



React para iniciantes





<title>

AGENDA DO </> </> </> MÓDULO

</title>



Agenda do módulo



Anotações



{Consumindo a API "The Movie DB"}

{Deixando tudo mais dinâmico}

{Criando página de busca}

{Criando o componente de Card}



O site ProgrammableWeb
tem mais de **24.000**
APIs cadastradas em seu
diretório.

<title>

MISSÃO DO MÓDULO

</title>

Configurar uma **API** e criar o
componente movieCard.





{01.}

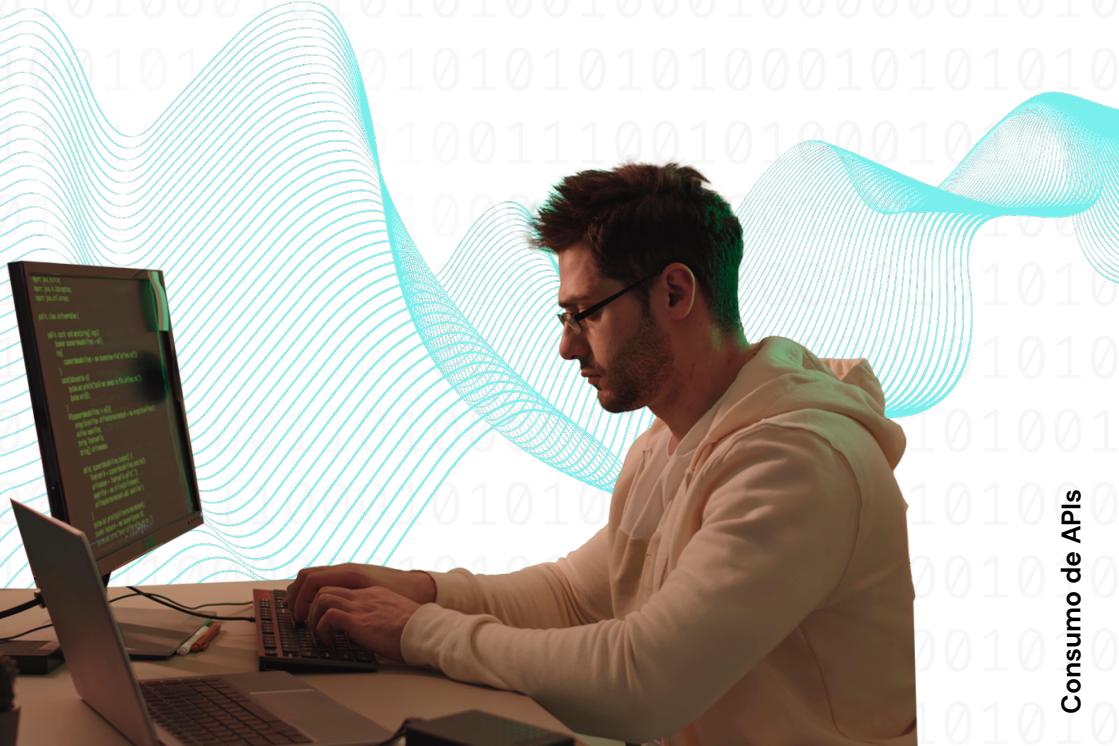
Consumindo a API "The Movie DB"

API

Uma API é uma **forma de comunicação entre sistemas**, funcionando como um formato de dados universal que permite que os aplicativos **troquem essas informações uns com os outros**, de maneira padronizada, independentemente da linguagem em que foram criados.



Consumo de APIs





+

+

+

+

+

+



Uma API que não é
compreensível não é utilizável.

James Gosling



◦ Axios

O Axios é uma biblioteca que auxilia diretamente no consumo da **API**, ajudando na **requisição** das informações recebidas.

◦ Requisições HTTP

Ele é considerado uma **biblioteca de requisições HTTP**. Essas requisições são mensagens enviadas pelo cliente para iniciar uma **ação no servidor**.



jQuery e API Fetch

Outros recursos parecidos e famosos são jQuery e API Fetch. Todos utilizam as Promises, que são classes presentes de forma nativa no JavaScript ES6. A Promise é um objeto que representa uma requisição de operação assíncrona e a eventual conclusão ou falha da mesma.

Síncrono

Cada comando espera o outro concluir para poder executar.

Assíncrono

É necessário terminar as operações em tempo oportuno.

A promise pode estar em diferentes estados:

- > **pending (pendente)**: estado inicial, em que não foi realizada nem rejeitada, ou seja, em processamento.
- > **rejected (rejeitada)**: falha na operação.
- > **fulfilled (realizada)**: sucesso na operação.



DotEnv

O DotEnv é um pacote usado para **carregar automaticamente variáveis de ambiente** através de um arquivo **.env** (ponto env).

Quando um programa é executado, ele recebe informações de fora, vindas do ambiente em que ele está sendo executado. Essas informações de ambiente são passadas implicitamente **via variáveis de ambiente**.

```
<title>
```

PULO DO GATO

```
</title>
```

- › No GitHub, adicione ao arquivo **.gitignore** o arquivo **.env** e ele fará esse trabalho para você.

React para iniciantes
Consumo de APIs

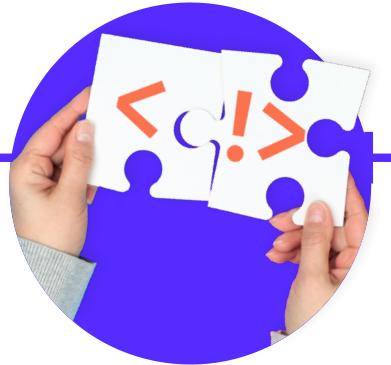




<title>

MÃO + NA MASSA

</title>



> Instale o Axios em seu projeto. Lembre-se que é necessário rodar no projeto o comando **npm install axios**.

> Instale o DotEnv e crie um arquivo .env na raiz do projeto. Depois, crie uma conta no site <https://www.themoviedb.org/> para obter a API Key e a URL para imagens.

{02.}

Deixando tudo mais dinâmico



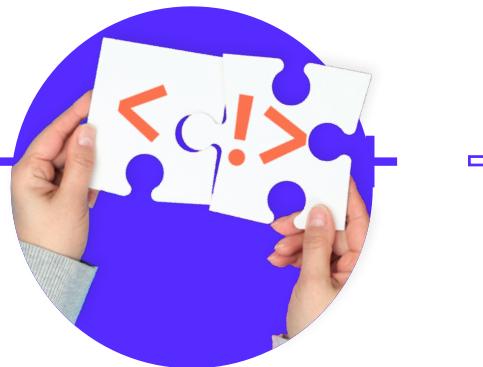
Para deixar a aplicação mais dinâmica, é necessário **importar a API**. Você pode importar em diferentes páginas, como:

- > **home**;
- > **search**;
- > **movie**.

```
<title>
```

MÃO + + +
NA MASSA

```
</title>
```



Agora é a sua vez!

A partir do que foi ensinado na aula, dinamize a **página Home**, utilizando as importações necessárias e criando a interface da Home que será utilizada.



{03.}

Criando página de busca

Elementos que compõem a página de busca:

- useEffect.
- useState.
- useSearchParams.



<title>

MÃO + + NA MASSA

</title>



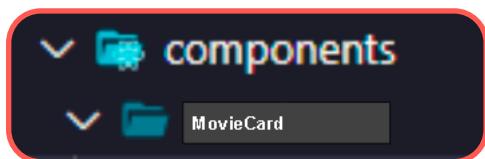
Crie sua página de busca, a partir do que foi visto em aula. O arquivo Search.tsx deve ser encontrado dentro da pasta **pages**.

{04.}

Criando o componente de card



Por meio do **componente MovieCard**, é possível apresentar o filme em uma caixa, contando com a **sinopse**, **tempo de duração** e se o **usuário deseja ver mais sobre o filme** ou adicioná-lo à sua lista de desejos por meio de botões.



Para iniciar, é preciso criar uma pasta com o **nome MovieCard**, dentro da pasta components.

Dentro da pasta MovieCard, é preciso criar três arquivos: o primeiro arquivo será em **tsx**, o segundo arquivo será o de **estilos em ts** e o terceiro arquivo será o de **tipos**.



Depois, faz-se necessário
**importar as bibliotecas,
componentes, tipagens e
também o arquivo de estilos.**

```
1 import { useNavigate } from "react-router-dom"
2 import { FiCameraOff, FiCheck, FiPlus } from "react-icons/fi"
3
4 import { Button } from "../Button"
5
6 import { ButtonVariants } from "../Button/types"
7 import { IMovieCardProps } from "./types"
8
9 import * as Styles from "./styles"
```

Além disso, é preciso usar alguns ícones, como: **FiCameraOff**, **FiCheck** e **FiPlus**; além do componente **Button**.



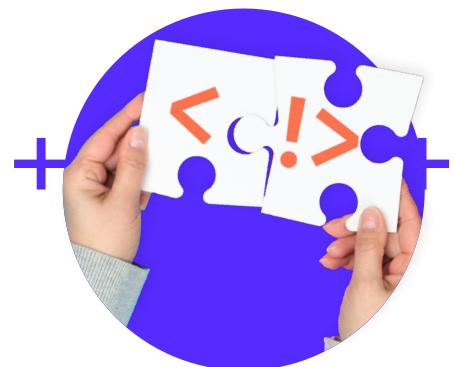
<title>

MÃO + + NA MASSA

</title>

Agora é a sua vez!

Crie o seu componente **MovieCard**!





<title>

DESAFIO CONQUER

</title>



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

<body>

Crie uma Pokédex, uma lista onde podemos buscar os Pokémon existentes e ver suas informações.

Crie uma página com um campo de busca que irá consumir da API o termo buscado.

Crie um componente que conterá os detalhes das informações de cada Pokémon.

No seu componente de detalhes do Pokémon, devem constar:

- Imagem do Pokémon.
- Número na Pokédex.
- Tipo.

Utilize a API: <https://pokeapi.co/>.

</body>



<title>

QUERO MAIS

</title>



Guia de informações
sobre API



```
if (interested)  
// gimmemore
```



Anotações

-
-
-
-
-
-

