Navegando entre os componentes

Photo by Quino Al on Unsplash

Módulo 3 - Aula 7 - HARD

RESILIA | Senac

Axios - PUT

```
import axios from "axios";

function App() {
  function putAPI() {
    axios
      .put("LINK", {BODY})
      .then((resposta) => console.log(resposta))
      .catch((erro) => console.log(erro));
  }
}
```

Axios - DELETE

```
import axios from "axios";

function App() {
  function deleteAPI() {
    axios
      .delete("LINK")
      .then((resposta) => console.log(resposta))
      .catch((erro) => console.log(erro));
  }
}
```

Hook de Efeito

useEffect

É um Hook para lidar com efeitos colaterais do componente e, portanto, está diretamente atrelado ao ciclo de vida.

Com ele podemos lidar com todas as fases do ciclo de vida.

useEffect

É uma função que recebe como argumento uma função callback e um array de dependências, o qual você incluirá variáveis das quais o useEffect dependerá e, assim, a função ficará aguardando a mudança dessa variável.

SPAs

SPA significa Single Page Application, ou seja, uma aplicação de apenas uma página.
É uma implementação de aplicação web que carrega em apenas um documento (HTML) e, em seguida, atualiza o conteúdo do documento por meio de APIs JavaScript, quando um conteúdo diferente deve ser mostrado.

SPAs

Isso permite que usuários usem sites sem precisar carregar páginas totalmente novas do servidor, o que pode resultar em ganhos de desempenho, experiência, etc.

Ex.: Gmail

SPAs

O React foi planejado para atender à necessidade de criação de SPAs mais dinâmicas e performáticas e ele cumpre isso, assim como outras tecnologias.

SPAs

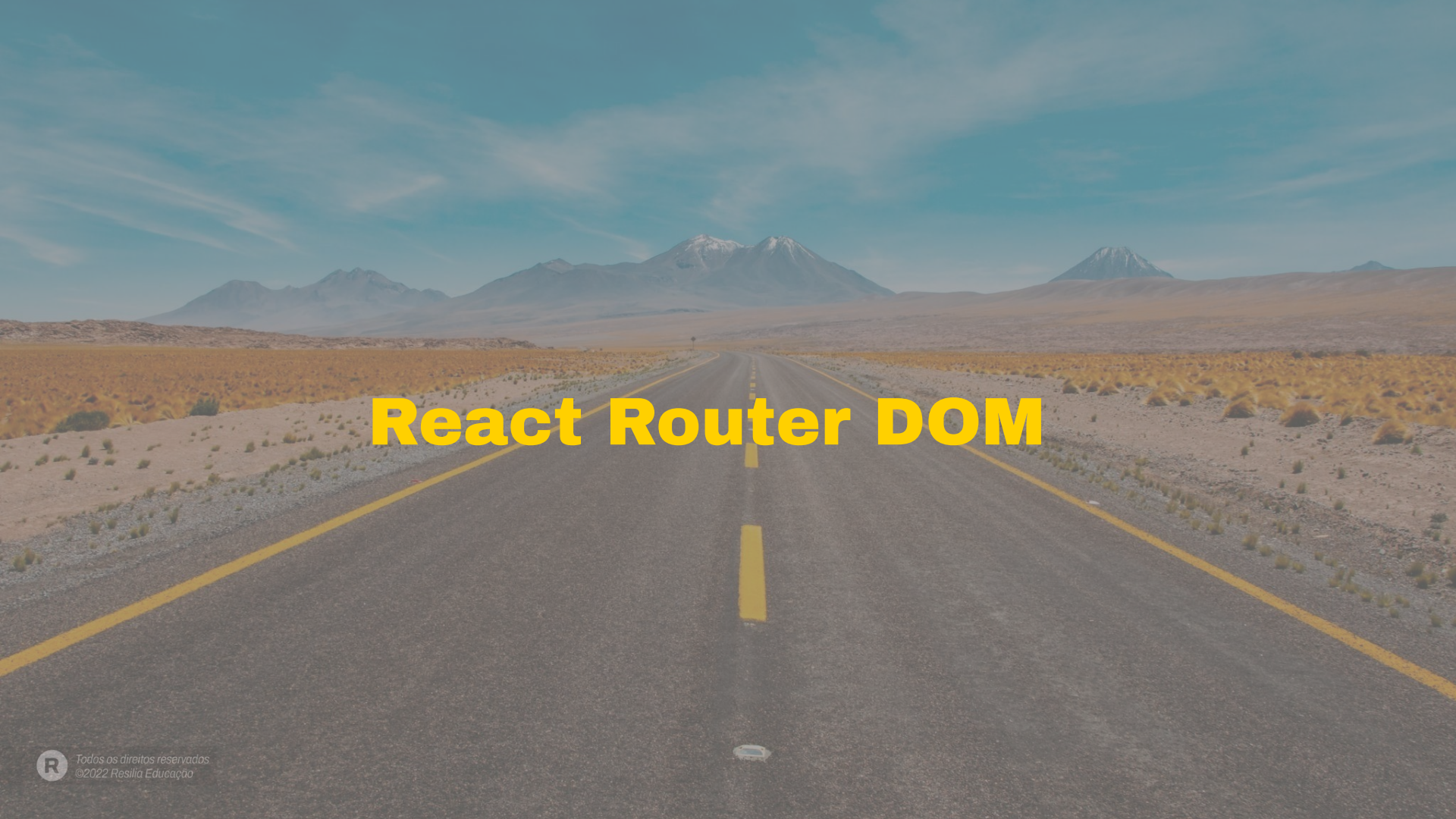
Porém, SPAs também trazem desvantagens.
Manter o estado exige um maior esforço, implementar a navegação é trabalhoso e manter uma performance significativa também não é fácil.

TODO: Save Effects

Dentro do nosso componente Tarefas.jsx:

- Iremos usar o useState para:
  - Executar a função getApi
  - A função deve ser executada somente na primeira vez que entramos na página.

- No componente NovaTarefa.jsx:
  - Criar um botão de “Enviar”
  - Adicionar a função postApi, que realizará um post para nosso json server
  - deve-se salvar a tarefa em nosso json server ao clicar no botão.



react-router-dom

- Permite roteamento dinâmico
- Navegação dentro da aplicação sem recarregar a página
- Baseado em componentes (quase tudo é componente)

react-router-dom

Roteamento dinâmico vs Roteamento estático

No estático as rotas são declaradas antes da renderização da aplicação, ou seja, na inicialização

No dinâmico o roteamento é feito durante a renderização da aplicação, sem necessitar de uma convenção ou configuração fora da aplicação em execução

react-router-dom - Instalação

Na pasta do projeto React execute

```
npm install react-router-dom@6
```

Vamos usar a versão 6!

react-router-dom - Como usar?

São 3 componentes principais:
<BrowserRouter />
<Routes />
<Route />

Podemos declarar no arquivo main.jsx (ou index.js, se tiver usando CRA) já que ele que faz a injeção do código no HTML.

Ou no próprio componente App, transformando-o em um grande gerenciador de rotas

react-router-dom - BrowserRouter

Componente pai que deve envolver tudo que o react-router for fazer, ou seja, tudo que for relacionado a rotas deve estar declarado dentro dele.

react-router-dom - Routes

Componente que envolve todas as rotas da aplicação, responsável por fazer a troca de componentes de acordo com a rota.

react-router-dom - Route

Componente de rota em si, é onde declaramos o path (caminho) e se esse caminho form o mesmo da URL ele irá renderizar o componente contido em element

```
<Route path="/rota-exemplo" element={<Exemplo />} />
```

react-router-dom - Link

Para navegação podemos usar o componente Link, ele irá renderizar no HTML uma tag <a> com atributo href, o link/URL para outra rota é posto no atributo **to**, formando a expressão “link to”, que significa “link para”.

```
<Link to="/rota-exemplo">Exemplo</Link>
```

react-router-dom - Rotas com parâmetros

Para se criar rotas com parâmetros, basta apenas colocar dois pontos (:) e o nome do parâmetro que vamos passar

```
// parametro id
<Route path="/task/:id" element={<UpdateTask />}>
```

react-router-dom - Rotas com parâmetros

Para ler o parâmetro devemos usar o hook useParams do react-router-dom.
É usual declarar uma variável `let params` que recebe a execução desse hook, também podemos desestruturar o parâmetro

```
import { useParams } from 'react-router-dom'
function UpdateTask() {
  let params = useParams()

  return <p>Tarefa com ID {params.id}</p>
}
```

Atividade em pares