Trabalho 2 - Redes de Computadores 2023/2 - Turma 2

Lucas de Almeida Abreu Faria - 170016668
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
lucasaafaria@gmail.com

***Abstract*—This document is a report of the second practical project of the computer networks subject from Universidade de Brasília, which had the goal of exploring the use of containers in creating applications. The author chose to implement the base of an e-commerce app, using the containers to run an nginx server, a node server, a mysql database and a phpMyAdmin admin panel.**

***Index Terms*—Containers, Docker, docker-compose, web app, e-commerce**

I. Introdução

Este relatório visa documentar o desenvolvimento do segundo projeto prático da disciplina de Redes de Computadores da Universidade de Brasília, ministrada pelo Professor Gabriel Ferreira. O projeto se trata da criação de um arquivo *docker-compose.yml* contendo as configurações para 4 serviços ou interdependentes entre si, como fariam em uma aplicação comercial.

Para este projeto, o autor optou por utilizar serviços presentes na construção de e-commerces, visto que aplicações desse tipo são amplamente utilizadas no mercado e representam uma parcela extremamente significativa das compras realizadas no mundo, movimentando anualmente bilhões de dólares.

Os 4 serviços escolhidos foram:

- Um servidor nginx, para poder servir os arquivos frontend do projeto
- Um servidor Nodejs utilizando o framework express, para realizar a comunicação com o banco de dados e prover os endpoints necessários
- Um banco de dados MySql, contendo os dados dos produtos vendidos pela aplicação
- Painel de gerenciamento phpMyAdmin, que permite o controle do banco de dados via interface gráfica no browser

II. Detalhamento dos Serviços

A. Frontend

Para servir o frontend do e-commerce, utilizou-se um servidor nginx que retorna o arquivo INDEX.HTML para requisições feitas ao root (”/”) da aplicação. No arquivo INDEX.HTML, temos a definição dos estilos utilizados na página, da estrutura dos elementos HTML, e também código javascript que faz um requisição HTTP para o endpoint /PRODUCTS do servidor web com o intuito de receber os dados dos produtos, que são então mostrados na tela para o usuário.

Configuração do frontend no *docker-compose.yml*:

```yaml
frontend:
  image: nginx:latest
  ports:
    - "80:80"
  volumes:
    - ./frontend:/usr/share/nginx/html
  depends_on:
    - backend
```

Fig. 1. Configuração do servidor nginx

Código de requisição dos produtos e display na interface:

```javascript
document.addEventListener('DOMContentLoaded', () => {
  const productList = document.getElementById('productList');

  fetch('http://localhost:3000/products')
    .then(response => response.json())
    .then(products => {
      products.forEach(product => {
        const productItem = document.createElement('li');
        productItem.classList.add('product-item');
        productItem.innerHTML = `
          <h3>${product.name}</h3>
          <p>Descrição: ${product.description}</p>
          <p>Preço: $${product.price}</p>
          <p>Quantidade: ${product.quantity}</p>
        `;
        productList.appendChild(productItem);
      });
    })
    .catch(error => {
      console.error('Error fetching products:', error);
    });
});
```

Fig. 2. Display dinâmico de produtos no Frontend

B. Backend

Foi escolhido para o backend um servidor Nodejs com apoio do framework express para criar as rotas/endpoints da aplicação e lidar com a comunicação com o banco de dados MySql, utilizando a biblioteca MYSQL2.

Configuração do backend para o container gerado:

```yaml
backend:
  build: ./backend
  ports:
    - "3000:3000"
  environment:
    - NODE_ENV=production
  depends_on:
    - database
  restart: always
```

Fig. 3. Configuração do servidor Nodejs no docker-compose.yml

```
FROM node:14
WORKDIR /usr/src/app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
CMD ["npm", "start"]
```

Fig. 4. Configuração do servidor Nodejs no Dockerfile

Foi definido um endpoint /PRODUCTS que deve retornar ao cliente a lista de todos os produtos registrados no banco MySql, e foi determinado que a porta de execução do servidor é a de número 3000, então para confirmar o funcionamento do servidor backend, pode-se acessar (após a inicialização dos containers) a url http://localhost:3000/products. Essa ação deve mostrar na tela um JSON com as informações dos produtos.

C. Banco de Dados

Para esse projeto, utilizou-se um banco de dados MySql com o intuito de armazenar os dados dos produtos a serem vendidos no e-commerce.

O container desse serviço foi configurado de tal forma que, após sua inicialização, automaticamente um script SQL é executado para criar a Database no banco (caso seja a primeira execução), criar a tabela PRODUCTS caso ela ainda não exista, e por fim adicionar os dados dos produtos.

Configuração do servidor MySql no *docker-compose.yml*:

```yaml
database:
  image: mysql:latest
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: admin
    MYSQL_DATABASE: ecommerce
    MYSQL_USER: root
    MYSQL_PASSWORD: admin
  command: --init-file /docker-entrypoint-initdb.d/init.sql
  volumes:
    - ./mysql-init:/docker-entrypoint-initdb.d
```

Fig. 5. Configuração do servidor de banco de dados

D. Painel Admin

Por fim, o último serviço escolhido foi o phpMyAdmin, que provê uma interface gráfica via browser para gerenciamento do banco de dados, eliminando assim a necessidade de usar a linha de comando ou um SGBD como MySQL Workbench. Esse tipo de painel de administrador é muito comum no mercado para que pessoas que não são da área de tecnologia possam gerenciar o estoque de produtos de maneira mais intuitiva e amigável.

Configuração do phpMyAdmin no *docker-compose.yml*:

```yaml
admin_panel:
  image: phpmyadmin/phpmyadmin:latest
  ports:
    - "8080:80"
  environment:
    PMA_HOST: database
    MYSQL_ROOT_PASSWORD: admin
```

Fig. 6. Configuração do servidor de banco de dados

III. CONCLUSÃO

O projeto foi uma excelente oportunidade para exercitar os conceitos de conteinerização e virtualização, e a aplicação desenvolvida funcionou corretamente para os requisitos propostos. Um vídeo explicando em maiores detalhes o código produzido e a execução do programa pode ser acessado em https://www.youtube.com/watch?v=unlUWgyPHxQ. O código fonte do projeto está disponível em https://github.com/lucasaafaria/Trabalhos-Redes-de-Computadores

REFERENCES

[1] Docker Official Documentation, disponível em https://docs.docker.com/get-started/, acessado em 17/12/2023

[2] G. C. Ferreira, documentação própria disponível em https://gabrielcarvfer.github.io/ORAN_testbed_docs/container-tools.html, acessado em 17/12/2023.