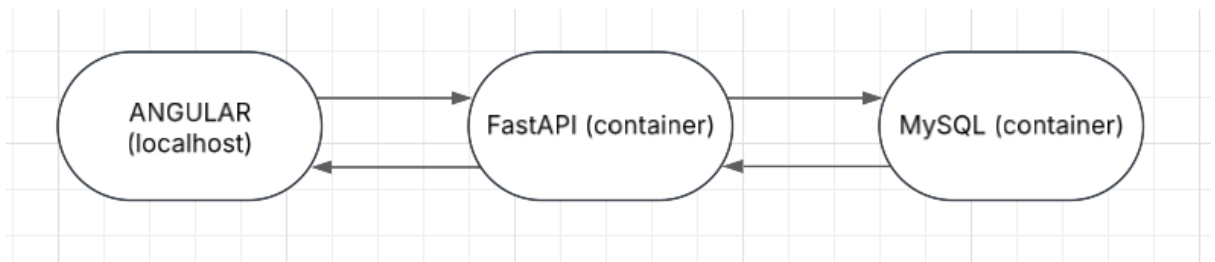


Objetivo:

O intuito desse projeto é estudar o envio e o consumo de requisições HTTP, ao final do projeto é para ser realizado um CRUD completo e a integração entre as ferramentas Angular, FastAPI (DOCKER), MySQL(DOCKER).

Arquitetura geral do projeto:



Arquitetura Backend (está na pasta backend):

~/backend

Pasta que contém todos os arquivos necessários para rodar o back-end em um container.

~/backend/app

Pasta que contém a aplicação Python.

~/backend/Dockerfile

Arquivo responsável pela configuração do container FastAPI. Como não existe uma imagem oficial para o FastAPI, o Dockerfile define o ambiente do container:

Configurações do Dockerfile:

```
FROM python:3 # Configura o container para executar Python 3.x.x
WORKDIR /usr/src/app # Define o diretório de trabalho do container
COPY . /usr/src/app # Copia os arquivos do diretório local para o WORKDIR
RUN pip install --no-cache-dir -r dependenciaspython.txt # Instala as dependências do FastAPI
EXPOSE 8000 # Expõe a porta 8000 para comunicação
```

~/backend/dependenciaspython.txt

Arquivo com as dependências do FastAPI e outras bibliotecas necessárias.

~/backend/init.sql

Arquivo que será executado automaticamente sempre que o container for iniciado ou perder seu volume. Ele configura o banco de dados inicial.

~/backend/docker-compose.yml

Arquivo de orquestração dos containers. Configura todos os aspectos do backend e interage com o banco de dados MySQL.

Principais configurações para o container Python:

```
context: . # Localiza o arquivo Dockerfile e usa suas configurações
volumes: ./usr/src/app # Espelha o diretório local com o container
ports: "8000:8000" # Expõe a porta 8000 do container para a máquina local
depends_on: - mysql # O serviço Python depende do MySQL
networks: my_network # Rede interna para comunicação entre containers
command: ["uvicorn", "main:app", "--host", "0.0.0.0", "--reload"] # Inicia o FastAPI no servidor Uvicorn
```

Principais configurações para o container Docker:

```
restart: no # Não reinicia automaticamente em caso de erro
environment:
  MYSQL_ROOT_PASSWORD: root # Senha do root no MySQL
ports:
  - "3306:3306" # Porta padrão do MySQL
  - "33060:33060" # Porta para conexão no MySQL
volumes:
  - mysql_data:/var/lib/mysql # Volume para persistência dos dados do MySQL
networks:
  - my_network
```

~/backend/main.py

Arquivo principal do Python, responsável por configurar as rotas, iniciar a instância do FastAPI e configurar o CORS. Este é o arquivo que será executado no servidor Uvicorn

~/backend/app

Contém a aplicação Python composta por três pastas principais:

- **~/backend/app/core**
Contém a configuração da conexão com o banco de dados e a "base" de modelagem ORM.
- **~/backend/app/models**
Contém a modelagem ORM para o banco de dados.
- **~/backend/app/**
Define as rotas da API e mantém a persistência de dados utilizando PyDantic.

Arquitetura frontend (está na pasta frontend):

A arquitetura front end segue o modelo de projetos Angular, logo temos os seguintes elementos:

[em processo ...]

Casos de uso:

- Pré requisitos:
 - iniciar o container -> sudo docker compose up -d
 - iniciar o servidor angular -> ng server -0

Situação 1 - Login e cadastro:

Rota de acesso: localhost:4200/login

Página esperada:

Projeto: Estudo de API Rest com FastAPI • [Descritivo](#)

Email:

Senha:

Enviar Login

Por padrão o arquivo init.sql inicia o banco de dados com 1 usuário já cadastrado, podemos verificar o usuário acessando o container:

- sudo docker exec -it mysqlapp mysql -u root -p //senha: 5912
- use projetoapi;
- select * from usuario;

```
marques@marques: ~... x  marques@marques: ~... x
mysql> select * from usuario
-> ;
+----+-----+-----+-----+
| id | nome  | email           | senha |
+----+-----+-----+-----+
|  1 | teste | teste@teste.com | teste |
+----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> 
```

Utiliza email e senha na aplicação e verificar os LOGS no navegador e na API:

The image shows a terminal window on the left and a web browser on the right. The terminal displays the command to start the application using Docker Compose, followed by logs indicating the application is running on http://0.0.0.0:8000. The web browser shows the login page with fields for Email (teste@teste.com) and Senha (password), and a button to 'Enviar Login'. The browser's developer console shows the successful login response as a JSON object.

```
marques@marques: ~/Documentos/desafios/projetoAPIRest/backe...$ sudo docker logs -f fastapiapp
[sudo] senha para marques:
INFO: Will watch for changes in these directories: ['/usr/src/app']
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO: Started reloader process [1] using StatReload
INFO: Started server process [8]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 172.18.0.1:40128 - "GET /usuario/login?email=teste@teste.com&senha=teste HTTP/1.1" 200 OK
```

Projeto: Estudo de API Rest com FastAPI

Email:

Senha:

```
ok { usuario: { id: 1, nome: "teste", email: "teste@teste.com" } }
```

Retornou 200 pois o usuário está cadastrado.

Agora na situação de realizar um cadastro:

Rota de acesso: localhost:4200/cadastro

The image shows a terminal window on the left and a web browser on the right. The terminal displays the command to start the application using Docker Compose, followed by logs indicating the application is running on http://0.0.0.0:8000. The web browser shows the registration page with fields for Nome (Juan), Email (juan@teste.com), and Senha (password), and a button to 'Cadastrar'. The browser's developer console shows the successful registration response as a JSON object.

```
marques@marques: ~/Documentos/desafios/projetoAPIRest/backe...$ sudo docker logs -f fastapiapp
[sudo] senha para marques:
INFO: Will watch for changes in these directories: ['/usr/src/app']
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO: Started reloader process [1] using StatReload
INFO: Started server process [8]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 172.18.0.1:40128 - "GET /usuario/login?email=teste@teste.com&senha=teste HTTP/1.1" 200 OK
INFO: 172.18.0.1:48652 - "OPTIONS /usuario/cadastro HTTP/1.1" 200 OK
INFO: 172.18.0.1:48652 - "POST /usuario/cadastro HTTP/1.1" 200 OK
```

Cadastro

Nome:

Email:

Senha:

```
ok { usuario: { id: 2, nome: "Juan", email: "juan@teste.com", senha: "password" } }
```