Perfil EL - Engenharia de Linguagens
(1º ano do MEI)
Representação e Processamento de Conhecimento na Web

Projeto Final

Relatório de Desenvolvimento

João Gonçalves
PG46535

Sara Queirós
PG47661

30 de junho de 2022

Conteúdo

Capítulo 1

Introdução

Neste projecto, foi desenvolvida uma aplicação Web que implementa e dá suporte a um repositório de recursos didáticos. A aplicação possui algumas das caraterísticas de uma aplicação web: autenticação de utilizadores, informação pública e privada, possibilidade de comentar os recursos disponíveis, etc. Através desta aplicação, um professor ou qualquer entidade com necessidades semelhantes, pode publicar conteúdos de forma organizada e de fácil acesso. Desta forma, um aluno/cliente poderá ter acesso a esses recursos desque que esteja ligado à internet, podendo ainda deixar avaliações e comentários sobre os mesmos.
Paralelo a isto, existe uma interface que permite a um administrador de sistema gerir recursos e utilizadores.

Capítulo 2

Arquitetura da Aplicação

O serviço que a aplicação fornece é conseguido através de 3 componentes distintas:

1. API de dados.
2. Aplicação.
3. Servidor de Autenticação.

Estes componentes estão conectados a uma base de dados MongoDB que garante a integridade dos dados necessários para o correto funcionamento da aplicação.
Toda a aplicação foi desenvolvida usando a framework **Express** do **Node.js**, permitindo utilizar a mesma linguagem de programação em todo o stack do projeto. De modo a atender ao critério de geração dinâmica de páginas web, dependendo do conteúdo a ser apresentado, as views foram desenvolvidas em **Pug**, com recurso a icons **Bootstrap** empregados em conjunto com **W3.css**.

2.1 Utilizadores

2.1.1 User

O user é o utilizador que toma o papel de "consumidor"dos recursos da aplicação. Este utilizador pode vizualizar e descarregar os recursos presentes no RRD, desde que possua as permissões corretas para o efeito.

2.1.2 Producer

O producer é o utilizador que possui permissões para carregar ficheiros na aplicação.

2.1.3 Admin

O admin é o utilizador com maior nível de acesso na aplicação, sendo o único que pode fazer gestão dos restantes utilizadores, assim como gerir recursos disponíveis na aplicação.

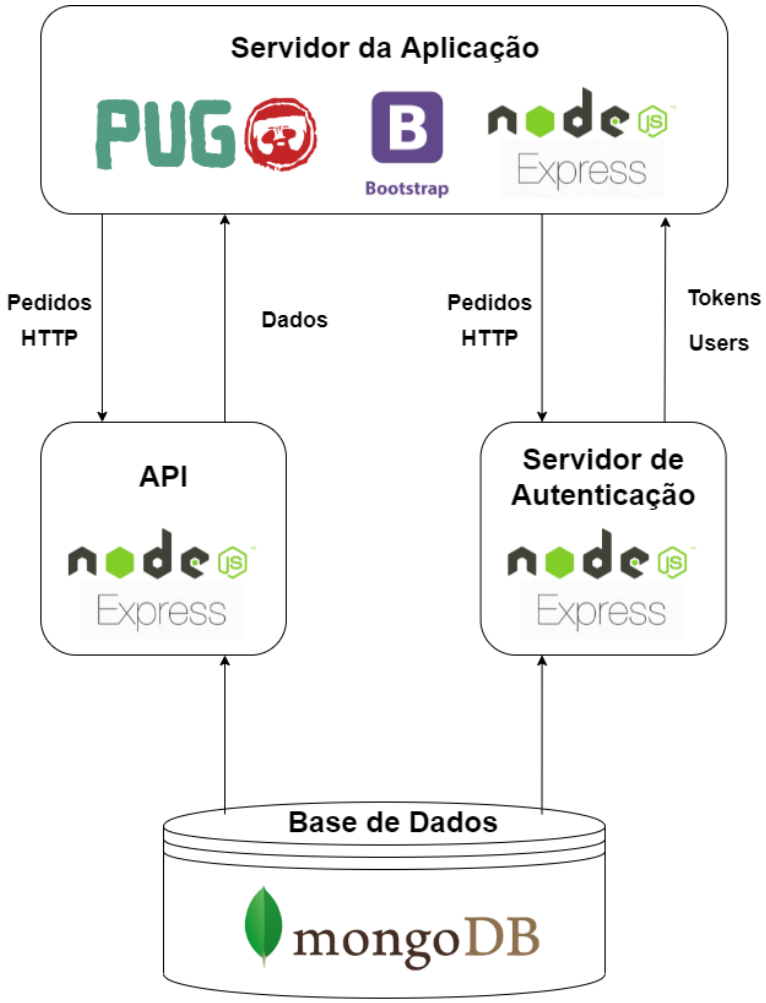

Figura 2.1: Arquitetura da Aplicação

2.2 Componentes

2.2.1 Base de Dados

A base de dados que dá suporte à aplicação é do tipo NoSQL, permitindo uma fácil gestão de metadados em formato JSON e simplificando processos que seriam desnecessariamente complexos caso se tratasse de uma Base de Dados relacional. Nesta aplicação, a base de dados RRD2022 dá suporte à API e ao Servidor de Autenticação através das coleções **recursos** e **users** respetivamente.

2.2.2 Servidor de Autenticação

Este servidor quando iniciado, fica ativo na porta 8002, sendo o responsável pela autenticação dos utilizadores que efetuam login na aplicação, fornecendo também informações relativas aos níveis de permissão que cada utilizador possui, permitindo que o servidor de aplicação faça a gestão da informação à qual cada utilizador tem acesso.
Estes processos ocorrem com recurso a pedidos HTTP por parte do servidor da aplicação, cujas respostas consistem em tokens JWT, que serão processados do lado a aplicação.
O servidor acede à coleção **users** da base de dados através do *mongoose*, utilizando o *Schema* definido nos *models* com o nome **user.js**:

```
const mongoose = require('mongoose')

var userSchema = new mongoose.Schema({
    username: { type: String, required: true },
    password: { type: String, required: true },
    level: { type: String, required: true }
});

module.exports = mongoose.model('user', userSchema)
```

2.2.3 API

Este servidor quando iniciado, fica ativo na porta 8001, sendo o responsável pelo fornecimento dos dados relativos aos recursos que aplicação disponibiliza, sendo por isso a ligação da aplicação com a Base de Dados e com o sitema de ficheiros que permite o armazenamento dos recursos.
É neste servidor que se encontra o sistema de ficheiros, dividindo os recursos que são carregados na aplicação por utilizadores, facilitando o acesso e gestão desses mesmos recursos.
O servidor acede à coleção **recursos** da base de dados através do *mongoose*, utilizando o *Schema* definido nos *models* com o nome **recurso.js**:

```
const mongoose = require('mongoose')

var recursoSchema = new mongoose.Schema({
    criacao: String,
```

```
    submissao: String,
    produtor: String,
    submissor: String,
    titulo: String,
    tipo: String,
    desc: String,
    size: String,
    path: String,
    downloads: String,
    comentarios: [{
        comm: String,
        user: String,
        aval: String,
    }]

});

module.exports = mongoose.model('recurso', recursoSchema)
```

2.2.4 Servidor da Aplicação

Este servidor quando iniciado, fica ativo na porta 8003, sendo o responsável pela gestão dos interfaces que vão ser gerados, assim como a que utilizadores estes devem ser apresentados. É neste servidor que os utilizadores interagem com os recursos da aplicação, podendo efetuar o carregamento, descarregamento e modificação de recursos, dependendo das suas permissões. É também neste servidor que um Administrador pode efetuar a gestão de recursos e utilizadores.
Para isso, este servidor realiza pedidos à API de dados para obter as informações relativas aos recursos disponíveis na aplicação, gerando com esses pedidos páginas dinâmicas **Pug** que apresentam os recursos aos utilizadores. Um produtor pode realizar *POSTs* na API de dados, permitindo o carregamento de ficheiros na aplicação. Para além disto, os utilizadores podem efetuar avaliações e deixar comentários nos recursos.
No que diz respeito à gestão de utilizadores, apenas utlizadores do tipo Admin podem realizar essa gestão, sendo que as restantes informações sobre os utilizadores são usadas internamente pela própria aplicação para determinar se um utilizador com login efetuado tem ou não acesso a determinada página ou recurso.

Capítulo 3

Recursos Disponíveis

GET /api/recursos	Devolve um array em JSON com a informacao dos recursos
GET /api/recurso/:id	Apresenta as informações sobre um recurso específico
DEL /api/recursos/eliminar/:id	Elimina um recurso específico
GET /api/recursos?tipo=X	Devolve um array em JSON com a informacao dos recursos do tipoX;
GET /api/recursos?q=pal	Devolve um array em JSON com a informacao dos recursos que contem pal no titulo
POST /api/recursos	Regista um recurso no repositorio
POST /login	Permite efetuar o Login
POST /registar	Permite que um novo utilizador se registe
GET /logout	Permite efetuar Logout
POST /editarPass/:id	Permite editar a palavra-passe de um utilizador
POST /editarPerm/:id	Permite editar as permissões associadas a um utilizador

Capítulo 4

Execução

Para ser possível executar o programa, é necessário inicializar os 3 servidores da seguinte forma:

```
npm start
```

Ao consultar a página principal em 'localhost:8003/api', é possível , através da interface gráfica aceder a todas as funcionalidades que o programa possui, de forma intuitiva.

Caso se pretenda inicializar o programa com informação relativa a utilizadores, basta introduzir esses dados na Base de Dados Mongo, através de um ficheiro JSON, na DB RRD2022, na collections 'users'. Para tal sugere-se o seguinte comando:

```
mongoimport -d RRD2022 -c users ficheiro.json
```

Capítulo 5

Conclusão

Após a elaboração deste projeto, podemos dizer, como balanço geral, que grande parte do conhecimento obtido ao longo desta UC foi aplicado, desde o emprego de autenticação, ao uso de views geradas dinâmicamente e à comunicação entre servidores.
A escala do projeto permitiu não só conjugar os diversos conceitos aprendidos em RPCW, como também perceber em parte os processos que hoje em dia são utilizados no desenvolvimento de aplicações Web.
Contudo, ficaram algumas partes por implementar, sendo elas a gestão de ficheiros segundo o modelo OASIS e o gestor de notícias e notificações, que iriam acrescentar funcionalidades importantes ao projeto.