Representação e Processamento
de Conhecimento na Web (1º ano de MEI)

Trabalho Prático

Relatório de Desenvolvimento

Ana Sofia Gomes
(PG47003)

Pedro Pereira
(A80376)

19 de junho de 2022

Conteúdo

Capítulo 1

Introdução

No âmbito da unidade curricular Representação e Processamento de Conhecimento na Web, o docente desafiou os alunos a edificarem uma aplicação Web que implementasse e desse suporte a um repositório de recursos didáticos. Apesar disso, permitiu que estes selecionassem uma temática distinta.

Assim, optou-se por desenvolver uma aplicação à semelhança do IMDb e da Letterboxd que permitisse aos utilizadores consultar as informações de uma panóplia diversificada de filmes, bem como anotar e partilhar a sua opinião acerca dos mesmos. Esta foi construída em JavaScript, recorrendo à *framework* Express.

Estrutura do Relatório

O presente relatório explicita detalhadamente o processo de desenvolvimento do projeto e a abordagem escolhida para solucionar o problema em mãos.

Inicialmente, em 2.1 são apresentados os requisitos que foram considerados como essenciais para a aplicação.

Depois em 2.2 é discutida a estrutura utilizada para o armazenamento de dados da base de dados e a maneira como esta foi povoada. De seguida são expostos os servidores que servem de API de dados e de autenticação, também como a sua forma de utilização.

No capitulo 2.5 são descritos os vários tipos de utilizadores da aplicação e as varias funcionalidades a que tem acesso.

Por ultimo é feito um pequeno guia sobre como instalar a aplicação desenvolvida utilizando *Docker*.

Capítulo 2

Conceção/Desenho da Resolução

2.1 Requisitos

Dado que se escolheu uma temática distinta da proposta, começou-se por definir um conjunto de requisitos que deveriam ser assegurados pelo sistema. É de notar que grande parte destes são uma adaptação daqueles que se encontravam no enunciado disponibilizado.

Primeiramente, estipulou-se que a aplicação teria de interagir com três tipos diferentes de atores, sendo estes:

- **Consumidor:** Utilizadores que não se encontram autenticados no sistema. Estes têm acesso às seguintes funcionalidades:
  - Consultar notícias relativas ao mundo cinematográfico;
  - Consultar a lista de todos os filmes presentes no catálogo;
  - Realizar pesquisa de filmes por título;
  - Consultar a informação de cada um dos filmes, bem como a opinião dos utilizadores relativamente aos mesmos;
  - Criar uma conta.
- **Produtor:** Utilizadores que se encontram autenticados no sistema, mas que não têm permissões de administrador. Para além das ações acima explicitadas, estes podem também:
  - Registar a sua opinião relativamente a um determinado filme;
  - Consultar o seu perfil. Este é composto por uma secção dedicada às informações pessoais e outra reservada aos *logs*[1] registadas, bem como por uma foto de perfil;
  - Editar o seu perfil;

[1]Um log é a opinião de um dado utilizador relativamente a um filme. Este é composto por uma cotação, um comentário e uma indicação de se o filme pertence ou não aos filmes favoritos do utilizador.

  - Importar e exportar os seus *logs*.

- **Administrador:** Utilizadores autenticados no sistema que têm permissões de administrador. Estes dispõem de todas as funcionalidades enumeradas anteriormente, contudo têm também acesso a uma área reservada na qual podem:
  - Administrar os utilizadores:
    * Consultar listagem dos utilizadores;
    * Criar, editar e remover utilizadores;
    * Consultar estatísticas relativas ao número de exportação de *logs*.
  - Administração de *logs*:
    * Consultar listagem dos *logs*;
    * Editar e remover os *logs* de qualquer utilizador presente no sistema.
  - Administração de notícias:
    * Consultar listagem de notícias;
    * Criar, editar e remover notícias;

É de realçar que, quando um utilizador se regista no sistema, é-lhe atribuído o perfil de produtor. Já quando um administrador procede à criação de um utilizador, este pode selecionar o perfil deste (produtor ou administrador).

2.2 Base de dados

Tendo definidos os requisitos do sistema, ao desenvolvimento de uma base de dados em MongoDB de forma a garantir a persistência dos dados da aplicação.

2.2.1 Estrutura

Considerando que era imperativo armazenar os dados relativos aos utilizadores, aos filmes, aos *logs* e às notícias, optou-se por criar a base de dados `FilmLog` que contém uma coleção para cada uma destas entidades. Abaixo, encontra-se a composição das coleções geradas.

Coleção dos utilizadores:

- *username*
- *password*
- nome completo
- biografia
- perfil
- número de vezes que exportou os seus *logs*

Coleção dos filmes:

- identificador
- titulo
- realizador
- ano
- duração
- sinopse
- estúdios
- países
- linguagens
- elenco
- géneros

Coleção dos *logs*:

- *username* do utilizador que realizou o *log*
- identificador do filme
- nome do filme
- cotação
- comentário
- indicação de se o filme pertence ou não aos filmes favoritos do utilizador
- data em que foi realizado o *log*

2.2.2 Povoamento

De modo a enriquecer a aplicação, optou-se povoar a base de dados com um catálogo grande e diversificado de filmes. Para tal, utilizou-se dois *datasets* presentes no Kaggle que continham a informação relativa a aproximadamente 5 mil filmes.

Dado que a informação dos filmes se encontrava dividida em dois *datasets* distintos no formato CSV, tornou-se imperativa a criação de uma *script* avulsa em Python que reúne toda a informação presente nestes num único ficheiro JSON, composto por uma lista de objetos.

O ficheiro gerado através deste conversor encontra-se pronto para ser importado para o MongoDB através do seguinte comando:

```
mongoimport -d FilmLog -c movies --file new_movies.json --jsonArray
```

onde:

- `FilmLog`: nome da base de dados
- `movies`: nome da coleção onde serão armazenados os filmes
- `new_movies.json`: nome do ficheiro gerado pelo conversor

2.3 Servidor de autenticação

Tendo concluída a preparação da base de dados, principiou-se por edificar um servidor responsável pela autenticação dos utilizadores. Este é composto por um único modelo de acesso à base de dados

(`userSchema`) e por um controlador que implementa as operações de *query* (`User`), bem como por um roteador.

Este apresenta apenas duas rotas:

- **POST user/login**: recebe as credenciais introduzidas na aplicação pelo utilizador e verifica se estas estão corretas. Em caso afirmativo, autentica o utilizador através da geração de um *token*, caso contrário apresenta um erro.
- **POST user/signup**: recebe as credenciais introduzidas pelo utilizador e verifica se o *username* selecionado está disponível e se cumpre os requisitos predefinidos[2]. Em caso afirmativo, insere o utilizador na base de dados, caso contrário retorna uma mensagem de erro.

Na primeira rota, a autenticação dos utilizados é feita através de uma estratégia local do Passport que recorre aos JSON Web Tokens (JWT).

Para correr este servidor, basta utilizar o comando `npm start` dentro da diretoria `auth-server`. Executando este comando, o servidor fica a correr na porta 4444, podendo ser acedido pelo *browser* através do endereço `http://localhost:4444/`.

2.4 API de dados

Em seguida, continuou-se o desenvolvimento da aplicação passando para a criação de uma API de dados que recebe pedidos, acede à base de dados e envia as respostas em formato JSON.

Esta é composta por quatro modelos abstratos de acesso à base de dados (`userSchema`, `movieSchema`, `logSchema` e `newSchema`), tendo um controlador para cada um destes (`User`, `Movie`, `Log` e `New`). A acrescer a isto, possui um roteador que recebe os pedidos e envia os dados relativos aos mesmos.

Sempre que a API recebe um pedido, a primeira ação que esta realiza é verificar se este traz consigo com um *token* de acesso. A partir deste, a API é capaz de averiguar o perfil do utilizador que efetuou o pedido e, deste modo, determinar a informação a que este tem acesso. Isto é possível uma vez que a API tem conhecimento do segredo utilizado na criação do *token*. Se o pedido não possui um *token* ou se possui um *token* inválido, o utilizador trata-se de um consumidor.

A API disponibiliza uma panóplia de rotas que permitem a inserção, edição e remoção de utilizadores, *logs* e notícias. Para além disso, possibilita ainda a procura de filmes, utilizadores, *logs* e notícias através do valor de diferentes campos e a listagem dos mesmos. A acrescer a isto, possui uma rota que retorna o *username* e o perfil do utilizador que se encontra autenticado num dado modelo. Na secção seguinte, será referida a utilidade desta última rota.

[2]Definiu-se que um username válido apenas pode conter caracteres alfanuméricos, maiúsculos ou minúsculos, pontos, hífens e *underscores*

Destas, as rotas de edição e remoção de notícias, bem como as de listagem e a remoção de utilizadores são reservadas aos administradores. Já a edição de utilizadores é reservada aos utilizadores com o perfil de administrador e de produtor.

Para correr este servidor, basta utilizar o comando `npm start` dentro da diretoria `api-server`. Executando este comando, o servidor fica a correr na porta 4445, podendo ser acedido pelo *browser* através do endereço `http://localhost:4445/`.

2.5 Servidor da aplicação

Por fim, implementou-se o servidor responsável por apresentar uma interface gráfica aos utilizadores. Este recebe os pedidos, pede à API as informações necessárias para os satisfazer, recebe os dados e mostra-os aos utilizador através da interface gráfica.

Nesta secção, explicita-se os detalhes mais interessantes relativamente a este servidor, começando pelo comportamento dinâmico da interface gráfica.

2.5.1 Comportamento dinâmico da interface gráfica

Para o correto funcionamento da aplicação, foi necessário ter em consideração que a interface gráfica tem de se adaptar tendo em conta o perfil do utilizador que a está a utilizar. Um caso notório deste comportamento é o cabeçalho da aplicação.

Este último sofre alterações conforme o perfil do utilizador que está a utilizar o sistema, tal como podemos constatar nas imagens abaixo.

Quando estamos perante um consumidor, o cabeçalho apenas contém as opções de criar uma conta e de iniciar sessão numa conta já existente, independentemente da página na qual este se encontro.

Figura 2.1: Cabeçalho de um consumidor

No caso em que um produtor está autenticado, o cabeçalho muda. Quando o utilizador se encontra no seu perfil, o cabeçalho contém a sua foto de perfil e o seu nome de utilizador, seguida de uma hiperligação para a página com a listagem dos filmes e com a opção de encerrar sessão

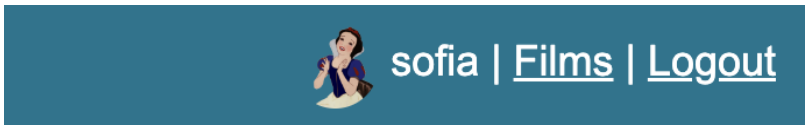

Figura 2.2: Cabeçalho do perfil de um produtor

Já os administradores, estando no seu perfil, deparam-se um cenário semelhante ao dos produtores, contudo o seu cabeçalho possui uma opção que os conduz à área reservada dos administradores. Nesta podem efetuar as operações referidas nos requisitos.

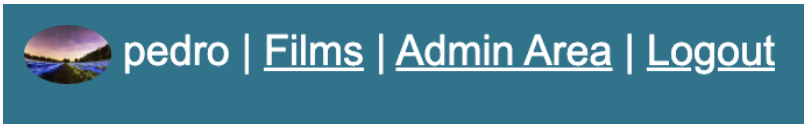

Figura 2.3: Cabeçalho do perfil de um administrador

Nas rotas em que é necessário conhecer o perfil do utilizador que está a utilizar a aplicação, o roteador deste servidor encarrega-se de pedir à API de dados as informações acerca do mesmo. Sempre que possível, estas são enviadas juntamente com as informações de outros pedidos. Estas informações são posteriormente enviadas para um *template* em Pug que, através de condicionais, mostra a página correspondente.

Outro caso em que se verifica este comportamento é na página de cada um dos filmes. Um utilizador não autenticado apenas consegue visualizar as informações do filme e as opiniões deixadas por outros utilizadores. Já um utilizador autenticado tem acesso à mesma informação, bem como a um formulário onde pode registar a sua opinião sobre o filme em questão.

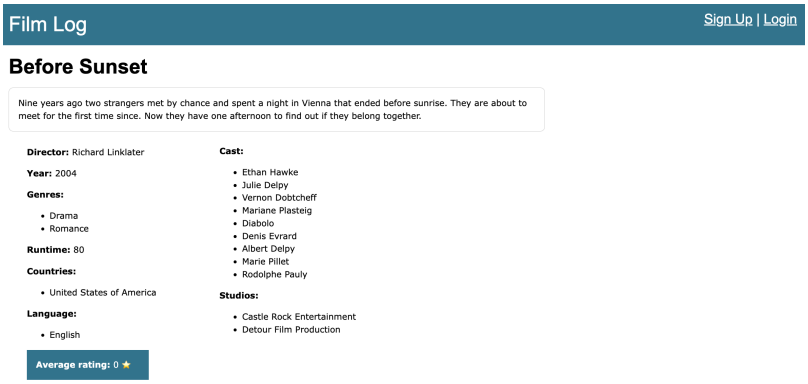

Figura 2.4: Página de um filme de um utilizador não autenticado

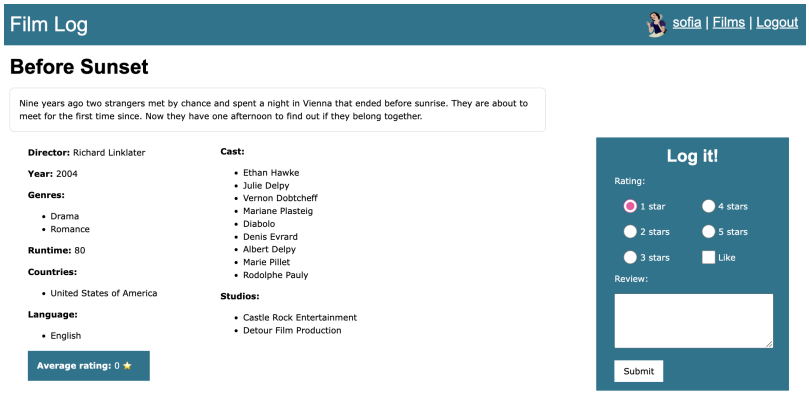

Figura 2.5: Página de um filme de um utilizador autenticado

2.5.2 Foto de perfil

Na página da edição do perfil de cada utilizador é permitido aos utilizadores fazer *upload* de uma imagem no formato `.jpeg`, que será utilizada como a sua foto de perfil. Esta foto, depois de ser guardada no *file system*, é movida para uma pasta que contém as imagens de todos os utilizadores e o nome da mesma alterado para ser o nome de quem a submeteu.

Quando é feito um da pedido de imagem de um utilizador ao servidor, este verifica se existe o ficheiro com o mesmo nome. Caso este não esteja presente, é devolvida uma imagem que é utilizada por defeito.

2.5.3 Importar e Exportar dados de utilizador

De forma a dar ao utilizador mais controlo sobre os seus dados, permitiu-se ao mesmo fazer *download* de um ficheiro com todas as avaliações que este fez na forma de um CSV. Para gerar este CSV, é feito um pedido à API, que devolve de uma vez todos os *logs* que o utilizador gerou. Cada um destes representa uma linha do CSV.

O campo `_id` é ignorado já que serve apenas para uso interno à base de dados e não representa informação útil para o utilizador. Da mesma forma que é permitido a um utilizador exportar os seus dados, também é possível importar um ficheiro no formato CSV que, por sua vez, é interpretado pelo servidor e todas as avaliações lá presentes são enviadas para a API.

Este processo não dá ao utilizador nenhum privilégio extra, por isso independentemente do que está presente no ficheiro fornecido, o campo `user` é ignorado e é utilizado o nome do utilizador que fez o pedido. O campo `date` é também ignorado e é utilizada a data da submissão.

Como forma de exportar todos os dados gerados durante a utilização da aplicação, existe uma rota exclusiva a administradores que cria um ficheiro no formato Bagit e permite o *download* do mesmo. Este ficheiro é constituído por:

- Um *manifest file* que contém uma entrada para cada ficheiro presente na pasta `data`;
- Um ficheiro de texto que contém informações tais como o tipo de codificação dos ficheiros de texto, tamanho total, entre outros;
- Uma pasta `data` que por sua vez contém:
  - Uma pasta `user` que contém ficheiros que armazenam a informação de cada utilizador em formato JSON
  - Uma pasta `log` que contém ficheiros semelhantes, mas com as avaliações de cada filme;

2.6 Docker

Para facilita a instalação da aplicação foram criados *Dockerfiles* para cada um dos servidores bem como um *dockercompose* que faz o *build* e configura as portas para cada um dos servidores e gera também um *container* que contem a base de dados

Capítulo 3

Trabalho Futuro

Relativamente ao trabalho futuro, o primeiro ponto a ser atacado deveria ser o design da aplicação, uma vez que este não se adapta conforme o tamanho da janela em que está a ser visualizado.

Em seguida, seria interessante aumentar as opções de procura, passando a permitir pesquisar filmes pelo género, pelo ano, entre outros, bem como estender estas procuras para as tabelas da área reservada dos administradores.

Para além disso, poderia-se diversificar as estatísticas apresentadas quer na área reservada quer na página de cada um dos filmes. Um exemplo seria a colocação de um gráfico de barras na página de cada filme que permitisse observar a distribuição das cotações do filme.

Outras possibilidades seriam permitir que os administrações importem informação para o programa utilizando um Bagit e a criação de uma página individual para cada membro do elenco dos diversos filmes.

Capítulo 4

Conclusão

No decorrer deste relatório, percorreu-se o processo que esteve por detrás da implementação de uma aplicação Web que possibilita a consulta de informações sobre filmes e noticias do mundo cinematográfico, bem como da partilha de opiniões.

O projeto em mãos possibilitou a solidificação de conceitos lecionados no decorrer da unidade curricular, tais como a criação de APIs de dados, autenticação de utilizadores utilizando o Passport e o JWT, a edificação de interfaces gráficas dinâmicas e a utilização do MongoDB.

Durante o processo de desenvolvimento, a maior dificuldade foi garantir que todas as rotas se encontravam devidamente protegidas.

Apesar da aplicação ainda ter uma infinidade de funcionalidades que poderiam ser implementadas, considera-se que os objetivos estipulados para este projeto foram atingidos. A equipa está satisfeita com o resultado final obtido.