

Trabalho Prático

Programação na Internet

Grupo 6

Trabalho realizado por:

Daniel Nunes N° 50882

João Pires N° 50876

Professor Filipe Freitas

Lisboa, 5 de janeiro 2025

Índice

Índice	2
Introdução	3
Configuração do Projeto	4
Instalação de Dependências.....	4
Configuração do Elasticsearch.....	4
Criação dos Índices do Elasticsearch	4
Execução da aplicação	4
Estrutura da aplicação	5
Server-side.....	5
Servidor ExpressJS	5
Camada Web	5
Autenticação	5
Camada de Serviços.....	6
Camada de Dados	6
Módulo de erros	6
Client-side.....	6
Estrutura de dados	8
Conclusão.....	9

Introdução

No âmbito da unidade curricular de Programação na Internet, este trabalho aborda conceitos sobre o desenvolvimento de aplicações web e APIs.

A aplicação FOCCACIA (**FOotball Complete Clubs API and Chelas Internet Application**) foi desenvolvida ao longo do semestre e permite pesquisar equipas de futebol (através da *Football API*), bem como as ligas em que participaram nas demais temporadas. Permite ainda, para utilizadores autenticados, a organização de equipas em grupos.

A aplicação fornece uma API para gerir os grupos de um utilizador, pesquisar equipas e ligas e criar utilizadores, a aplicação fornece ainda uma interface gráfica que permite interagir com todas as funcionalidades da API.

A API foi documentada utilizando o padrão da *OpenAPI* e contém uma descrição detalhada dos *endpoints*, essa documentação está disponível no ficheiro [docs/foccacia-api-spec.json](#). É possível também ver exemplos de pedidos à API no ficheiro [docs/foccacia-api-requests.http](#).

Configuração do Projeto

Como referido anteriormente, a aplicação foi desenvolvida utilizando a linguagem *JavaScript* e o ambiente de execução [Node.JS](#), para o armazenamento dos dados foi utilizado o [Elasticsearch](#). Ambos devem ser instalados, antes de iniciar a configuração do projeto.

Instalação de Dependências

Primeiramente devem ser instaladas as dependências do projeto utilizando um gestor de pacotes como o [npm](#) que vem instalado junto com o Node.JS. Para isso utilizamos o seguinte comando:

```
npm install
```

Configuração do Elasticsearch

Depois de transferido o *Elasticsearch*, deve ser aberto o ficheiro de configuração [config/elasticsearch.yml](#) e desativar todas as entradas [xpack.security](#). Para iniciar o *Elasticsearch*, executamos o programa [bin/elasticsearch](#) através da linha de comandos.

Criação dos Índices do Elasticsearch

Para poderem ser armazenados dados no *Elasticsearch*, é necessário criar os índices que o FOCCACIA irá utilizar para guardas as informações (os índices assemelham-se a tabelas SQL). No ficheiro [docs/foccacia-elasticsearch.http](#) encontram-se os pedidos HTTP que permitem criar esses índices, bem como outros pedidos para controlar os dados armazenados no *Elasticsearch*.

Execução da aplicação

Para executar a aplicação utiliza-se o comando:

```
npm run start
```

Para executar os testes utiliza-se o comando:

```
npm run test
```

Estrutura da aplicação

A aplicação está dividida em duas grandes componentes: o lado do servidor (gere a API e a UI) e o lado do cliente (as páginas web apresentadas ao utilizador).

Server-side

O lado do servidor é dividido em 6 partes:

- Servidor *ExpressJS*;
- Camada web (API e UI);
- Autenticação (Configuração do *PassportJS*);
- Camada de serviços;
- Camada de dados (*Elasticsearch* e *Football API*);
- Módulo de erros.

Servidor ExpressJS

foccacia-server.js

O servidor é a parte principal da aplicação e é a partir deste que todas as outras componentes são interligadas e montadas (utilizando injeção de dependências).

Consiste num servidor *ExpressJS* que utiliza *PassportJS* para autenticação e *handlebars* para a definir o layout das páginas web (apresentadas ao cliente).

Camada Web

foccacia-web-api.js

foccacia-web-ui.js

Contém as funções que tratam de todos os pedidos feitos aos *endpoints* da REST API e do website. Servem como “pontes” entre o servidor e os serviços.

Autenticação

passport-config.js

Contém as configurações necessárias para o funcionamento do *PassportJS*.

Camada de Serviços

foccacia-services.js

Implementação lógica das funcionalidades da aplicação (criação de grupos e utilizadores, inserção de equipas em grupos, etc.), inclui também verificação e validação de dados.

Camada de Dados

foccacia-elastic.js

fapi-teams-data.js

A camada de dados é composta por duas partes: o módulo dos dados da aplicação (onde os grupos e utilizadores são armazenados) e o módulo para pedidos à *Football API* (para realizar pesquisas de equipas, ligas e temporadas).

Ambos os módulos oferecem uma variante que é utilizada nos testes unitários, estas variantes são:

foccacia-data-mem.js – Oferece a opção de guardar dados em memória ao invés de utilizar uma base de dados.

fapi-teams-data-fake.js – Contém respostas a algumas dos pedidos à *Football API*, diminuindo a quantidade de pedidos efetuados.

Módulo de erros

Módulo que contém todos os erros possíveis da API e que constrói as respostas da mesma.

Client-side

O lado do cliente contém tudo o que é apresentado e servido ao utilizador. Existem 2 diretórias principais:

- **views/** – Onde estão localizadas todas as páginas e os componentes que as compõem;
- **public/** – Contém todos os ficheiros extras, necessários para as páginas web (no caso da aplicação, as folhas de estilo CSS).

A diretoria **views/**, como referido anteriormente, contém os templates para todas as páginas web apresentadas pela aplicação. Essas páginas muitas vezes contêm elementos comuns entre elas, por isso mesmo, a diretoria **views/** foi dividida em 4 subdiretorias:

- **pages/** – Contem os templates das páginas a ser mostradas ao utilizador;
- **componentes/** – Inclui as *partials* que são comuns a várias páginas (ex. botões, caixas de texto, etc.);
- **common/** – Inclui as *partials* que são comuns a várias páginas, mas que não são necessariamente componentes (ex. conteúdo da tag **<head>**);
- **icons/** – Inclui os ícones utilizados nas páginas da aplicação.

Estrutura de dados

Para a persistência dos dados dos utilizadores na aplicação, utilizamos o *Elasticsearch*, um sistema de pesquisa distribuído capaz de gerir grandes quantidades de dados. Esses dados são guardados em índices (semelhantes a tabelas em SQL), no caso da aplicação FOCCACIA, utilizamos 2 índices: `users` e `groups`.

Abaixo encontram-se os modelos de dados de ambos os índices:

users	
username	<i>string</i>
password	<i>string</i>
token	<i>string</i>

groups	
name	<i>string</i>
description	<i>string</i>
userId	<i>string</i>
teams	<i>Team[]</i>

Uma equipa é composta por:

- **name** (*string*) – Nome da equipa;
- **leagueld** (*number*) – ID da liga;
- **league** (*string*) – Nome da liga;
- **season** (*number*) – Ano da temporada;
- **stadium** (*string*) – Nome do estádio da equipa;
- **logo** (*string*) – URL para o logotipo da equipa.

Os modelos de dados utilizados pela aplicação assemelham-se aos modelos dos dados no *Elasticsearch*, exceto na obtenção de grupos (onde o campo `userId` é removido) e em operações sobre utilizadores como o registo e início de sessão (onde o campo `password` é removido).

Conclusão

A aplicação FOCCACIA atingiu os objetivos previamente estabelecidos, a mesma tem uma interface de utilizador intuitiva e reativa que facilita a criação e gestão de grupos.

A API foi documentada usando o formato *OpenAPI* o que facilita os desenvolvedores na integração da mesma nas suas aplicações. Verificou-se ainda o funcionamento da API através de testes unitários ao serviço, facilitando a manutenção do projeto.

A autenticação é efetuada com a ajuda do *PassportJS* que facilita a implementação de um sistema de início de sessão com persistência de sessão.

Existem possíveis melhoramentos que podem ser feitos na aplicação:

- Implementação de um sistema de cache de forma a reduzir os pedidos à *Football API*;
- Página de utilizador para controlo de conta (alteração de palavra-passe, apagar conta, etc.);
- Melhorar o *website* de forma a poder ser visto em ecrãs mais pequenos (*smartphones* e *tablets*);
- Adicionar mais temas ao *website* (nomeadamente o modo escuro);
- Melhorar a acessibilidade do site para utilizadores sem rato.