API com ReactJS

Programa
3000 TALENTOS TI

O que vamos ver hoje?

- API
- Fetch
- Axios

API com fetch

Fetch

- É a função nativa do Javascript para o consumo de api’s.
- Faz uma consulta e resulta em uma promessa de que algo será retornado, essa promessa é chamada de Promisse.
- Essa promessa pode tanto ser boa, ter retornado os dados, quanto ter falhado por algum motivo - como no caso da conexão com o servidor cair.

Fetch

```
useEffect(() => {
    async function carregaRepositorios() {
      const response = await fetch(
        "https://api.github.com/users/rafaelkasper/repos"
      );
      const repositorios = await response.json();

      setRepositorio(repositorios);
    }
    carregaRepositorios();
  }, []);
```

API com Axios

Axios

- É um cliente HTTP baseado em promessas totalmente agnóstico, ou seja, que não depende de frameworks e bibliotecas.
- Ele é super maleável, podendo ser utilizado do lado do cliente (React, Vue, etc) e do lado do backend (express, nest, etc).

Axios

- Fornece vários atalhos para cada tipo de requisição:
  - axios.get('url')
  - axios.delete('url')
  - axios.post('url', { ...dados })
  - axios.put('url', { ...dados })

Axios

- Para termos o axios disponível em nossa aplicação, basta instalá-lo com via NPM:
  - npm i axios
- Depois é necessário importá-lo no componente que fará requisições:
  - import axios from 'axios'

Axios

- Como é baseado em promessas, podemos lidar com o sucesso e falha da chamada, usando o bloco then e o catch.

Axios

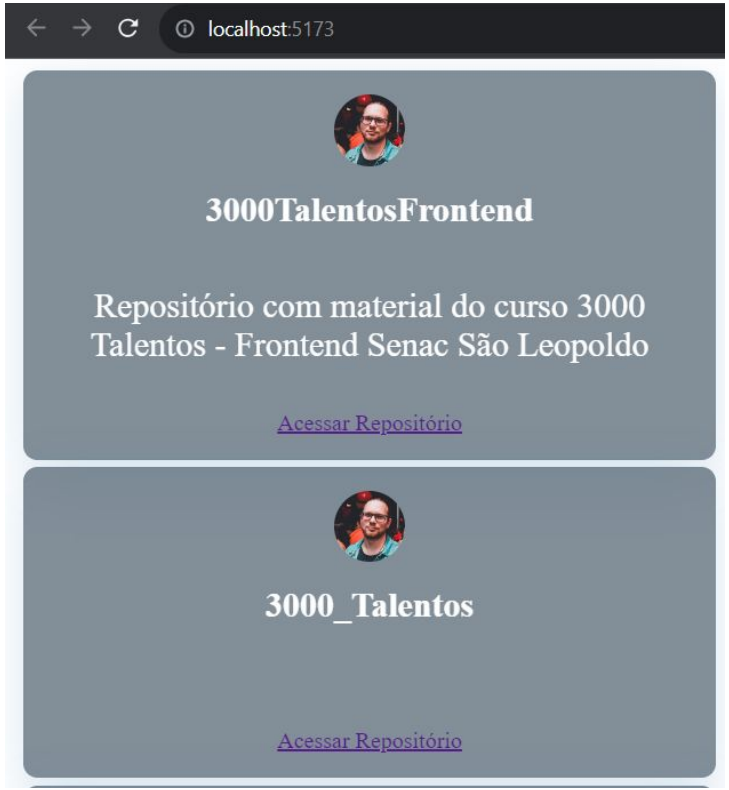
- Uma requisição completa com useEffect, Axios, useState e jsx ficaria assim:

Axios

```jsx
import { useState, useEffect } from "react";
import axios from "axios";
import Card from "../../components/card";

function Inicial() {
  const [repositoriosList, setRepositoriosList] = useState([]);
  useEffect(() => {
    async function carregaRepositorios() {
      await axios
        .get("https://api.github.com/users/rafaelkasper/repos")
        .then(function (response) {
          setRepositoriosList(response.data);
        })
        .catch(function (error) {
          console.log(error);
        });
    }
    carregaRepositorios();
  }, []);

  return (
    <div>
      {repositoriosList.map((repositorio) => (
        <Card
          key={repositorio.id}
          title={repositorio.name}
          content={repositorio.description}
          image={repositorio.owner.avatar_url}
          link={repositorio.html_url}
        />
      ))}
    </div>
  );
}

export default Inicial;
```

```jsx
import "./index.css";
const Card = ({ title, content, image, link }) => {
  return (
    <article className="card">
      <img src={image} alt="imagem do card" />
      <div>
        <h2>{title}</h2>
        <p>{content}</p>
      </div>
      <a href={link}>Acessar Repositório</a>
    </article>
  );
};

export default Card;
```

Axios


localhost:5173
3000TalentosFrontend
Repositório com material do curso 3000 Talentos - Frontend Senac São Leopoldo
Acessar Repositório
3000_Talentos
Acessar Repositório

Axios

- Podemos também resumir a requisição da seguinte forma:

```
useEffect(() => {
  async function carregaRepositorios() {
    try {
      const response = await axios.get(
        "https://api.github.com/users/rafaelkasper/repos"
      );
      setRepositoriosList(response.data);
    } catch (error) {
      console.log(error);
    }
  }
  carregaRepositorios();
}, []);
```

Resumo

Resumo 📓

- Podemos utilizar Fetch ou Axios para fazer requisições.
- Com o uso de useState e useEffect podemos criar telas de visualização dinâmica de acordo com os dados retornados.

Dúvidas? 

Programa

3000 TALENTOS TI

Obrigado(a)!