Prova final 2a Etapa Framework

Teórico

**-> Hooks:** Foram adicionados para simplificar o código, sem necessidade de usar uma classe.

*Exemplo*: useEffect, useState.

**-> UseState:** Adiciona um estado a componentes, que pode ser atualizado ao longo do tempo.

Exemplo:

const [count, setCount] = useState(0); // Inicializa o estado `count` com o valor 0

setCount(count + 1); // Atualiza o estado `count`

**-> UseEffect:** Ele executa algo quando reporta que algum componente mudou.

Exemplo:

const [count, setCount] = useState(0); // Seta um contador

useEffect(() => {

document.title = `Você clicou ${count} vezes`; // Atualiza o título da página

}, [count]); // Executa o efeito sempre que `count` mudar

return (

<div>

<p>Você clicou {count} vezes</p>

<button onClick={() => setCount(count + 1)}>Clique aqui</button> </div>

);

Prático

1. Crie a pasta no C:
2. Abra a pasta no VSCode
3. Escreva no terminal:

npm create vite@latest nomedoprojeto

cd nomedoprojeto

npm install

npm install react-router-dom

*// copiar o código App.jsx para o App.jsx do seu projeto*

npm run dev

App.jsx

```javascript

import { useState, useEffect } from 'react';

import { BrowserRouter as Router, Routes, Route, useNavigate } from 'react-router-dom';

import './App.css';

// Componente 1: Exemplos de useState e useEffect

function Exemplos() {

const [count, setCount] = useState(0);

const [message, setMessage] = useState('');

useEffect(() => {

const fetchData = () => {

setTimeout(() => {

setMessage('Dados carregados!');

}, 2000);

};

fetchData();

}, []);

useEffect(() => {

document.title = `Você clicou ${count} vezes`;

}, [count]);

return (

<div>

<h1>Exemplos</h1>

<div className="card">

<h3>Exemplo de useState:</h3>

<button onClick={() => setCount(count + 1)}>

Você clicou {count} vezes.

</button>

<h3>Exemplo de useEffect:</h3>

<p>1. {message ? message : 'Carregando...'}</p>

<p>2. Olhe o título da página</p>

</div>

</div>

);

}

// Componente 2: Revisão de previsão do tempo

function Revisao() {

const [city, setCity] = useState(''); // Declara a variavel `city` como string

const [weather, setWeather] = useState(null); // Declara a variavel `weather` como string

const [error, setError] = useState(''); // Declara a variavel `error` como nulo

const API\_KEY = 'c966e4dcdb03ee1244e7abcd81963546'; // Substitua pela sua chave de API

const fetchWeather = async (cityName) => {

try {

const response = await fetch(

`https://api.openweathermap.org/data/2.5/weather?q=${cityName}&appid=${API\_KEY}&units=metric` // Url da api

);

if (!response.ok) { // Se a resposta do fetch for diferente de 'ok',...

throw new Error('Cidade não encontrada'); // ... joga um erro.

}

const data = await response.json(); // Dados pegos da resposta transformados em json

console.log(data) // Mostra os dados no console para poder lidar com a estrutura

setWeather(data); // Seta e preenche a variável `weather` da data.

setError(''); // Seta a variável `erro` para vazia, já que não foi encontrado nenhum erro.

} catch (err) { // Se algo der errado no fetch, ...

setWeather(null); // ... a váriavel `weather` recebe nulo

setError(err.message); // e a váriavel `error` recebe a mensagem de erro do catch

}

};

const handleSearch = () => { // Cria uma função anônima e temporaria chamada `handleSearch` que seá chamada ao clique do botão de busca

if (city) { // Se `city` estiver setada, ou oo input com alguma cidade ...

fetchWeather(city); // `fetchWeather` vai receber a cidade do input

} else {

setError('Por favor, digite o nome da cidade'); // O erro vai aparecer para indicar ao usuário que ele precisa digitar alguma cidade

}

};

return (

<div>

<h1>Previsão do Tempo</h1>

<input type="text" placeholder="Digite o nome da cidade"

value={city} // Faz o valor do input receber a `city` escrita

onChange={(e) => setCity(e.target.value)} // Sempre que o valor de entrada mudar, altera o valor do campo, e atualiza o valor de city

/>

<button onClick={handleSearch}>Buscar</button>

<p>{error}</p> {/\* Mostra o errro \*/}

{weather && ( // Renderiza apenas se weather não for nulo

<div>

<h2>{weather.name} ({weather.sys.country})</h2> {/\* Mostra o dado.name que agora esta alocado em `weather` e o país. OBS: Tudo depende da estrutura do json \*/}

<p>Temperatura: {weather.main.temp}°C</p> {/\* Mostra a temperatura \*/}

<p>Condições: {weather.weather[0].description}</p> {/\* Mostra as condições \*/}

<img

src={`http://openweathermap.org/img/w/${weather.weather[0].icon}.png`}

alt={weather.weather[0].description}

/> {/\* Recebe o incone que vem do json. Só lendo a documentação da API para saber qual a url \*/}

</div>

)}

</div>

);

}

// Componente inicial

function Principal(){

const navigate = useNavigate();

const handleClickRevisao = () => {

navigate('/revisao'); // Navega para a rota de revisão

};

const handleClickExemplos = () => {

navigate('/exemplos'); // Navega para a rota de revisão

};

return (

<section>

<h1>Página Inicial</h1>

<div>

<h3>Exemplos de useEffect e useState:</h3>

<button onClick={handleClickExemplos}>Clique aqui</button>

</div>

<div>

<h3>Revisão da prova:</h3>

<button onClick={handleClickRevisao}>Clique aqui</button>

</div>

</section>

);

}

// Componente Principal: App

function App() {

return (

<Router>

<Routes>

<Route path="/" element={<Principal />} />

<Route path="/revisao" element={<Revisao />} />

<Route path="/exemplos" element={<Exemplos />} />

</Routes>

</Router>

);

}

export default App;

```