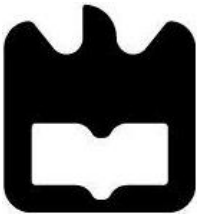
TDW – Mini Projeto

Universidade de Aveiro

Departamento de Comunicação e Arte

Mestrado em Comunicação e Tecnologias Web



Ricardo Cruz nº 93118

17 de Novembro de 2021

Índice

1. Introdução

Neste mini projeto, a aplicação que desenvolvi chama-se CeMovies. Como o próprio nome indica, é referente ao tema de filmes. Nesta aplicação web, é possível ver uma lista de filmes aparecendo em diferentes formatos, como filmes recomendados, upcoming movies, ou então através de filtros. Na lista de filmes, o utilizador pode clicar no filme que pretender, e ver informações mais detalhadas do mesmo.

Esta aplicação faz pedidos á API TMDB (https://developers.themoviedb.org/3/getting-started/introduction), que fornece todos os dados que são apresentados no website.

Para aceder ao website, pode ser feito através de dois URL's diferentes:

- **https://ricardocruz29.github.io/TDW_mod1b/Projeto/ (hospedado em GitHub pages)**
- **https://labmm.clients.ua.pt/deca_21TDW/deca_21TDW_07/Projeto/ (hospedado servidor UA)**

PS : A box shadow que adiciona cores ao fundo da página, não funciona corretamente no Safari. É aconselhado a ser testado no Chrome ou Firefox, só pela estética.

2. Estratégia de Implementação

As páginas html deste website possuem todas o evento **onload**, onde é chamada a função **main(page)** do ficheiro script.js. Esta função recebe um argumento de entrada, que funciona como identificador da página que foi loaded. Nesta função, a primeira ação a executar é renderizar o menu lateral, através da chamada á função **renderMenu(page)** definida no ficheiro renderMenu.js.

De seguida, na função main() existe um switch que para a página passada no argumento, vai fazer ações diferentes.

- **index.html:** Chama a função loadRecommendedMovies() e a função getUpcomingMovies(). A primeira função pega num objeto javascript que contém informações de filmes, e adiciona a informação de cada filme dinamicamente ao carrossel de imagens na home page (index.html). A segunda função faz um pedido á API que retorna os resultados com os filmes que estão para ser lançados. São apresentados apenas 6 filmes, por efeitos estéticos.
- **movies.html:** Primeiro, chama a função ClearFilters() para limpar os filtros caso estes não se encontrem no estado default. De seguida chama a função **getMovies("all")** que vai fazer um pedido á API para obter todos os filmes, e gera também um **paginator**. Por fim, são chamadas as funções que adicionam os event listeners aos elementos de filtragem.
- **moviePage.html:** Esta página vai ser loaded sempre que um utilizador clica para ver mais informação de um certo filme. Para indicar qual é o filme, é passado nos parâmetros do url o id do filme clicado. Seguidamente, a primeira coisa que vai ser feita neste caso, é ir buscar o id do filme aos parâmetros do url. Com o id do

filme, é chamada a função **getMovieInfo(id),** que vai buscar toda a informação daquele filme. Na resposta do pedido, não vem toda a informação necessária, e consequentemente são efetuados depois mais três pedidos á API. Um pedido é para ir buscar o elenco do filme, outro é para ir buscar o id id do trailer do filme do youtube, e finalmente é pedido também filmes semelhantes, de forma a sugerir novos filmes ao utilizador.

Nota: As funções chamadas nestas páginas estão definidas no ficheiro correspondente i.e, index.js, movie.js, moviePage.js. Uma exceção a isto são as funções que começam por “get”, que nesse caso são referentes a pedidos á API, e estão definidas no ficheiro APIfetch.js

3. Elementos dinâmicos implementados

3.1. Menu

O menu aparece em todas as páginas, e consequentemente faz todo o sentido que não seja repetido o código do menu em cada ficheiro html. Desta forma, criei uma função **renderMenu(page)** que renderiza todo o menu dinamicamente na página que foi loaded. Assim sendo, o menu tem um comportamento semelhante a uma componente, não existindo repetição de código.

3.2. Recommended Movies

Para fazer os recommended movies, optei por criar um objeto javascript que contém informação sobre os filmes que são adicionados dinamicamente ao carrossel da home page. Escolhi ser eu a criar o objeto, porque o endpoint que retornava filmes recomendados não retornava filmes muito conhecidos, e também porque a imagem que era retornada era na vertical. Como eu uso o carrossel de forma a atrair o utilizador, convém as imagens terem qualidade alta, e para efeitos estéticos que estivessem na horizontal.

Desta forma, mantenho a facilidade de manualmente aceder ao objeto e adicionar, remover ou editar filmes, conjuntamente com uma melhor estética.

3.3. Upcoming Movies

Os upcoming movies funcionam numa lógica muito semelhante aos recommended movies, com a pequena alteração desta informação ser obtida através de um endpoint da API. Para os filmes retornados, limito a serem apresentados apenas 6, também por questões estéticas.

3.4. Movie Page

A página de um filme em específico, vai renderizar dinamicamente toda a informação referente á informação do filme que foi escolhido. Num primeiro pedido á API, vai ser adicionado dinamicamente a imagem do filme, o título, ano, a duração

do filme (vem em minutos, mas faço o parse para horas) e o rating. Num próximo pedido, vou buscar o elenco do filme, onde aproveito o nome do ator e também a sua fotografia (aparece a fotografia quando se faz hover no nome). Para além disto, faço também um pedido á API para obter o id do trailer do filme no Youtube. Com este id, num modal já previamente criado no html, que contém um elemento iframe, adiciono a source do trailer. Caso o utilizador clique fora da modal, a source é retirada. Por fim faço um pedido para obter filmes semelhantes, e adiciono também dinamicamente á pagina.

3.5. Movies

Na página dos filmes temos 3 tipos de pedidos á API. Um pedido onde vamos buscar todos os filmes, que ocorre quando o utilizador entra na página, ou quando o utilizador não insere nada no campo de pesquisa e clica no Enter ou no botão para pesquisar. O segundo pedido, é quando o utilizador insere algo no campo de pesquisa do nome do filme, e pesquisa. Neste caso é feito um pedido para ir buscar os filmes que contém aquele valor passado no campo. O terceiro pedido, é quando o utilizador usa os filtros de ano, género, ou sort by, que sempre que o utilizador muda um destes valores, é feito um pedido á API, onde primeiramente verifica os valores de todos estes dropdowns, e insere no URL aqueles que não estão no valor default.

Todos estes pedidos são feitos por default para a página número 1, e na resposta do pedido vem indicado o número total de página. Com esta informação retornada, conseguimos adicionar dinamicamente todos os filmes á página, e criar ainda um paginator.

3.6. Paginator

O paginator é possível ser feito porque na resposta vem indicado o número total de páginas com informação. Posto isto, faz todo o sentido que, de forma semelhante ao menu, este paginator funcione como uma componente, de forma que possa ser utilizado de forma geral em mais do que um pedido á API, e em mais do que uma página do website, de forma a não serem repetidas linhas de código.

O paginator desenvolvido segue basicamente uma estrutura de < 1 2 3 4 5 6 … NR_TOTAL_PÁGINAS > . Esta estrutura é feita apenas se o número total de páginas for superior a 7. Quando é inferior ou igual a 7, adicionamos só esses números á estrutura.

3.7. Genres

Como os géneros de filmes são praticamente estáticos, ou seja, é raro um novo género ser adicionado, considerei que seria melhor, manualmente fazer um pedido á API e obter todos os géneros, e depois criar um objeto javascript com essa informação. Desta forma, continuamos com a dinamização da inserção de cada género no dropdown, no entanto não estamos necessitados de fazer mais um pedido á API.

4. Desafios

O maior desafio foi sem dúvida a implementação do paginator, devido ás exceções e á lógica que tem de ser implementada num paginator com a estrutura < 1 2 3 4 5 6 … NR_TOTAL_PÁGINAS > .

Para ultrapassar este desafio, foi basicamente fazer vários testes, e encontrar novos bugs, e ir desenvolvendo um algoritmo generalizado para qualquer pedido, e que tivesse em conta as exceções existentes.

5. Obstáculos não ultrapassados

Não acho que tenha existido concretamente um obstáculo que não tenha conseguido ultrapassar, mas provavelmente o maior foi a própria API que utilizei, que possuía alguns problemas como não permitir pesquisa simultânea de nome de filmes com os outros filtros, e também a filtragem por ano não estar a funcionar como esperado pela API.

6. Conclusão

Concluindo, acho que no tempo estipulado para o desenvolvimento do projeto, consegui desenvolver um produto estável e pronto para ser lançado ao público. O número de funcionalidades pode não ser a maior, mas todo o código está feito de forma dinâmica e geral, levando a um menor custo de implementação das novas funcionalidades.

Com este mini projeto e este módulo, consegui organizar todas as ideias de javascript que estavam um pouco espalhadas, e também consolidar e aprender novos conceitos de javascript.