UNIVERSIDADE FEDERAL DE SÃO CARLOS
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação

DEVOPS 2025/1
Turma A

TodoApi DevOps - Descrição da Aplicação
Prática DevOps

Nome: Pedro Cassiano Coleone
RA: 793249

São Carlos - SP

1. Visão Geral da Aplicação

A aplicação **TodoApi DevOps** é um sistema simples para gerenciamento de tarefas (*to-do list*), que permite criar, listar, editar e excluir tarefas. Foi desenvolvida utilizando arquitetura de múltiplos contêineres Docker, com backend em **ASP.NET Core 8.0** e frontend em **React**. O sistema utiliza um banco de dados **PostgreSQL** para armazenamento persistente.

Esta aplicação atende à atividade acadêmica de containerização, demonstrando a integração e comunicação entre múltiplos contêineres em um ambiente orquestrado via **Docker Compose**.

---

2. Arquitetura e Contêineres Utilizados

2.1 Contêiner Backend

- Implementado em **ASP.NET Core 8.0**
- Responsável por expor a **API REST** para operações **CRUD** de tarefas
- Baseado na imagem oficial: **mcr.microsoft.com/dotnet/aspnet:8.0**
- Contêiner nomeado: **backend**
- Comunicação interna com frontend e banco de dados via nomes de serviço

2.2 Contêiner Frontend

- Implementado em **React**
- Responsável pela interface web que consome a API do backend
- Utiliza **Node.js** para build e **Nginx** para servir arquivos estáticos

O **Nginx**, neste caso, foi utilizado para servir os arquivos estáticos do frontend e atuar como proxy reverso para o backend, permitindo o encaminhamento das requisições da interface para a API. Dessa forma, **evita-se problemas de CORS**, centralizando o acesso na mesma origem.

- Baseado nas imagens oficiais: **node:18-alpine** & **nginx:alpine**
- Contêiner nomeado: **frontend**

2.3 Contêiner Database

- Banco de dados: **PostgreSQL 15**
- Responsável pelo armazenamento das tarefas
- Baseado na imagem oficial: **postgres:15-alpine**
- Volume nomeado: **database** (para persistência dos dados)
- **Observação:** não possui código-fonte próprio, apenas configuração no **docker-compose.yml**.

---

3. Tecnologias Utilizadas

- **Backend:** ASP.NET Core 8.0
- **Frontend:** React 18 com Node.js e Nginx
- **Banco de Dados:** PostgreSQL 15
- **Orquestração:** Docker Compose

---

4. Manual de Instalação e Execução

Requisitos

- Docker e Docker Compose instalados
- Sistema operacional: Windows, Linux ou macOS
- Acesso com permissão para execução de comandos Docker

Passos para rodar a aplicação

Clone o repositório do projeto:

```shell
git clone https://github.com/pedro-coleone/TodoApi-devops.git
cd TodoApi-devops
```

Execute o comando para subir os contêineres:

```shell
docker-compose up --build
```

1. **Acesse o frontend no navegador:**
   http://localhost:3000

2. **A API backend estará disponível em:**
   http://localhost:5000/api

Parar os contêineres

Para interromper a execução e remover os contêineres:

```shell
docker-compose down
```

---

5. Estrutura do Repositório

```
/TodoApi-devops
  /backend
        Código-fonte backend (.NET)
        Dockerfile
  /frontend
        Código-fonte frontend (React)
        Dockerfile
        nginx.conf
  /database
        (não possui código, apenas configuração no docker-compose.yml)
  docker-compose.yml
  Descrição da Aplicação (este documento em PDF)
  README.md (tutorial para implementação)
```

---

6. Observações e Considerações Finais

- A comunicação entre os contêineres é feita via **nomes de serviços** definidos no **docker-compose.yml**, respeitando o requisito de **não usar localhost** para intercomunicação entre contêineres.

- O contêiner de banco de dados **não possui código-fonte**, pois utiliza o serviço oficial do **PostgreSQL**, configurado no **Docker Compose** com volume para persistência.

- O projeto foi desenvolvido para **fins acadêmicos**, demonstrando o uso prático de **Docker**, **Docker Compose** e **multi-contêineres** para uma aplicação web completa.

- Possibilidade futura de extensão para ambientes de produção utilizando **Kubernetes**.