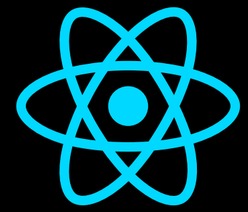
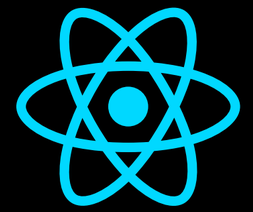


useParams e consumo de API com react

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

O que iremos aprender?



- O que é useParams?

O que é useParams ?

O useParams é um hook do React Router que permite acessar os parâmetros de rota dinâmicos em componentes funcionais. Ele é utilizado quando você tem rotas com segmentos variáveis, que podem mudar dependendo da URL.

Por exemplo, considere uma rota como esta:

```
<Route path="/produto/:id">
  <Produto />
</Route>
```

Aqui, :id é um parâmetro de rota dinâmico. Quando a rota /produto/123 é acessada, useParams permite acessar o valor de id (123, no exemplo acima) dentro do componente Produto.

Como usar o useParams?

Importação: Primeiro, importe o hook useParams do React Router:

```
import { useParams } from 'react-router-dom';
```

Acesso aos parâmetros: Em seguida, dentro do componente funcional, chame o hook useParams. Ele retornará um objeto contendo os parâmetros de rota.

```
function Produto() {
    const { id } = useParams();
    // Use o valor de id...
  }
```

Aqui, id corresponde ao nome do parâmetro de rota especificado na definição da rota.

Exemplo completo

Vamos criar um exemplo completo para ilustrar como usar o useParams.
Suponha que temos a seguinte rota em nossa aplicação:

```
<Route path="/produto/:id">
  <Produto />
</Route>
```

E queremos exibir detalhes do produto com base no id na URL.
Podemos fazer isso usando useParams:

Exemplo completo

```jsx
import { useParams } from 'react-router-dom';

function Produto() {
  const { id } = useParams();

  return (
    <div>
      <h2>Detalhes do Produto: {id}</h2>
      {/* Aqui você pode usar o id para buscar os detalhes do produto na sua fonte de dados */}
    </div>
  );
}
```

Neste exemplo, quando a rota /produto/123 é acessada, o componente Produto renderizado exibirá "Detalhes do Produto: 123".

Exercício

Exercício: Detalhes do Usuário
Neste exercício, você criará uma aplicação React que exibe detalhes de um usuário com base no ID fornecido na URL. Você usará o useParams para acessar o ID do usuário na rota e exibir seus detalhes.

Passos:

1. Configure a aplicação React com o React Router para lidar com diferentes rotas.
2. Crie uma rota que corresponda a um caminho como **/usuario/:id**, onde **:id** é um parâmetro de rota dinâmico que representa o ID do usuário.
3. Crie um componente chamado **DetalhesUsuario** que será renderizado quando a rota **/usuario/:id** for acessada.

Exercício

- Use o hook useParams dentro do componente DetalhesUsuario para acessar o ID do usuário da URL.
- Simule uma fonte de dados de usuários, como um array de objetos, onde cada objeto representa um usuário com propriedades como id, nome, email, etc.
- Utilize o ID do usuário obtido por meio do useParams para encontrar o usuário correspondente na fonte de dados de usuários.
- Exiba os detalhes do usuário, como nome, email, etc., na tela.

Exercício

```jsx
// App.js
import React from 'react';
import { BrowserRouter as Router, Route } from 'react-router-dom';
import DetalhesUsuario from './DetalhesUsuario';

function App() {
  return (
    <Router>
      <Route path="/usuario/:id">
        <DetalhesUsuario />
      </Route>
      <Route path="/">
        <h1>Página Inicial</h1>
      </Route>
    </Router>
  );
}

export default App;
```

Consumo de API com React

O consumo da API não muda aqui no react caso seja feita com o fetch. Vamos relembrar e colocar em prática!

```
fetch('link da api')
.then(response => response.json())
.then(dados => console.log(dados))
```

fetch siginifica captar. Cada api tem um método. Get (pegar), Post (enviar), Delete (deletar) e Update (atualizar). Veremos todos os tipos de requisições aqui.

Consumo de API com React

```
fetch('link da api')
.then(response => response.json())
.then(dados => console.log(dados))
```

O link da API, nós pegamos na documentação. then, significa então. Capte a resposta desse link, e então, foi colocado aqui como response, mas pode ser o nome que vocês quiserem!
Agora, uma reflexão: imagine um banco de dados feito em python, e o front feito com a tríade (html,css e javascript). Como os dados conversariam entre o back e o front, já que as linguagens são diferentes?

Consumo de API com React

```
fetch('link da api')
.then(response => response.json())
.then(dados => console.log(dados))
```

Simples. Os dados são enviados da única forma que todas as linguagens reconheceriam: string. Então nesse exemplo, response é uma string gigante, e quando passo o comando response.json(), estou convertendo essa string gigante, para o formato de objeto javascript!

Então, outro then, dados agora, é a minha string grandona convertida, preciso dar um nome diferente, já que agora é algo diferente (um objeto javascript,e não uma string). e então defini o que quero fazer com esses dados, quero mostrar no console.

Consumo de API com React

Atneção: Essa é a estrutura para fazer o consumo da api em requisições do tipo get que não exijam autenticação. E quando queremos fazer mais de uma coisa com os dados, precisamos colocar chaves, para que dados seja reconhecido.

```
fetch('link da api')
.then(response => response.json())
.then(dados => {
    console.log(dados)
//outras comandos que envolvam o uso da variável dados
})
```

Requisições do tipo Post

Requisições do tipo post são para enviar algo para o banco de dados. Esse tipo de requisição tem parâmetros, coisas que definitivamente precisamos enviar para que a requisição seja feita com sucesso. Como se trata de enviar algo para o banco de dados, a grande maioria das requisições do tipo post, precisam de autenticação.

Frequentemente, existe uma rota do tipo get para que você consiga pegar o que chamamos de token. Uma sequencia de letras e numeros aleatórios, mas que representa a sua permissão para mexer nas outras APIs.

Requisições do tipo Post

```
const token = localStorage.getItem('token');
            const response = await fetch('http://34.125.197.110:3333/user', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                },
                body: JSON.stringify({
                    username: nome,
                    email: email,
                    password: senha
                })
            });
```

Requisições do tipo Update

```
const response = await fetch(`http://34.125.197.110:3333/user/${id}`, {
        method: "PUT",
        headers: {
          Authorization: `Bearer ${token}`,
          "Content-Type": "application/json"
        },
        body: JSON.stringify(body)
      });
```

Requisições do tipo Delete

```
try {
    const token = localStorage.getItem("token");
    const response = await fetch(`http://34.125.197.110:3333/user/${estagiario.id}`, {
      method: "DELETE",
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
```

Projeto

Exercício: Catálogo de Filmes da Disney

Neste exercício, você criará um aplicativo React que consome a API da Disney para exibir informações sobre filmes, personagens e outros conteúdos relacionados à Disney.

Passos:

1. Explorar a Documentação da API:
    - Visite a documentação da API da Disney para entender os endpoints disponíveis e os dados que você pode acessar.
2. Configuração do Projeto:
    - Configure um novo projeto React usando Create React App ou a ferramenta de sua escolha.

Projeto

1. Consumo da API:
    - Utilize o método fetch ou uma biblioteca como Axios para fazer solicitações à API da Disney e obter informações sobre filmes, personagens, etc.
2. Exibição dos Dados:
    - Crie componentes para exibir os dados obtidos da API na interface do usuário. Por exemplo, você pode ter um componente para exibir a lista de filmes da Disney, outro para detalhes de personagens, etc.
3. Navegação:
    - Implemente navegação entre as diferentes seções do seu aplicativo, como uma lista de filmes, detalhes de um filme específico, lista de personagens, etc. Você pode usar React Router para isso.

Projeto

1. Recursos Adicionais:
2. Adicione funcionalidades extras, como filtros para encontrar filmes por categoria, barra de pesquisa para encontrar personagens específicos, etc.
3. Implemente a capacidade de favoritar filmes ou personagens e exibir uma lista de favoritos.
4. Integre recursos interativos, como trailers de filmes da Disney do YouTube.