



React para iniciantes





<title>

AGENDA DO </> </> </> MÓDULO

</title>

 Agenda do módulo

 Anotações



{Criando a página de filme I}

{Criando a página de filme II}

{Criando a página de Wishlist}

{Criação de componente de loading
e refinamento da aplicação}

React para iniciantes:
Refinando a aplicação



<title>

MISSÃO DO MÓDULO

</title>

Criar a página do filme e loading e implementar a função WishList.



{01.}

Criando a página de filme I



Neste módulo, vamos criar a [página Movie](#), que é a página que deve ser aberta quando clicamos no botão “Ver mais” de cada filme.

Vamos criar também a [página WishList](#), que irá conter todos os filmes que adicionarmos à nossa lista de favoritos. E, por último, criaremos o componente Loading, que deve aparecer enquanto aguardamos as informações serem carregadas.



Coloque o React na sua lista de favoritos!



Para iniciar a criação da nossa página de Movie, dentro da **pasta pages**, vamos criar um novo arquivo Movie.tsx. Dentro desse arquivo, vamos iniciar exportando a function Movie().

```
1 export function Movie()
```

É importante que você fique ligado e acompanhe o passo a passo desta etapa na videoaula.



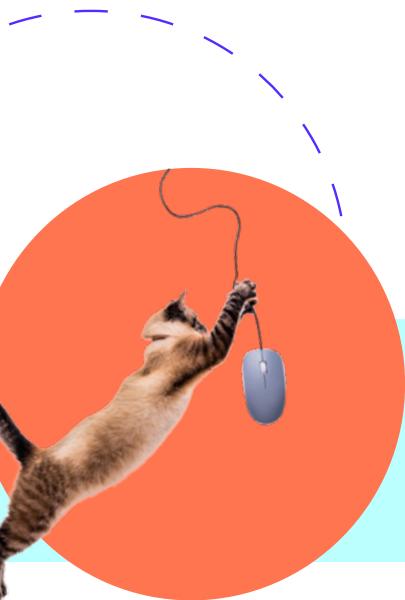
Depois de realizada a etapa inicial, partimos para a próxima. Para simular um filme que fosse recebido pela API, vamos mockar um filme estático na nossa página Movie a fim de desenvolver as regras para o display de cada informação no local correto e substituir alguns textos que temos. Mas, antes disso, precisamos configurar a interface de um filme a ser recebido. Então vamos criar um arquivo **Movie.ts na pasta Interface**.

Inicialmente, vamos criar uma interface para os gêneros, pois alguns filmes possuem mais do que um gênero. Baseados em um resultado da API e seus elementos, vamos criar a interface IGenresProps. Não se esqueça de acompanhar esse passo a passo na videoaula.

```
10010100010101010100010101010010  
01101001111010011100101000101010101  
1 interface IGenresProps {  
2   id: number  
3   name: string  
4 }
```

Depois de criadas as interfaces, vamos voltar ao arquivo Movie.tsx e criar uma constante movie, de forma que seja igual à resposta da API padrão do TMDB. Para mockarmos esse movie, iremos fazer essa declaração antes do return da função Movie().

Lembre-se de ver na videoaula esse passo a passo e também como fazer a estilização para a página.



<title>

PULO DO GATO

</title>

- › Quando estiver trabalhando com APIs, use **optional chaining**, caso algum dos parâmetros corra o risco de não existir.

No h4, vamos criar um map para passar por todos os gêneros de movie.genres, transformando em string e substituindo as vírgulas simples por uma vírgula com um espaço, para facilitar a visualização.

```
1 <h4>
2   {movie ?.genres
3     .map((genre) => genre.name)
4     .toString()
5     .replaceAll(", ", ", ")
6     <span></span>
7     Ano de lançamento
8     <span></span>
9     Tempo de duração
10    </h4>
```



Para termos as horas de duração do filme, devemos dividir o tempo de duração por 60 e remover qualquer resto de minutos, então a const movieHours deve ficar da seguinte forma:

```
1 const movieHours = movie && Number(Math.trunc(movie.runtime / 60).toFixed(0))
```

Para os minutos, vamos subtrair o tempo total de **movieHours**.

```
1 const movieMinutes = movie && movieHours && (movie.runtime - movieHours * 60).toFixed(0)
```

Agora que temos as três const criadas, vamos inseri-las na nossa **div about**.

```
1 <h4>
2   {movie?.genres
3     .map((genre) => genre.name)
4     .toString()
5     .replaceAll(", ", ", ")}
6   <span>•</span>
7   {movieYearRelease}
8   <span>•</span>
9   {`${
10    movieHours
11    movieMinutes
12  }`}
13 </h4>
```

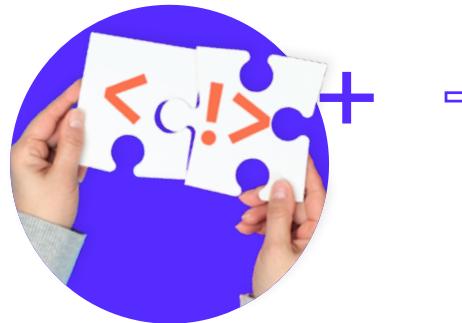


<title>

MÃO + + + NA MASSA

</title>

Crie a estrutura inicial da sua
página Movie.



{02.}

Criando a página de filme II

React para iniciantes:
Refinando a aplicação



Para continuar completando a página Movie, é preciso implementar recursos importantes. Para isso, faz-se necessário retornar aos botões criados e incrementá-los.

Para que a URL não fique indefinidamente copiada no clipboard, é preciso adicionar uma regra para que ela permaneça copiada apenas durante 2,5 segundos.

Não se esqueça de acompanhar esse passo a passo na videoaula.

Após isso, é hora de implementar as funções que foram criadas, introduzindo-as no primeiro botão da **div actions**. Não se esqueça de acompanhar esse passo a passo na videoaula.

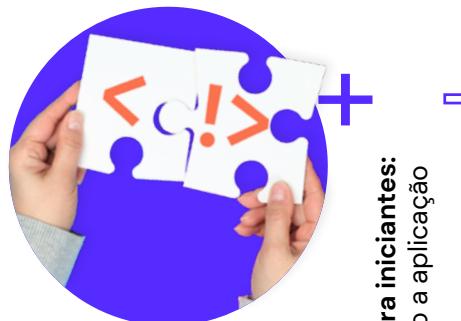
Para configurar sua página Movie de maneira assertiva, consulte a videoaula.

```
<title>
```

MÃO + + + NA MASSA

```
</title>
```

Dê continuidade na criação da estrutura da sua página Movie.





{03.}

Criando a página de Wishlist



Nesta aula, vamos criar a estrutura estática da nossa página de Wishlist (favoritos).

Para isso, iremos mockar um movie na página para usá-lo como base para o desenvolvimento desse recurso.

Na versão final, esses dados devem vir da API e do contexto de `useWishlist` – que será criado no próximo módulo.

Moviecard

Devemos saber o que deve conter na Wishlist: um conjunto de todos os filmes que serão adicionados pelo usuário à lista.



Em primeiro lugar, você deve **importar** o **MovieCard**.

```
1 import { MovieCard } from "../components/MovieCard"
```

Em seguida, você deve **criar e exportar** a **function WishList()**.

```
1 export function WishList() {  
2   //  
3 }
```

Insira o **return** e um **fragment** nele.
Posteriormente, vamos inserir o **styled component** no lugar do fragment.

```
1 export function WishList() {  
2   return (  
3     //  
4     <>  
5     //  
6     </>  
7   )  
8 }
```



Dentro do **fragment**, insira um h1 com o texto Minha Lista e, abaixo dele, uma div com **className="cards"**, que é onde estarão todos os cards dos movies que forem adicionados à Wishlist.

```
1 export function WishList() {  
2  
3   return (  
4     <>  
5       <h1>Minha lista</h1>  
6  
7       <div className="cards">  
8  
9         </div>  
10    </>  
11  )  
12}
```

Dentro da **div**, insira um componente MovieCard.

```
1 export function WishList() {  
2  
3   return (  
4     <>  
5       <h1>Minha lista</h1>  
6  
7       <div className="cards">  
8         <MovieCard/>  
9       </div>  
10      </>  
11  )  
12}
```



Nesse momento, a IDE estará acusando erro no MovieCard por não ter especificado o movie. Para solucionar, é preciso mockar o mesmo movie que foi utilizado durante a criação da página Movie. Portanto, insira-o antes do return e, após isso, importe o IMovieRequestProps.

```
1 const movie : IMovieRequestProps = {genres:[{"id":18,"name":"Drama"}],  
2   "homepage":"http://www.foxmovies.com/movies/fight-club","id":558,  
3   "overview":"A ticking-time-bomb insomniac and a slippery soap salesman channel primal male aggression into a shocking new form of therapy.",  
4   "poster_path":"/p88BW7pdSp6B6Ih7Q24rQJPhJK.jpg","release_date":"1999-10-15","revenue":100853753,  
5   "runtime":139,"title":"Fight Club","vote_average":8.432,"vote_count":24722}
```



<title>

PULO DO GATO

</title>

- Ao mockar algum elemento ou alguma variável, dê preferência por usar dados reais ou o mais próximo possível do dado que será recebido na versão final.



Para continuar com a programação da Wishlist, aconselhamos que assista à aula para melhor compreensão.

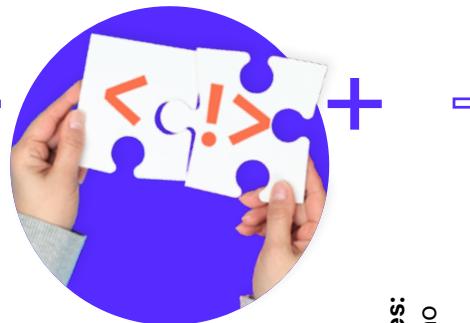
```
1 return (
2   <h1>Minha lista</h1>
3
4   <div className="cards">
5     <MovieCard
6       key={movie.id}
7       movie={movie}
8     >
9   </div>
10 )
11 </div>
12 }
13 }
```

<title>

MÃO + + +
NA MASSA

</title>

- Crie a estrutura da sua página Wishlist.





{04.}

Criação de componente de loading e refinamento da aplicação

Nesta aula, você aprenderá a criar o **componente Loading**, que deve ser usado enquanto aguarda a promise da API ser devolvida. Ele é usado em diversas páginas, como Movie, Search e Home. A ideia é que ele seja um componente com 3 bounces que possuem uma animação.



Vamos iniciar **criando uma pasta Loading em components**.

Dentro dela, criaremos um arquivo index.tsx. Após isso, vamos exportar a function Loading(). Depois, adicione o return dentro da função.

```
1 export function Loading() {  
2  
3 }
```

Em seguida, **insira um fragment que será o Styles.Container** futuramente.

```
1 export function Loading() {  
2   return (  
3     <>  
4     </>  
5   )  
6 }
```

O componente Loading possui uma estrutura bem simples, com apenas 3 divs bounce dentro do fragment. Então, é preciso adicioná-los. Depois, você deve **criar um arquivo styles.ts dentro de Loading**.

```
1 export function Loading() {  
2   return (  
3     <>  
4       <div className="bounce1" />  
5       <div className="bounce2" />  
6       <div className="bounce3" />  
7     </>  
8   )  
9 }
```



Em styles.ts, importe styled from styled-components.

```
1 import styled from "styled-components"
```

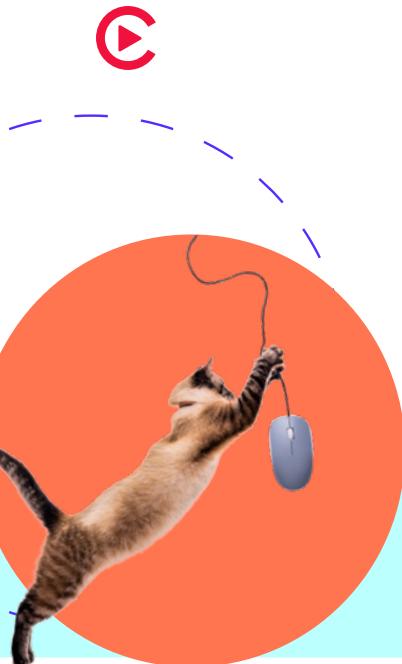
bounces

Para a criação do formato para os bounces, aconselhamos que [assista à aula para uma melhor compreensão](#).

Após isso, vamos importar como Styles no nosso componente Loading e substituir o fragment.

Implementando o componente Loading

Para implementar o componente Loading nas páginas, aconselhamos que [assista à aula para uma melhor compreensão](#).



<title>

PULO DO GATO

</title>

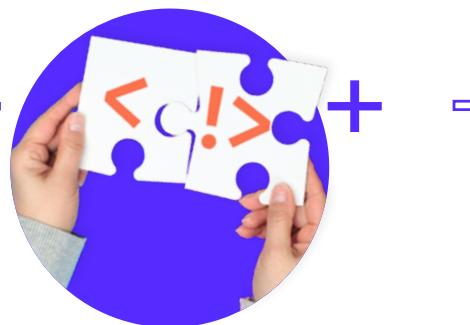
- É importante pensar previamente em todos os locais que um componente pode ser reutilizado.

<title>

MÃO + NA MASSA

</title>

- Crie a estrutura do seu componente Loading e implemente nas páginas Movie, Home e Search.





<title>

DESAFIO CONQUER

</title>



```
if (challenge)
  // start
```

<body>

- Transforme em link o título do filme e a imagem da sua capa para a homepage do filme.
- Mostre ao lado da nota média a quantidade de votos enviados.

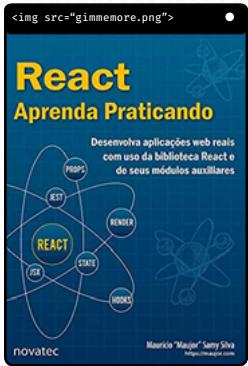
</body>



<title>

QUERO MAIS

</title>



**React - Aprenda
Praticando**
Maurício Samy
Silva



**Primeiros Passos
com React**
Stoyan Stefanov



```
if (interested)  
// gimmelmore
```



Anotações

-
-
-
-
-
-

