'Pedidos' section contains five requests: 'Listar Pedidos' (GET), 'Obter Pedido por ID' (GET), 'Criar Pedido' (POST), 'Atualizar Pedido' (PUT), and 'Apagar Pedido' (DELETE).

Método	Ação
GET	Listar Pratos
GET	Obter Prato por ID
POST	Criar Prato
PUT	Atualizar Prato
DELETE	Apagar Prato
GET	Listar Ingredientes de um Prato
POST	Adicionar Ingrediente a um Prato
DELETE	Remover Ingrediente de um Prato
GET	Listar Pedidos
GET	Obter Pedido por ID
POST	Criar Pedido
PUT	Atualizar Pedido
DELETE	Apagar Pedido

# Postman

## Restaurante ↔ Pratos (1:N)

1. Criar um restaurante.
2. Criar um 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 o restaurante.
6. Verificar as alterações.
7. Eliminar o restaurante.



# Postman

## Cliente ↔ Entregas (1:N)

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

Extra: Verificar o estado do servidor.

The screenshot shows the Postman interface with a dark theme. It displays three main sections of API documentation:

- Clientes**:
  - `GET Listar Clientes`
  - `GET Obter Cliente por ID`
  - `POST Criar Cliente`
  - `PUT Atualizar Cliente`
  - `DEL Apagar Cliente`
- Entregas**:
  - `GET Listar Entregas`
  - `GET Obter Entrega por ID`
  - `POST Criar Entrega`
  - `PUT Atualizar Entrega`
  - `DEL Apagar Entrega`
- Health Check**:
  - `GET Health Check`

# Film