

# Rotas e Componentes

Site: [Geração Tech](#)  
Curso: Formação em Desenvolvedor Web - Online  
Livro: Rotas e Componentes

Impresso por: JOÃO VITOR DE MELO FREITAS  
Data: quinta-feira, 18 jul. 2024, 22:42

# Índice

## 1. React

- 1.1. Vídeo Aula
- 1.2. Paginação e Rotas
- 1.3. Vídeo Aula
- 1.4. Trabalhando com Rotas e Componentes
- 1.5. Vídeo Aula
- 1.6. Componentes de Layout e Melhoria de Performance com Rotas
- 1.7. Vídeo Aula
- 1.8. Tratamento de Rotas Inválidas e Parâmetros de URL
- 1.9. Vídeo Aula

# 1. React

## Introdução ao useEffect

Hoje, vamos continuar aprendendo sobre hooks, focando no `useEffect`. Esse hook é essencial para gerenciar efeitos colaterais em componentes funcionais, como buscar dados, assinar streams ou manipular o DOM. O [React](#) tem vários hooks, mas alguns são mais importantes, e esses são os que vamos aprender.


## Quando Utilizar o useEffect

O `useEffect` é útil em várias situações. Um exemplo comum é buscar dados de um banco de dados quando o componente é montado ou atualizado. Vamos criar um exemplo prático para entender melhor como ele funciona.

## Estrutura do useEffect

Vamos para a parte lógica do nosso componente. Para usar o `useEffect`, você deve importá-lo do [React](#):

jsx

 Copiar código

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

O `useEffect` recebe dois parâmetros: uma função e uma lista de dependências. Vamos começar com um exemplo básico sem a lista de dependências:

jsx

 Copiar código

```
useEffect(() => { console.log('useEffect foi chamado'); });
```

## Renderizações e o useEffect

O `useEffect` é chamado em todas as renderizações do componente. Vamos ver isso na prática:

jsx

 Copiar código

```
import React, { useState, useEffect } from 'react'; const App = () => { const [count, setCount] = useState(0); useEffect(() => { console.log('useEffect foi chamado'); }); return ( <div> <p>Contador: {count}</p> <button onClick={() => setCount(count + 1)}>Incrementar</button> </div> ); }; export default App;
```

Toda vez que o estado `count` é atualizado, o componente é renderizado novamente, e o `useEffect` é chamado.

## Lista de Dependências

O segundo parâmetro do `useEffect` é a lista de dependências. Vamos ver um exemplo de como usá-lo para controlar quando o efeito deve ser chamado:

jsx

 Copiar código

```
useEffect(() => { console.log('useEffect foi chamado uma vez'); }, []);
```

Com a lista de dependências vazia, o `useEffect` será chamado apenas uma vez, quando o componente for montado.

## Exemplo com Dependências

Agora, vamos ver um exemplo onde o `useEffect` depende de uma variável de estado específica:

jsx

 Copiar código

```
import React, { useState, useEffect } from 'react'; const App = () => { const [count, setCount] = useState(0); const [count2, setCount2] = useState(0); useEffect(() => { console.log('useEffect chamado para count'); }, [count]); return ( <div> <p>Contador 1: {count}</p> <button onClick={() => setCount(count + 1)}>Incrementar Contador 1</button> <p>Contador 2: {count2}</p> <button onClick={() => setCount2(count2 + 1)}>Incrementar Contador 2</button> </div> ); }; export default App;
```

Aqui, o `useEffect` será chamado apenas quando `count` for atualizado. Alterações em `count2` não ativarão o `useEffect`.

## Conclusão

Hoje, aprendemos sobre o `useEffect`, um hook fundamental do [React](#) para gerenciar efeitos colaterais. Vimos como utilizá-lo em diferentes cenários e como a lista de dependências pode controlar quando o efeito é chamado. Na próxima aula, continuaremos explorando mais sobre hooks e outras funcionalidades do [React](#).

Vejo vocês na próxima aula!

## 1.1. Vídeo Aula

dia 20 video1



## 1.2. Paginação e Rotas

Tudo bem com vocês? Vamos continuar aqui nossas aulas de [React](#). Agora, quero entrar em um outro tema: paginação. Em toda aplicação, vai ser necessário, em algum momento, implementar paginação. Vamos ver como fazer isso no [React](#).


### Introdução às Rotas

No HTML, CSS e JavaScript, a paginação é feita criando múltiplos arquivos HTML e utilizando a tag `a` para linkar entre eles. No [React](#), é um pouco diferente porque não atualizamos a página inteira. Em vez disso, utilizamos um "container" onde tudo acontece.

### Estrutura de Arquivos

No nosso `index.html`, temos uma `div` com o ID `root`. Tudo o que renderizamos no [React](#) vem para dentro dessa `div`.

html


 Copiar código

```
<div id="root"></div>
```

### Configurando o [React Router](#)

Para gerenciar a navegação em nossa aplicação [React](#), vamos utilizar a biblioteca [react-router-dom](#). Vamos instalá-la:

bash


 Copiar código

```
npm install react-router-dom
```

### Verificando a Instalação

Após a instalação, você pode verificar se o pacote foi adicionado no `package.json`.

json

 Copiar código

```
"dependencies": { "react-router-dom": "^5.2.0" }
```

## Criando Componentes de Página


Vamos criar duas páginas: `Home` e `Products`.

### Estrutura de Pastas

- `src`
  - `pages`
    - `Home.jsx`
    - `Products.jsx`

### Componente Home

jsx

 Copiar código

```
import React from 'react'; const Home = () => { return ( <div> <h1>Home</h1> </div> ); }; export default Home;
```

### Componente Products

jsx

 Copiar código

```
import React from 'react'; const Products = () => { return ( <div> <h1>Products</h1> </div> ); }; export default Products;
```

## Configurando as Rotas


Vamos criar um componente `Routes` para gerenciar nossas rotas.

### Estrutura de Pastas

- `src`
  - `routes`
    - `Routes.jsx`

### Componente Routes

jsx


 Copiar código

```
import React from 'react'; import { BrowserRouter as Router, Route, Routes } from 'react-router-dom'; import Home from '../pages/Home'; import Products from '../pages/Products'; const AppRoutes = () => { return ( <Router> <Routes> <Route path="/" element={<Home />} /> <Route path="/products" element={<Products />} /> </Routes> </Router> ); }; export default AppRoutes;
```

## Utilizando o Componente de Rotas

Vamos importar e utilizar o componente `AppRoutes` no nosso `App.jsx`.

jsx

 Copiar código


```
import React from 'react'; import AppRoutes from './routes/Routes'; const App = () => { return ( <div> <AppRoutes /> </div> ); }; export default App;
```

## Navegando Entre Páginas

Para navegar entre as páginas, vamos criar um componente `Header` com links para `Home` e `Products`.

### Componente Header

jsx


 Copiar código

```
import React from 'react'; import { Link } from 'react-router-dom'; const Header = () => { return ( <nav> <ul> <li> <Link to="/">Home</Link> </li> <li> <Link to="/products">Products</Link> </li> </ul> </nav> ); }; export default Header;
```

## Utilizando o Header

Vamos adicionar o `Header` ao nosso `App.jsx`.

jsx

 Copiar código

```
import React from 'react'; import AppRoutes from './routes/Routes'; import Header from './components/Header'; const App = () => { return ( <div> <Header /> <AppRoutes /> </div> ); }; export default App;
```

## Testando a Navegação

Agora, ao iniciar a aplicação, você deve ser capaz de navegar entre as páginas `Home` e `Products` sem recarregar a página inteira.

## Conclusão

Aprendemos a configurar rotas no [React](#) utilizando a biblioteca [react-router-dom](#). Criamos componentes de página e um componente de navegação (Header) para facilitar a navegação entre as páginas. Na próxima aula, vamos continuar explorando mais funcionalidades do [React](#) Router e melhorar a nossa aplicação.

Vejo vocês na próxima aula!



### 1.3. Vídeo Aula

dia 20 video2




## 1.4. Trabalhando com Rotas e Componentes

Fala, Devs! Tudo bem com vocês? Vamos continuar nossa aula de [React](#), dando continuidade ao nosso último conteúdo sobre rotas.

### Configuração Inicial

Criamos uma rota para o nosso componente chamado **Home**:

jsx

 Copiar código

```
<Route path="/" element={<Home />} />
```

E também uma rota para o nosso componente **Produtos**:

jsx

 Copiar código

```
<Route path="/produtos" element={<Produtos />} />
```

Quando acessamos a barra (/), caímos na página **Home**. Agora, ao acessar **/produtos**, vamos para a página de produtos. No entanto, falta ainda algo importante: um **header** que permitirá navegar facilmente entre essas páginas.

### Criando o Header

Vamos criar um header que contenha links para nossas páginas. Comece criando uma nova pasta chamada **components** dentro de **src**. Dentro dessa pasta, crie um novo arquivo chamado **Header.jsx**.

### Estrutura do Header

Dentro de **Header.jsx**, vamos usar um snippet para criar a estrutura inicial do componente. Caso você não tenha o snippet instalado, recomendo instalar a extensão [React Snippets](#) no VS Code.

jsx

 Copiar código

```
import React from 'react'; import { Link } from 'react-router-dom'; const Header = () => { return ( <header> <h1>Logo</h1>  
<nav> <ul> <li><Link to="/">Home</Link></li> <li><Link to="/produtos">Produtos</Link></li> </ul> </nav> </header> ); };  
export default Header;
```

### Utilizando o Header

Agora que o **Header** está criado, precisamos usá-lo no nosso aplicativo. Vamos importar e incluir o **Header** no nosso componente principal, garantindo que ele apareça em todas as páginas.

### Integrando o Header no App

Abra o arquivo onde está definido o componente principal (geralmente **App.jsx** ou **App.tsx**) e importe o **Header**:

jsx

 Copiar código

```
import React from 'react'; import { BrowserRouter as Router, Route, Routes } from 'react-router-dom'; import Home from  
'./components/Home'; import Produtos from './components/Produtos'; import Header from './components/Header'; const App = ()  
=> { return ( <Router> <Header /> <Routes> <Route path="/" element={<Home />} /> <Route path="/produtos" element={<Produtos  
</Route> /> </Routes> </Router> ); }; export default App;
```

## Corrigindo Erros Comuns

Se você encontrar uma tela branca, verifique o console para ver mensagens de erro. Um erro comum é não usar o componente `Link` dentro de um componente contêiner da biblioteca `react-router-dom`, como o `Router`. Assegure-se de que todos os componentes de rota estejam dentro do `Router`.

## Verificando o Funcionamento

Agora, atualize o navegador e verifique se o header está aparecendo corretamente e se os links funcionam. Clique no link `Home` para ir para a página inicial e no link `Produtos` para ir para a página de produtos.

## Conclusão

Nesta aula, criamos um header e configuramos rotas básicas em nosso aplicativo `React`. No próximo episódio, continuaremos a explorar mais sobre rotas e outras funcionalidades do `React`. Até lá, pratique bastante e nos vemos na próxima aula!

## 1.5. Vídeo Aula

dia 20 video 3 converted



## 1.6. Componentes de Layout e Melhoria de Performance com Rotas

Tudo bem com vocês? Vamos continuar com nossas aulas de [React](#) e ainda estamos falando sobre rotas. Quero mostrar um detalhe importante sobre a estrutura de componentes e rotas no [React](#) para melhorar a performance da aplicação.

### Problema Inicial

Atualmente, temos um arquivo de rotas (`Routes.jsx`) onde colocamos um componente `Header` e nossas rotas. Além disso, podemos adicionar um componente `Footer` após essas rotas. O problema é que todas as páginas acabam tendo o `Header` e o `Footer`, mesmo quando isso não é necessário.

### Solução: Componentes de Layout

Vamos criar um componente de layout para isolar o `Header` e o `Footer`, de modo que apenas as páginas que realmente precisam desses componentes os utilizem. Dessa forma, nossa aplicação fica mais performática, pois apenas a área de conteúdo muda ao navegar entre as páginas.

### Estrutura de Pastas


Vamos criar uma estrutura de pastas que reflete essa nova organização:

- `src`
  - `components`
    - `Header.jsx`
    - `Footer.jsx`
  - `layouts`
    - `PageLayout.jsx`
  - `pages`
    - `Home.jsx`
    - `Products.jsx`
  - `routes`
    - `Routes.jsx`

### Passo a Passo

#### Componente Header


jsx

 Copiar código

```
import React from 'react'; const Header = () => { return ( <header> <h1>Header</h1> </header> ); }; export default Header;
```

#### Componente Footer

jsx

 Copiar código

```
import React from 'react'; const Footer = () => { return ( <footer> <h1>Footer</h1> </footer> ); }; export default Footer;
```

#### Componente PageLayout

Vamos criar um layout de página que inclui o `Header`, o `Footer` e um espaço para o conteúdo dinâmico.

jsx

 Copiar código

```
import React from 'react'; import { Outlet } from 'react-router-dom'; import Header from '../components/Header'; import Footer from '../components/Footer'; const PageLayout = () => { return ( <div> <Header /> <main> <Outlet /> </main> <Footer /> </div> ); }; export default PageLayout;
```

### Configurando as Rotas

Agora vamos configurar nossas rotas para usar o `PageLayout`.

## Componente Routes

jsx

 Copiar código

```
import React from 'react'; import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'; import Home from
'../pages/Home'; import Products from '../pages/Products'; import PageLayout from '../layouts/PageLayout'; const AppRoutes =
() => { return ( <Router> <Routes> <Route path="/" element={<PageLayout />}> <Route index element={<Home />} /> <Route
path="products" element={<Products />} /> </Route> </Routes> </Router> ); }; export default AppRoutes;
```

## Utilizando o Componente de Rotas

Vamos importar e utilizar o componente `AppRoutes` no nosso `App.jsx`.

jsx

 Copiar código

```
import React from 'react'; import AppRoutes from './routes/Routes'; const App = () => { return ( <div> <AppRoutes /> </div>
); }; export default App;
```

## Testando a Navegação

Ao iniciar a aplicação, você deve ser capaz de navegar entre as páginas `Home` e `Products`. A única parte que será atualizada é o conteúdo dentro do `Outlet`, enquanto o `Header` e o `Footer` permanecem estáticos.

## Conclusão

Aprendemos a criar componentes de layout para isolar o `Header` e o `Footer`, melhorando a performance da nossa aplicação [React](#). Esse padrão é muito útil para manter a consistência da UI e reduzir a carga de renderização desnecessária.

Vejo vocês no próximo episódio, onde continuaremos explorando mais funcionalidades do [React Router](#).

## 1.7. Vídeo Aula

dia 20 video 4 converted



## 1.8. Tratamento de Rotas Inválidas e Parâmetros de URL

Fala, devs! Tudo bem com vocês? Continuando com nosso estudo de [React](#) e ainda falando sobre rotas, hoje vamos abordar dois pontos importantes: tratamento de rotas inválidas e como trabalhar com parâmetros de URL. Vamos lá!

### Tratamento de Rotas Inválidas

#### Problema

Quando um usuário tenta acessar uma rota que não existe, a aplicação não responde de forma adequada. Por exemplo, se digitarmos uma URL inválida, nada acontece e o console mostra um erro dizendo que nenhuma rota combinou com a localização solicitada.


#### Solução

Vamos criar uma página para exibir quando uma rota não for encontrada, com uma mensagem amigável e um link para retornar à página inicial.

#### Passo a Passo

##### 1. Crie o Componente de Página Não Encontrada

jsx

 Copiar código

```
import React from 'react'; import { Link } from 'react-router-dom'; const NotFound = () => { return ( <div> <h3>Página não encontrada</h3> <Link to="/">Voltar</Link> </div> ); }; export default NotFound;
```

##### 2. Adicione a Rota de Página Não Encontrada

No arquivo de rotas (`Routes.jsx`), adicione uma rota coringa para capturar todas as URLs que não correspondem a nenhuma rota definida.

jsx

 Copiar código

```
import React from 'react'; import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'; import Home from '../pages/Home'; import Products from '../pages/Products'; import NotFound from '../pages/NotFound'; import PageLayout from '../layouts/PageLayout'; const AppRoutes = () => { return ( <Router> <Routes> <Route path="/" element={ <PageLayout /> } <Route index element={ <Home /> } /> <Route path="products" element={ <Products /> } /> <Route path="*" element={ <NotFound /> } /> </Route> </Routes> </Router> ); }; export default AppRoutes;
```

#### Teste

Acesse uma URL inválida, como `/nada`, e verifique se a mensagem "Página não encontrada" é exibida com um link para retornar à página inicial.

### Trabalhando com Parâmetros de URL

#### Problema

Às vezes, precisamos passar informações específicas na URL, como IDs de produtos, para que possamos renderizar diferentes conteúdos com base nesses parâmetros.

#### Solução

Vamos criar uma rota dinâmica que captura parâmetros da URL e os utiliza dentro do componente.

#### Passo a Passo

##### 1. Crie o Componente de Produto

jsx

 Copiar código



```
import React from 'react'; import { useParams } from 'react-router-dom'; const Product = () => { const { id, name } = useParams(); return ( <div> <h1>Produto {name}</h1> <p>ID do Produto: {id}</p> </div> ); }; export default Product;
```

## 2. Adicione a Rota Dinâmica

No arquivo de rotas (`Routes.jsx`), adicione uma rota que capture os parâmetros `id` e `name`.

jsx



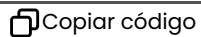
Copiar código

```
import React from 'react'; import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'; import Home from '../pages/Home'; import Products from '../pages/Products'; import Product from '../pages/Product'; import NotFound from '../pages/NotFound'; import PageLayout from '../layouts/PageLayout'; const AppRoutes = () => { return ( <Router> <Routes> <Route path="/" element={ <PageLayout /> }> <Route index element={ <Home /> } /> <Route path="products" element={ <Products /> } /> <Route path="products/:id/:name" element={ <Product /> } /> <Route path="*" element={ <NotFound /> } /> </Route> </Routes> </Router> ); }; export default AppRoutes;
```

## 3. Atualize o Componente de Produtos

Adicione links para cada produto que direcionem para a nova rota com parâmetros.

jsx



Copiar código

```
import React from 'react'; import { Link } from 'react-router-dom'; const Products = () => { const productList = [ { id: 1, name: 'Banana' }, { id: 2, name: 'Maçã' }, { id: 3, name: 'Abacate' }, { id: 4, name: 'Espinafre' }, { id: 5, name: 'Cenoura' } ]; return ( <div> <h1>Produtos</h1> <ul> {productList.map(product => ( <li key={product.id}> <Link to= `/products/${product.id}/${product.name}`>{product.name}</Link> </li> ))} </ul> </div> ); }; export default Products;
```

## Teste

Acesse a página de produtos e clique em um dos links. Verifique se a página do produto é exibida com o nome e o ID corretos.

## Conclusão

Hoje aprendemos a tratar rotas inválidas e a trabalhar com parâmetros de URL no [React](#). Essas técnicas são essenciais para criar uma aplicação robusta e amigável ao usuário. Vejo vocês no próximo episódio, onde continuaremos explorando mais funcionalidades do [React Router](#). Valeu, pessoal!

## 1.9. Vídeo Aula

dia 20 video5 converted

