Unidade Curricular – Projeto Físico de Redes
Prof. Augusto da Rosa Muniz

Exercício Guiado: "Minha Primeira Stack Docker-Compose"

Objetivo: rodar docker compose up e ver um site simples lendo/escrevendo em um banco de dados rodando em outro container, escrever o arquivo **docker-compose.yml e apresentar em aula.** O trabalho deverá ser apresentado em aula de forma de pitch rápida de no máximo de 10 min, o trabalho pode ser realizado em dupla.

Contexto Teórico

Conceito	Para que serve na prática
Docker	Empacota uma aplicação e suas dependências em um container isolado.
docker-compose	Orquestra **vários** containers descritos em um único arquivo docker-compose.yml.
Redes entre containers	Containers na *mesma* rede user-defined enxergam-se por nome e trocam dados pela camada TCP/UDP sem expor portas ao host.
Fluxo típico (backend + DB)	Navegador → container Web (app) → container DB (db) → volume (dados persistentes).

Desafio Prático

Crie dois containers com Compose:

1. Backend (Python + Flask) na porta 5000
2. Banco de Dados (PostgreSQL 15)

- **A rota GET /visitantes deve registrar e retornar o número de visitas armazenado no banco.**

Estrutura de Pastas Recomendadas

```
meu-projeto-compose/
├── app/
│   ├── requirements.txt
│   └── app.py
└── docker-compose.yml
```

Passo-a-Passo — Orientações:

Unidade Curricular – Projeto Físico de Redes
Prof. Augusto da Rosa Muniz

1- **Crie o código Flask** (app/app.py):

```
from flask import Flask, jsonify
import os, psycopg2

DB_CFG = {
   "dbname": os.environ["POSTGRES_DB"],
   "user": os.environ["POSTGRES_USER"],
   "password": os.environ["POSTGRES_PASSWORD"],
   "host": os.environ["DB_HOST"],  # resolverá para 'db'
   "port": 5432,
}

app = Flask(__name__)

def get_conn():
   return psycopg2.connect(**DB_CFG)

@app.route("/visitantes")
def visitantes():
   with get_conn() as conn:
      with conn.cursor() as cur:
         cur.execute("CREATE TABLE IF NOT EXISTS visitas (n INT);")
         cur.execute("INSERT INTO visitas VALUES (1);")
         cur.execute("SELECT SUM(n) FROM visitas;")
         total, = cur.fetchone()
   return jsonify(total_visitas=total)

if __name__ == "__main__":
   app.run(host="0.0.0.0", port=5000)
```

2- **Lista de dependências (app/requirements.txt)**

```
Flask==3.0.2
psycopg2-binary==2.9.9
```

3- **Dockerfile para a aplicação (app/Dockerfile):**

```
FROM python:3.12-slim
WORKDIR /src
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

4- **Escreva o docker-compose.yml**

   a. **Neste capítulo o aluno deve escreve o arquivo docker-compose.yml.**

5- **Critérios de Sucesso**

   a. **Build e subida: docker compose up -d –build**
   b. **Verificar containers: docker compose ps**

6- **Testar endpoint**

   a. **curl http://localhost:5000/visitantes**
   b. **# → {"total_visitas": 1}, depois 2, 3, …**

7- **Persistência**

   a. **Rode:** docker compose down
   b. **Rode:** docker compose up -d **novamente**
   c. **O contador deve continuar incrementando (graças ao volume db_data).**

8- Explicação de Cada Bloco Comandos Úteis que o Aluno Deve Saber

Bloco	Explicação em linguagem acessível
`services:`	Define **quem** vai rodar.
`db:`	PostgreSQL pronto para uso; variáveis criam usuário e banco.
`volumes:` dentro de `db`	Guarda arquivos do banco fora do container.
`networks:`	Cria rede `backend-net` onde `app` e `db` se enxergam usando DNS interno.
`depends_on:`	Garante que o container `db` inicie antes de `app`.
`ports:`	Expõe a porta 5000 do Flask para o host (navegador).

9- Comandos Úteis que o Aluno Deve Saber

```
docker compose up -d            # sobe em segundo plano
docker compose logs -f app      # segue logs do Flask
docker compose exec db psql -U guto visitasdb -c "SELECT COUNT(*) FROM visitas;"
docker compose down             # derruba containers, mantém volumes
docker compose down -v          # derruba e apaga volumes (remove dados)
```

10- O que o aluno deve demonstrar em aula

- Executa docker compose up -d --build sem erros.
- Acessa http://localhost:5000/visitantes e mostra JSON com contador aumentando.
- Explica (em suas palavras) como app alcança db (DNS interno na rede backend-net).
- Mostra que os dados persistem após docker compose down/up.
- Demonstração ao vivo: inserção → listagem → persistência.

Boa sorte, e caprichem na organização e nos detalhes do projeto!

Qualquer dúvida estou à disposição.

Prof. GutoMuniz