Victor Silvestre França de Oliveira  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Código Original:**

import React, { useState, useEffect } from 'react';

function App() {

const [informacao, setInformacao] = useState([]);

const [isLoading, setIsLoading] = useState(true);

useEffect(() => {

fetch('https://api.example.com/data [api.example.com]')

.then(response => response.json())

.then(informacao => {

setInformacao(informacao)

setIsLoading(false);

})

.catch(error => {

console.error("Error fetching data: ", error);

});

}, []);

if (isLoading) {

return <p>Loading...</p>;

}

return (

<div>

<h1>Data List</h1>

<ul>

{informacao.map(item => (

<li key={item.id [item.id]}>{item.name [item.name]}</li>

))}

</ul>

</div>

);

}

export default App;

**Código Refatorado:**

import React, { useState, useEffect } from 'react';

function App() {

const [informacao, setInformacao] = useState([]);

const [isLoading, setIsLoading] = useState(true);

useEffect(() => {

const fetchData = async () => {

try {

const res = await fetch('https://api.example.com/data');

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const data = await response.json();

setInformacao(data);

} catch (error) {

console.error("Error fetching data:", error);

} finally {

setIsLoading(false);

}

};

fetchData();

}, []);

if (isLoading) {

return <p>Loading...</p>;

}

return (

<div>

<h1>Data List</h1>

<ul>

{informacao.map(item => (

<li key={item.id}>{item.name}</li>

))}

</ul>

</div>

);

}

export default App;

**O que foi refatorado:**

1. Removi a parte extra da url API: [api.example.com]
2. Criei uma constante ‘data’ para evitar confusão e acrescentei a lógica para função assíncrona (`async/await`) para tratamento dos dados
3. if (!response.ok) { throw new Error(`HTTP error! Status: ${response.status}`); }

**R**: Adicionei uma verificação para garantir que a resposta da API é bem sucedida, se não for é lançado com erro com status HTTP

1. Then e catch usado diretamente no useEffect

Inseri a lógica da chamada API em uma função assíncrona(‘fetchData’). Lidando com a requisição e tratamento dos dados, enquanto ‘useEffect’ só chama a função.

1. <li key={item.id}>{item.name}</li>

**R:** A utilização de [item.id] e [item.name] estava errada, foi removido os colchetes e ‘key’ deve ser um valor único para cada item na lista, e ‘item.name’ é o texto que será exibido

1. Adição do bloco finally