UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e Tecnologia

Departamento de Computação

DevOps - Prática DevOps

Professor: Dr Delano Beder

Autor

Cleidson F Almeida - 761766

São Carlos, 21 de Maio de 2025

Sumário

1. Introdução

O objetivo desta atividade foi aplicar práticas de conteinerização de aplicações utilizando Docker, com foco na composição de múltiplos contêineres integrados via Docker Compose. A aplicação escolhida simula uma livraria virtual com funcionalidades completas de CRUD (criar, ler, atualizar e deletar livros), sendo composta por três partes principais: uma interface frontend em Vue.js, uma API backend em FastAPI, e um banco de dados PostgreSQL. O projeto foi desenvolvido de forma a respeitar as boas práticas de comunicação entre contêineres por nome e separação de responsabilidades.

2. Estrutura e Tecnologias Utilizadas

A solução foi dividida nos seguintes contêineres, cada um com seu Dockerfile (exceto o banco, que utiliza imagem oficial):

- **Frontend:** Vue 3 + Vite, servido via Nginx
- **Backend:** FastAPI com SQLAlchemy e Uvicorn
- **Banco de Dados:** PostgreSQL 15

Todos os serviços são orquestrados por um arquivo docker-compose.yml, garantindo comunicação entre os serviços via nomes dos contêineres (backend, db).

3. Desenvolvimento

O backend foi implementado com FastAPI, utilizando SQLAlchemy para interação com o banco PostgreSQL. A aplicação possui rotas REST para listar, adicionar, editar e excluir livros, com validação via Pydantic. O frontend foi desenvolvido com Vue 3 e Vite, gerando uma SPA (Single Page Application) servida via Nginx. Um proxy reverso foi configurado no Nginx para encaminhar chamadas /api/* ao backend, evitando o uso de localhost e respeitando a arquitetura de rede Docker. Ao iniciar a aplicação, um script de inicialização (seed.py) popula o banco com 10 livros, permitindo testes imediatos.

4. Execução e Testes

4.1. Instalação

1. git clone https://github.com/cleidsonfalmeida/DevOps-Studies.git
2. cd livraria-docker
3. docker compose up --build -d

4.2. Acessos

- Frontend: http://localhost:8080
- API (Swagger): http://localhost:8000/docs

4.3. Estrutura do Projeto

```
livraria-docker/
├── backend/
│   ├── app/
│   ├── Dockerfile
├── frontend/
│   ├── src/
│   ├── public/favicon.png
│   ├── nginx.conf
│   ├── Dockerfile
├── docker-compose.yml
└── README.md
```

5. Conclusão

Esta prática permitiu compreender o fluxo completo de desenvolvimento e deploy de uma aplicação moderna em múltiplos contêineres. Utilizando ferramentas como FastAPI, Vue, PostgreSQL e Nginx, foi possível construir uma solução web funcional, portável e organizada. O uso de docker-compose facilitou o gerenciamento e integração dos serviços, reforçando a importância de boas práticas DevOps desde a fase de desenvolvimento.

6. Referências

- Documentações oficiais do Docker, Docker Hub, FastAPI, Vue.js e PostgreSQL
- Slides e materiais da disciplina