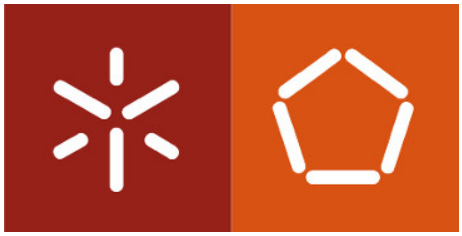
Virtualização de Redes

23 de Junho de 2021

TP2

a83899 André Morais

Docker Microservices



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1 Introdução

Neste trabalho prático, o objetivo era implementar um serviço de autenticação e um *file server* e a posteriori, dar *deploy* da aplicação num *docker container*.

Nestas próximas secções, aquilo que pretendo é explicar cada um dos passos para a conceção deste projeto e para a sua estruturação.

2 Conceção/Desenho da resolução

Na Figura seguinte, aprenseta-se um pequeno esquema de como está estruturado este projeto

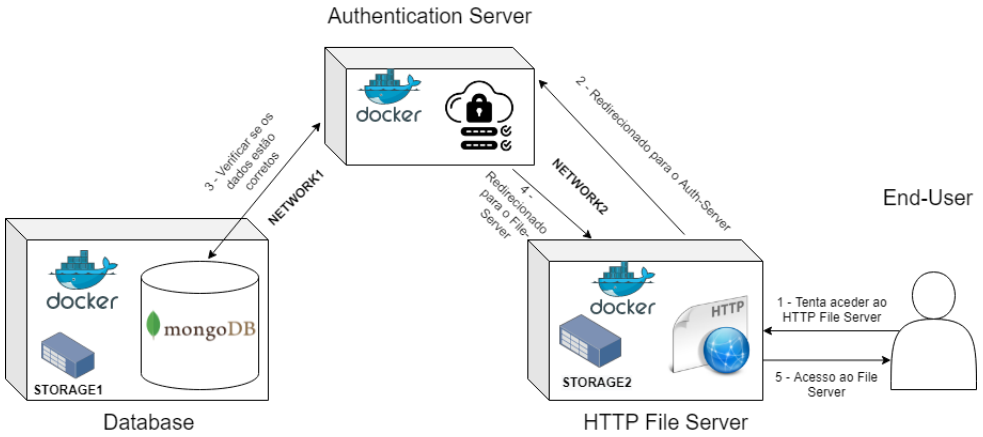


Figura 1: Modelo do Sistema

2.1 Descrição da arquitetura

Através do modelo do sistema podemos observar que existem 3 componentes principais, sendo estas:

- **Database**: Base de dados em **MongoDB**, que armazena os dados dos utilizadores que criam conta no *Authentication Server*.
- **Authentication Server**: Programado em **ExpressJS** com **PUG** para a interface gráfica. É um servidor que fornece uma página de login,de registar e de autenticação para redirecionar para o *File Server*.
- **HTTP File Server**: Programa em **Javascript**. Servidor de ficheiros que redireciona para o servidor de autenticação se este nao tiver um token.

Para além destas componentes há também duas netowrks diferentes: a **Network1** que liga o mongo ao *Auth Server* e a **Network2** que liga o *Auth Server* ao *File Server*.

3 Authentication Server

Neste servidor, são tratados inicialmente os pedidos de login, se já tiver conta. Em caso contrário, é necessário registar-se primeiro, sem necessidade de token. Quando os dados de um login são inseridos corretamente é guardado um token e redirecionado para uma página para confirmação de autorização, como mostra a figura 2.

Para a geração do *token* é usado o padrão *JSON Web Token*, onde está presente o id de um utilizador, que neste caso é o email (é cifrado para não haver fuga de informção). Este token tem uma duração de 30 minutos.

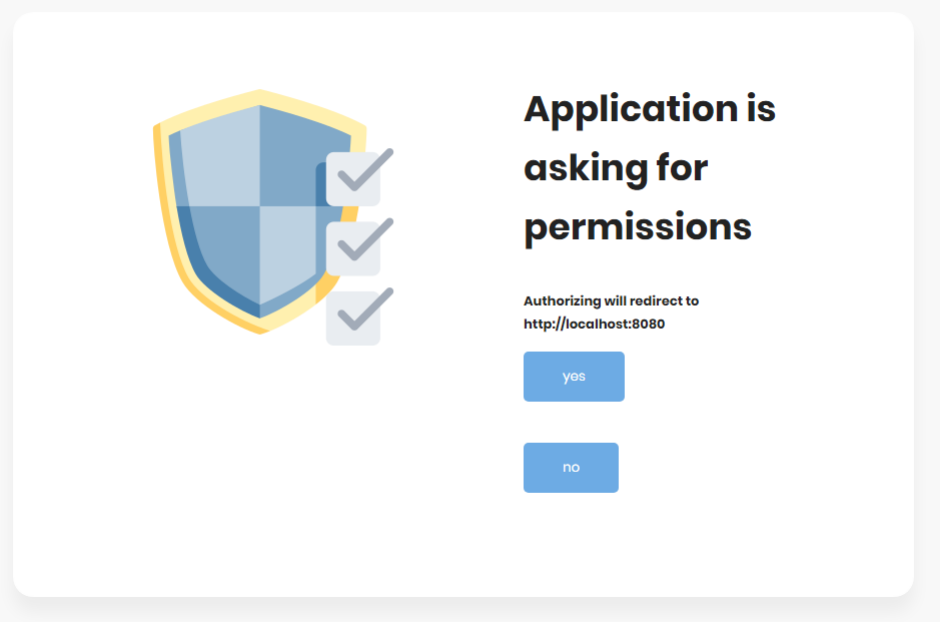


Figura 2: Página de Permissão

4 HTTP File Server

Neste servidor, os pedidos têm de possuir um token válido, para que este consiga proceder naturalmente. Se não houver token, este servidor redireciona para o server de autenticação para o utlizador conseguir dar login.

Figura 3: File Server

5 Base de dados

Na Base de dados apenas é guardado o email e a respetiva password como pode ver no exemplo seguinte

```
[
    {
        "email" : "andre@gmail.com",
        "password" : "123"
    }
]
```

6 Docker Containers

As aplicações instaladas nos container são criadas através de *docker files*, com o objetivo de que *docker-compose.yml* possa ser excutável em qualquer computador e reproduzindo a arquitetura com a sua própria *network* e *volumes* estabelecidos.

6.1 DockerFiles

Um dockerfile é um documento texto que contem todos os comandos que um utilizador pode usar na linha de comandos para criar uma imagem, neste caso de duas *web apps*

6.1.1 Authentication Server

```
FROM node:14

WORKDIR /usr/tp2/auth_server

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 7777
CMD [ "node", "./bin/www" ]
```

6.1.2 HTTP File Server

```
FROM node:14

WORKDIR /usr/tp2/http-file-server

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8080
CMD [ "node", "./bin/http-server", "../http-file-server
/home/server/"]
```

6.2 Docker-Compose

Vão ser criados os tais 3 serviços: o **MongoDB**, o **Auth Server** e o **File Server**. Para além disso foram criadas duas networks e duas storages. Como ja foi falado anteriormente, a **network1** comunica através do **mongoose** com o *Auth Server* e a **network2** através de pedidos http. Os storage é para armazenar os dados da base de dados e dos ficheiros, respetivamente

```
version: "3"

services:
    mongo:
        image: mongo
        container_name: mongo
        volumes:
            - storage1:/home
        networks:
            - network1
    auth-server:
        build: ./auth-server
        container_name: auth-server
        networks:
            - network1
            - network2
        ports:
            - '7777:7777'
    http-file-server:
        build:  ./http-file-server
        container_name: http-file-server
        volumes:
            - storage2:/home
            - ../:/usr/tp2/http-file-server/home/server
        networks:
            - network2
        ports:
            - '8080:8080'
volumes:
    storage1:
    storage2:
networks:
    network1:
    network2:
```

7 Conclusão

Em suma, a realização deste projeto permitiu-me consolidar a aprendizagem da UC de Virtualização de Redes assim como utilizar e perceber a importante função dos *dockers containers*.

O objetivo principal do trabalho foi alcançado apesar de um ou outro ponto expostos no enunciado nao foram totalmente abordados.