FIAP

RESPONSIVE WEB DEVELOPMENT

VITE + REACT

REACT – ROUTER { ***ROTAS*** }

Prof. Alexandre Carlos profalexandre.jesus@fiap.com.br

Prof. Luís Carlos lsilva@fiap.com.br

Prof. Wellington Cidade profwellington.tenorio@fiap.com.br

REACT - ROTAS

FIAP

O QUE SÃO ROTAS?

Em nossas aplicações com o React, não podemos simplesmente criar links em nossas aplicações para direcionar para outras páginas internas, como fazemos em projetos com HTML simples. O React é um **Single Page Application**, ou seja, uma aplicação que **funciona em uma única página**. Isso é possível porque ele cria uma página a partir da montagem de componentes, que podem tanto fazer o papel de um container, quanto de um simples botão.

Quando trabalhamos com rotas, podemos definir quais componentes irão compor a página, dando um novo caminho (URI) para eles, sendo assim, **cada rota será uma página**, e podemos criar quantas páginas forem necessárias. O usuário nem irá perceber o que está acontecendo.

REACT - ROTAS

CRIANDO A APLICAÇÃO

Para conseguirmos criar rotas em nossa aplicação vamos precisar importar o pacote de uma biblioteca a **react-router-dom**, quando criar o projeto, siga a sequência abaixo:

1.

```
PS D:\projetos\base> npm create vite@latest
? Project name: » vite-project
```

2.

```
PS D:\projetos\base> npm create vite@latest
√ Project name: ... my-app
? Select a framework: » - Use arrow-keys. Return to submit.
    Vanilla
    Vue
>   React
    Preact
    Lit
    Svelte
    Solid
    Qwik
    Others
```

3.

```
PS D:\projetos\base> npm create vite@latest
√ Project name: ... my-app
√ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
>   TypeScript
    TypeScript + SWC
    JavaScript
    JavaScript + SWC
    Remix ↗
```

React Router

VITE + REACT

REACT - ROTAS

CRIANDO A APLICAÇÃO

Para conseguirmos criar rotas em nossa aplicação vamos precisar importar o pacote de uma biblioteca a **react-router-dom**, quando criar o projeto, siga a sequência abaixo:

```
PS D:\projetos\base> npm create vite@latest
√ Project name: ... my-app
√ Select a framework: » React
√ Select a variant: » TypeScript

Scaffolding project in D:\projetos\base\my-app...

Done. Now run:

  cd my-app
  npm install
  npm run dev

PS D:\projetos\base>
```

abre a pasta do projeto

instala a pasta de dependencias node_modules.

inicia o projeto(coloca no ar)

4 .

Instala o pacote da biblioteca das rotas

package.json > {} dependencies > react-router-dom

```
11    },
12    "dependencies": {
13      "react": "^18.3.1",
14      "react-dom": "^18.3.1",
15+     "react-router-dom": "^6.26.1"
16    },
```

arquivo onde se localiza as dependencias instaladas.

após ser instalado

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR COMMENTS

```
PS D:\projetos\base\my-app> npm i  react-router-dom

up to date, audited 199 packages in 2s

42 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

comando a ser executado.

5 .

React Router

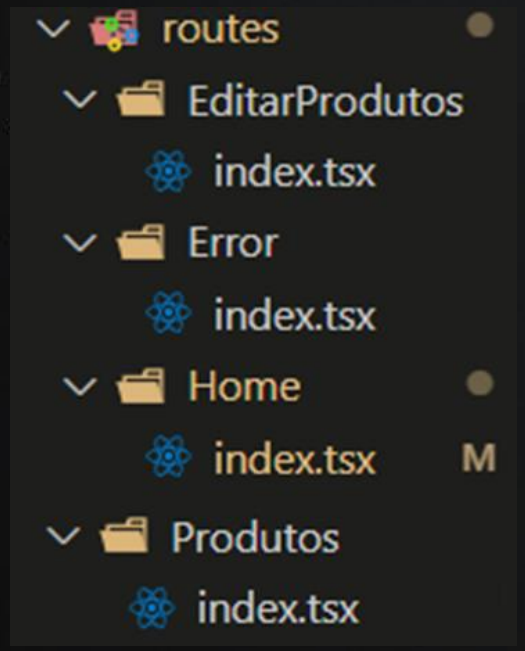
VITE + REACT

REACT - ROTAS

CRIANDO AS PÁGINAS DO NOSSO EXEMPLO

Quando criamos os componentes que representarão as nossas páginas devemos fazer algumas boas práticas para organizarmos melhor nosso projeto:

1. Vamos criar uma pasta chamada **routes**, ela irá guardas as páginas que vamos criar.
2. Para cada página vamos criar uma pasta com o nome da página e dentro um arquivo chamado index.tsx;



Teremos as páginas:

- Home
- Produtos
- EditarProdutos
- Error

React Router

REACT - ROTAS

CONTEÚDO INICIAL DOS COMPONENTES

Home

```
export default function Home() {
    return (
        <main>
            <h2>Home</h2>
        </main>
    )
}
```

Produtos

```
export default function Produto() {
    return (
        <div>
            <h1>Produto</h1>
        </div>
    );
}
```

EditarProduto

```
export default function EditarProdutos() {
    return (
        <section>
            Editar Produtos
        </section>
    )
}
```

Error

```
export default function Error(){

    return(
        <>
            <h1>Erro 404 - Página não encontrada</h1>
        </>
    )
}
```

React Router

REACT - ROTAS

IMPORTANDO COMPONENTES

A primeira coisa que devemos fazer após criar as páginas é importar os componentes que iremos necessitar no main.tsx, será a partir de lá que iremos fazer esse controle.

```
5   import Home from "./routes/Home/index.tsx";
6   import Produtos from "./routes/Produtos/index.tsx";
7   import EditarProdutos from "./routes/EditarProdutos/index.tsx";
8   import Error from "./routes/Error/index.tsx";
9   import { createBrowserRouter, RouterProvider } from 'react-router-dom';
```

React Router

REACT - ROTAS

CONFIGURANDO AS ROTAS

Para configurarmos as rotas precisamos utilizar o método createBrowserRouter, ele recebe um objeto com os dados de App e dentro do atributo children, cada objeto representa uma das rotas possíveis em nosso projeto.

errorElement – define o componente que será apresentado caso o usuário coloque uma URI inexistente.

Children – atributo de App que armazena em um array os objetos de todas as páginas que podem ser acessadas.

Path - define a **URI** da página logo após o endereço do projeto no browser.

Element - define o **Componente** que será chamado quando a URI for requisitada no browser.

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <App/>,
    errorElement: <Error/>,
    children:[
      {
        path: "/",
        element: <Home/>
      },
      {
        path: "/produtos",
        element: <Produtos/>
      },
      {
        path: "/produtos/editar/:id",
        element: <EditarProduto/>
      }
    ]
  }
])
```

React Router

REACT - ROTAS

CONFIGURANDO AS ROTAS

Após configurarmos nosso createBrowserRouter, que demos o nome de router, vamos usar o **RouterProvider** para controlar a renderização das páginas no **ReactDOM**. Para isso precisamos também passar o nosso router como atributo.

```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>,
)
```

React Router

REACT - ROTAS

CONFIGURANDO AS ROTAS

Como configuramos o **App.tsx** como componente principal agora devemos prepara-lo para trabalhar junto com o **RouterProvider**. O App vai receber o **Outlet** que vai disponibilizar a passagem das páginas por dentro dele. Faça conforme o código abaixo:

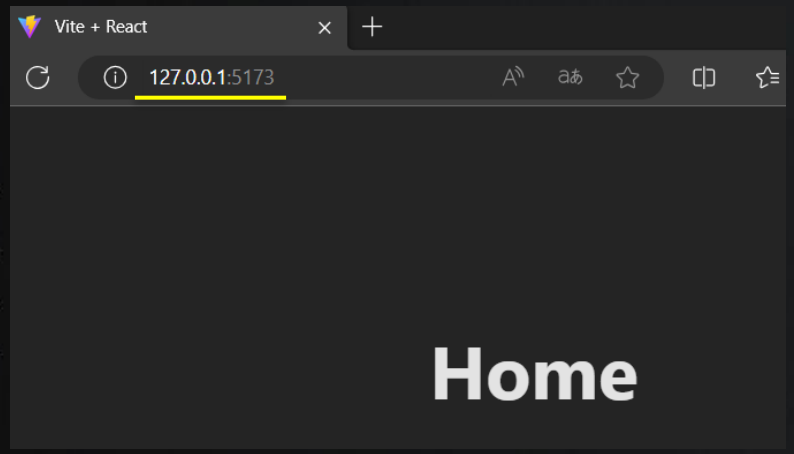
```
import './App.css'
import { Outlet } from 'react-router-dom'

function App() {

  return (
    <>
      <Outlet/>
    </>
  )
}
export default App
```

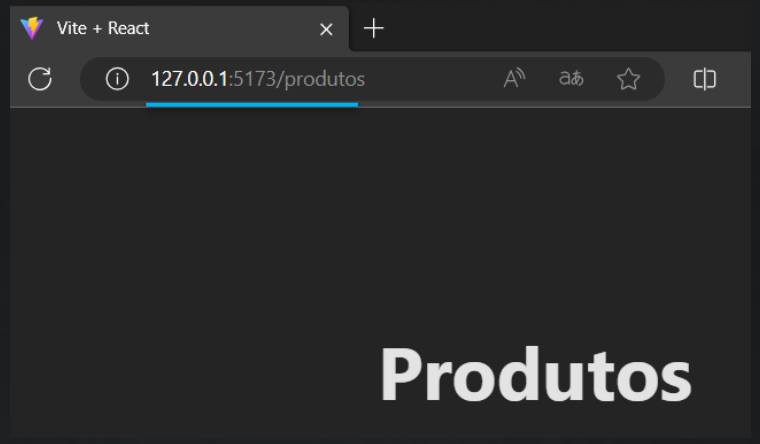
React Router

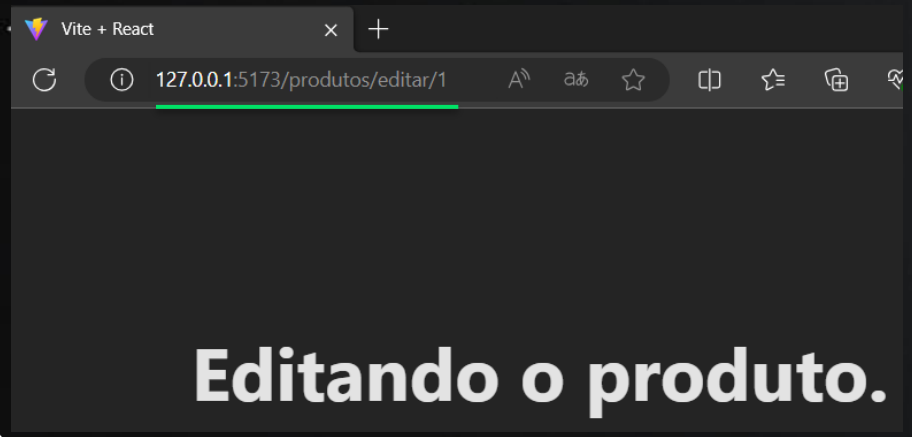
REACT - ROTAS

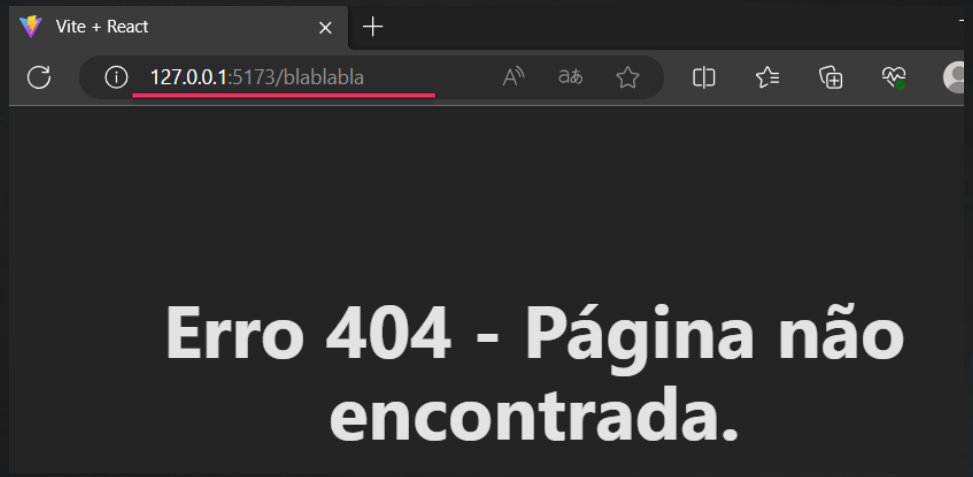
NAVEGANDO USANDO AS URI'S

Com essa primeira parte implementada já podemos testar nossas rotas.









React Router

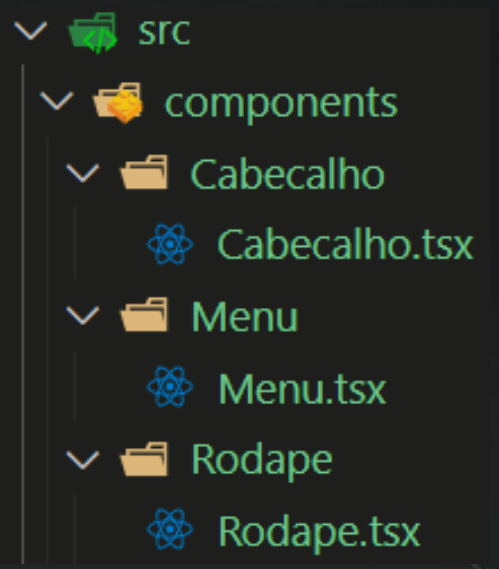
REACT - ROTAS

CRIANDO LINKS PARA NAVEGAR

Não podemos deixar uma aplicação sem uma forma de navegação, vamos adicionar em nosso App.tsx, mais 03(três) componentes, Cabecalho, Rodape e Menu.
Como estamos em uma SPA, eles irão ficar estáticos disponíveis em todas as páginas.
Crie uma pasta chamada **components** dentro de **src** e crie os componentes dentro desta pasta.

src > App.tsx > ...
You, 1 second ago | 1 author (You)

```tsx
import { Outlet } from "react-router-dom";
import Rodape from "./components/Rodape/Rodape.tsx";
import styles from "./App.module.css";
import Cabecalho from "./components/Cabecalho/Cabecalho.tsx";

function App() {
  return (
    <div className={styles.container}>
      <Cabecalho />
      <Outlet />
      <Rodape />
    </div>
  );
}
export default App;
```



Obs. Não se esqueça de chamar o componente **Menu** dentro de **Cabecalho**.

React Router

REACT - ROTAS

CRIANDO LINKS PARA NAVEGAR

Em nosso componente Menu.tsx vamos criar links através do componente <***Link***>do router-dom para nossas rotas.

Importando o componente Link do router dom.

```tsx
import { Link } from "react-router-dom"

export default function Menu(){

    return(
        <nav className="menu">
            <Link to='/'>Home</Link>
            <span> | </span>
            <Link to='/produtos'>Produtos</Link>
        </nav>
    )
}
```

Componente Link passando a rota.

Obs. Não se esqueça de chamar o componente **Menu** dentro de **Cabecalho**.

React Router

REACT - ROTAS

ESTILIZANDO COM CSS.MODULES

CSS Modules é uma técnica de escopo de estilo em projetos de front-end que permite criar classes CSS localizadas por padrão. Isso significa que os estilos definidos em um CSS Module são aplicados apenas ao componente em que são importados, evitando colisões de estilos e garantindo que as regras CSS não vazem para outros componentes.

Como Funcionam os CSS Modules?

1. **Escopo Local**:
   - Quando você cria uma classe em um arquivo `.module.css`, o CSS Modules transforma automaticamente os nomes das classes em identificadores únicos, garantindo que elas sejam aplicadas somente ao componente correspondente.
   - Exemplo:

```css
/* styles.module.css */
.button {
    background-color: blue;
}
```

React Router

REACT - ROTAS

ESTILIZANDO COM CSS.MODULES

O nome da classe `.button` será transformado em algo como `.styles_button__3XK1Y`, onde `styles` é o nome do arquivo, `button` é o nome da classe original, e `3XK1Y` é um hash gerado automaticamente.

2. **Importação em JavaScript/TypeScript**:

- Você pode importar as classes CSS diretamente no seu componente React:

```
1  import styles from './styles.module.css';
2
3  function MyButton() {
4    return <button className={styles.button}>Click Me</button>;
5  }
```

REACT - ROTAS

ESTILIZANDO COM CSS.MODULES

- Nesse caso, a classe `styles.button` se refere à classe escopada gerada pelo CSS Module.

3. **Benefícios:**
   - **Isolamento de Estilos**: Como cada classe é transformada em um identificador único, não há risco de sobrescrita acidental de estilos entre diferentes componentes.
   - **Facilidade de Manutenção**: Como as classes são locais, você pode usar nomes simples e significativos sem se preocupar com conflitos globais.

Vantagens e Uso Comum

- **Reutilização**: CSS Modules são úteis em projetos grandes onde múltiplos desenvolvedores podem estar criando estilos. Isso previne conflitos de nomenclatura e torna o código mais modular e fácil de manter.
- **Popularidade em frameworks modernos**: CSS Modules são frequentemente usados em conjunto com frameworks como React, Vite, e Next.js, onde o escopo de componentes é fundamental.

REACT - ROTAS

AJUSTANDO COMPONENTES REAPROVEITÁVEIS

Vamos reforçar a ideia de que podemos ter componentes que ficam fixos, sendo reutilizados em todas as estruturas de página, assim vamos criar em **components** um componente **Rodape.tsx** e estilizar ele e o nosso menu. Sendo assim faça a chamada do css correspondente de acordo com a classe do elemento.

div principal do App = .container
Menu = .nav
Home = .main
Rodape = .footer

Crie um arquivo ***App.module.css*** e inclua as seguintes estilizações.

src > App.module.css > ...

```css
.container {
    height: 95vh;
    width: 100%;
    display: flex;
    flex-direction: column;
    justify-content: space-between;

    .nav {
        display: flex;
        width: 100vw;
        justify-content: center;
        align-items: center;
        height: 8vh;
        background-color: royalblue;

        &
        a {
            color: white;
            font-size: 1.5em;
            padding: 15px;
        }
    }
    .main {
        display: flex;
        width: 100vw;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        height: 70vh;
        background-color: bisque;
    }
    .footer {
        display: flex;
        width: 100vw;
        justify-content: center;
        align-items: center;
        height: 10vh;
        background-color: darkblue;
        color: #fff;
        font-size: 32px;
    }
}
```

React Router

REACT - ROTAS

AJUSTANDO COMPONENTES REAPROVEITÁVEIS

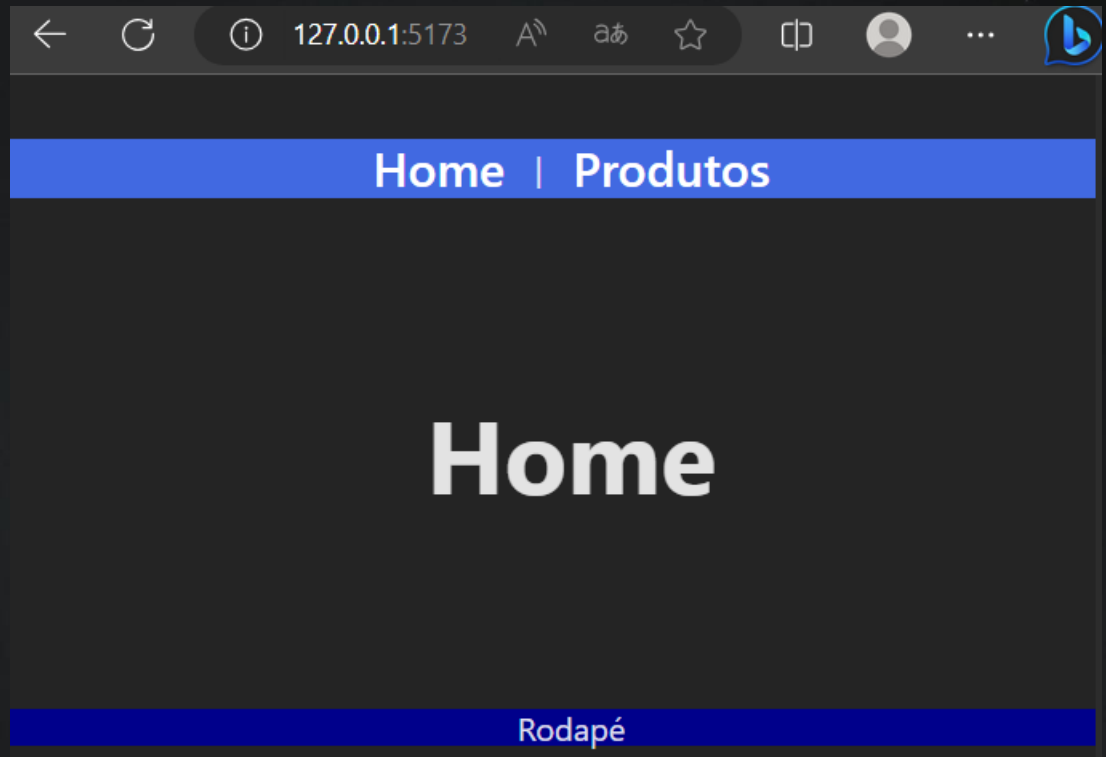
Não esqueça de chamar os componentes **Cabecalho** e **Rodape** no **App**.

src > App.tsx > ...

You, 1 second ago | 1 author (You)

```tsx
import { Outlet } from "react-router-dom";
import Rodape from "./components/Rodape/Rodape.tsx";
import styles from "./App.module.css";
import Cabecalho from "./components/Cabecalho/Cabecalho.tsx";

function App() {
  return (
    <div className={styles.container}>
      <Cabecalho />
      <Outlet />
      <Rodape />
    </div>
  );
}
export default App;
```

Nossa página deve ficar assim. Lembra que já estamos aproveitando um pouco o CSS do template.



React Router

REACT - ROTAS

PASSANDO VALORES PELA URI

Quando precisamos passar valores de uma página para outra, como por exemplo o ID de um produto, podemos usar um Hook chamado UseParams. Vamos criar um arquivo TS com um array de produtos, simulando dados vindos do backend. Crie o arquivo **listaProdutos.ts**. Não esqueça de tipar a lista.

src > TS listaProdutos.ts > ...

```ts
export const listaprodutos:{id:number,nome:string,preco:number}[] = [
    {id:1, nome:'Teclado', preco:158},
    {id:2, nome:'Mouse', preco:25},
    {id:3, nome:'Monitor', preco:450},
    {id:4, nome:'Fone de ouvido', preco:80},
    {id:5, nome:'Cadeira Gamer', preco:2590},
    {id:6, nome:'Mouse Pad', preco:45},
    {id:7, nome:'Web Cam', preco:675},
    {id:8, nome:'Filtro de Linha', preco:120},
];
```

REACT - ROTAS

PASSANDO VALORES PELA URI

Vamos carregar os dados de nossa lista na página **Produtos.tsx**. Você reparou que nosso array tinha um export? Usaremos ele para acessar a lista e carregar, usando o método map para criar os links.

Vamos passar o ID do produto pela URI.

src > routes > Produtos > index.tsx > ...

You, 1 second ago | 1 author (You)

```tsx
import { Link } from "react-router-dom";
import styles from "../../App.module.css";
import { listaprodutos } from "../../listaProdutos";

export default function Produto() {
    return (
        <div className={styles.produto}>
            <h1>Produto</h1>
            {listaprodutos.map((prod)=>(
                <div key={prod.id}>
                    {prod.nome} - <Link to={`/editar-produtos/:${prod.id}`}>Editar o Produto</Link>
                </div>
            ))}
        </div>
    );
}
```

A passagem do id do produto está configurado lá no ***main.tsx***, no objeto que presenta a página EditarProduto, temos **:id** representando que passaremos dados através dele .

```tsx
{path: '/editar-produtos/:id', element: <EditarProdutos />},
```

O próximo passo será configurar o recebimento dele.

React Router

REACT - ROTAS

PASSANDO VALORES PELA URI

Na página EditarProduto.jsx vamos receber o ID do produto que devemos selecionar acessando os dados da lista.
Para isso precisaremos usar:

useParams – método do router para pegar os dados passados na URI;

useNavigate – nos permite redirecionar como um link, só que de dentro do código.

Obs. Como os dados que estamos acessando são estáticos, não estamos realmente editando eles, somente acessando.

```jsx
import { useParams, useNavigate } from "react-router-dom"
import { listaProdutos } from "../../components/listaProdutos"

export default function EditarProduto(){

    const lista = listaProdutos
    const navegacao = useNavigate()
    const {id} = useParams()

    const proc = lista.filter(prod => prod.id == id)
    const produto = proc[0]

    const salvar = ()=>{
        alert(`Produto: ${produto.nome} editado com sucesso!`)
        return navegacao('/produtos')
    }

    return(
        <main>
            <h1>Editando o produto </h1>
            <p>Editando os dados do produto: {produto.nome}</p>
            <button onClick={salvar}>Salvar</button>
        </main>
    )
}
```

React Router

REACT - ROTAS

REDIRECIONANDO UM ENDEREÇO INUTILIZADO

Um caso que pode acontecer é precisarmos desativar uma página temporariamente ou definitivamente e não termos o tempo hábil de remover todas as chamadas dela na aplicação, para isso podemos usar o Navigate diretamente no **Main.tsx** para trazer o usuário de volta a página principal.

Importando o Navigate do pacote Router-DOM.

```tsx
import { createBrowserRouter, RouterProvider, Navigate } from 'react-router-dom'
import Home from './routes/Home/index.tsx'
import Produtos from './routes/Produtos/index.tsx'
```

```tsx
    path: "/produtos/editar/:id",
    element: <EditarProduto/>
  },
  {
    path: "/antiga",
    element: <Navigate to="/"/>
  }
]
```

Novo objeto em nosso array de rotas. Assim se alguém tentar ir para o endereço antigo, será direcionado para a página home.

React Router

EXERCÍCIO

FIAP

Crie uma aplicação para um site de vendas de Smartphones e Tablets.
Ele deve conter as seguintes páginas:

- Home – Na página inicial deve ter duas propagandas de promoções;
- Aparelhos – Nesta página deverá ter uma lista de aparelhos, os dados deverão vir de uma lista criada em um arquivo JS a parte, no mínimo 5 aparelhos;
- VisualizarAparelho – Após o cliente escolher um dos aparelhos, ele deverá ser direcionado para esta página, onde poderá ver a foto e todos os detalhes do aparelho.

- O Cabeçalho e rodapé deverão ficar fixos, sendo aproveitados por todas as páginas.
- Deverá ter uma página para orientar o usuário caso ocorra o Erro 404, e ele deverá ter a opção de voltar para página Home.
- Todas as páginas devem estar estilizadas.

OBRIGADO

FIAP


FIAP