INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Tecnologias Web e Ambientes Móveis

Projeto - Fase de Implementação

Diogo Patusca /23925

Beja, 2024

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Tecnologias Web e Ambientes Móveis

Logflix

Elaborado por:

Diogo Patusca /23925

Docentes:

Luís Carlos Bruno
Henrique Água-Doce

Descrição do trabalho da fase de implementação do projeto da unidade curricular "Tecnologias Web e Ambientes Móveis" e que é parte integrante da sua metodologia de avaliação

Beja, 2024

Índice

Lista de quadros

1 Introdução

Este projeto tem como objetivo desenvolver um software de forma rigorosa, simulando assim as várias fases de desenvolvimento de um trabalho comercial.

O projeto tem o nome de LogFlix, o tema enquadra-se na área de Media e Entretenimento. A finalidade deste software é permitir ao utilizador consultar todos os filmes/séries que já viu e os detalhes dos mesmos.

As API utilizadas foram:

- OMDb API – API de acesso público que fornece informações sobre filmes, programas de televisão e outros conteúdos de entretenimento, para a realização do presente trabalho foram cruciais as funcionalidades de recolha de informação de filmes. Esta API permitiu aceder a detalhes como título, elenco, classificação, ano de lançamento, entre outros, de uma vasta coleção de media. A autenticação é realizada através de uma chave de API (API key);
- RESTfull API – fundamental para a realização das tarefas propostas, permitindo ações de GET, POST, DELETE e PATCH. Através dessas quatro ações será possível consultar a lista pessoal, adicionar filme á lista pessoal, adicionar filme á lista total (admin), adicionar comentário a um filme, ver os comentários sobre um filme e remover o próprio comentário ou remover qualquer comentário (admin).

As tarefas propostas a realizar inicialmente foram consultar filmes/series, gerir filmes/series, ler comentários dos utilizadores sobre os filmes e series e publicar um comentário sobre um filme/serie, sendo atribuídas as duas primeiras da responsabilidade de Diogo Patusca e as restantes duas tarefas da responsabilidade de João Costa.

Durante a realização deste projeto, foram feitas partes em conjunto, as partes de "suporte" que ambos íamos precisar para a realização de cada tarefa e a API RESTFull, porém as partes relacionadas com cada tarefa foram realizadas independentemente por cada membro do grupo.

Como as tarefas que me foram atribuídas possuíam mais funcionalidades ficou da responsabilidade de João Costa implementar as funcionalidades de autenticação e barra de pesquisa.

As funcionalidades desenvolvidas em relação á gestão e consulta de filmes foram as seguintes:

- Listas atualizarem no offcanvas;
- Criar uma lista;
- Adicionar um filme á lista pessoal;
- Remover um filme da lista pessoal;
- Ao entrar na lista mostrar os filmes já adicionados;
- Eliminar uma lista criada;
- Adicionar um filme á lista total (admin).

2 Decisões Globais de Implementação

Para o desenvolvimento deste projeto as bibliotecas e plataformas utilizadas foram:

- Axios versão 1.7.2 – esta biblioteca foi útil para realizar pedidos http á API;
- Bootstrap versão 5.3.3 – permitiu deixar o site responsivo e esteticamente mais agradável;
- React versão 18.2.0 – permitiu construir as interfaces e criar as componentes;
- React-bootstrap versão 2.10.2 – permitiu o uso de estilos e componentes do Bootstrap de forma mais integrada com o react;
- React-dom versão 18.2.0 – permitiu renderizar os componentes no navegador;
- React-router-dom versão 6.23.1 – permitiu a navegação entres as diferentes páginas e componentes.

Para garantir a responsividade do site, foi utilizado o bootstrap que forneceu diversas ferramentas e componentes. O bootstrap permitiu utilizar o "Grid System", o que ajudou a organizar os componentes de forma flexível nas páginas, além disso utilizou-se a navbar do bootstrap que se adapta automaticamente a diferentes tamanhos de ecrã e implementou-se um "Off Canvas Layout" em vez de uma sidebar, assim esta opção fica acessível através do menu hambúrguer ajudando na navegação de ecrãs menores.

Para assegurar a acessibilidade para pessoas com necessidades especiais foram contemplados os seguintes aspetos:

- Alternativas de texto – qualquer elemento não textual tem uma alternativa em texto garantindo que os leitores de ecrã possam ler esses elementos aos utilizados;
- Navegação apenas com o teclado – permitindo assim que pessoas com dificuldades motoras não necessitem de utilizar um rato para auxiliar na navegação;
- Tempo suficiente de leitura e de uso – foi implementado que cada popup tivesse um timer ajustado ao tempo necessário para a leitura do mesmo;
- Ausência de conteúdo sensível para epiléticos – garantiu-se que não exista conteúdo que possa provocar ataques epiléticos no utilizador;
- Previsibilidade – as páginas aparecem e funcionam de forma esperada;
- Texto legível
- Compatibilidade tecnológica.

Todas estas definições e decisões permitem criar uma experiência inclusiva na utilização, permitindo que todas as pessoas, independentemente da forma, possam utilizar o Logflix.

Para avaliar a acessibilidade do site foi utilizado o site do governo, disponibilizado pelo docente, de forma a garantir todos os níveis de acessibilidade do nível A. Através do access monitor, foram introduzidos os vários ficheiros com o HTML das várias páginas obtendo assim os seguintes resultados de acessibilidade num máximo de 10:

- Homepage – 8.6
- Homepage com o off canvas aberto – 8.4
- SearchPage sem pesquisa – 8.3
- SearchPage com pesquisa – 8.5
- MoviePage – 8.4
- Lista com filmes – 9.1
- Lista sem filmes – 9.1

No código do projeto, a estrutura das pastas e ficheiros está organizada de forma a facilitar o desenvolvimento e manutenção, de seguida vai ser descrita a estrutura.

React:

- Pasta “src” – contem todo o código base da aplicação
- Pasta “src/auth” – contém o ficheiro que faz o tratamento da autenticação dos utilizadores, é nesta pasta que está o código que verifica o login e o logout.
- Pasta “src/componentes” – é onde está o código de todos os componentes que são reutilizados nas páginas
- Pasta “src/routes” – é onde está o código referente às páginas da aplicação.
- Ficheiro “App.tsx” – é onde está definida a estrutura das rotas das páginas da aplicação;
- Ficheiro “main.tsx” é onde a aplicação é inicializada, e é neste ficheiro que renderiza o componente principal no DOM
- Ficheiro “App.css” é onde são definidos alguns estilos globais da aplicação.

Api:

- Pasta “Api”, que contem toda a informação da API realizada para nos auxiliar na criação das tarefas;
- “api/controlers”, que tem as funcionalidades de login, signup, dar get aos users e dar delete dos users;
- “api/middleware” faz a verificação das permissões através da role do utilizador;
- “api/models” tem os modelos e os esquemas dos dados pretendidos;
- “api/routes” tem todas as funcionalidades dos GET, POST, PATCH e DELETE.

3 Decisões de Implementação Específicas

Em seguida são explicadas as principais decisões de implementação para as funcionalidades que suportam cada uma das seguintes partes do sistema.

3.1 Homepage

A Homepage apresenta uma barra de navegação com um menu hambúrguer, uma barra de pesquisa e um botão para dar login ou logout. Na parte superior da secção do conteúdo da página é apresentada uma breve descrição sobre o projeto realizado, já na parte inferior são apresentados alguns filmes em destaques. Na parte do rodapé é apresentado o nome dos autores e o ano e nome do site.

Na HomePage encontram-se alguns componentes como a NavBar, o TextCard, o Destaques , o Item, este encontra-se dentro dos Destaques e por fim o Footer, como se pode ver pelo diagrama da figura 1 e pelo código apresentado na figura 3 e 4.

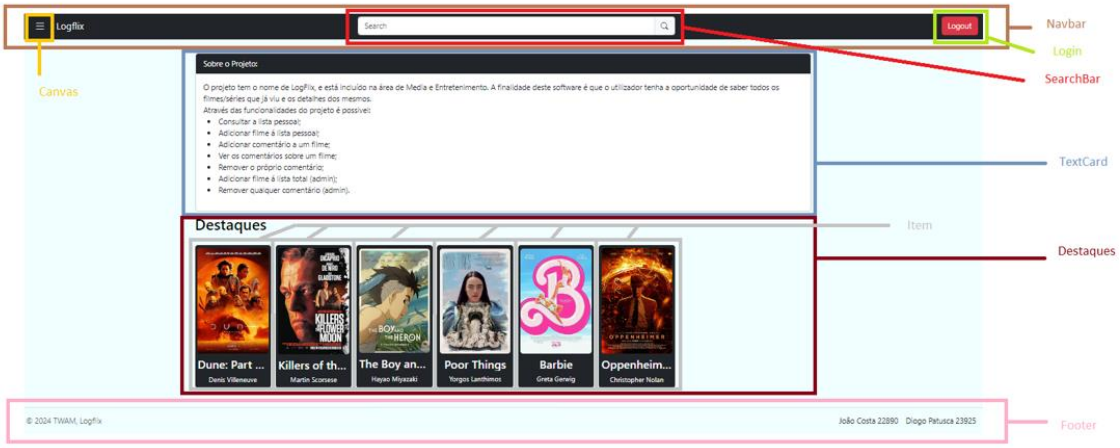

Figura 1- Diagrama da HomePage

```
function HomePage() {
    return (
        <>
            <NavBar />
            <div className="page-background">
                <TextCard />
                <Destaques />
                <Footer />
            </div>
        </>
    );
}
```

Figura 2- Código HomePage

3.1.1 Funcionalidades de suporte para a tarefa 1

Na HomePage, alguns dos componentes da NavBar, como o off canvas e a barra de pesquisa, servem para ajudar a realização da tarefa 1, assim como o componente dos Destaques.

Como é possível observar na figura 3, através das opções do canvas é possível aceder á SearchPage, pressionando o botão "Procurar" o utilizador é redirecionado para a página de pesquisa, caso não pretenda pesquisar o filme pela barra de pesquisa. Já o componente dos Destaques permite selecionar um dos filmes em exibição e consultá-lo, cujo código foi explicado no tópico 3.2.2.

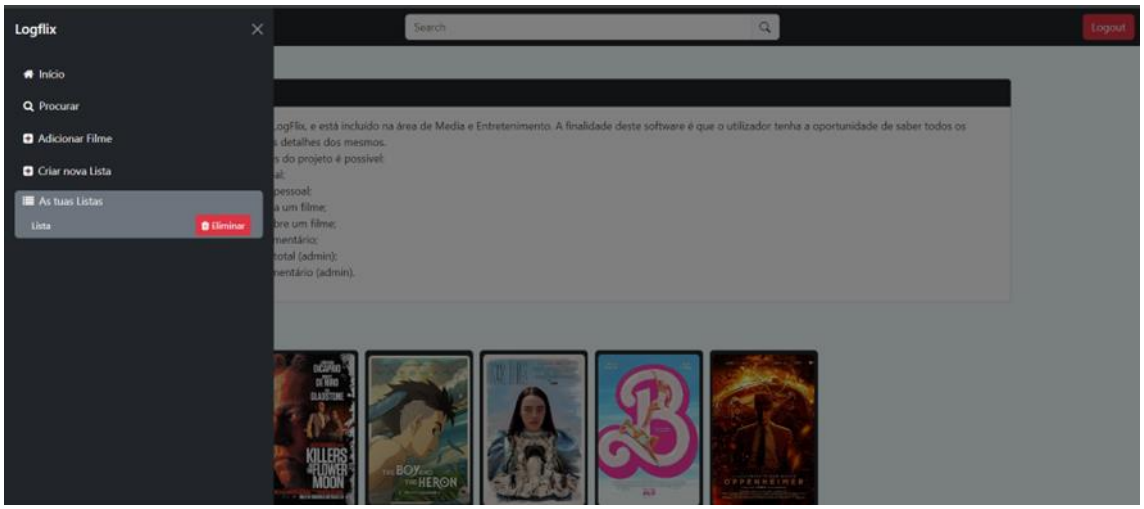

Figura 3- HomePage com off canvas (admin)

```
<ul className="list-unstyled mb-0">
    <li>
        <Button variant="dark" className="button-style" onClick={() => handleClick("/")}>
            <span className="fa fa-home me-2"></span>
            <span>Início</span>
        </Button>
    </li>
    <li>
        <Button variant="dark" className="button-style" onClick={() => handleClick("/search")}>
            <span className="fa fa-search me-2"></span>
            <span>Procurar</span>
        </Button>
    </li>
    <li>
        {(isAdmin) && (
            <Button variant="dark" className="button-style" onClick={handleShowMovieForm}>
                <span className="fa fa-plus-square me-2"></span>
                <span>Adicionar Filme</span>
            </Button>
        )}
    </li>
    <li>
        {(isLoggedIn) && (
            <Button variant="dark" className="button-style" onClick={handleShowList}>
                <span className="fa fa-plus-square me-2"></span>
                <span>Criar nova Lista</span>
            </Button>
        )
        }
    </li>
```

Figura 4- Código Botões do Off Canvas

```
function Destaques() {
    return (
        <div className="container justify-center page-background my-2">
            <h2 className="text-black mb-4">Destaques</h2>
            <ul
                style={{
                    listStyleType: 'none',
                    padding: 0,
                    display: 'flex',
                    flexWrap: 'wrap',
                    gap: '10px',
                }}
            >
                <Item id="tt15239678" />
                <Item id="tt5537002" />
                <Item id="tt6587046" />
                <Item id="tt14230458" />
                <Item id="tt1517268" />
                <Item id="tt15398776" />

            </ul>
        </div>
    );
}
```

Figura 5- Código Destaques

3.1.2 Funcionalidades de suporte á tarefa 2

Para suporte da tarefa 2, todos dos componentes presentes na HomePage encontram-se no off canvas.

Como é possível observar na figura 3, através das opções do canvas é possível criar uma lista, ver as listas criadas, eliminar a lista e caso o utilizador seja administrador pode adicionar um filme á lista do site, as ações referidas podem ser realizadas pressionando os botões ”Criar nova Lista”, após a criação é possível observar a lista criada, para eliminar a lista basta pressionar o botão “eliminar” e confirmar a ação e no caso do administrador pode adicionar o filme carregando no respetivo botão. Passo agora a explicar os excertos do código necessários:

Para criar a lista, comecei por fazer um post na API da lista criada, com o nome escolhido no Modal de criação da list, que foi chamado de listName e verifiquei se o utilizador tinha permissão para realizar essa ação, caso a ação seja concluída com sucesso a lista é criada e as listas do utilizador são atualizadas, caso contrário é exibida uma mensagem de erro. Para atualizar as listas do utilizador é utilizada a const fetchUserLists que faz um GET á API das listas do utilizador, como se pode observar na figura 7.

```
const handleCreateList = async () => {
    try {
        const response = await axios.post(
            'http://localhost:3000/lists',
            { ListName: listName },
            {
                headers: {
                    Authorization: `Bearer ${userData.token}`
                }
            }
        );
        setMessageListForm('Lista criada com sucesso!');
        setTimeout(() => {
            handleCloseList();
        }, 1000);
        fetchUserLists();
    } catch (error) {
        if (error.response && error.response.data) {
            setMessageListForm(error.response.data.message);
        } else {
            setMessageListForm('Não foi possível criar a lista.');
        }
    }
};
```

Figura 6- Código handleCreateList

```
const fetchUserLists = async () => {
    try {
        const response = await axios.get(
            'http://localhost:3000/lists/userLists',
            {
                headers: {
                    Authorization: `Bearer ${userData.token}`
                }
            }
        );
        setUserLists(response.data.Lists);
    } catch (error) {
        console.error('Erro', error);
    }
};

useEffect(() => {
    if (userData && userData.token) {
        fetchUserLists();
    }
}, [userData]);
```

Figura 7- Código fetchUserLists

Para eliminar as listas criadas, através do off canvas foi criada a const deleteList, que faz um DELETE na API, que elimina a lista através do id da lista se o utilizador tiver permissão, e depois as listas do utilizador são atualizadas, caso contrário é exibida uma mensagem de erro.

```
const deleteList = async () => {
    try {
        const response = await axios.delete(
            `http://localhost:3000/lists/${list._id}`,
            {
                headers: {
                    Authorization: `Bearer ${userData.token}`
                }
            }
        );
        fetchUserLists();
        setMessage("Lista deletada com sucesso");
        setTimeout(() => {
            handleClosePopUp();
            setMessage("")
          }, 1000);

    } catch (error) {
        if (error.response && error.response.data) {
            setMessage(error.response.data.message);
        } else {
            setMessage('Erro ao deletar a lista');
        }
    }
}
```

Figura 8- Código deleteList

Por fim, para o administrador pode adicionar um filme á aplicação comecei por meter os dados necessários do formulário na const formData, depois fiz um POST na API para adicionar o filme e passei essa const para o filme ficar com os campos de informação preenchidos, caso a ação seja realizada com sucesso o filme é adicionado, caso contrário é exibida uma mensagem de erro.

```
const handleAddMovie = async () => {
    try {
        setIsLoading(true);
        const formData = new FormData();
        formData.append('Title', movie.Title);
        formData.append('Plot', movie.Plot);
        formData.append('Year', movie.Year);
        formData.append('Type', movie.Type);
        formData.append('Genre', movie.Genre);
        formData.append('Runtime', movie.Runtime);
        formData.append('Rated', movie.Rated);
        formData.append('Director', movie.Director);
        formData.append('Actors', movie.Actors);
        formData.append('Writer', movie.Writer);
        formData.append('Released', movie.Released);
        formData.append('Country', movie.Country);
        formData.append('Awards', movie.Awards);
        formData.append('Metascore', movie.Metascore);
        formData.append('imdbRating', movie.imdbRating);
        formData.append('imdbVotes', movie.imdbVotes);
        formData.append('BoxOffice', movie.BoxOffice);
        formData.append('Poster', movie.Poster);

        const response = await axios.post(
            'http://localhost:3000/movies',
            formData,
            {
                headers: {
                    'Content-Type': 'multipart/form-data',
                    Authorization: `Bearer ${userData.token}`
                }
            }
        );
        setMessageListForm('Filme adicionado com sucesso!');
        setIsLoading(false);
        setTimeout(() => {
            handleCloseMovieForm();
        }, 1000);
    } catch (error) {
        setIsLoading(false);
        if (error.response && error.response.data) {
            setMessageListForm(error.response.data.message);
        } else {
            setMessageListForm('Não foi possivel adicionar o filme.');
        }
    }
};
```

Figura 9- Código handleAddMovie

3.1.3 Acessibilidade da HomePage

Para a página relacionadas com a HomePage, foram implementadas todas as precauções de modo a todas as conformidades do nível de acessibilidade A fossem cumpridas, isto foi conseguido através de vários fatores, como a escolha do contraste entre o fundo negro e o texto branco para facilitar durante a leitura, todos os botões são possíveis de utilizar apenas com o uso do teclado e todos os elementos não textuais tem um texto alternativo para permitir a leitores de ecrã a leitura desses elementos.

Esta página foi devidamente testada no access monitor do governo, obtendo o resultado seguinte:

- Homepage – 8.6
- Homepage com o off canvas aberto – 8.4

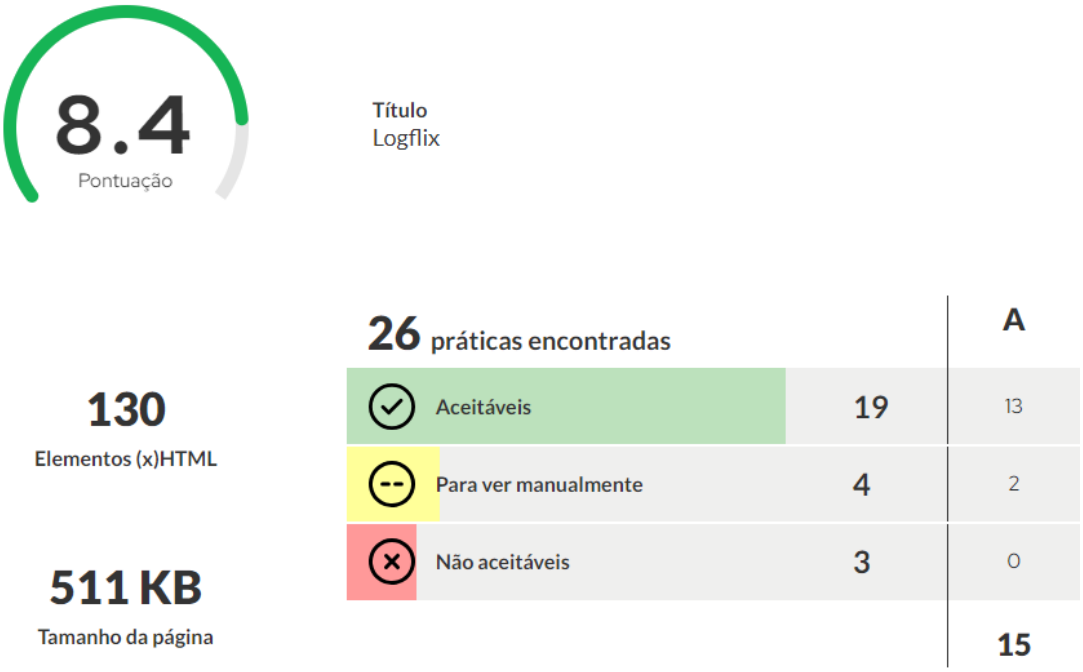

Figura 10- Acessibilidade HomePage com canvas aberto

3.2 Páginas que suportam a Tarefa 1

As páginas que suportam a realização da tarefa 1, que consiste na consulta de filmes, são a HomePage, cujo diagrama de componentes se encontra na figura 1, a SearchPage e a MoviePage. A consulta de um filme pode ser realizada através da barra de pesquisa, para ver o filme pretendido, ou através da lista pessoal, caso já tenha lá o filme que deseja ver.

De seguida irei mostrar os diagramas de componentes da SearchPage e da MoviePage, respetivamente.

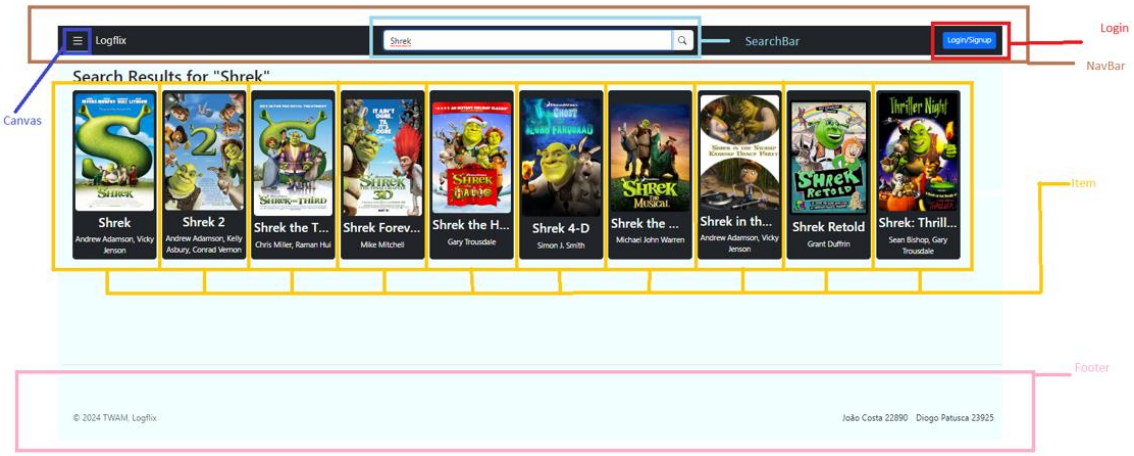

Figura 11- Diagrama SearchPage

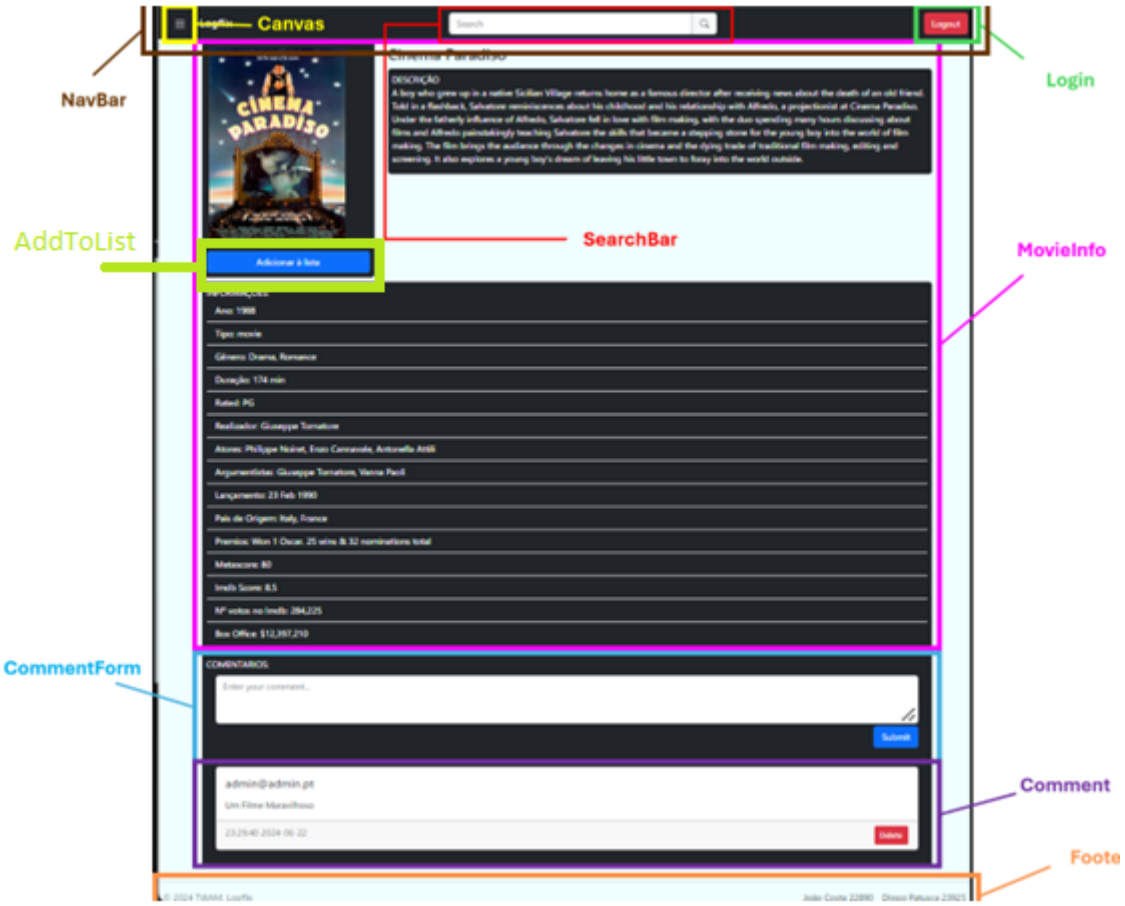

Figura 12- Diagrama da MoviePage

3.2.1 Funcionalidades de suporte da SearchPage

Na página da **SeachPage**: o método de pesquisa. Este método consiste em meter num mapa os resultados que correspondem aos títulos pesquisados e para cada um desses resultados chamar o componente Item com o ID do filme, para estes serem apresentados e ser possível consultá-los.

```
{results.length > 0 ? (
    results.map((result) => (
        <Item id={result.imdbID} />
    ))
) : (
    <li>No results found</li>
)}
```

Figura 13- Código método de Pesquisa

3.2.2 Funcionalidades de suporte da MoviePage

Para o componente **Item**, comecei por declarar o navigate, que será responsável por levar o utilizador para outra página, por declarar o estado do filme que inicialmente é null e o setMovie, que irá atualizar esse estado do filme. Com o uso do useEffect a const fetchMovie é chamada para atualizar a informação do filme através do getMovie que obtém os dados do filme com a ajuda da API e depois atualiza o estado do filme, já a const handleMovie serve para redirecionar o utilizador para outra página. No componente Item é retornado o botão que tem a const handleMovie, que leva o utilizador para a página do filme escolhido, nesse botão é colocado o poster do filme, assim como o título e os diretores.

```
function Item({ id }) {
    const navigate = useNavigate();
    const [movie, setMovie] = useState(null);

    useEffect(() => {
        const fetchMovie = async () => {
            const movieData = await getMovie(id);
            setMovie(movieData);
        };

        fetchMovie();
    }, [id]);

    const handleMovie = (id) => {
        navigate('/movie/' + id);
    };

    const getMovie = async (id) => {
        const response = await axios.get('http://localhost:3000/movies/?plot=full&i=' + id);
        return response.data.movie;
    }
```

Figura 14- Código Item

```
return (
    <>
        <button type="button" className="btn btn-dark p-1 rounded cursor-pointer transition-all" onClick={() =>
        handleMovie(id)} style={{ minWidth: '140px', width: '160px' }}>
            <img src={movie.Poster} className="img-fluid rounded mb-2" alt="Playlist" />
            <h4 className="text-white text-truncate mb-2">{movie.Title} </h4>
            <p className="text-white small mb-0">{movie.Director}</p>
        </button >
    </>
);
```

Figura 15- Return Item

No componente do **MovieInfo** o método de obtenção das informações é igual ao método explicado no componente Item, ou seja, através de uma chamada á API e posteriormente a atualização do estado do filme onde as informações ficam guardadas em "movie". Através do movie, é possível mostrar todos os dados do filme, através do código "{movie.info}", onde info deverá ser substituído pelo nome definido nos models da API.

```
    <div className="col-md-9">
        <h3 className="text-start mt-2">{!isLoading ? movie.Title : 'Loading...'}</h3>
        <div className="d-flex flex-column justify-content-between bg-dark p-2 rounded text-white mt-2" style={{ backgroundColor: '#333' }}
            <div className="text-box mb-2">
                <p className="m-0">DESCRIÇÃO</p>
                {!isLoading ? movie.Plot : 'Loading...'}
            </div>
        </div>
    </div>
</div>

<div className="row mt-2">
    <div className="col-12">
        <div className="text-box bg-dark p-2 rounded text-white" style={{ backgroundColor: '#333' }}>
            <p className="m-0">INFORMAÇÕES:</p>
            <ul className="list-group list-group-flush">
                {!isLoading && (
                    <>
                        <li className="list-group-item bg-dark text-white">Ano: {movie.Year}</li>
                        <li className="list-group-item bg-dark text-white">Tipo: {movie.Type}</li>
                        <li className="list-group-item bg-dark text-white">Gênero: {movie.Genre}</li>
                        <li className="list-group-item bg-dark text-white">Duração: {movie.Runtime}</li>
                        <li className="list-group-item bg-dark text-white">Rated: {movie.Rated}</li>
                        <li className="list-group-item bg-dark text-white">Realizador: {movie.Director}</li>
                        <li className="list-group-item bg-dark text-white">Atores: {movie.Actors}</li>
                        <li className="list-group-item bg-dark text-white">Argumentistas: {movie.Writer}</li>
                        <li className="list-group-item bg-dark text-white">Lançamento: {movie.Released}</li>
                        <li className="list-group-item bg-dark text-white">Pais de Origem: {movie.Country}</li>
                        <li className="list-group-item bg-dark text-white">Premios: {movie.Awards}</li>
                        <li className="list-group-item bg-dark text-white">Metascore: {movie.Metascore}</li>
                        <li className="list-group-item bg-dark text-white">Imdb Score: {movie.imdbRating}</li>
                        <li className="list-group-item bg-dark text-white">Nº votos no Imdb: {movie.imdbVotes}</li>
                        <li className="list-group-item bg-dark text-white">Box Office: {movie.BoxOffice}</li>
                    </>
```

Figura 16- Parte do return do MovieInfo

3.2.3 Acessibilidade da tarefa 1

Para todas as páginas relacionadas com a concretização da tarefa 1, foram implementadas todas as precauções de modo a todas as conformidades do nível de acessibilidade A fossem cumpridas, isto foi conseguido através de vários fatores, como a escolha do contraste entre o fundo negro e o texto branco para facilitar durante a leitura, todos os botões são possíveis de utilizar apenas com o uso do teclado e todos os elementos não textuais tem um texto alternativo para permitir a leitores de ecrã a leitura desses elementos. As páginas relacionadas com esta tarefa foram devidamente testadas no access monitor do governo, obtendo o resultado seguinte:

- SearchPage sem pesquisa – 8.3/10
- SearchPage com pesquisa – 8.5/10
- MoviePage – 8.4/10

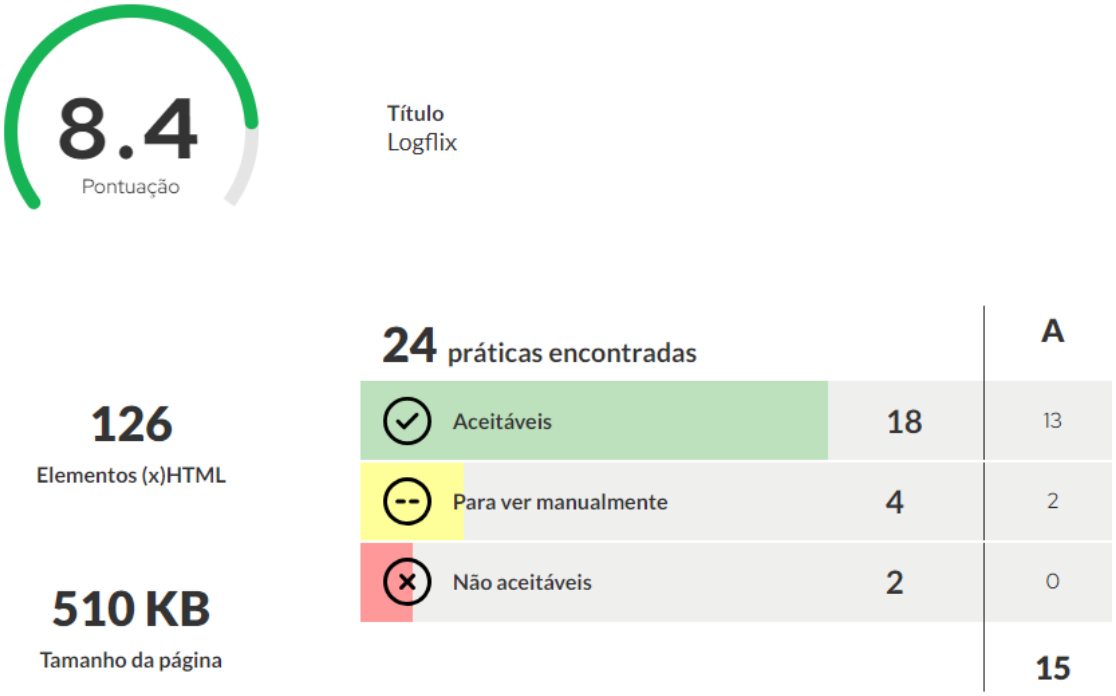

Figura 17- Acessibilidade MoviePage

3.3 Páginas que suportam a Tarefa 2

As páginas que suportam a realização da tarefa 2, que consiste na gestão de filmes, são a HomePage, MoviePage ,cujos diagramas de componentes já foram apresentados nas figuras 1 e 5, respetivamente, e a ListPage. A gestão do filme pode ser feita ao criar uma lista pessoal, eliminar uma lista pessoal, adicionar um filme á lista pessoal, remover um filme da lista pessoal e também de adicionar um filme ao site, sendo que apenas o administrador pode realizar esta última funcionalidade.

De seguida irei mostrar o diagramas de componentes da ListPage.

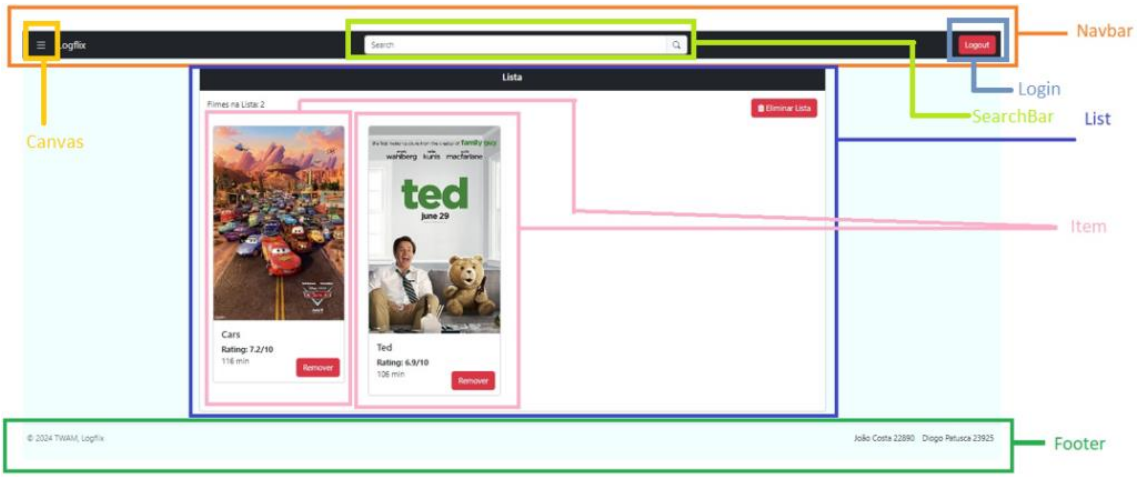

Figura 18- Diagrama da ListPage

3.3.1 Funcionalidades de suporte da ListPage

De seguida, irei apresentar e explicar o código do componente List que poderá causar algumas dúvidas.

Primeiramente foram declaradas todas as variáveis, constantes e o useEffect para atualizar os filmes da lista, depois implementou-se a parte do código responsável por dar get dos filmes na lista, que é realizada de forma semelhante ao get da informação dos filmes, explicado na MoviePage. A parte de eliminar a lista de filmes é apresentada na figura 12, esse excerto de código consiste em chamar o método delete da API para eliminar a lista apenas se a lista for do próprio utilizador, caso isso se verifique então a lista é eliminada, caso contrário aparece um erro. Por fim, retorno a lista.

```
const deleteList = async () => {
  try {
    const response = await axios.delete(
      `http://localhost:3000/lists/${listId}`,
      {
        headers: {
          Authorization: `Bearer ${userData.token}`
        }
      }
    );
    setMessage("Lista deletada com sucesso");
    setTimeout(() => {
      handleClosePopUp();
      setMessage("");
      navigate('/');
    }, 500);

  } catch (error) {
    if (error.response && error.response.data) {
      setMessage(error.response.data.message);
    } else {
      setMessage('Erro ao deletar a lista');
    }
  }
};
```

Figura 19- Código deleteList

3.3.2 Funcionalidades de suporte para a MoviePage

Na MoviePage, apenas resta explicar como funciona o botão de "Adicionar á Lista". Ao selecionar este botão é pretendido para escolher a lista que pretende adicionar o filme, através de um modal, ao confirmar esse model o filme é adicionado á sua lista escolhida, isto foi realizado através do componente AddToList, que faz um POST do filme na lista caso a lista seja do próprio utilizador.

Caso tudo seja realizado com sucesso, é feito um fetch á lista para ser possível ver de imediato a lista atualizada.

```
const addToList = async (e) => {
    e.preventDefault();
    try {
        await axios.post(
            'http://localhost:3000/lists/' + listId,
            {
                "movieId": movieId
            },
            {
                headers: {
                    'Authorization': `Bearer ${userData.token}`,
                }
            }
        );
        setMessage('Adicionado com sucesso');
        setTimeout(() => {
            handleCloseModal();
        }, 1000);
    } catch (error) {
        setError('Erro ao adicionar à lista');
    }
};
```

Figura 20- Código addToList

```jsx
return (
    <>
        <button type="button" className="btn btn-primary" onClick={handleOpenModal}>
            Adicionar à lista
        </button>

        <div className={`modal ${showModal ? 'd-block' : 'd-none'}`} tabIndex={-1}>
            <div className="modal-dialog">
                <div className="modal-content">
                    <div className="modal-header">
                        <h5 className="modal-title">Escolha a lista</h5>
                        <button type="button" className="btn-close" data-bs-dismiss="modal" aria-label="Close" onClick={handleCloseModal}></button>
                    </div>
                    <div className="modal-body">
                        <form onSubmit={handleSubmit}>
                            <div className="form-floating mb-3">
                                <select className="form-select" id="floatingSelect" aria-label="Floating label select example" onChange={handleChange}>
                                <option selected>Selecione a lista</option>
                                    {userLists.map(list => (
                                        <option key={list._id} value={list._id}>{list.ListName}</option>
                                    ))}
                                </select>
                                <label htmlFor="floatingSelect">Listas</label>
                            </div>
                            <div className="modal-footer">
                                <button type="button" className="btn btn-secondary" data-bs-dismiss="modal" onClick={handleCloseModal}>Close</button>
                                <button type="submit" className="btn btn-primary">Submit</button>
                            </div>
                        </form>
                        {error && <p className="text-danger">{error}</p>}
                        {message != "" && <p className="text-success">{message}</p>}
                    </div>
                </div>
            </div>
        </div>
    </>
);
```

Figura 21- Return do AddToList

3.3.3 Acessibilidade da tarefa 2

Para todas as páginas relacionadas com a concretização da tarefa 2, foram implementadas todas as precauções de modo a todas as conformidades do nível de acessibilidade A fossem cumpridas, isto foi conseguido através de vários fatores, como a escolha do contraste entre a cor vivida do botão e o fundo negro para facilitar durante a leitura, todos os botões são possíveis de utilizar apenas com o uso do teclado e todos os elementos não textuais tem um texto alternativo para permitir a leitores de ecrã a leitura desses elementos. As páginas relacionadas com esta tarefa foram devidamente testadas no access monitor do governo, obtendo o resultado seguinte:

- List com filmes - 9.1/10
- List sem filmes – 9.1/10
- MoviePage – 8.4/10

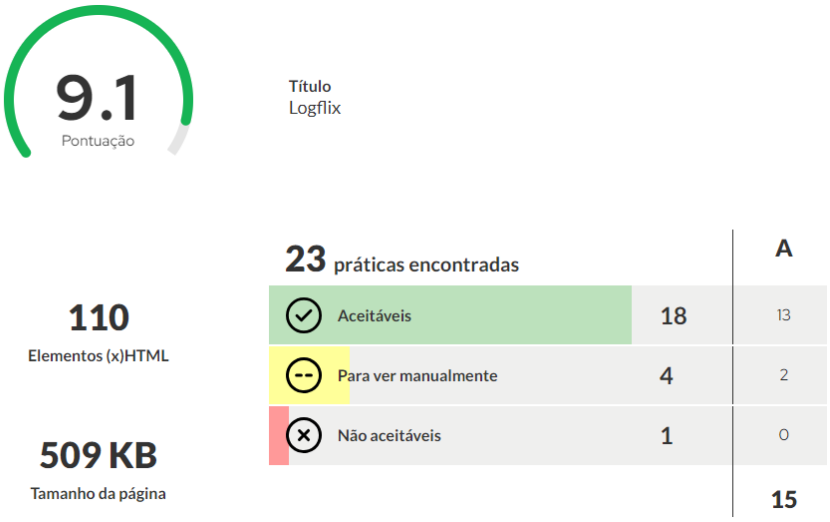

Figura 22- Acessibilidade List com filmes

4 Avaliação das Interfaces / Testes com Utilizadores

Nesta parte do trabalho, descrevem-se os vários tipos de tarefas que foram realizadas antes, durante e após a realização dos testes, de forma a obter feedback por parte dos utilizadores que realizaram os testes de usabilidade.

4.1 Planeamento e desenho dos testes

Nesta secção, são detalhadas as ações desenvolvidas antes dos testes das interfaces gráficas do protótipo funcional. Isso inclui definir objetivos claros, planear as atividades necessárias, tomar decisões sobre o processo de avaliação, criar um guia de testes e produzir a documentação necessária para apoiar essas atividades.

Os objetivos definidos foram identificar as dificuldades de navegação e usabilidade do sistema, receber feedback sobre a experiência geral dos utilizadores, avaliar a eficácia das funcionalidades desenvolvidas e a compreensão do sistema pelos utilizadores. O perfil dos utilizadores a serem recrutados seriam pessoas que gostam de filmes e series, que representem o público-alvo do sistema, se possível, com uma variedade de perfis de modo a obter opiniões mais abrangentes.

As tarefas vão ser avaliadas seguindo algumas métricas como a taxa de conclusão de tarefas, o número de cliques, o tempo demorado e uma avaliação subjetiva do utilizador quanto á dificuldade da tarefa.

Antes da realização do teste, os utilizadores preencheram e assinaram um documento de consentimento e preencheram um inquérito pré teste de modo a fornecerem a sua idade, género, experiência com aplicações WEB, conhecimento sobre filmes, entre outros.

Durante o teste, foi pedido aos utilizadores para realizarem as seguintes tarefas:

- Criar uma conta com role a User : email: numero@numero.user
- Criar uma lista com o nome (número de aluno)
- Adicionar o filme Cars á lista
- Remover a lista criada inicialmente
- Criar uma conta com a role Admin : email: numero@numero.admin
- Adicionar um filme á lista do site

Após a realização dos testes, os utilizadores preencheram um inquérito sobre o seu nível de satisfação ao utilizar o site, a sua opinião sobre a estrutura da aplicação, facilidade de aprendizagem e de utilização, facilidade nas tarefas, entre outros.

Todas as provas de realização dos testes encontram-se disponíveis na seguinte ligação: google drive.

4.2 Execução dos testes

Nesta secção, serão abordadas as partes relacionadas com a execução dos testes de usabilidade das interfaces gráficas do protótipo funcional. O teste foi aplicado a 5 participantes, o que é considerado suficiente para identificar cerca de 90% dos problemas de usabilidade.

Para a realização dos testes, foi escolhido um local tranquilo e confortável (H2O sábado á tarde), onde foram tiradas notas sobre as ações, foram também apontadas as opiniões dos utilizadores e foi gravado o ecrã durante a realização dos testes. O teste foi realizado através do meu computador portátil e tinha a previsão de duração total de 15 minutos, já com o preenchimento dos questionários.

O projeto foi apresentado de forma clara e concisa, destacando as suas funcionalidades principais. Durante realização das tarefas propostas, foi solicitado aos utilizadores para pensarem em voz alta e comentarem as suas ações, dificuldades e sugestões.

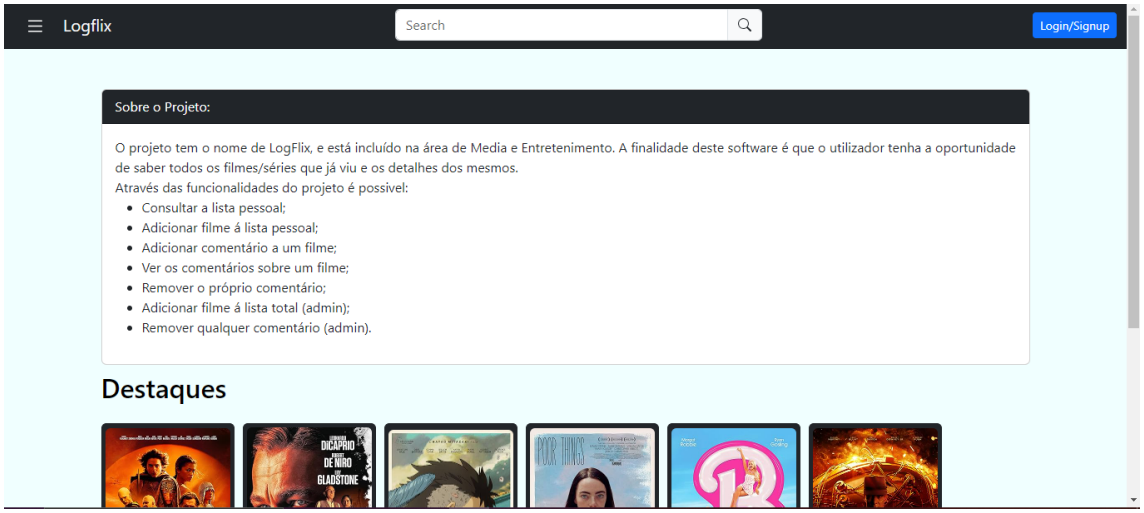

Figura 23- Página inicial para os Testes

4.3 Resultados

Resultados Pré Teste:

Ao analisar as respostas pré-testes, é possível verificar que as idades dos utilizadores testados variam entre os 19 e os 32 anos, onde 80% são do género masculino e 20% do género feminino. Quanto á experiência em aplicações WEB, os participantes dizem ter uma experiência considerável, obtendo uma média de 8.4 num máximo de 10. Quanto á experiência em aplicações semelhantes, todos os utilizadores dizem ter experiência moderada com apenas um participante a ter o nível máximo, tendo a média obtido 7.2 em 10 valores . Na parte de conhecimento de filmes a situação é semelhante ao nível obtido na experiência na questão anterior, tendo esta obtido 7.4 em 10 possíveis, o dispositivo preferível para navegar na WEB é o desktop e os utilizadores consideram ter uma capacidade de aprendizagem moderada/alta obtendo 7.8 /10.

A questão que poderá causar uma maior atenção é a utilização de sistemas semelhantes, uma vez que foi a pergunta com uma média nas respostas menor, por outro lado, a questão com melhor pontuação média foi a experiência na utilização de aplicações WEB.

A análise feita pode ser verificada através da seguinte sequência de imagens:

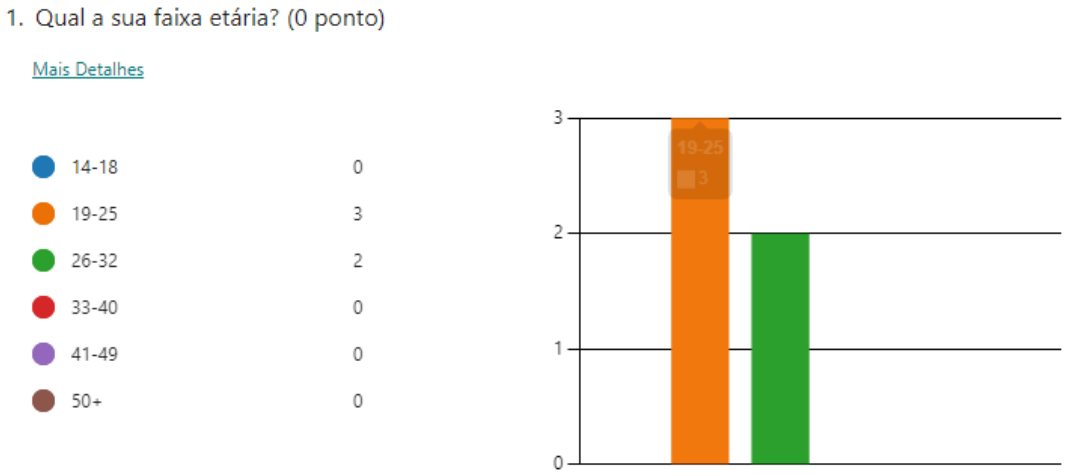

Figura 24- Resultado da Faixa etária

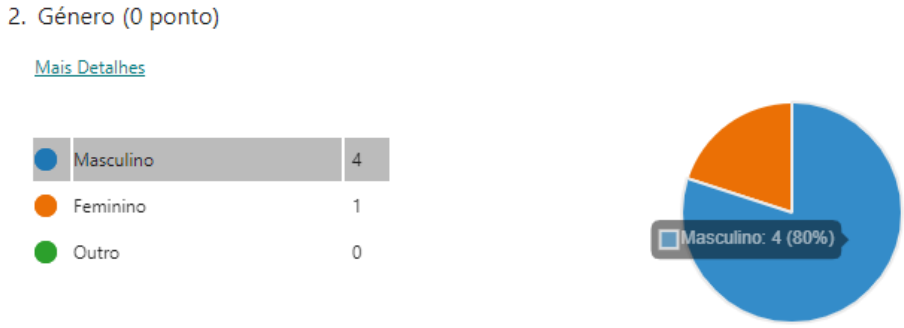

Figura 25-Resultado do Género

3. Qual considera o seu nível de experiência em aplicações WEB? (0 ponto)
   1 - mínimo
   10 - máximo

Mais Detalhes

8.40
Classificação Média

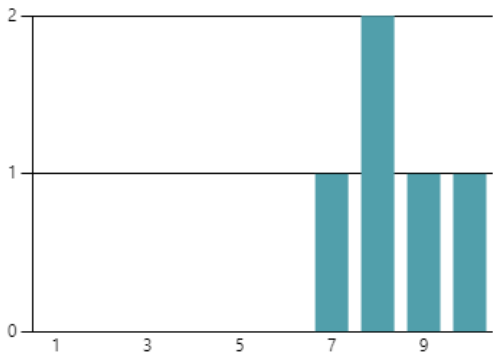

Figura 26- Resultado de experiência na WEB

4. Qual considera o seu nível de experiência em aplicações semelhantes? (0 ponto)
   1 - mínimo
   10 - máximo

Mais Detalhes

7.20
Classificação Média

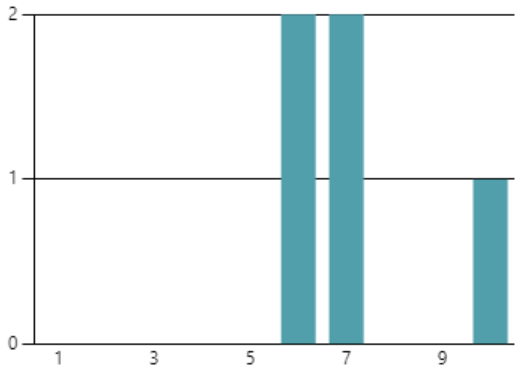

Figura 27- Resultado de experiência em aplicações semelhantes

5. Qual considera o seu conhecimento sobre filmes ou séries? (0 ponto)
   1 - mínimo
   10 - máximo

Mais Detalhes

7.40
Classificação Média

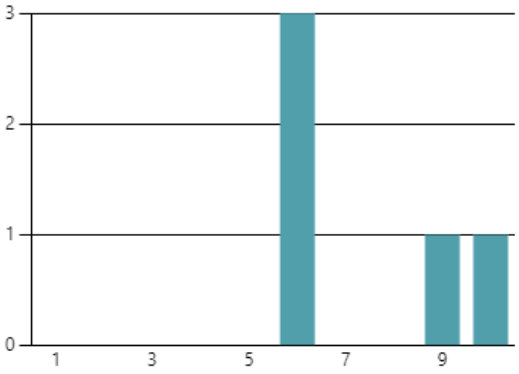

Figura 28- Resultados do conhecimento sobre filmes

6. Qual considera ser o seu dispositivo preferencial de navegação (0 ponto)

Mais Detalhes

Desktop	4
Laptop	1
Tablet	0
Smartphone	0
Outro	0

Figura 29- Resultados do dispositivo de navegação

7. Como avalia a sua habilidade em aprender novos sistemas ou aplicativos? (0 ponto)
   1 - mínimo
   10 - máximo

Mais Detalhes

7.80
Classificação Média

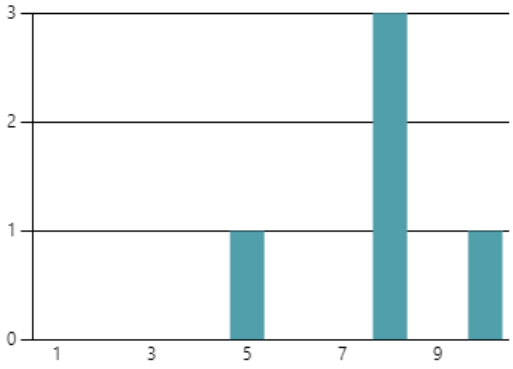

Figura 30- Resultados da capacidade de aprendizagem

Resultados Pós Testes:

Ao analisar as respostas pós-testes, consegui recolher a informação que todos os participantes conseguiram realizar as tarefas pretendidas, apesar de nem sempre com o caminho ótimo.

Ao analisar os resultados das tarefas, é possível observar que apenas 2 das 6 tarefas não foram realizadas com o número de cliques ótimo, sendo estas "Criar conta para o user" e "Adicionar um filme á lista". Os tempos de execução das tarefas estão dentro do esperado, pois as tarefas pretendidas eram de consultas simples, através da imagem que se segue pode-se também observar um decréscimo de cerda de 2,5 segundos ao criar uma conta pela segunda vez.

Tarefas	Media Cliques	Mediana Cliques	Desvio Padrão Cliques	Min Cliques	Max Cliques	Media Tempos	Mediana Tempos	Desvio Padrão Tempos	Min Tempos	Max Tempos
Criar uma conta com role a User: email: numero@numero.user	6,6	7	0,489897949	6	7	22,774	25,67	5,009677035	14,27	27,47
Criar uma lista com o nome = numero de aluno	4	4	0	4	4	8,536	8,98	1,921973985	6,41	11,59
Adicionar o filme Cars á lista	6,4	6	0,8	6	8	15,696	13,62	4,499607094	10,02	22,86
Remover a lista criada inicialmente	3	3	0	3	3	4,66	4,5	1,000039999	3,51	6,49
Criar uma conta com role a admin: email: numero@numero.admin	6	6	0	6	6	20,168	19,54	2,268738857	18,05	24,56
Adicionar um filme á lista do site	4	4	0	4	4	21,382	17,73	7,054925655	12,96	30,83

Figura 31- Resultados Testes

Ao nível das respostas dos utilizadores, todos conseguiram realizar todas as tarefas propostas, além disso o nível de satisfação ao utilizar o portal foi elevado, os utilizadores consideraram também o portal fácil de utilizar, que não é necessária a ajuda nem a aprendizagem de novos conhecimentos, que o portal está consistente e que tem uma aprendizagem fácil. A análise feita pode ser verificada através da seguinte sequência de imagens:

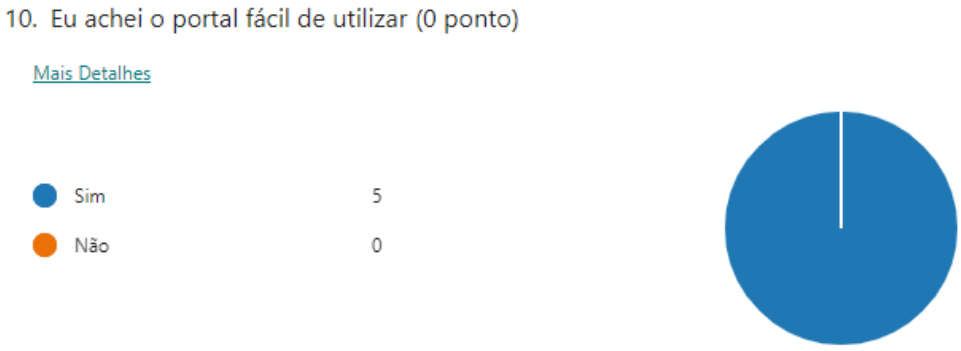

Figura 32- Resultados da facilidade da aplicação

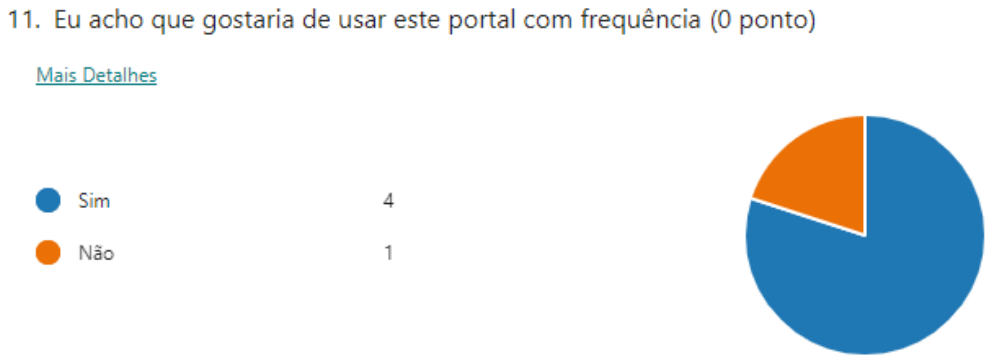

Figura 33- Resultados se gostaria de voltar a utilizar a aplicação

12. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o portal (0 ponto)

Mais Detalhes

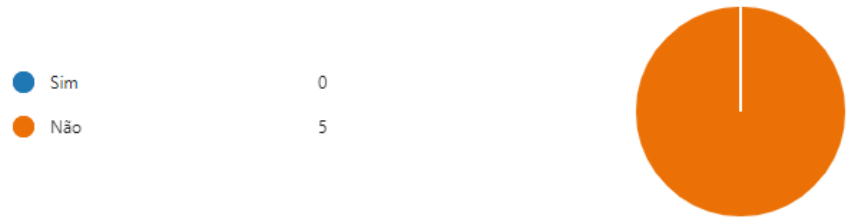

Figura 34- Resultados de se precisaria de ajuda

13. Eu acho que as várias opções do portal estão muito bem integradas (0 ponto)

Mais Detalhes

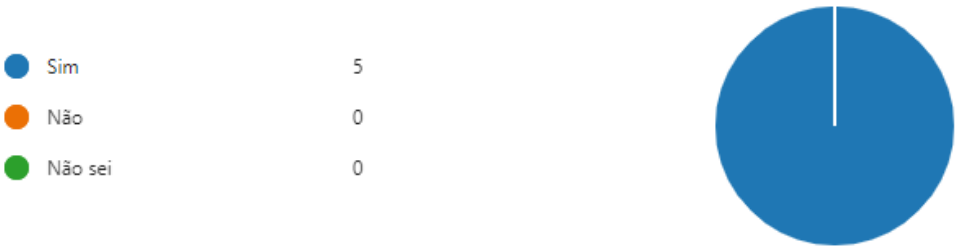

Figura 35- Resultados das opções estão integradas

14. Eu acho que o portal apresenta muita inconsistência (0 ponto)

Mais Detalhes

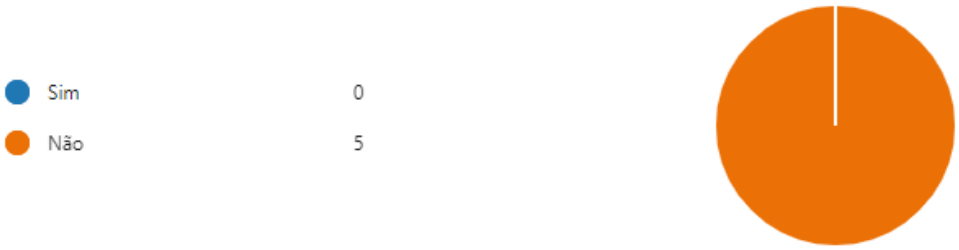

Figura 36-Resultados da inconsistência da aplicação

15. Eu imagino que as pessoas aprenderão como usar este portal rapidamente (0 ponto)

Mais Detalhes

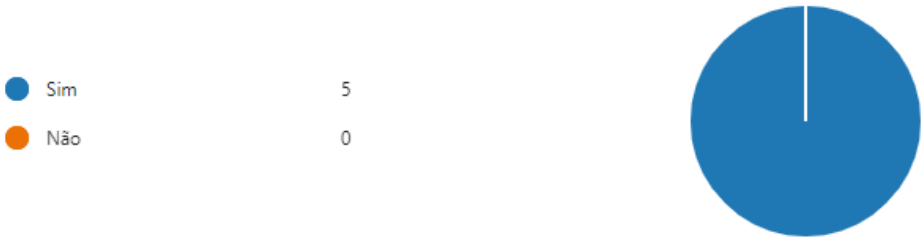

Figura 37- Resultados da facilidade de aprendizagem

16. Eu precisei de aprender várias coisas novas antes de conseguir usar o portal (0 ponto)

Mais Detalhes

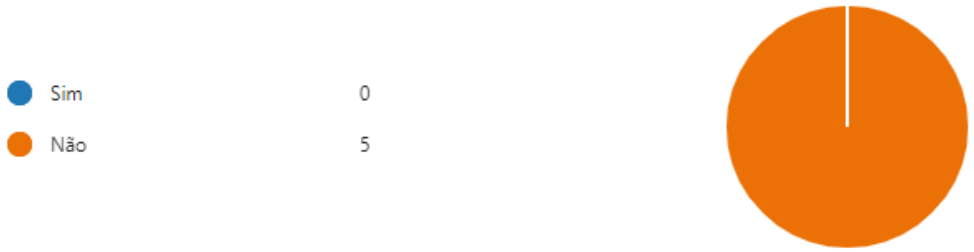

Figura 38- Resultados se precisou de aprender algo novo

5 Conclusões Finais e Trabalho Futuro

Antes de começar a realizar o trabalho em react, foi realizada a nossa restfull API, que foi criada com o apoio de tutorias, esta API foi fundamental uma vez que a API escolhida não tinha DELETE, PATCH e tinha poucos GET e POST, na minha opinião, o facto de ter de aprender rapidamente a realizar esta API considero ter sido a parte mais desafiante do projeto, visto que me sinto mais á vontade a trabalhar com react.

Depois da API, começámos por desenvolver em conjunto os componentes comuns a todas as tarefas, de modo também a existir maior consistência na aplicação, após esses componentes estarem todos criados passei para a parte da criação das páginas principais das minhas tarefas e da adição dos componentes necessários nas páginas em comum. Esta fase foi mais rápida do que o esperado uma vez que já tinha a estrutura completa e faltava apenas colocar os componentes no sítio a que correspondiam.

No final da fase de implementação do projeto LogFlix, testadas as várias funcionalidades principais, além de "testar" a experiência do utilizador. Os testes de usabilidade foram realizados por cinco participantes que indicaram que as funcionalidades desenvolvidas foram bem concebidas, com todos os participantes a conseguir realizar as tarefas propostas, embora que algumas não tenham sido realizadas com o número ótimo de cliques. Através da análise dos resultados pós-testes e da análise aos inquéritos preenchidos pelos utilizadores, concluo que o projeto atingiu os seus objetivos principais de fornecer uma plataforma intuitiva para a gestão de filmes e séries, com uma usabilidade que satisfaz as expectativas dos utilizadores.

Num trabalho futuro, gostaria de aumentar os níveis de acessibilidade do site, uma vez que apenas o nível A está garantido, criar mais funcionalidades para os utilizadores e para o admin, também gostaria de adicionar funcionalidades estéticas á aplicação como a escolha do tema de cores e também gostaria de adicionar uma opção de personalização dos utilizadores.

23/junho/2024

6 Bibliografia

https://www.youtube.com/playlist?list=PL55RiY5tL51q4D-B63KBnygU6opNPFk_q

https://www.youtube.com/playlist?list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d

https://accessmonitor.acessibilidade.gov.pt/

https://omdbapi.com/

Conteúdos lecionados nas aulas teóricas fornecidos pelo docente

https://getbootstrap.com/
https://www.w3schools.com/bootstrap/bootstrap_ver.asp
https://www.w3schools.com/react/default.asp