INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Projeto - Implementação

Pedro Pita, 19933

2021

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Projecto - Implementação

Elaborado por:

Pedro Pita, nº19933

Orientado por:

Luís Carlos da Silva Bruno

Relatório de projeto da unidade curricular Tecnologias para a Web e Dispositivos Móveis

Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

2021

Índice

Lista de Tabelas

Lista de Figuras

1. Introdução

Neste trabalho vou desenvolver uma aplicação web, denominada de “Next”. O principal objetivo passa por dar a conhecer uma imensa variedade de filmes aos utilizadores. Adicionalmente, estão ao dispor dos usuários ações como adição e remoção de filmes da lista de favoritos, permitindo-lhes, assim, decidir de forma eficiente qual será o próximo filme a assistir (conceito presente aquando a nomeação da app - “Next”), da mesma forma como poderá retirar um dos favoritos, quando já assistido.

Para este sistema poder ser alimentado, utilizei uma Rest API de um site já existente, denominado de “The movie db” (link). Esta API disponibilizou-me dados reais sobre filmes, tais como as avaliações, últimos lançamentos, entre várias outras informações. Além disso, também me permitiu criar as listas de favoritos, possibilitando ao utilizador adicionar e remover filmes dessa mesma lista.

Na fase de análise/desenho, recolhi informações relativamente às tarefas a serem implementadas, assim como desenhei um protótipo não funcional da interface, sendo que este apresentou-se como um suporte a às tarefas mencionadas. Além disso, fiz testes de usabilidade com futuros possíveis utilizadores da aplicação, aproveitando para recolher o seu feedback. Este passo permitiu-me melhorar a interface em termos de usabilidade.

Já na fase de implementação, desenvolvi um protótipo funcional para dar suporte às seguintes tarefas:

T1- Adicionar um filme a uma lista de favoritos: A primeira tarefa passa pelo utilizador poder adicionar um filme à sua lista de favoritos.

T2- Remover um filme a uma lista de favoritos: A segunda tarefa passa pelo utilizador poder remover um filme da sua lista de favoritos.

2. Decisões Globais de Implementação

2.1. Tecnologias

2.1.1. ReactJS

Para o desenvolvimento do protótipo funcional, decidi usar esta biblioteca de javascript por várias razões, destacando-se as três principais:

- **Fácil aprendizagem –** uma vez utilizar uma linguagem lecionada nas aulas, nomeadamente, o JavaScript, a adaptação é mais simples. Além disso, e por se tratar de uma bibiblioteca muito popular, acaba por haver muito conteúdo na internet que facilita a aprendizagem;
- **Procura de mercado –** Uma das razões que me levou a escolher o ReactJS foi a grande procura por programadores com experiência nesta biblioteca. Desta forma, vi aqui uma opurtunidade de enriquecer o meu currículo, bem como conhecimento;
- **Renderização assíncrona –** Permite alterar estados e partes/zonas da página sem ser necessário recarregar a página completamente.

Versão: 17.0.2

Documentação: https://reactjs.org/docs/getting-started.html

2.1.2. Node.js

Uma vez que o Node.js é a plataforma mais conveniente para hospedar, bem como executar um servidor web para uma aplicação ReactJS, usei-a para esses propósitos ou finalidades.

Versão: 12.13.1

Documentação: https://nodejs.org/en/docs/

2.1.3. JavaScript

Como a biblioteca ReactJS é toda ela baseada em JavaScript, foi necessário utiliza-la no desenvolvimento do protótipo functional.

Versão: ECMAScript 2018

Documentação: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

2.1.4. JQuery

Recorri ao JQuery por forma a facilitar a manipulação do DOM, sendo que o último verifica-se mais simples, comparativamente ao Javascript nativo.

Versão: 3.3.1

Documentação: https://api.jquery.com

2.1.5. APIs

Para este sistema poder ser alimentado, utilizei uma Rest API de um site já existente, o "The movie db". Esta API cedeu-me dados reais sobre filmes, tais como os títulos e capas, avaliações, últimos lançamentos, entre outras informações. Além disso, permitiu-me criar as listas de favoritos, possibilitando ao utilizador adicionar e remover filmes dessa mesma lista.

https://www.redhat.com/en/topics/api/what-is-a-rest-api JSON

2.1.6. HTML

Uma das características do ReactJS é o JSX, que nos permite poder escrever HTML e CSS dentro de um arquivo javascript. Assim sendo, HTML é a linguagem de marcação escolhida sem hesitar.

Versão: HTML5

Documentação: https://developer.mozilla.org/pt-BR/docs/Web/HTML

2.1.7. CSS

Pela mesma razão de ter escolhido HTML, decidi utilizar o CSS, mesmo sabendo que poderia ter utilizado uma tenoclogia mais avançada, como é o caso do SCSS. No entanto, e tendo em conta que a maior parte dos estilos que pretendo utilizar já estão disponiveis no bootstrap, entendi não ser necessário recorrer a uma tecnologia vanguardista, priorizando o CSS até ao nível 4.

Versão: 2.1: Level 4 Revision 1

Documentação: https://developer.mozilla.org/pt-BR/docs/Web/CSS

2.1.8. Bootstrap

A razão por que decidi utilizar o bootstrap passou pela facilidade que este oferece em tornar uma interface responsive. Além disso, apresenta várias classes que me permitem estilar a interface de forma muito mais simples e rápida.

Escolhi a versão 5 ao invés da 4, uma vez que a primeira já apresenta updates, tanto em termos de tamanhos de colunas, como em rows e containers, por exemplo. Enquanto que a versão 4 disponibiliza apenas 4 tamanhos de ecrãs (sm,md,lg,xl), no bootstrap 5 são-nos fornecidos 6 tamanhos de ecrã (xs,sm, md,lg,xl,xxl);

Versão: Bootstrap V5

Documentão: https://getbootstrap.com/docs/5.0/getting-started/introduction/

2.2. Páginas Modelo

Todas as páginas foram desenvolvidas com o máximo cuidado por forma a tornar o site responsivo em todos os tamanhos de ecrãs, ao longo deste tópico irei abordar apenas 3 ecrãs sendo eles:

- **Computador:**
  - **Dispositivo:** Legion Y540
  - **Resolução do dispositivo:** 1920 x 1080
  - **Grid correspondente no Bootstrap:** xll (≥1200)
- **Tablet:**
  - **Dispositivo:** IPad
  - **Resolução do dispositivo:** 768x1024
  - **Grid correspondente no Bootstrap:** md (≥768 e <1024)
- **Telemóvel:**
  - **Dispositivo:** IPhone X
  - **Resolução do dispositivo:** 375x812
  - **Grid correspondente no Bootstrap:** xs (<576)

2.2.1. Homepage

Nesta página são apresentados alguns filmes “top rated” ao utilizador, destacando-se 3 num carousel e 4 alinhados em baixo do carousel.

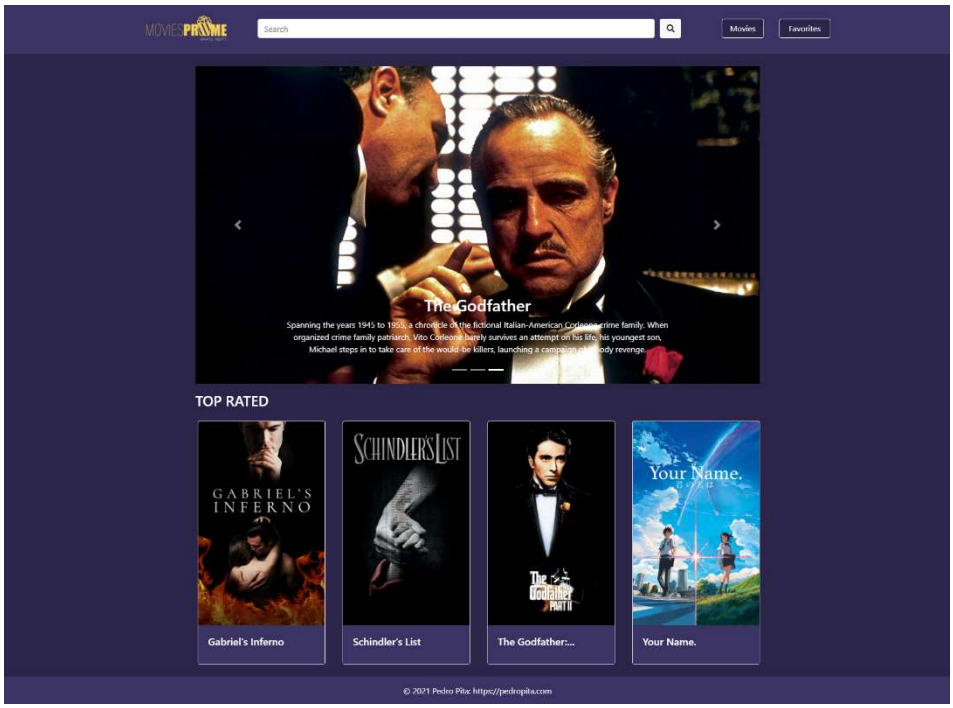

Figura 1-Homepage versão desktop

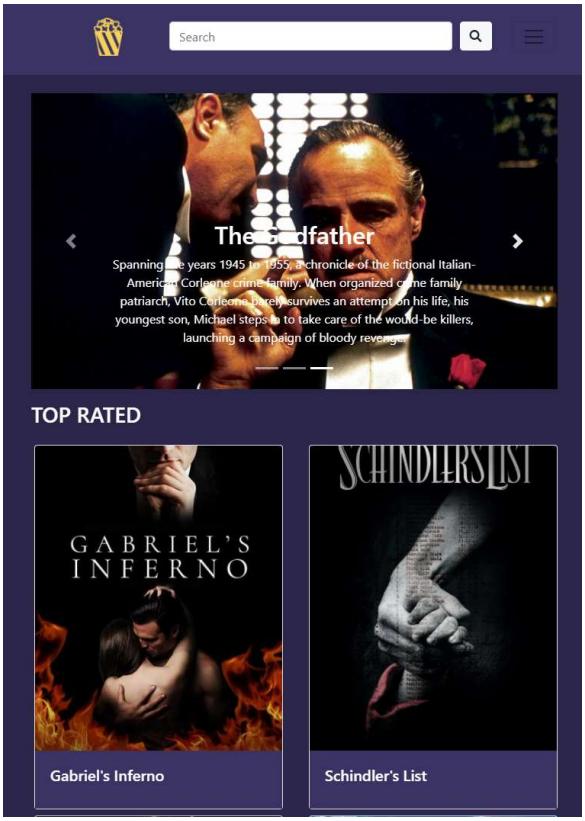

Figura 2-Homepage versão tablet

Figura 3-Homepage versão mobile

2.2.2. Listagem de filmes

Nesta página o utilizador pode consultar os vários filmes no sistema, bem como usar os filtros de pesquisa - 'sort by' / categorias / definição de um intervalo de datas de lançamento.

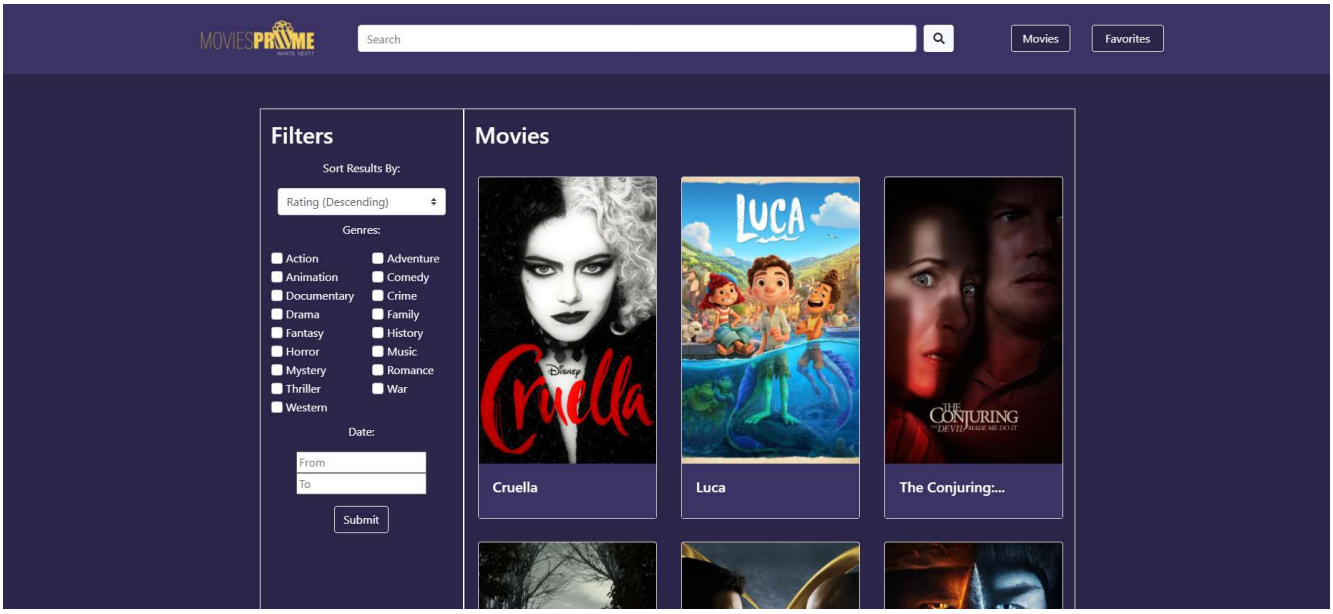

Figura 4-Listagem de filmes versão desktop

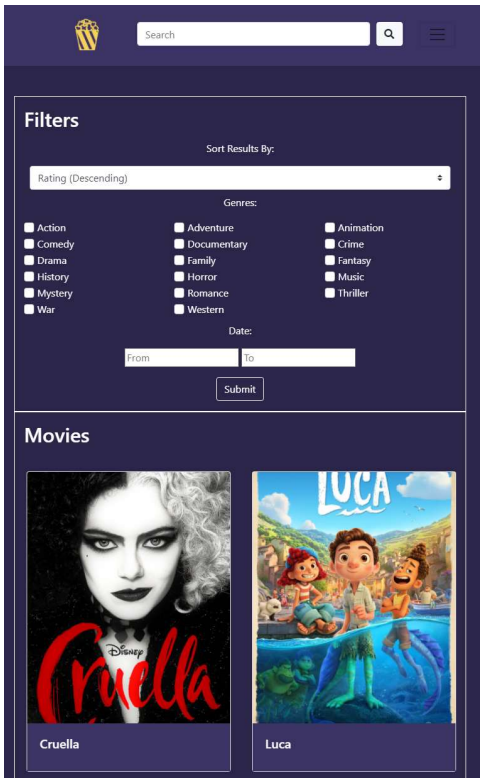

Figura 5-Listagem de filmes versão tablet

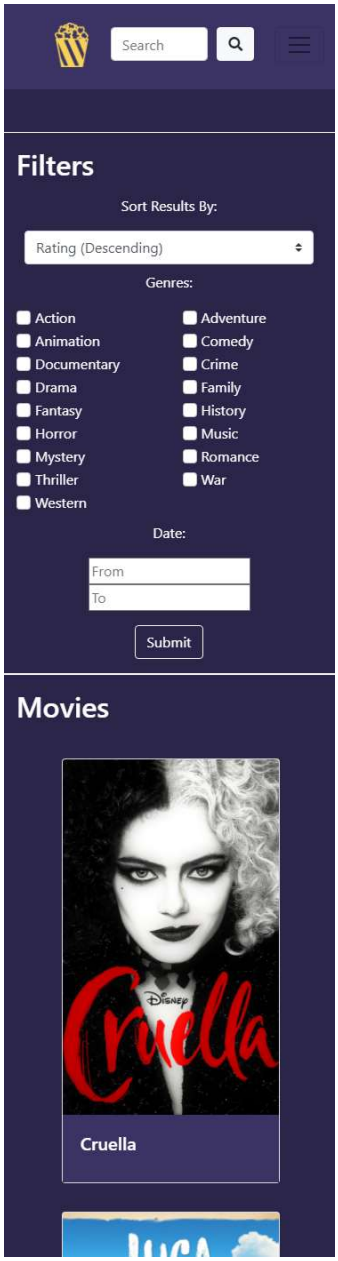

Figura 6-Listagem de filmes versão mobile

Figura 7- Paginação

2.2.3. Listagem de filmes favoritos

Aqui o utilizador pode consultar a sua lista de favoritos, com opção de recorrer aos filtros de pesquisa ‘sort by’ e categorias.

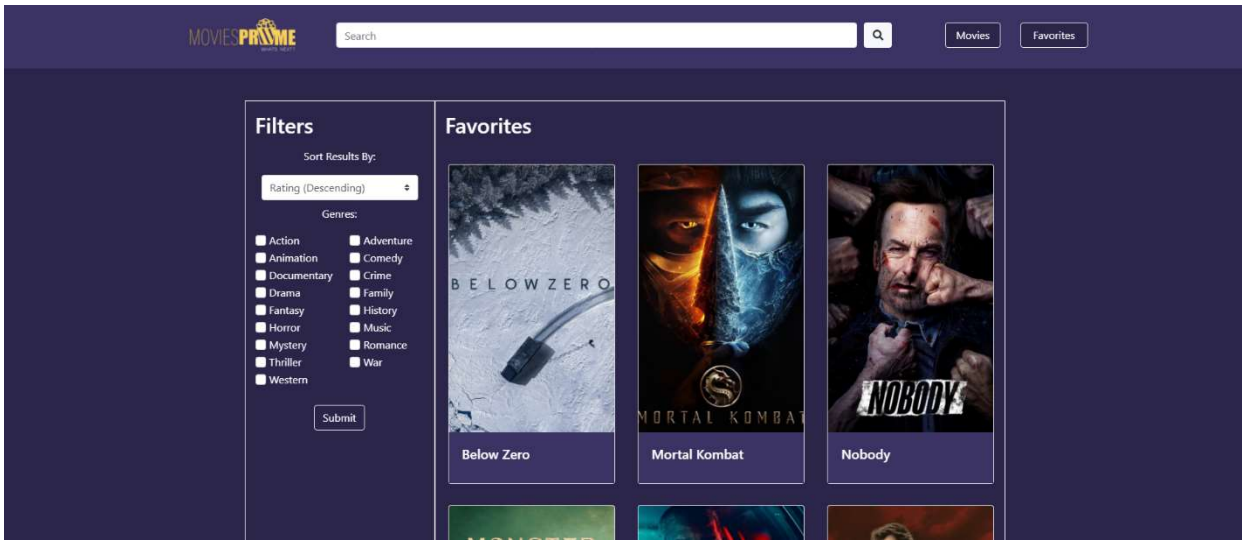

Figura 8-Listagem de filmes favoritos versão desktop

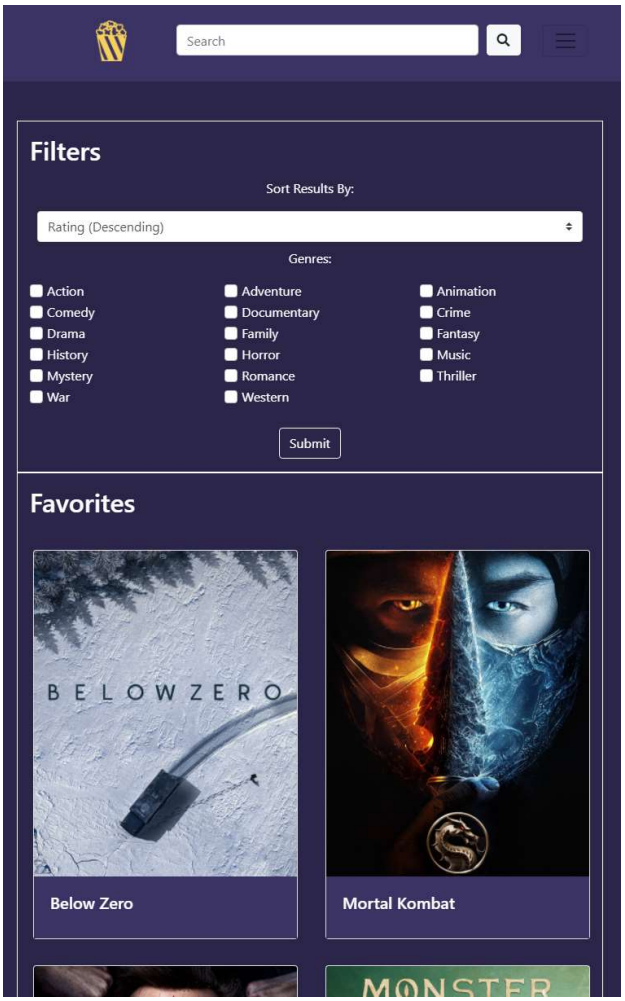

Figura 9-Listagem de filmes favoritos versão tablet

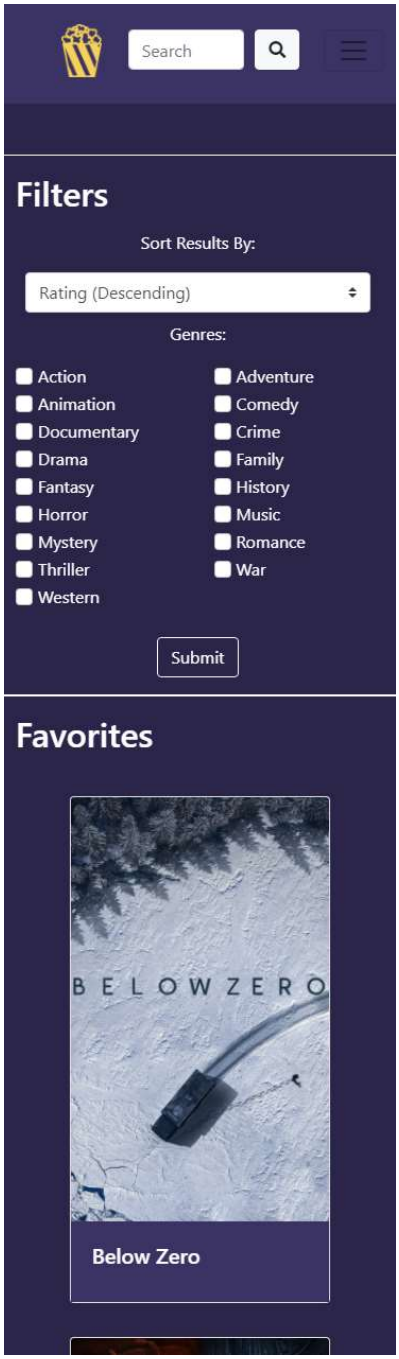

Figura 10-Listagem de filmes favoritos versão mobile

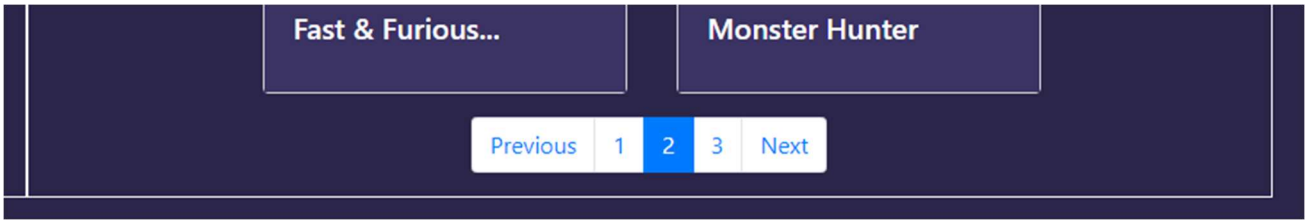

Figura 11-Paginação

2.2.4. Search

Após a pesquisa pelo título de um filme na navbar, são apresentados ao utilizador todos os filmes encontrados.

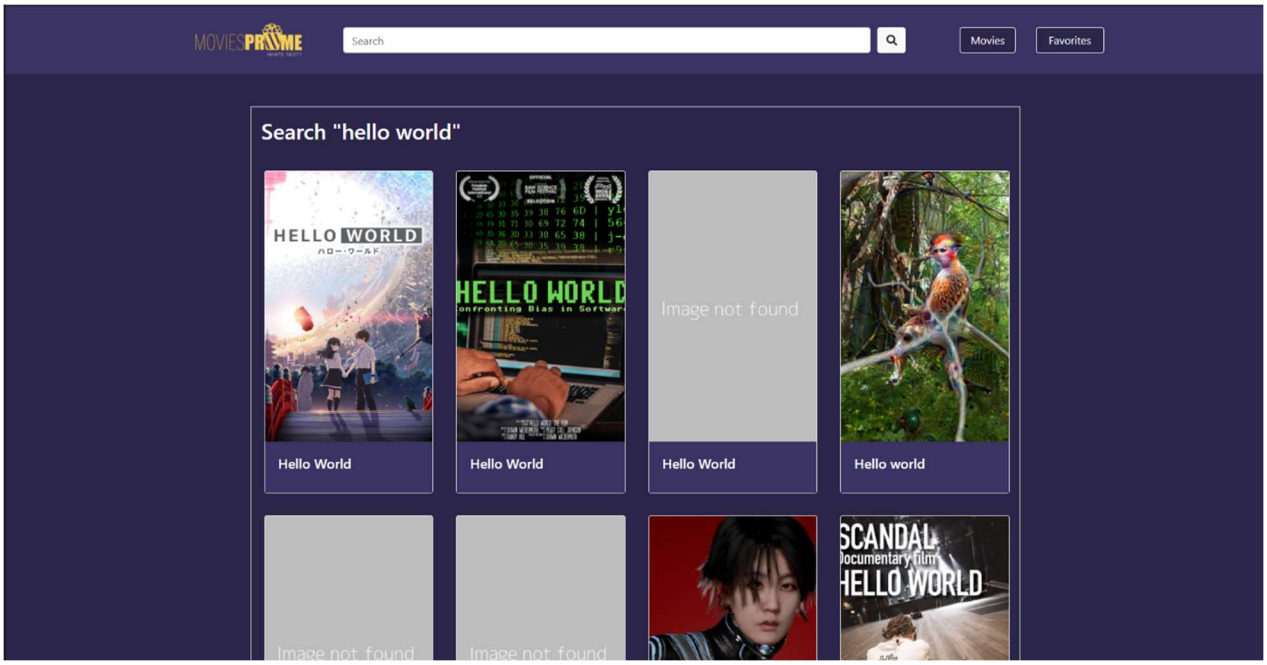

Figura 12- Search versão desktop

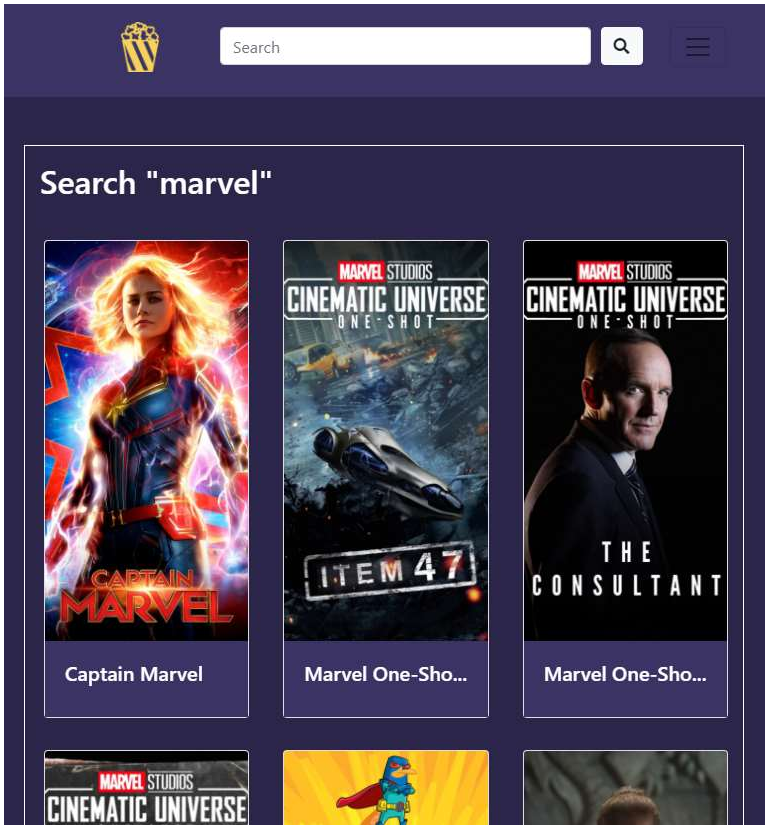

Figura 13-Search versão tablet

Figura 14-Search versão mobile

Figura 15-Paginação

2.2.5. Filme detalhado

Nesta página encontra-se o título, capa, data de lançamento, avaliação, categorias e sinopse de um filme escolhido pelo utilizador. É também aqui onde o usuário pode realizar as tarefas (T1 e T2).

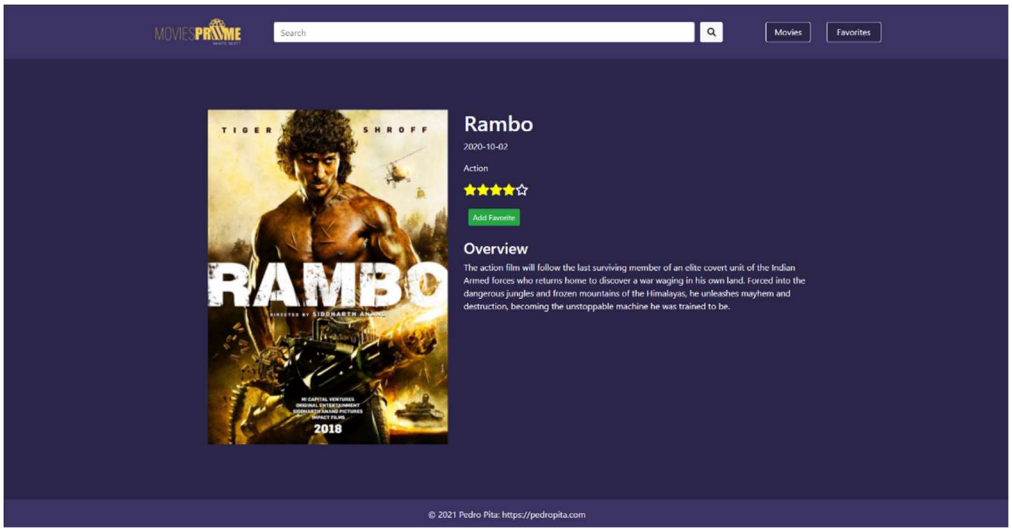

Figura 16-Filme detalhado versão desktop

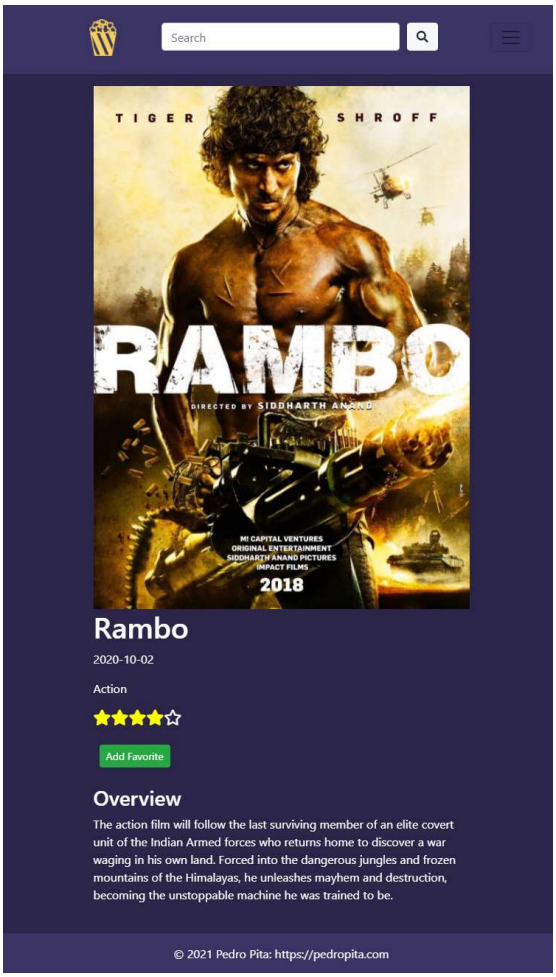

Figura 17-Filme detalhado versão tablet

Figura 18-Filme detalhado versão mobile

2.2.6. 404 – Página não encontrada

Quando o utilizador tenta aceder a uma rota que não é reconhecida pelo sistema, é-lhe apresentado um componente com uma imagem centrada, informando que ocorreu o erro 404 - página não encontrada.

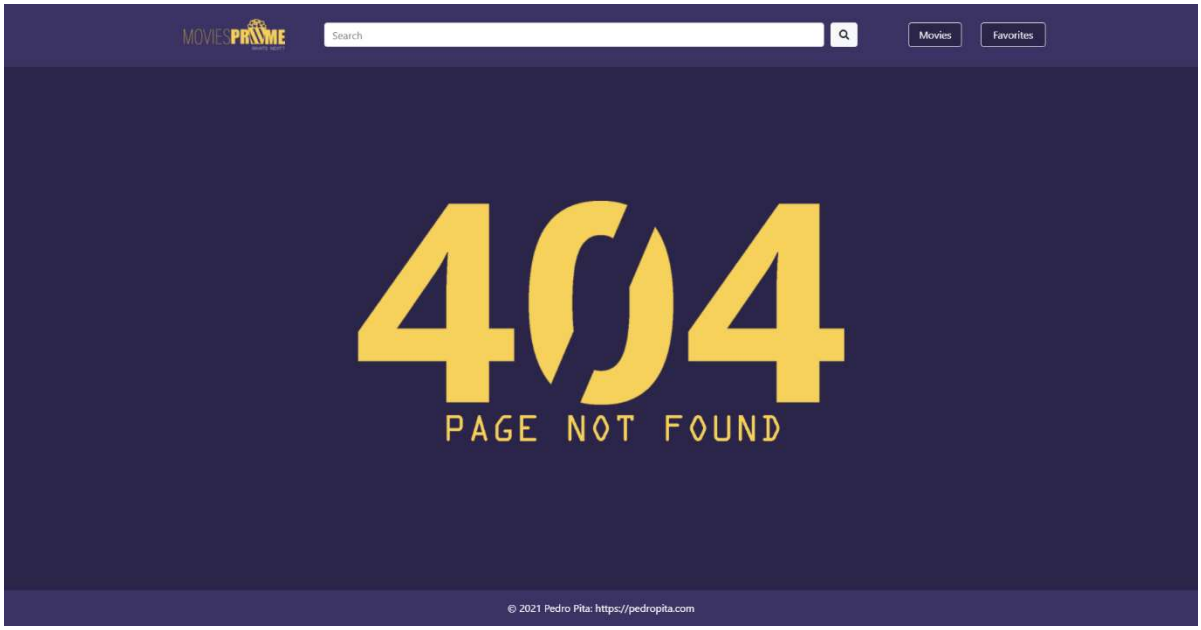

Figura 19- 404 versão desktop

Figura 20- 404 versão tablet

Figura 21-404 versão mobile

2.3. Principais estruturas de dados

2.3.1. Item filme

Este componente é dos mais reutilizados ao longo do sistema, sendo utilizado nas listagens de filmes (tanto de favoritos, search e listagem normal), bem como na homepage, na secção de top rated.

Quando a imagem não é encontrada é apresentada uma imagem default no seu lugar com o texto “Image dont found”

```
render(){
    return(
        <React.Fragment>
            <a className="ml-1 mt-2 d-flex justify-content-center" href={ "/detailed/" + this.props.id}>
                <div class="card secondary-background-color">
                    <img class="card-img-top img-fluid" src={this.props.imageURL} alt="Card image cap" style={ this.imageStyle()} onError={(e)=>{e.target.onerror = null; e.target.src="https://dummyimage.com/400x600/bdbdb
                    <div class="card-body secondary-background-color">
                        <h5 class="card-title">
                            { (this.props.title.length > 17) ? this.props.title.slice(0,14) + "..." : this.props.title}
                        </h5>
                    </div>
                </div>
            </a>
        </React.Fragment>
    )
}
```

Figura 22- Componente "Item filme"

2.3.2. Filtros

O componente filtro é utilizado em todas as listagens. Encontra-se na página de listagens de filmes favoritos e filmes normais, sendo composto por 4 componentes filho - as quais passo a explicar mais detalhadamente no tópico seguinte.

```
render(){
    return (
        <>
            <div className="row d-flex justify-content-center mt-2">
                <p className="col-12 d-flex justify-content-center">Sort Results By:</p>
                <Sort onChangeSort={this.handleSort} />
            </div>
            <div className="row mt-2">
                <p className="col-12 text-center">Genres:</p>
                <Genres onChangeGenre={this.handleGenres} />
            </div>
            <center>
                <div className="row mt-2">
                    {
                        (this.props.typeList != "favorites")
                        ?
                            <div>
                                <p className="col-12 text-center">Date:</p>
                                <div className="col-12">
                                    <DatePicker onChangeDatePicker={this.handleDatePicker}/>
                                </div>
                            </div>
                        :
                            ""
                    }

                    <div class="form-group mt-3" align="center">
                        <input onClick={this.handleSubmit} className="btn btn-light secondary-background-color" name="submit" type="submit" value="Submit" style={this.buttonStyle()}/>
                    </div>
                </div>
            </center>
        </>
    )
}
```

Figura 23-Componente "filtros"

2.3.3. Loading

Este componente é utilizado em todas as páginas. Enquanto o sistema está a fazer algum pedido externo a API, este componente renderiza um spinner até os dados serem carregados.

```
export default class Loading extends Component {
  render() {
    return (
          <div className="container" style={{height: "85vh"}}>
              <div class="d-flex align-items-center justify-content-center h-100">
                  <p class="align-middle"><Spinner animation="border" role="status"/></p>
              </div>
          </div>
      );
    }
}
```

Figura 24-Componente "Loading"

2.3.4. Navbar e Footer

O footer e o navbar são renderizados apenas uma vez, mantendo-se sempre presentes ao longo do sistema, uma vez ser alterado apenas o conteúdo localizado entre estes dois componentes.

```
render() {
    return (
        <>
            <nav class="navbar navbar-expand-lg navbar-light secondary-background-color mb-3" >
                <div class="container-fluid secondary-background-color">
                    <div className="col-12 col-lg-8 col-xl-9 ">
                        <div className="row">
                            <div class="input-group rounded">
                                <div className="col-3 col-sm-3 col-md-3 col-lg-5 col-xl-4" align="right">
                                    <div className="row">
                                        <div className="col-1 col-xxl-6"/>
                                        <div className="col-12 col-xxl-6">
                                            <a className="navbar-brand nav-link nav-items text-white" href="/" >{<img className="" src={this.state.url }/>}</a>
                                        </div>
                                    </div>
                                </div>
                                <div className="col-9 col-sm-9 col-md-9 col-lg-7 col-xl-8 align-self-center" align="left">
                                    <Search/>
                                </div>
                            </div>
                        </div>
                    </div>
                    <div className="col-12 col-lg-4 col-xl-3 secondary-background-color align-self-center" align="left" >
                        <div className="collapse navbar-collapse" id="navbarSupportedContent" >
                            <ul className="navbar-nav mr-auto ">
                                <li className="nav-item col-sm-12 col-md-12 col-lg-6">
                                    <form method="get" action="/movies/list">
                                        <input type="submit" className="btn btn-light btn-home col-sm-12 col-md-12 col-lg-12" value="Movies" />
                                    </form>
                                </li>
                                <li className="nav-item col-sm-12 col-md-12 col-lg-7">
                                    <form method="get" action="/movies/favorites">
                                        <input type="submit" className="btn btn-light btn-home col-sm-12 col-md-12 col-lg-12"value="Favorites" />
                                    </form>
                                </li>
                            </ul>
                        </div>
                    </div>
                </div>
            </nav>
        </>
    )
```

Figura 25-Componente "Navbar"

```
class Footer extends Component {
    render() {
        return (
            <>
                <footer className="text-center text-lg-start secondary-background-color mt-4">
                    <div className="text-center p-3">
                        <a className="text-white" href="https://pedropita.com/">© 2021 Pedro Pita: https://pedropita.com</a>
                    </div>
                </footer>
            </>
        )
    }
}

export default Footer;
```

Figura 26-Componente "Footer"

2.3.5. APP e rotas

Este componente é o 'componente raiz'. É ele quem trata de toda a gestão do sistema, ao receber os valores através do link e a chamar os componentes correspondentes. No caso de o sistema não reconhecer algum parâmetro passado, chama o componente 404.

```
return (
    <React.Fragment>
        <Nav />
        <Router>
            <Switch>
                <Route exact path="/" component={ Home } />
                <Route exact path="/movies/:value" component={ Movies } />
                <Route exact path="/movies/search/:value" component={ Movies } />
                <Route exact path="/detailed/:value" component={ Detailed } />
                <Route path="*" render= {() => <div className="container d-flex align-items-center justify-content-center"><img src="http://localhost:3000/404.png" width="70%"></img></div>} />
            </Switch>
        </Router>
        <Footer />
    </React.Fragment>
)
```

Figura 27- Componente que gere as rotas

2.4. Ficheiros utilizados

index.html

Este ficheiro é o único ficheiro html. Apenas o conteúdo do ficheiro é alterado em execução, através da troca de componentes;

App.js

Este componente é o 'componente raiz'. É ele quem trata de toda a gestão do sistema, garantindo a troca de html no ficheiro index.html. Além disso, é também ele quem recebe os valores através do URL e chama os componentes correspondentes. No caso de o sistema não reconhecer algum parâmetro passado, chama o componente 404.

css/style.css

Este ficheiro contém todo o CSS do sistema, para além do CSS utilizado pelo bootstrap.

pages/home.jsx

Este componente corresponde à página inicial do sistema. Nele é utilizado outro componente filho, neste caso, o **item_movie.jsx**.

pages/list_movies.jsx

Este componente corresponde à lista de filmes. Tem nele outros componentes, tratando-se eles do **filter_form.jsx** e o **item_movie.jsx.** Estes últimos, por sua vez, têm vários outros componentes que serão mencionados na descrição dos mesmos.

pages/detailed_movies.jsx

Este componente recebe um ID de um filme e faz um pedido à API, por forma a retornar os dados correspondentes a esse ID. É nele onde são efetuadas as tarefas principais do sistema.

components/navbar.jsx

Este componente é o navbar e é chamado assim que o sistema é inicializado. Dentro dele contém outro componente filho, o **search_form.jsx.**

components/footer.jsx

Este componente é o footer e é chamado no momento em que o sistema é inicializado.

components/filters_form.jsx

Este componente corresponde a todos os filtros das listagens, utilizando outros componentes filhos:

- **sort_filter.jsx** – componente do filtro "sort by";
- **genres_filter_form.jsx** – componente do filtro dos géneros;
- **date_picker.jsx** – componente que trata do date picker entre duas datas.

components/loading.jsx

Este componente é utilizado em todas as páginas. Ao mesmo tempo que o sistema está a fazer algum pedido externo a API, este componente renderiza um spinner até os dados serem carregados.

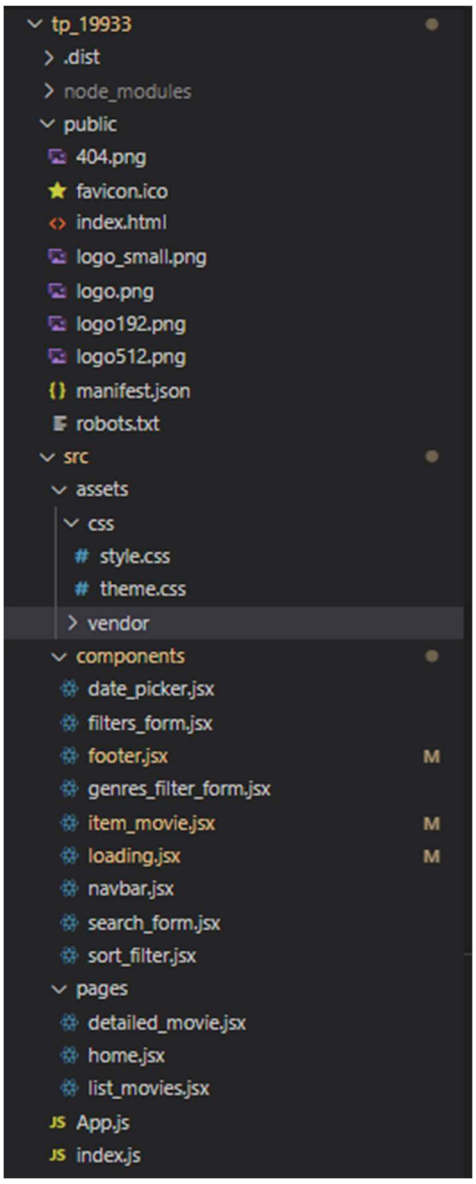

Figura 28-Hierarquia dos ficheiros

2.5. Papel da API

Para este sistema poder ser alimentado, utilizei uma Rest API de um site já existente, o "The movie db". Esta API foi fundamental no fornecimento dos seguintes dados:

API Call	Ação
GET /movie/top_rated?api_key={API_KEY}	Top Rated Filmes
POST /discover/movie?api_key={API_KEY}&conditions={CONDITIONS}	Listagem de filmes com filtros
GET /list/{LIST_ID}?api_key={API_KEY}	Consultar filmes da lista de favoritos
POST /list/{LIST_ID}/add_item?api_key={API_KEY}	Adicionar filme à lista de favoritos
DELETE /list/{LIST_ID}?api_key={API_KEY}	Remover filme da lista de favoritos
POST /list/{LIST_ID}/item_status?api_key={API_KEY}&movie_id={ID}	Verificar se um determinado filme está na lista de favoritos
GET /movie/{id}?api_key={API_KEY}	Consultar detalhes de um determinado filme

Tabela 1- API requests

2.6. Decisões gerais de acessibilidade

Algumas das decisões gerais de acessibilidade que implementei foram as seguintes:

- Adicionei "Alt Text" apropriado em todas as imagens;
- Usei títulos/cabeçalhos corretamente, bem como nas proporções adequadas;
- Indiquei corretamente a língua do website no head do HTML;
- Tornei todo o conteúdo acessível através do teclado, sendo possível percorrer a página através do tab (frente) e sfhit + tab (trás).

3. Decisões de Implementação Específicas

Seguidamente passo a explicar as principais decisões de implementação para a funcionalidade que suporta cada uma das duas tarefas.

3.1. Tarefa 1 - Adicionar um filme a uma lista de favoritos

3.1.1. Introdução

A primeira tarefa entende a adição de um filme à lista de favoritos, pelo utilizador.

A tarefa é executada na página do filme detalhado, onde são apresentados os detalhes correspondentes ao filme e questão, selecionado pelo utilizador.

Se o utilizador estiver perante um filme que não está na sua lista de favoritos, ser-lhe-á apresentado um botão a verde com o texto “Add Favorite”, como é possível constatar nas figuras 29 e 30.

Ao clicar no botão, é realizado um pedido à API - o de adicionar o filme a lista de favoritos. No caso de a API retornar uma mensagem de erro, entende-se que algo correu mal e o filme não foi adicionado. Por outro lado, e caso a API retorne uma mensagem de sucesso, será apresentada uma notificação temporária no canto superior direito, informando que o filme foi adicionado. Assim, o botão passa a vermelho com o texto “Remove Favorite”, como é possível verificar nas figuras 31 e 32.

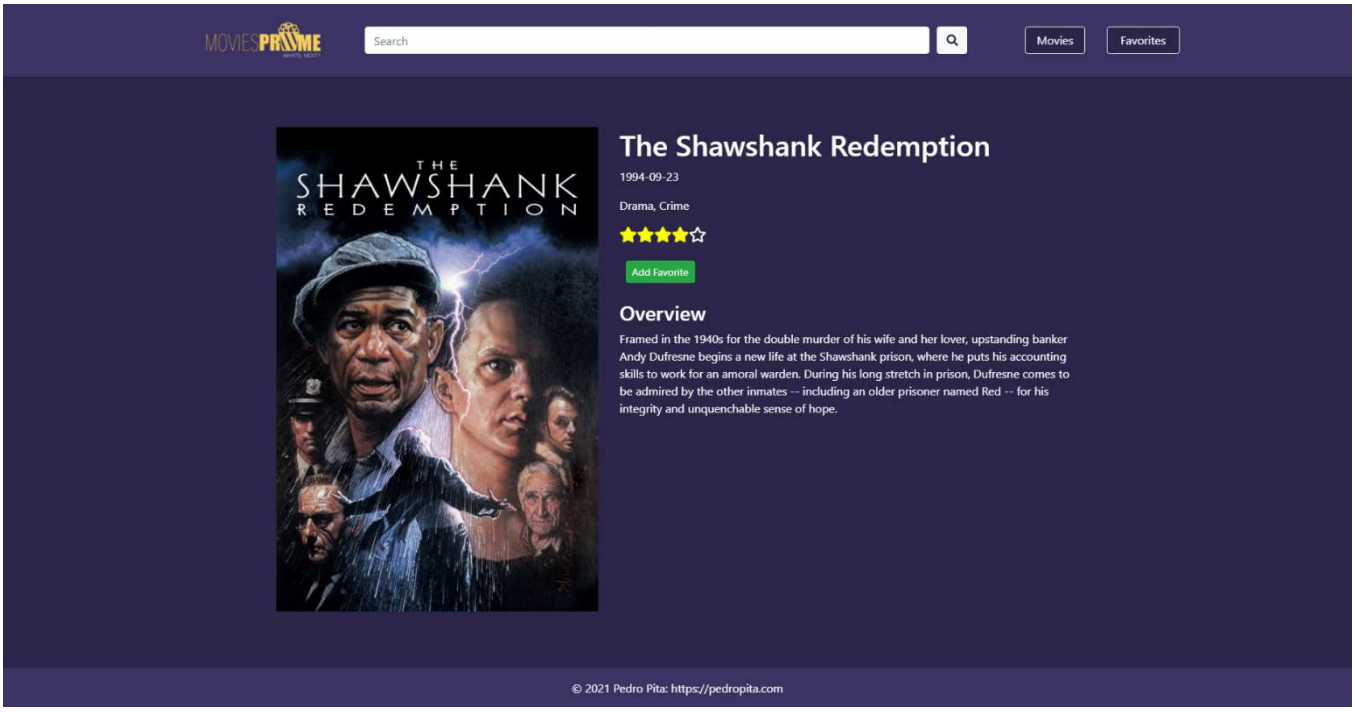

Figura 29- Filme detalhado versão desktop

Figura 30- Filme detalhado versão mobile

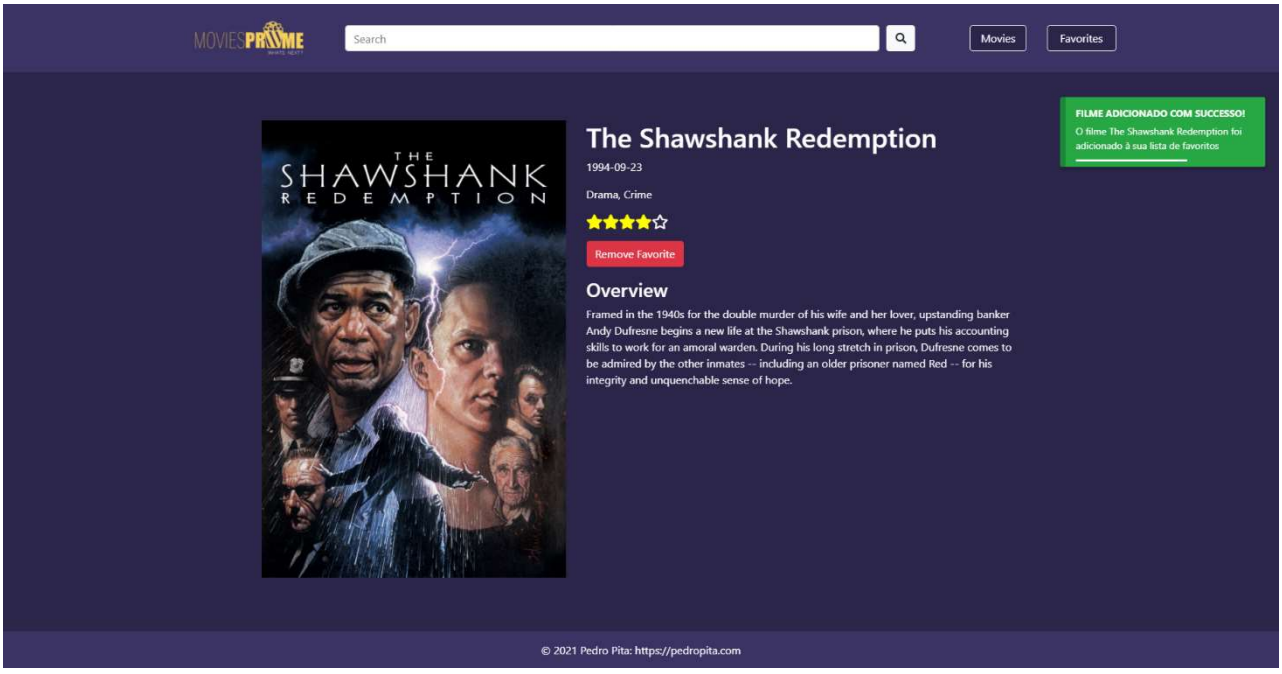

Figura 31-T1 executada na versão desktop

Figura 32-T1 executada na versão mobile

3.1.2. HTML

O HTML foi todo ele desenvolvido através do bootstrap. Assim, constitui-se por uma row que contém duas colunas lado a lado - uma para a imagem e outra para a informação sobre o filme -, sendo que na versão mobile é apresentada a imagem com as informações em baixo da imagem.

```
return (
    <div>
        <ReactNotification />
        <div className="container d-flex align-items-center justify-content-center">
            {
                <div className="row" >
                    <div className="col-md-5">
                        <img width="100%" src={ "https://image.tmdb.org/t/p/original/" + item.poster_path } alt={item.overview} />
                    </div>
                    <div className="col-md-7">
                        <h1>{ item.title}</h1>
                        <p>{ item.release_date }</p>
                        <div className="genres">
                            <p>{this.getGenres()}</p>
                        </div>
                        <div className="row">
                            <div class="col-md-12">
                                <p>{this.getStars()}</p>
                                <button value={this.state.item.id} className={ this.GetButtonClass()} onClick={ () => {this.swapFavoriteState()}}> { this.GetButtonText() }</button>
                            </div>
                        </div>
                        <div className="row mt-3">
                            <div className="col-md-12">
                                <h3>Overview</h3>
                                <p>{ item.overview }</p>
                            </div>
                        </div>
                    </div>
                </div>
            }
        </div>
    </div>
);
```

Figura 33- HTML da página de filme detalhado

3.1.3. CSS

Para poder perceber qual é a cor do botão e seu texto, utilizei duas funções - uma delas para obter o texto, e outra para obter as classes do bootstrap, que dão estilo ao botão.

```
GetButtonClass = () => {
    return this.state.favorite === true ? 'btn btn-danger' : 'btn btn-success' + " btn-sm m-2";
}

GetButtonText = () => {
    return this.state.favorite === true ? 'Remove Favorite' : 'Add Favorite';
}
```

Figura 34- Função de obter classes e texto do botão

3.1.4. JavaScript

Quando o onClick do botão é ativado, é chamado um método que trata de fazer o pedido à API, aguardando uma resposta. Quando essa resposta é obtida, é chamado o componente que mostra a notificação. Este componente permite-me adicionar vários tipos de notificação sendo possível personalizar a cor, tempo e tamanho da mesma.

```
swapFavoriteState = () => {
    const requestOptions = {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ media_id: this.state.item.id })
    };
    if(this.state.favorite)
        var link = 'https://api.themoviedb.org/3/list/7080650/remove_item?api_key=85b7f5dbd764003e3e05f18df89ff387&session_id=10cc9fc7513f93faba96a3042d349cc6e2b5d498';
    else
        var link = 'https://api.themoviedb.org/3/list/7080650/add_item?api_key=85b7f5dbd764003e3e05f18df89ff387&session_id=10cc9fc7513f93faba96a3042d349cc6e2b5d498';

    fetch(link, requestOptions)
    .then(async response => {
        this.setState( {favorite: !this.state.favorite });
        this.notificationGo(this.state.favorite === true ?  'FILME ADICIONADO COM SUCCESSO!' : 'FILME REMOVIDO COM SUCCESSO!',this.state.favorite === true ? 'O filme ' + this.state.item.title +' foi adicionado à sua list
    })
    .catch(error => {
        this.setState({
            isLoaded: true,
            error
        });
    });
}
```

Figura 35- Função que trata de executar a tarefa

```
notificationGo = (title, message) => store.addNotification({
    title: title,
    message: message,
    type: "success",
    insert: "top",
    container: "top-right",
    animationIn: ["animate__animated", "animate__fadeIn"],
    animationOut: ["animate__animated", "animate__fadeOut"],
    width:300,
    dismiss: {
      duration: 5000,
      onScreen: true
    }
});
```

Figura 36- Função que mostra a notificação

3.1.5. Acessibilidade

- Adicionei "Alt Text" apropriado em todas as imagens;
- Usei títulos/cabeçalhos corretamente e nas proporções adequadas;
- Indiquei corretamente a língua do website no head do HTML;
- Tornei todo o conteúdo acessível através do teclado, sendo possível percorrer a página através do tab (frente) e sfhit + tab (trás).

3.2. Tarefa 2 - Remover um filme a uma lista de favoritos

3.2.1. Introdução

A segunda tarefa passa habilitação do utilizador em remover um filme da sua lista de favoritos.

A tarefa é executada na página do filme detalhado, onde são apresentados os detalhes correspondentes ao filme selecionado pelo utilizador.

No caso de o utilizador estar perante um filme listado como favorito, é-lhe apresentado um botão a vermelho com o texto “Remove Favorite”, como podemos verificar nas figuras 37 e 38.

Ao clicar no botão, é feito um realizado um pedido à API para remover o filme da lista de favoritos. Caso a API retorne uma mensagem de erro, é evidenciado ao utilizador que algo correu mal e o filme não foi adicionado. Por outro lado, e no caso de a API retornar uma mensagem de sucesso, é apresentada uma notificação temporária no canto superior direito, informando que o filme foi removido e o botão passa a verde e com o texto “Add Favorite”, como podemos constatar nas figuras 39 e 40.

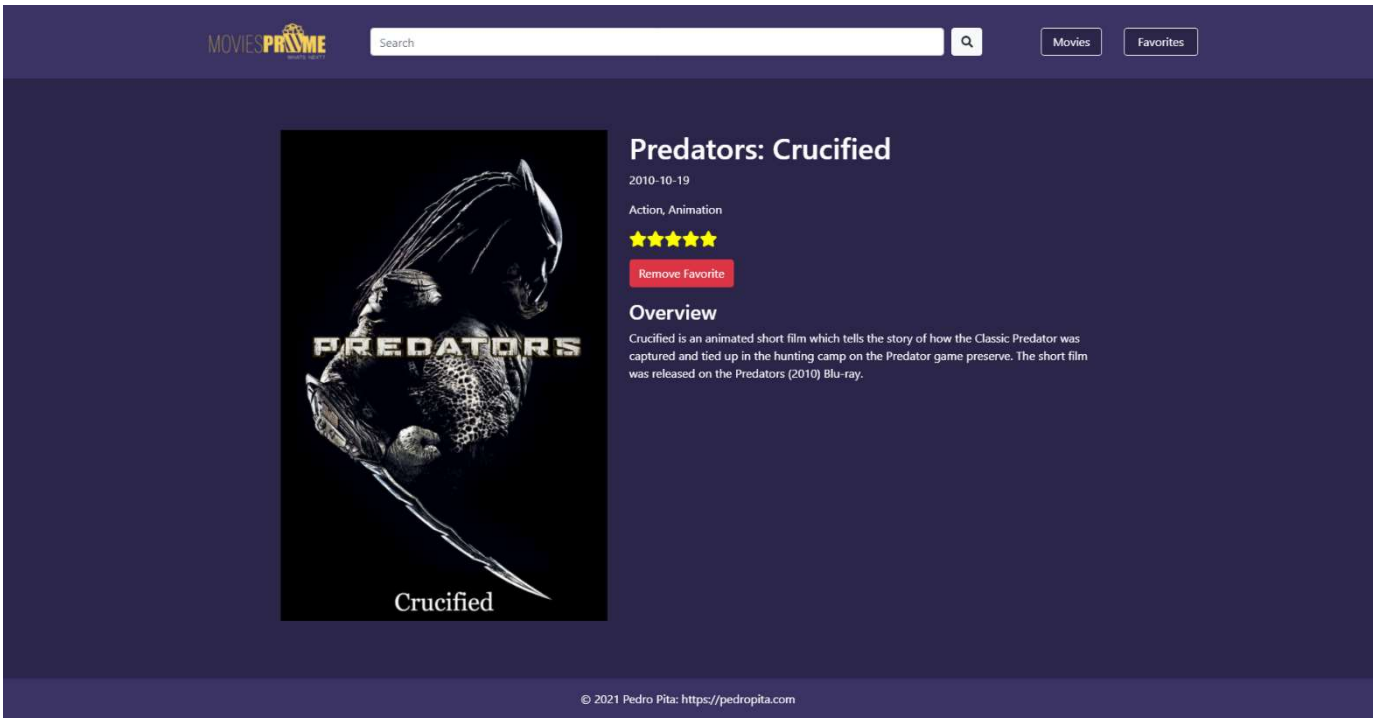

Figura 37-Filme detalhado versão desktop

Figura 38- Filme detalhado versão mobile

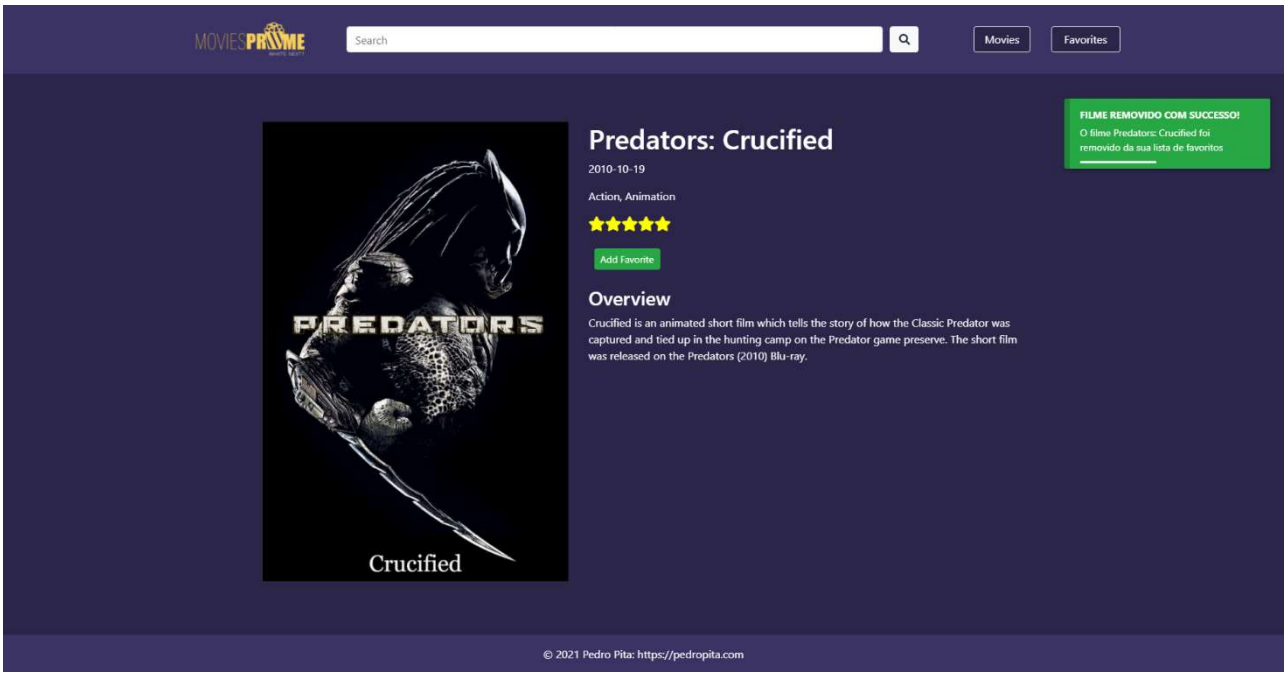

Figura 39- T2 executada na versão desktop

Figura 40-T2 executada na versão mobile

3.2.2. HTML

O HTML foi todo ele desenvolvido através do bootstrap, constituído por uma row com duas colunas lado a lado - uma para a imagem e outra para a informação sobre o filme - , sendo que na versão mobile é apresentada a imagem com as informações em baixo da imagem.

```
return (
    <div>
        <ReactNotification />
        <div className="container d-flex align-items-center justify-content-center">
            {
                <div className="row" >
                    <div className="col-md-5">
                        <img width="100%" src={ "https://image.tmdb.org/t/p/original/" + item.poster_path } alt={item.overview} />
                    </div>
                    <div className="col-md-7">
                        <h1>{ item.title}</h1>
                        <p>{ item.release_date }</p>
                        <div className="genres">
                            <p>{this.getGenres()}</p>
                        </div>
                        <div className="row">
                            <div class="col-md-12">
                                <p>{this.getStars()}</p>
                                <button value={this.state.item.id} className={ this.GetButtonClass()} onClick={ () => {this.swapFavoriteState()}}> { this.GetButtonText() }</button>
                            </div>
                        </div>
                        <div className="row mt-3">
                            <div className="col-md-12">
                                <h3>Overview</h3>
                                <p>{ item.overview }</p>
                            </div>
                        </div>
                    </div>
                </div>
            }
        </div>
    </div>
);
```

Figura 41-HTML da página de filme detalhado

3.2.3. CSS

Para poder perceber qual é a cor do botão e o texto por ele apresentado, utilizo duas funções - uma delas para obter o texto e outra para obter as classes do bootstra -, que vão estilizar o botão.

```
GetButtonClass = () => {
    return this.state.favorite === true ? 'btn btn-danger' : 'btn btn-success' + " btn-sm m-2";
}

GetButtonText = () => {
    return this.state.favorite === true ? 'Remove Favorite' : 'Add Favorite';
}
```

Figura 42- Função de obter classes e texto do botão

3.2.4. JavaScript

Quando o onClick do botão é ativado, é chamado um método que responsável por fazer o pedido à API, e aguardando uma resposta. Assim que a última é obtida, é chamado o componente que mostra a notificação. Este componente permite-me adicionar vários tipos de notificação, utilizando classes do bootstrap para informar qual a cor que pretendo, bem como adicionar o tempo que a notificação vai estar presente na tela e o seu tamanho.

```
swapFavoriteState = () => {
    const requestOptions = {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ media_id: this.state.item.id })
    };
    if(this.state.favorite)
        var link = 'https://api.themoviedb.org/3/list/7080650/remove_item?api_key=85b7f5dbd764003e3e05f18df89ff387&session_id=10cc9fc7513f93faba96a3042d349cc6e2b5d498';
    else
        var link = 'https://api.themoviedb.org/3/list/7080650/add_item?api_key=85b7f5dbd764003e3e05f18df89ff387&session_id=10cc9fc7513f93faba96a3042d349cc6e2b5d498';

    fetch(link, requestOptions)
    .then(async response => {
        this.setState( {favorite: !this.state.favorite });
        this.notificationGo(this.state.favorite === true ?  'FILME ADICIONADO COM SUCESSO!' : 'FILME REMOVIDO COM SUCESSO!',this.state.favorite === true ? 'O filme ' + this.state.item.title +' foi adicionado à sua list
    })
    .catch(error => {
        this.setState({
            isLoaded: true,
            error
        });
    });
}
```

Figura 43- Função que trata de executar a tarefa

```
notificationGo = (title, message) => store.addNotification({
    title: title,
    message: message,
    type: "success",
    insert: "top",
    container: "top-right",
    animationIn: ["animate__animated", "animate__fadeIn"],
    animationOut: ["animate__animated", "animate__fadeOut"],
    width:300,
    dismiss: {
      duration: 5000,
      onScreen: true
    }
});
```

Figura 44- Função que mostra a notificação

3.2.5. Acessibilidade

- Adicionei "Alt Text" apropriado em todas as imagens;
- Usei títulos/cabeçalhos corretamente e nas proporções adequadas;
- Indiquei corretamente a língua do website no head do HTML;
- Tornei todo o conteúdo acessível através do teclado, sendo possível percorrer a página através do tab (frente) e sfhit + tab (trás).

4. Avaliação das Interfaces/Testes com Utilizadores

4.1. Planeamento

4.1.1. Tarefas a ser testadas e objetivos

Nestes testes de usabilidade pretende-se que o utilizador realize as duas tarefas principais do sistema (T1 e T2). Assim, registo o feedback de possíveis futuros utilizadores do sistema. Através da avaliação deste feedback, é possível detetar eventuais fragilidades e até mesmo identificar melhorias de usabilidade. Desta forma, e de maneira a avaliar o desempenho e a experiência dos utilizadores através deste processo, anotei algumas informações. Entre elas, o tempo que o utilizador demorou a executar cada tarefa, a quantidade de cliques, e por último, se conseguiu ou não realizar com sucesso as várias ações necessárias para a realização da tarefa.

É importante referir que dei livre acesso aos utilizadores ao realizarem ambas as tarefas, podendo escolher entre ir pelo caminho mais logo ou pelo mais curto ou eficiente.

4.1.2. Elementos

Ao longo destes testes fui o facilitador. Auxiliei os participantes ao longo dos testes e além disso também tirei também algumas notas de acordo com o feedback do utilizador durante a realização das tarefas.

4.1.3. Escolha dos participantes

Para a realização dos testes escolhi 5 participantes. As escolhas tiverem em especial foco consumidores frequentes de sítios web, visto esse ser o perfil de possíveis futuros usuários.

4.1.4. Questionário aos participantes

Por forma a conhecer um pouco melhor os participantes, bem como a sua experiência com sites semelhantes, realizei um questionário. Tendo por base as suas respostas, organizei os dados numa tabela por forma a facilitar a análise. As perguntas do questionário foram as seguintes:

- Com que recorrência usa sítios web? (Diariamente, Semanalmente, Mensalmente, Nunca)
- Com que recorrência usa sítios web semelhantes? (sites de informação sobre filmes) (Diariamente, Semanalmente, Mensalmente, Nunca)

Além disso, realizei também algumas perguntas de forma a perceber o feedback acerca da sua experiência ao executar as tarefas

- Como avalia a sua experiência em termos de usabilidade?
- O que melhoraria na interface em termos de usabilidade?

Para responderem ao questionário criei um formulário no Microsoft Forms (link do formulário) e após a entrevista pedi ao utilizador que respondesse ao mesmo.

4.1.5. Onde foram feitos os testes?

Devido às circunstâncias em que nos encontramos por motivos pandémicos, decidi realizar os testes virtualmente, via chamada zoom com os utilizadores. Partilhei a minha tela com o sistema a correr no meu computador e deleguei o controlo ao utilizador. Desta forma, o usuário pôde ter autonomia nas decisões, enquanto anotava o seu feedback, com a chamada/sessão em gravação.

4.2. Execução dos testes

Os testes foram gravados por forma a comprovar a sua realização. Além disso, esta condição permitiu-me rever cada entrevista, e analisar mais calmamente o comportamento dos utilizadores.

Dei livre acesso aos utilizadores ao realizarem ambas as tarefas (T1 e T2). Escolhendo podem ir pelo caminho mais logo ou pelo mais curto.

Deixo anexado os links dos testes:

- Utilizador 1- https://www.youtube.com/watch?v=ecLozXFyPuI
- Utilizador 2- https://www.youtube.com/watch?v=MySZmY3quhI
- Utilizador 3- https://www.youtube.com/watch?v=ePBOeXnrBuI
- Utilizador 4- https://www.youtube.com/watch?v=nxEjA9NEvr4
- Utilizador 5- https://www.youtube.com/watch?v=frT0omnP2Bk

4.3. Resultados

4.3.1. Questionário

1. Com que recorrência usa sítios web?

Mais Detalhes

Diariamente	5
Semanalmente	0
Mensalmente	0
Nunca	0

2. Com que recorrência usa sítios web semelhantes? (sites de informação sobre filmes)

Mais Detalhes

Diariamente	0
Semanalmente	0
Mensalmente	1
Nunca	3

3. Como avalia a sua experiência em termos de usabilidade?

Mais Detalhes

Muito Má	0
Má	0
Razoável	0
Boa	0
Muito Boa	5

4. Melhoraria alguma coisa na interface em termos de usabilidade?

Mais Detalhes

Sim	0
Não	5

5. Se respondeu sim a pergunta anterior, o que melhoraria?

Mais Detalhes

0

Respostas

Respostas Mais Recentes

Figura 45- Respostas ao questionário

4.3.2. Analise Descritiva dos dados

i. Versão Desktop

Utilizador	Tempo (T1)	Nº Cliques (T1)	Tempo (T2)	Nº Cliques (T2)	Respostas Corretas (T1)	Respostas Corretas (T2)
1	00:00:10	2	00:00:12	3	100%	100%
2	00:00:10	4	00:00:10	3	100%	100%
3	00:00:05	2	00:00:08	3	100%	100%
4	00:00:10	2	00:00:07	3	100%	100%
5	00:00:09	3	00:00:07	3	100%	100%
Min	00:00:05	2	00:00:07	3	100%	100%
Média	00:00:08	2,50	00:00:08	3,00	100%	100%
Max	00:00:10	4	00:00:12	3	100%	100%
Desvio	00:00:02	0,80	00:00:02	0,00	0%	0%

Tabela 2- Analise descritiva dos dados na versão desktop

ii. Versão Mobile

Utilizador	Tempo (T1)	Nº Cliques (T1)	Tempo (T2)	Nº Cliques (T2)	Respostas Corretas (T1)	Respostas Corretas (T2)
1	00:00:07	2	00:00:14	4	100%	100%
2	00:00:09	2	00:00:14	4	100%	100%
3	00:00:22	4	00:00:12	4	100%	100%
4	00:00:08	2	00:00:12	4	100%	100%
5	00:00:14	4	00:00:11	4	100%	100%
Min	00:00:07	2	00:00:11	4	100%	100%
Média	00:00:11	2,67	00:00:12	4,00	100%	100%
Max	00:00:22	4	00:00:14	4	100%	100%
Desvio	00:00:06	0,98	00:00:01	0,00	0%	0%

Tabela 3- Analise descritiva dos dados na versão mobile

4.3.3. Conclusões sobre os dados obtidos

A partir dos dados recolhidos, posso observar um feedback dos utilizadores muito positivo.

Alguns dos utilizadores não têm por costume a utilização de sites deste género. No entanto, são navegadores web diariamente. Assim, posso entender que a interface está intuitiva, não só para utilizadores frequentes de sites semelhantes como para utilizadores que utilizam sítios web diariamente.

O tempo médio de execução de cada tarefa apresenta um valor inserido numa avaliação muito boa ou positiva, mantendo-se dentro do tempo expectado por mim à priori. No entanto, avalio uma extensão no tempo de realização das tarefas na versão mobile. Todavia, compreendo e identifico este aspeto como "nada muito significativo" ou "anormal", permanecendo, ainda assim, dentro tempo que havia sido atendido.

Em relação aos dados obtidos na fase de desenho e apesar de não apresentar um problema, posso observar uma extensão nos tempos de execução também por conta dos tempos de espera ao realizar os pedidos à API.

5. Esclarecimentos e conclusões finais

Ainda na fase de desenho, foi-me sugerido pelos utilizadores a implementação de uma forma mais prática no que concerne ou dizia respeito à realização/execução das tarefas. Esta sugestão passava pela adição de um botão em cada filme na listagem, onde através dele, o usuário podia adicionar ou remover o filme diretamente. No entanto, e após analisar a situação, decidi não implementar esta funcionalidade, uma vez que a mesma entendia a execução de 1 pedido por filme, por forma a poder averiguar se este estaria presente na lista de favoritos ou não, levando assim a cerca de 20 pedidos por página podendo causar uma maior lentidão no carregamento das listagens.

Tentei prevenir alguns erros, sendo que:

- Quando uma rota não é reconhecida, é apresentado o erro 404;
- Quando a API retorna algum erro, o utilizador é informado;
- Quando uma imagem não é encontrada, é salientada uma imagem default.

Para o desenvolvimento do projeto utilizei um repositório no github onde pode ser visto todos os commits efetuados por mim ao longo do desenvolvimento do projeto, desde a fase de proposta de trabalho até ao momento https://github.com/pedro-pita/TP_TWAM.

Bibliografia

[1] Material disponibilizado pelo docente da unidade curricular.
https://cms.ipbeja.pt/course/view.php?id=1566

[2] Ferramenta utilizada para criação do questionário
https://www.office.com/launch/forms

[3] Ferramenta utilizada para efetuar os testes
https://zoom.us/

[4] Ferramenta utilizada para efetuar os testes
https://www.youtube.com