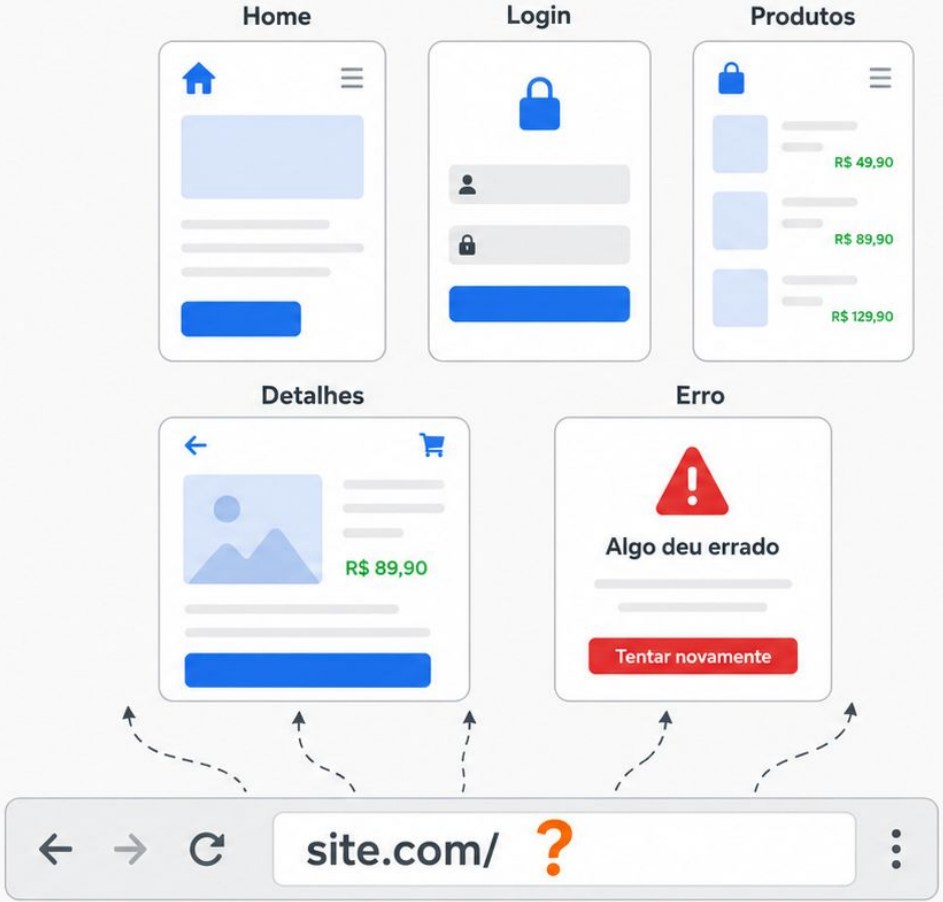
Navegação com React Router

O que vamos ver hoje?

- SPA: Single Page Application
- Revisão de rotas
- React Router
- Componentes do Router
- Navegação entre páginas
- Parâmetros na URL
- Boas práticas
- Deploy de SPA

Antes de falar de rotas...

- Em um **sistema real**, normalmente temos várias telas
- Exemplo:
  - Tela inicial
  - Tela de login
  - Tela de produtos
  - Tela de detalhes
  - Tela de erro
- O **problema** é:
  - Como trocar de tela?
  - Como representar cada tela na URL?
  - Como permitir que o usuário acesse uma tela diretamente?



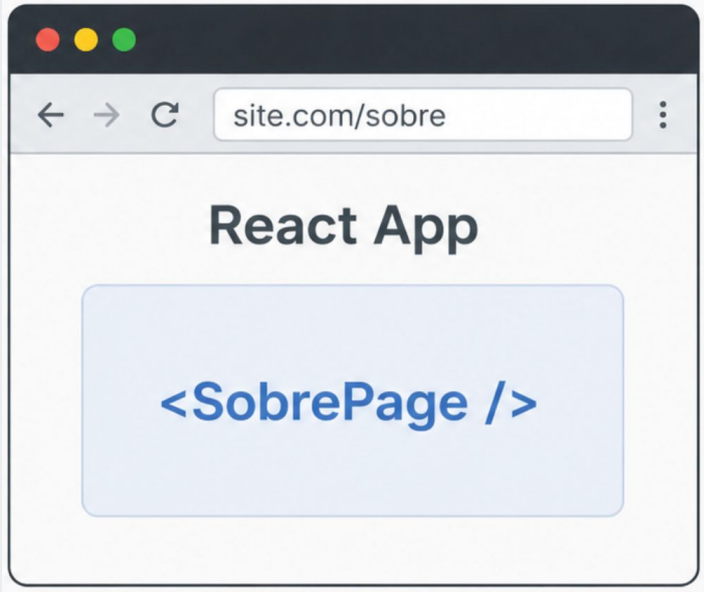
Aplicações tradicionais 🌐

- Em sites tradicionais, cada página pode ser um arquivo diferente
- Exemplo:
  `/index.html` → Página inicial
  `/login.html` → Página de login
  `/sobre.html` → Página sobre
- Ao clicar em um link, o navegador carrega outra página
- Isso pode gerar um recarregamento completo do site

Single Page Application

O que é uma SPA?

- SPA significa **Single Page Application**
- O navegador carrega uma página principal
- Depois disso, o **React** troca os componentes sem recarregar o site inteiro
- A URL muda, mas quem decide o que aparece é o **React Router**



/ → <HomePage />

/sobre → <SobrePage />

/login → <LoginPage />

SPA não significa “uma tela só” ⚡

- O nome pode confundir um pouco
- SPA não quer dizer que o sistema tem apenas uma tela
- Quer dizer que o navegador carrega uma página principal
- Depois disso, o **React** controla quais componentes aparecem

Sem Router: renderização condicional ⚡

- Até agora, poderíamos trocar telas usando estado
- Exemplo:

  `telaAtual === "home"` → mostra **Home**

  `telaAtual === "sobre"` → mostra **Sobre**

  `telaAtual === "login"` → mostra **Login**

- Funciona, mas começa a ficar trabalhoso

```jsx
import { useState } from 'react'

function App() {
  const [telaAtual, setTelaAtual] = useState("home")

  return (
    <div>
      {telaAtual === "home" && <Home />}
      {telaAtual === "sobre" && <Sobre />}
      {telaAtual === "login" && <Login />}
    </div>
  )
}
export default App
```

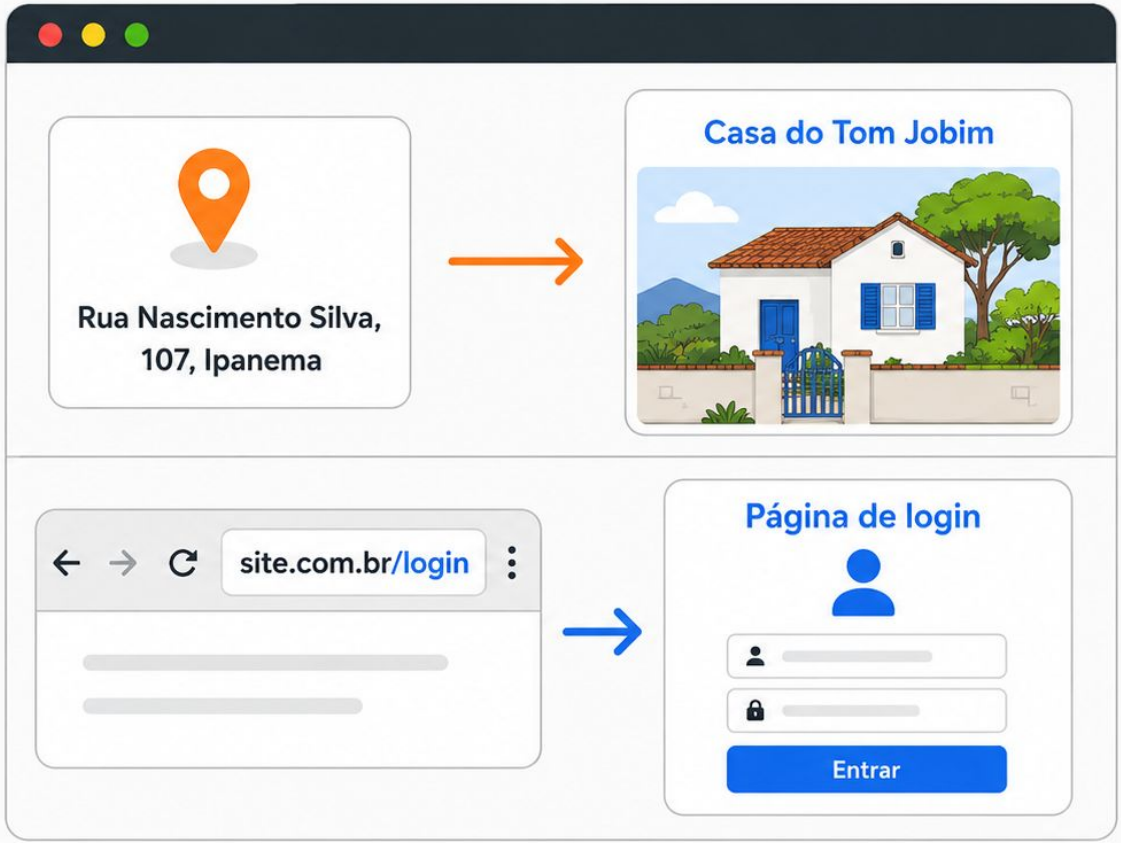
Estado que controla qual tela mostrar

Renderização condicional baseada no estado

Revisão - Rotas

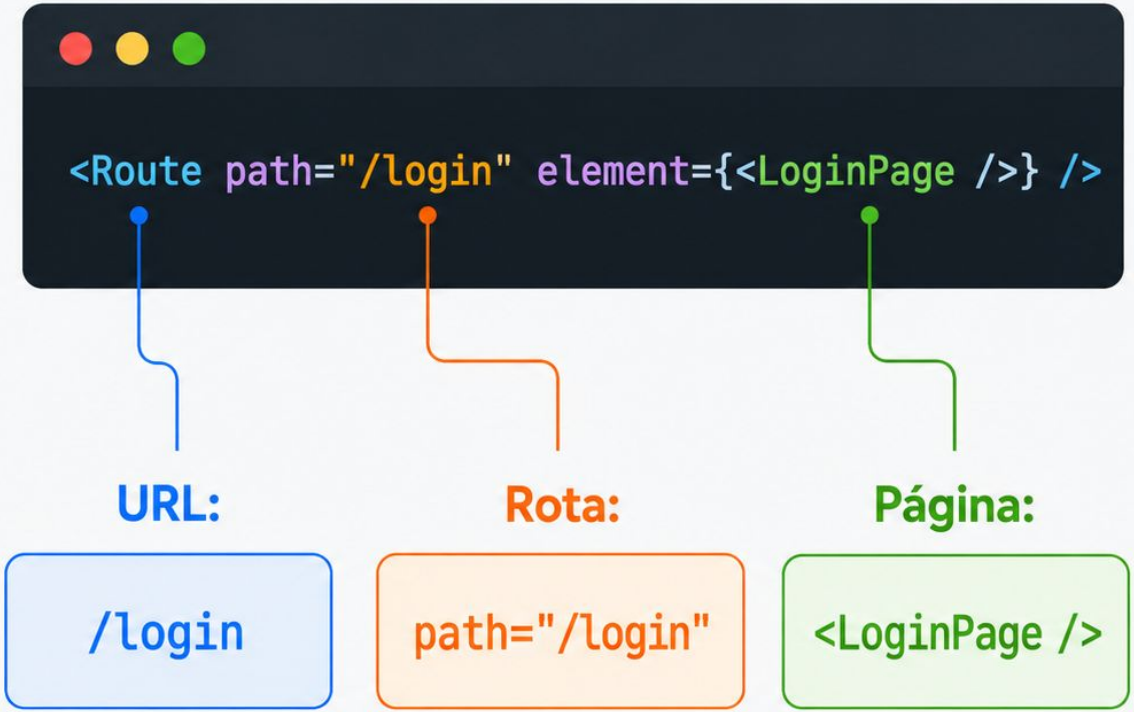
Revisão - Rotas

- **Rota** é uma relação entre um **endereço** e um **resultado**
- Exemplo do mundo real:
  - → Vá até a Rua Nascimento Silva, 107, Ipanema
  - → Você encontrará a casa do Tom Jobim
- No nosso contexto:
  - → Acesse site.com.br/login
  - → Você encontrará a página de login



URL, rota e página

- **URL:**
  é o endereço que aparece no navegador
- **Rota:**
  é a regra que associa uma URL a algum conteúdo
- **Página:**
  é o componente React exibido naquela rota



Exemplo mental ⚡

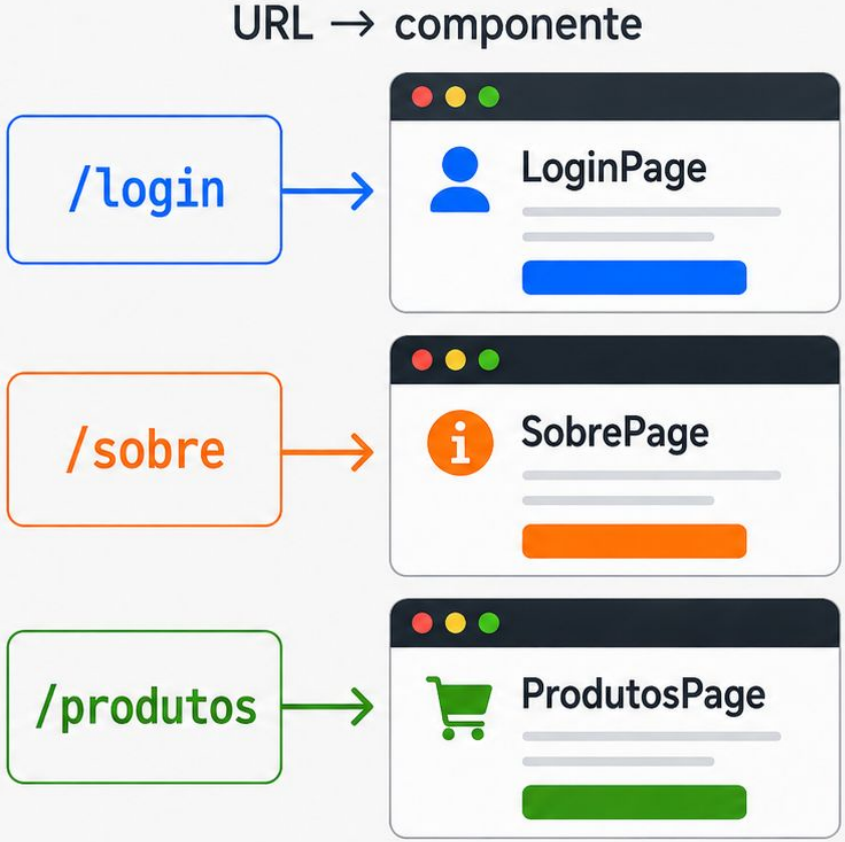
- Quando o usuário acessa:
  `/sobre`
- O React Router verifica:
  Existe alguma rota para `/sobre`?
- Se existir, ele renderiza:
  `<SobrePage />`

URL → rota → página

React Router 🔀

React Router

- É uma biblioteca para controlar navegação em aplicações React
- Ela permite mapear URLs para componentes
- Exemplo:
  `/login` → `LoginPage`
  `/sobre` → `SobrePage`
  `/produtos` → `ProdutosPage`

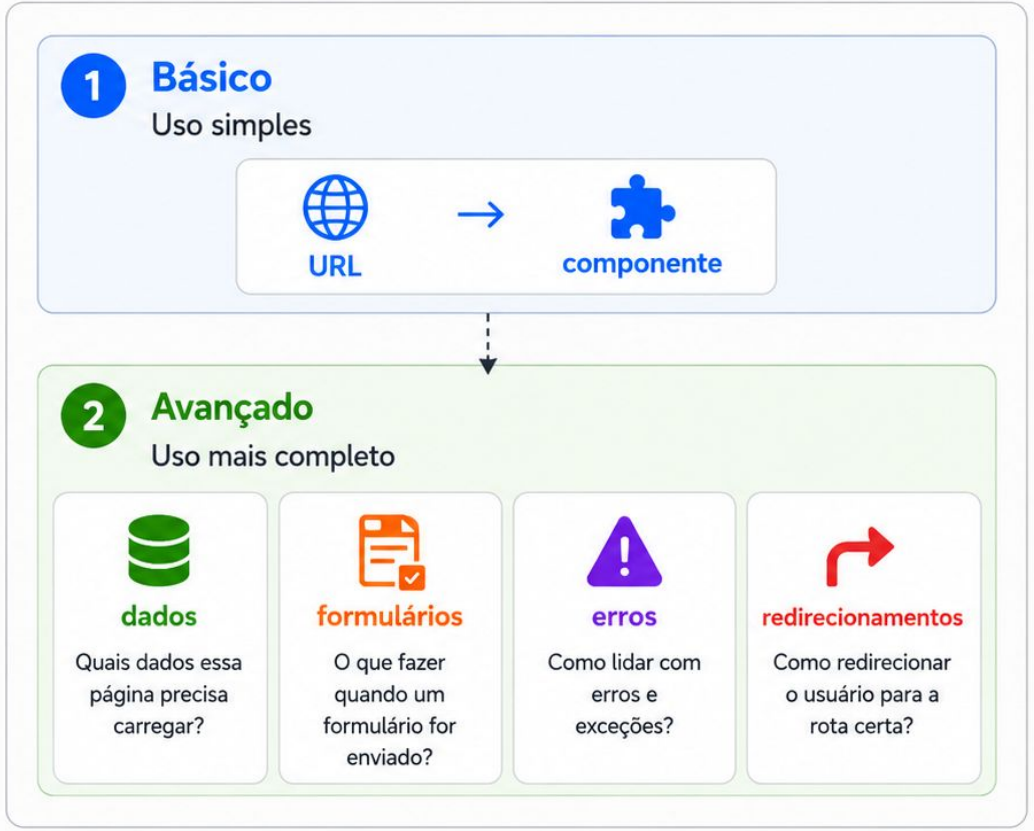


React Router pode ser usado em níveis

- No uso mais simples, ele responde:

  "Qual componente devo mostrar para esta URL?"

- Em usos mais avançados, ele também pode ajudar com:
  - Quais dados essa página precisa carregar?
  - O que fazer quando um formulário for enviado?
  - Como lidar com erros e redirecionamentos?



React Router atua em diferentes níveis de complexidade

React Router - Vantagens

- Facilita a gestão de múltiplas páginas
- Permite acessar páginas diretamente pela URL
- Integra a aplicação com o histórico do navegador
- Evita controlar tudo manualmente com estado
- Ajuda a organizar melhor sistemas com várias telas

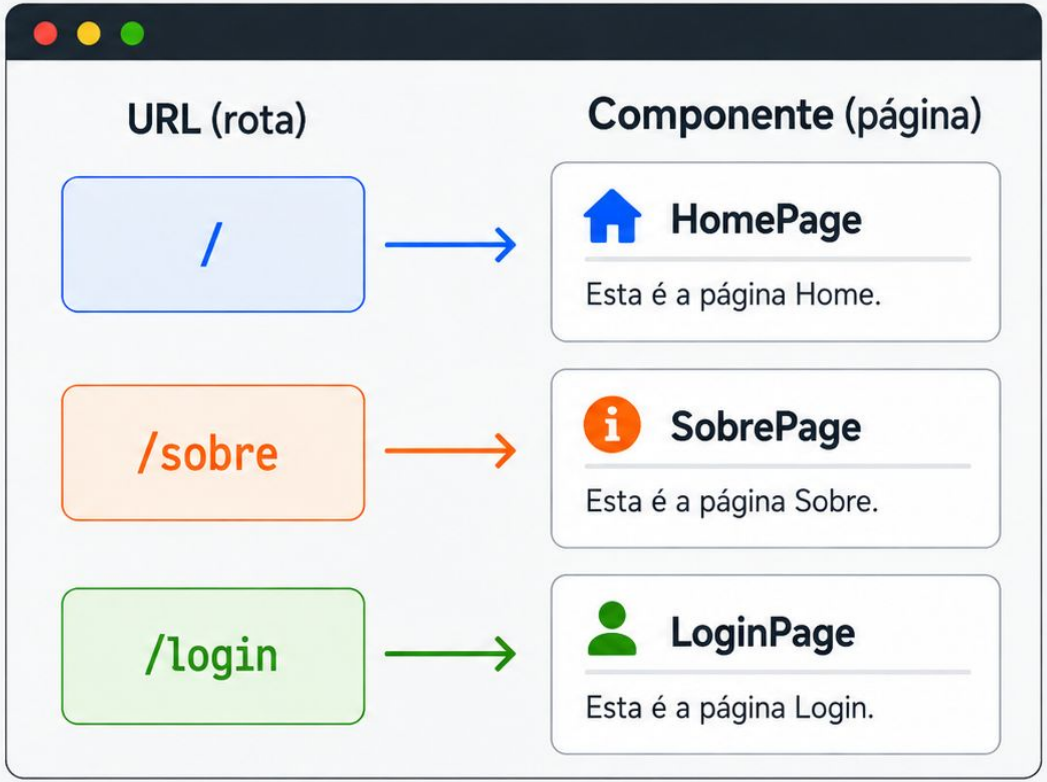
Nível 1: URL → componente

- Este é o nível que vamos dominar primeiro
- A rota diz qual componente deve aparecer
- Exemplo:

  / → HomePage

  /sobre → SobrePage

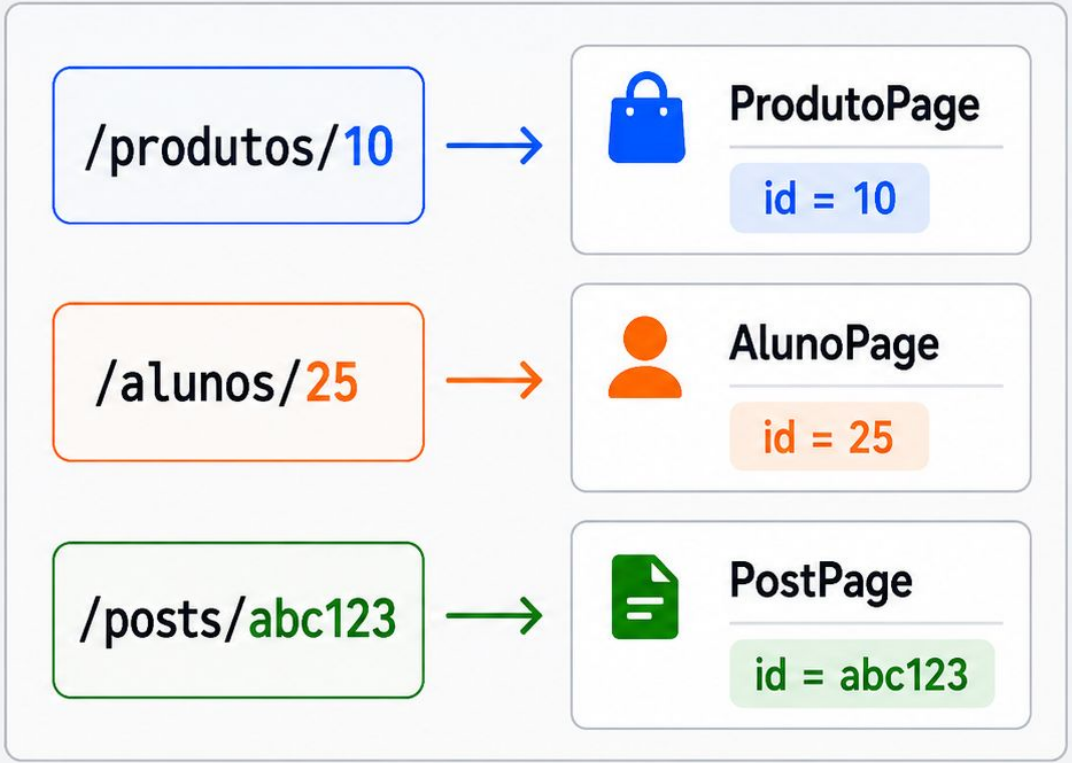
  /login → LoginPage



URL → componente

Nível 2: URL → componente + informação

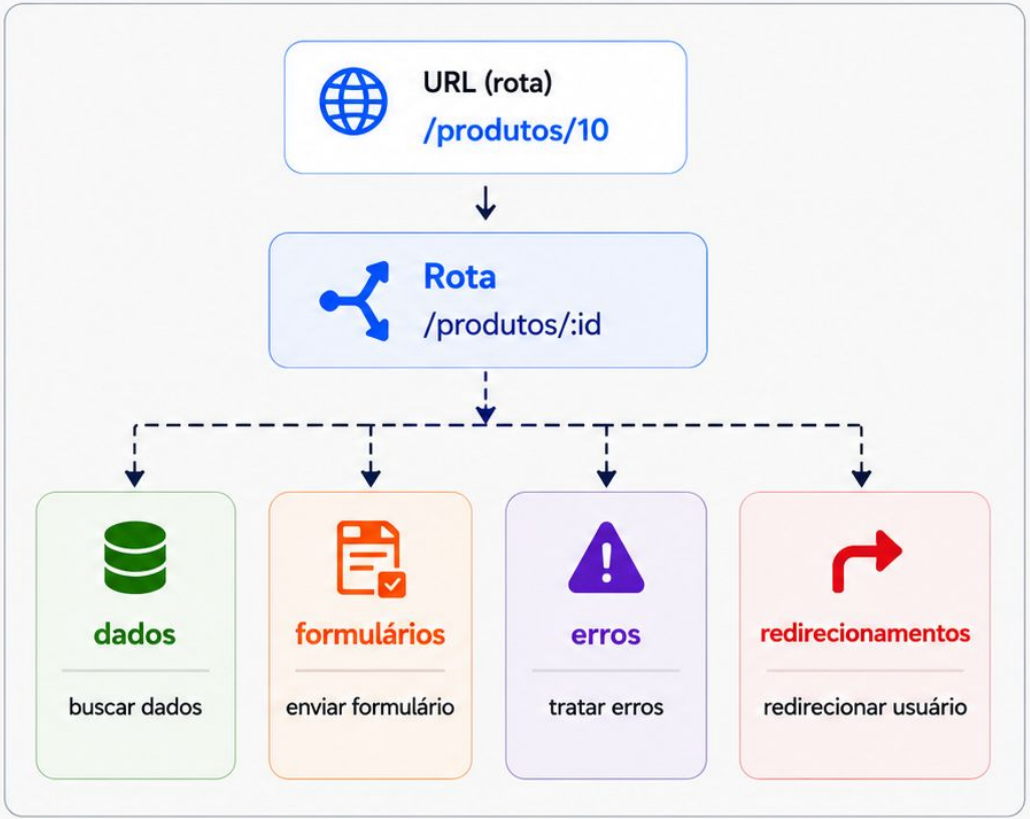
- Algumas URLs carregam informações importantes
- Exemplos:
  - /produtos/10
  - /alunos/25
  - /posts/abc123
- A URL informa qual página abrir
- E também qual item deve ser mostrado



URL → componente + informação

Nível 3: URL → dados e ações

- Em projetos maiores, a rota pode participar de mais coisas
- Por exemplo:
  - Buscar dados antes de mostrar a página
  - Tratar erros
  - Enviar dados de formulários
  - Redirecionar depois de uma ação
- Nesta aula, vamos apenas entender que isso existe



Uso mais avançado do React Router

Quando usar cada nível?

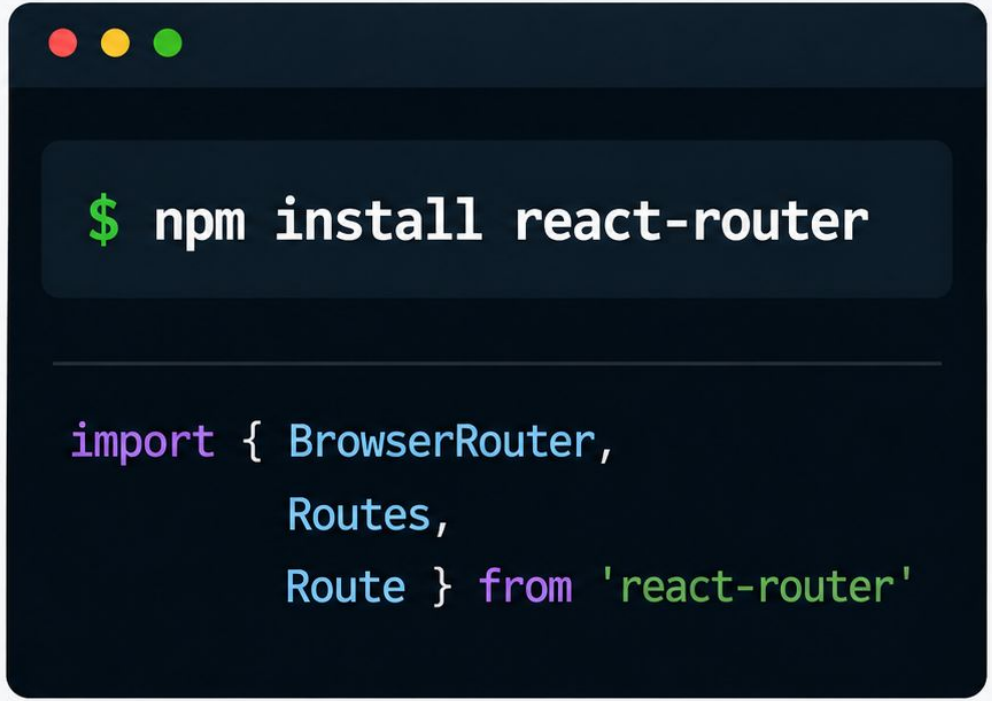
- **Projeto simples:**
  rotas básicas
- **Tela de detalhes:**
  rotas com parâmetros
- **Tela com busca ou filtro:**
  parâmetros de busca na URL
- **Sistema maior:**
  rotas também podem organizar dados, erros e ações

React Router - Instalação

- Como é uma biblioteca, precisamos instalar
- Na raiz do projeto, execute:

```
npm install react-router
```

- Depois disso, já podemos importar seus recursos no React



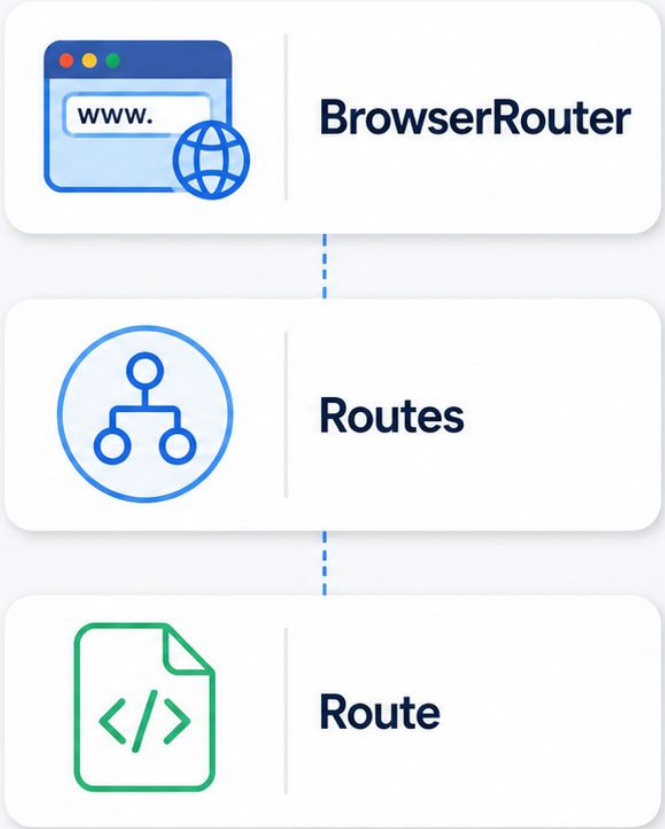
Observação: em materiais antigos, é comum encontrar react-router-dom.
Hoje o pacote principal recomendado é react-router.

Componentes Principais

Componentes do Router

- A biblioteca disponibiliza componentes para facilitar o roteamento

- Vamos começar com 3 principais:
  - BrowserRouter
  - Routes
  - Route



Componentes - BrowserRouter

Componente responsável por habilitar o roteamento no navegador

Ele deve envolver a parte da aplicação que usa rotas

Normalmente fica por fora das rotas

BrowserRouter

```
import { BrowserRouter } from "react-router";

function App() {
  return (
     <BrowserRouter>
        {/* aqui dentro vai a aplicação com rotas */}
     </BrowserRouter>
  );
}
```

Componentes - Routes

Componente responsável por agrupar as rotas

Dentro dele colocamos vários componentes Route

O React Router escolhe qual rota combina com a URL atual

```
<Routes>

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

   <Route path="/sobre" element={<SobrePage />} />

</Routes>
```

Componentes - Route

- Componente responsável por definir uma rota
- Recebe uma prop **path**
  - representa a URL
- Recebe uma prop **element**
  - representa o componente que será exibido

```
<Route path="/sobre" element={<SobrePage />} />
```

`path="/sobre"` → URL

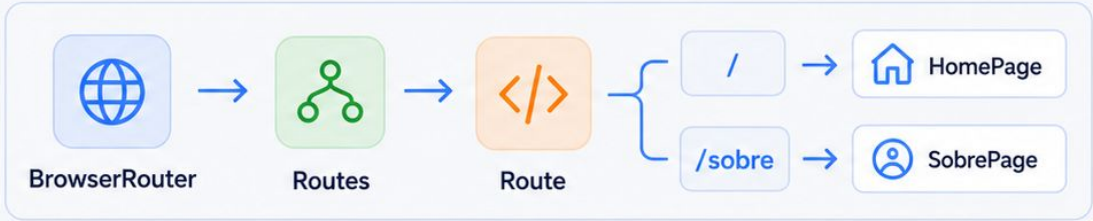
`element={<SobrePage />}` → componente exibido

Quando a URL for `/sobre`, mostre o componente **SobrePage**.

Juntando as peças

- BrowserRouter habilita o roteamento
- Routes agrupa as rotas
- Route define cada URL e sua página correspondente

```jsx
import { BrowserRouter, Routes, Route } from "react-router";

import HomePage from "./pages/HomePage";
import SobrePage from "./pages/SobrePage";

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/sobre" element={<SobrePage />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```



Página de erro

- Também podemos criar uma rota para URLs não encontradas
- Para isso, usamos: `path="*"`
- Essa rota funciona como uma página 404

```
<Route path="*" element={<NotFoundPage />} />
```

/qualquer-coisa ↓ <NotFoundPage />

Exemplo completo

- Vamos criar 3 páginas:
  - HomePage
  - SobrePage
  - NotFoundPage

- E configurar as rotas:

/	→	HomePage
/sobre	→	SobrePage
*	→	NotFoundPage

Exemplo de rotas

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<HomePage />} />
    <Route path="/sobre" element={<SobrePage />} />
    <Route path="*" element={<NotFoundPage />} />
  </Routes>
</BrowserRouter>
```

Exercício 1

- Crie um site usando React Router com 3 páginas:
  - Home
  - Sobre este site
  - Página de erro

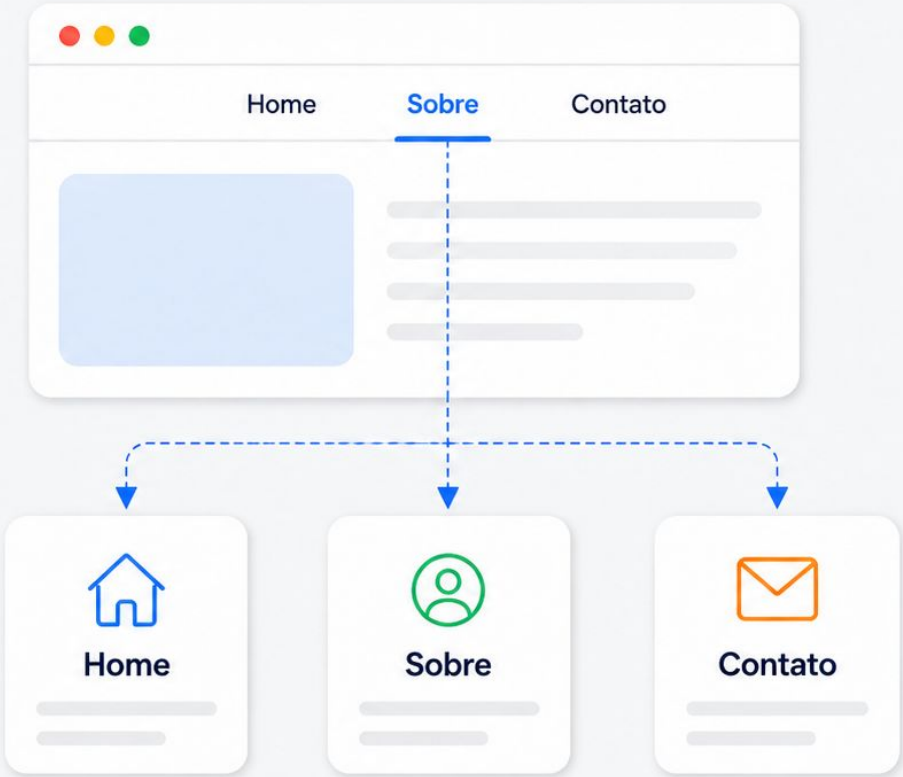
Requisitos:

- A página Home deve estar em `/`
- A página Sobre deve estar em `/sobre`
- Qualquer rota inexistente deve mostrar a página de erro
- Organize os arquivos em uma pasta `pages`

Navegação com links

Como navegar entre páginas?

- Agora temos rotas
- Mas o usuário precisa conseguir clicar e navegar
- Para isso, podemos usar:
  - Link
  - NavLink
  - useNavigate



Por que não usar <a>?

- Em HTML, usamos a tag <a> para navegar
- Exemplo:

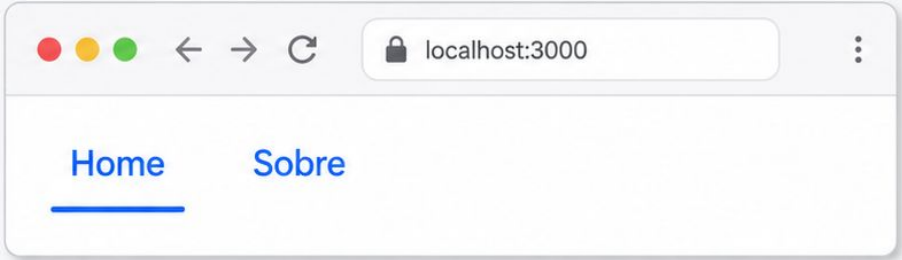
```
<a href="/sobre">Sobre</a>
```

- Em uma SPA, isso pode recarregar a página inteira
- Com React Router, usamos <Link>

Link

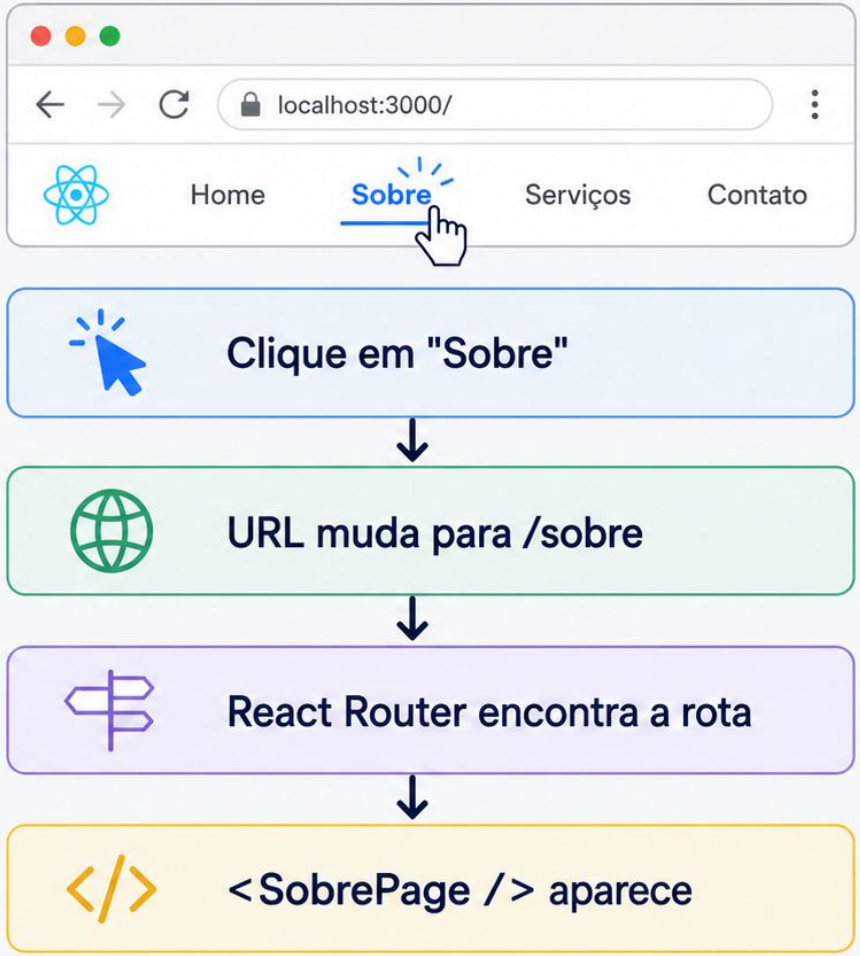
- O componente **Link** cria links internos na aplicação
- Ele muda a URL sem recarregar o site inteiro
- É ideal para menus, botões de navegação e links entre páginas

```jsx
import { Link } from "react-router";

function Menu() {
  return (
    <nav>
      <Link to="/">Home</Link>
      <Link to="/sobre">Sobre</Link>
    </nav>
  );
}
```



Link na prática

- O usuário clica no link
- A URL muda
- O React Router identifica a nova rota
- O componente correto aparece



NavLink

- O NavLink é parecido com o Link
- A diferença é que ele sabe se a rota atual está ativa
- É muito útil para menus
- Exemplo:
  destacar a página atual no menu

```jsx
import { NavLink } from "react-router";

function Menu() {
  return (
    <nav>
      <NavLink to="/">Home</NavLink>
      <NavLink to="/sobre">Sobre</NavLink>
    </nav>
  );
}
```

Link ou NavLink?

- Use **Link** quando quiser apenas navegar
- Use **NavLink** quando quiser saber qual link está ativo
- Exemplos:
  - **Link:** botão "Ver detalhes"
  - **NavLink:** menu principal do sistema

Link

Link → navegação simples

NavLink

NavLink → navegação com estado ativo

Menu fora das rotas

- O menu normalmente aparece em várias páginas
- Por isso, ele pode ficar fora do Routes
- As páginas ficam dentro do Routes
- O menu permanece fixo

```jsx
function App() {
  return (
    <BrowserRouter>
      <Menu />

      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/sobre" element={<SobrePage />} />
        <Route path="*"  element={<NotFoundPage />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Exercício 2

Adicione um menu ao projeto

O menu deve ter links para:

- Home
- Sobre

Requisitos:

- Use Link ou NavLink
- O menu deve aparecer em todas as páginas
- A navegação não deve recarregar a página inteira

useNavigate

Navegação programática

- Nem toda navegação acontece por clique em link
- Às vezes, precisamos navegar depois de uma ação
- Exemplos:
  - Depois de fazer login
  - Depois de salvar um cadastro
  - Depois de excluir um item
  - Depois de clicar em um botão específico

useNavigate

- O hook useNavigate permite navegar usando JavaScript
- Ele retorna uma função chamada navigate
- Podemos usar essa função para mudar de página

```jsx
import { useNavigate } from "react-router";

function HomePage() {
  const navigate = useNavigate();

  function irParaSobre() {
    navigate("/sobre");
  }

  return (
    <button onClick={irParaSobre}>
      Ir para Sobre
    </button>
  );
}
```

Navigate

A função **navigate** pode ser usada de algumas formas:

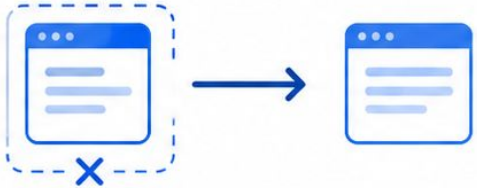
```
navigate("/rota")
```

1. vai para a página indicada

```
navigate(-1)
```

2. volta para a página anterior

```
navigate("/rota", { replace: true })
```



3. troca a página atual no histórico

useNavigate - Exemplo

- Neste exemplo, um botão leva o usuário para a página Sobre

```jsx
function HomePage() {
  const navigate = useNavigate();

  function irParaSobre() {
    navigate("/sobre");
  }

  return (
    <div>
      <h1>Home</h1>
      <button onClick={irParaSobre}>
        Ir para Sobre
      </button>
    </div>
  );
}
```

Link ou useNavigate?

- Use `Link` quando a navegação for parte da interface
- Use `useNavigate` quando a navegação depender de lógica

Exemplos

- Menu do site: use `Link`
- Depois de login bem-sucedido: use `useNavigate`

	Clique em menu → `Link`
	Botão simples → `Link` ou `useNavigate`
	Depois de salvar → `useNavigate`
	Depois de login → `useNavigate`
	Voltar página → `useNavigate(-1)`

Exercício 3

Agora faremos navegação com `useNavigate`

Tela Home:

- Adicionar um botão que vá para a tela **Sobre**

Tela Sobre:

- Adicionar um botão que vá para a tela **Home**
- Adicionar um botão que volte para a última página

Path params e useParams

O problema dos detalhes

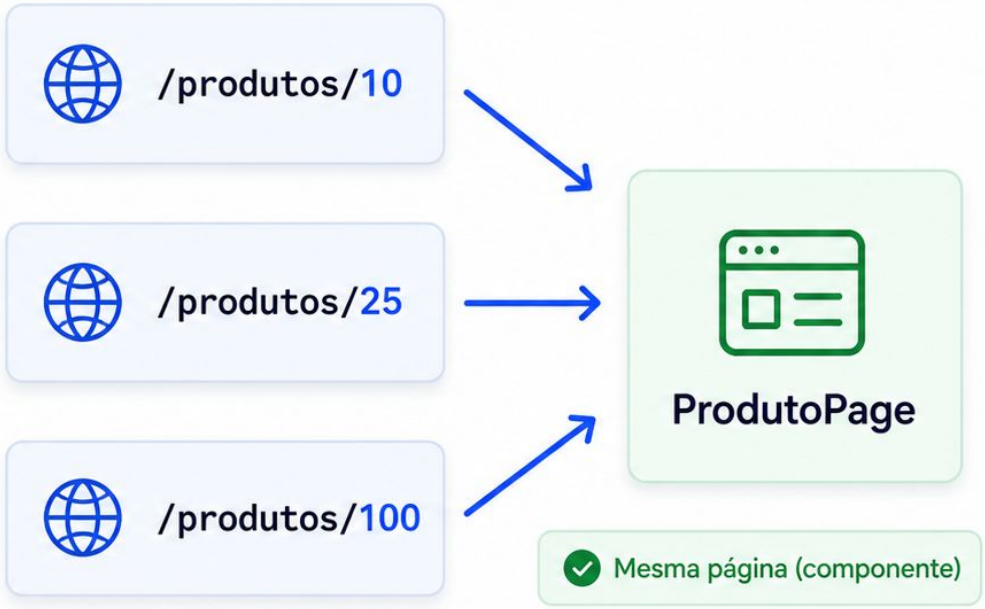
- Imagine uma tela de produtos
- Cada produto precisa ter uma página de detalhes
- **Exemplo:**
  - Produto 1
  - Produto 2
  - Produto 3
- Não faz sentido criar uma rota manual para cada produto

EXEMPLO RUIM

```
<Route path="/produto1" element={<Produto1 />} />
<Route path="/produto2" element={<Produto2 />} />
<Route path="/produto3" element={<Produto3 />} />
```

Path Params

- **Path params** são informações colocadas dentro do caminho da URL
- **Exemplo:**
  `/produtos/10`
  `/produtos/25`
  `/produtos/100`
- O número pode mudar
- Mas a página continua sendo a mesma



Criando uma rota com parâmetro

- Para criar um parâmetro na rota, usamos dois pontos

- **Exemplo:**

  `/produtos/:id`

- O `:id` indica que aquela parte da URL é variável

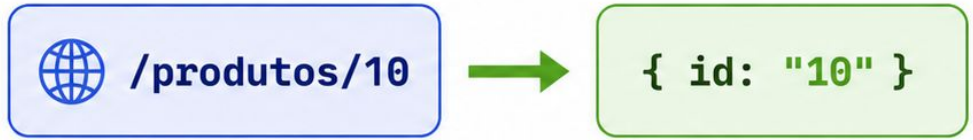
```
<Route path="/produtos/:id"
  element={<ProdutoPage />} />
```

`/produtos/10`	→	`id = 10`
`/produtos/25`	→	`id = 25`

useParams

- O hook useParams permite acessar os parâmetros da URL
- Ele retorna um objeto
- Cada parâmetro da rota vira uma propriedade desse objeto

```jsx
import { useParams } from "react-router";

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

  return (
    <h1>Produto selecionado: {id}</h1>
  );
}
```



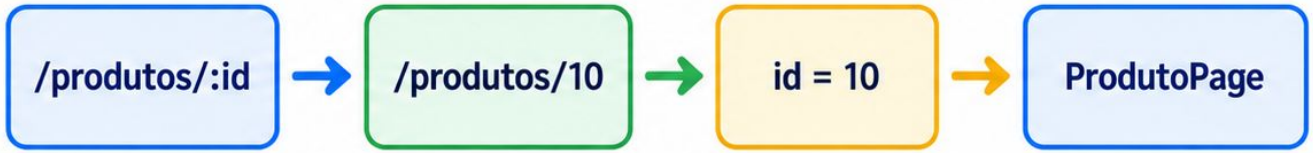
Exemplo completo

- **Rota:**
  /produtos/:id

- **URL acessada:**
  /produtos/10

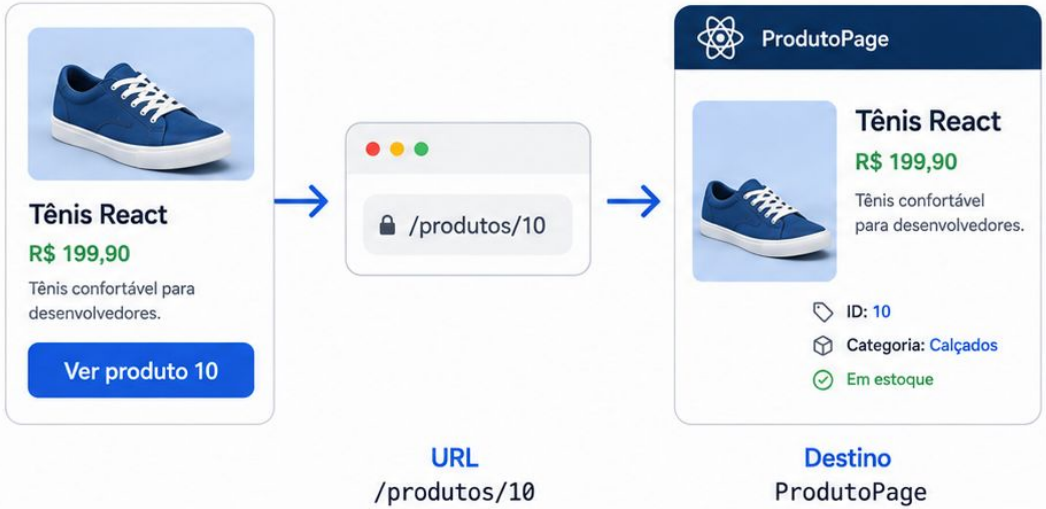
- **Resultado:**
  ProdutoPage recebe id = 10

```jsx
<Route path="/produtos/:id" element={<ProdutoPage />} />

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

  return <h1>Produto {id}</h1>;
}
```



Navegando com parâmetro

- Podemos navegar para uma rota dinâmica usando Link
- Exemplo:



Link

```
<Link to="/produtos/10">
  Ver produto 10
</Link>
```

useNavigate

```
navigate("/produtos/10");
```

Exercício 4

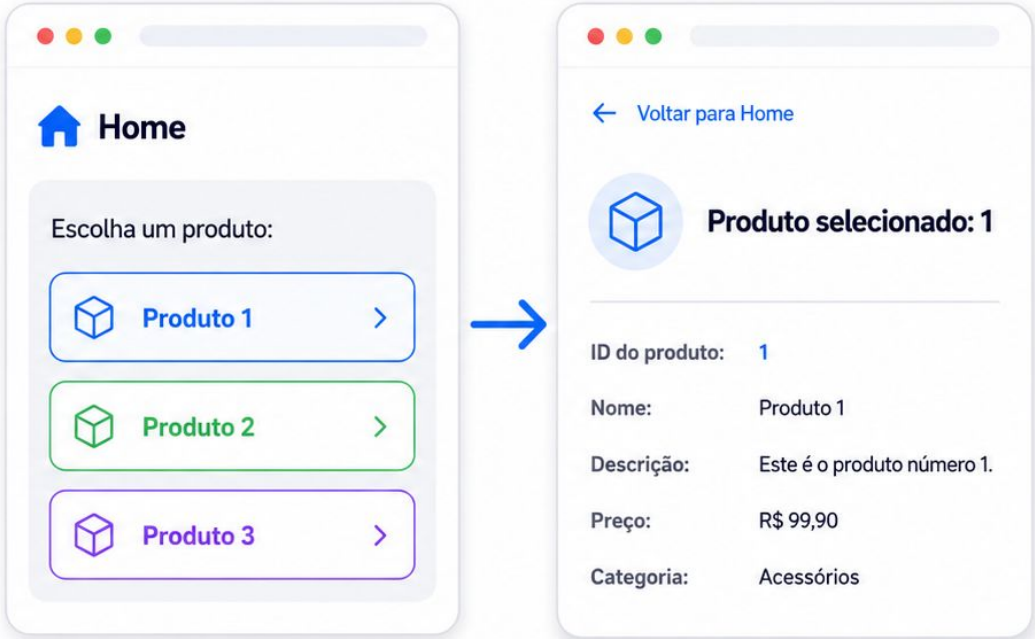
Crie uma página de detalhes de produto

Rotas:

- /produtos/1
- /produtos/2
- /produtos/3

Requisitos:

- Criar a rota /produtos/:id
- Criar ProdutoPage
- Usar useParams para mostrar o id do produto
- Criar links na Home para acessar produtos diferentes



Search params e useSearchParams

Path params não resolvem tudo

- Path params são ótimos para identificar um item
- Exemplo:

  `/produtos/10`

- Mas e quando queremos representar uma busca?

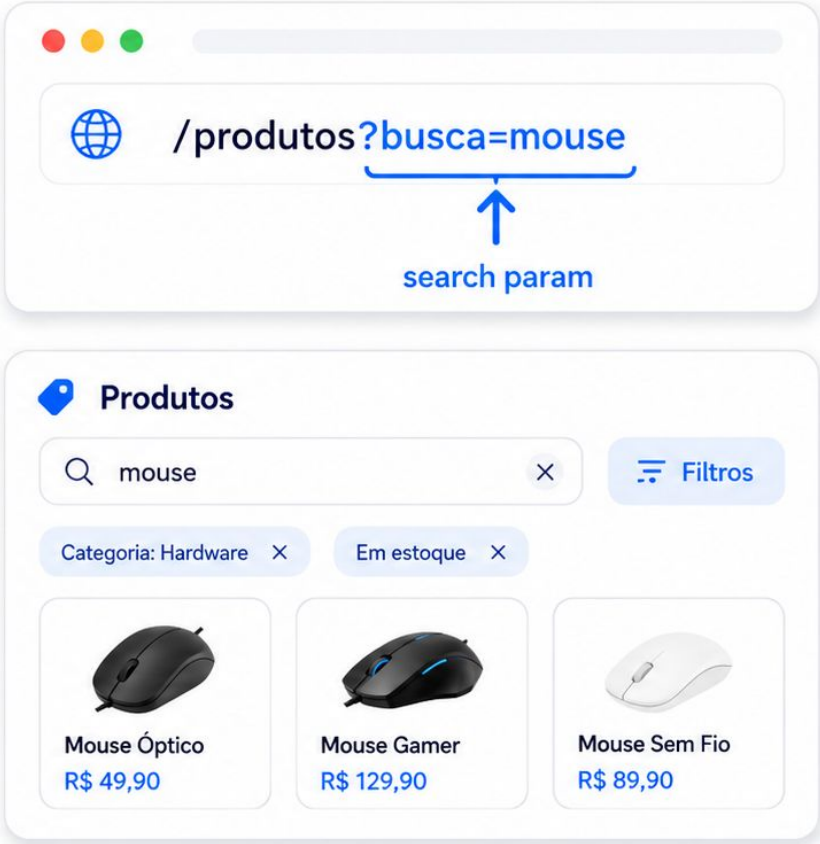
  `/produtos?busca=mouse`

- Para isso usamos search params

Search Params

- Search params são parâmetros depois do sinal `?`
- Eles representam filtros, buscas e configurações da tela
- **Exemplo:**

```
/produtos?busca=mouse
/produtos?categoria=hardware
/produtos?pagina=2
```



Path params vs Search params

- **Path params identificam algo principal**

  `/produtos/10`

- **Search params filtram ou configuram a visualização**

  `/produtos?busca=mouse`

- **Os dois podem aparecer juntos**

Exemplos

`/produtos/10`	→	detalhes do produto 10
`/produtos?busca=tv`	→	lista filtrada por tv

Path params	Search params
Identificam um recurso Representam algo único e específico.	**Filtram ou configuram** Ajustam a visualização ou o comportamento.
Fazem parte da URL Alteram o caminho da rota. Ex.: `/produtos/10`	**Vêm após o "?"** Não alteram o caminho da rota. Ex.: `?busca=mouse`
Usados para detalhes Ideal para páginas de detalhe ou itens individuais.	**Usados para busca e filtros** Perfeito para listas, ordenação, paginação, etc.
Definidos nas rotas São declarados como parâmetros dinâmicos. Ex.: `/produtos/:id`	**Acessados via useSearchParams** API do React Router para ler e atualizar os parâmetros.

useSearchParams

- O hook useSearchParams permite ler parâmetros de busca da URL
- Ele funciona com valores depois do ?
- **Exemplo:**

`/produtos?busca=teclado`

```
import { useSearchParams } from "react-router";

function ProdutosPage() {
  const [searchParams] = useSearchParams();

  const busca = searchParams.get("busca");

  return <h1>Busca: {busca}</h1>;
}
```

/produtos?busca=teclado → Busca: teclado

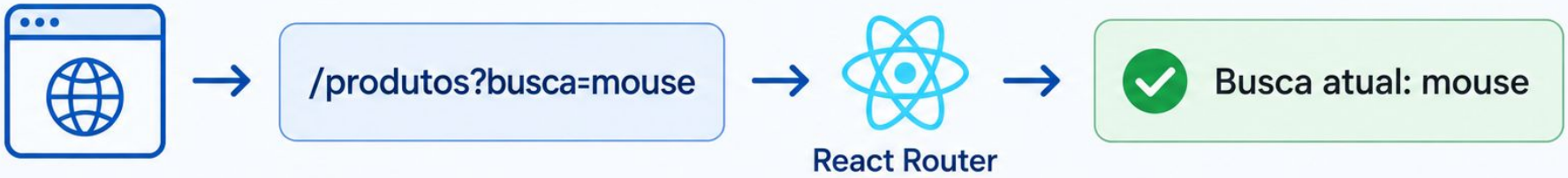
Exemplo com busca

URL:
/produtos?busca=mouse

Código:
searchParams.get("busca")

Resultado:
"mouse"

```jsx
function ProdutosPage() {
  const [searchParams] = useSearchParams();

  const busca = searchParams.get("busca");

  return (
    <div>
      <h1>Produtos</h1>
      <p>Busca atual: {busca}</p>
    </div>
  );
}
```



Exercício 5

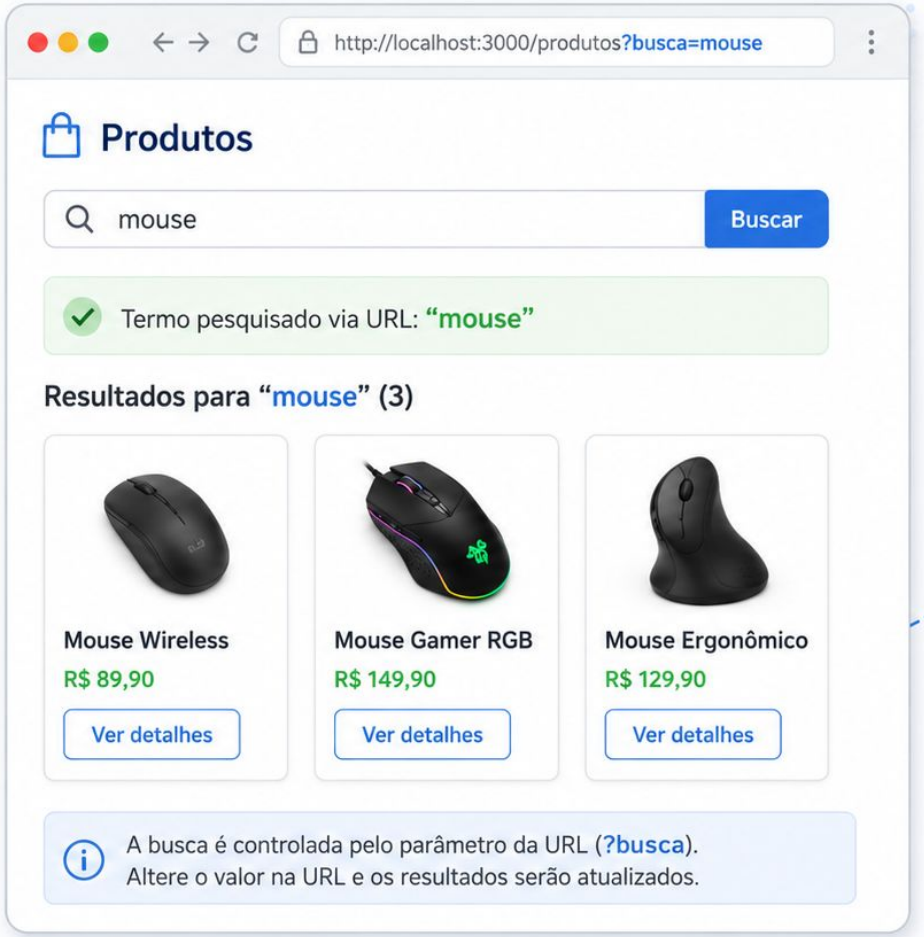
Crie uma página de produtos com busca pela URL

Exemplos de URL:

- `/produtos?busca=mouse`
- `/produtos?busca=teclado`
- `/produtos?busca=monitor`

Requisitos:

- Criar ProdutosPage
- Usar useSearchParams
- Mostrar na tela o termo pesquisado



Organização do projeto

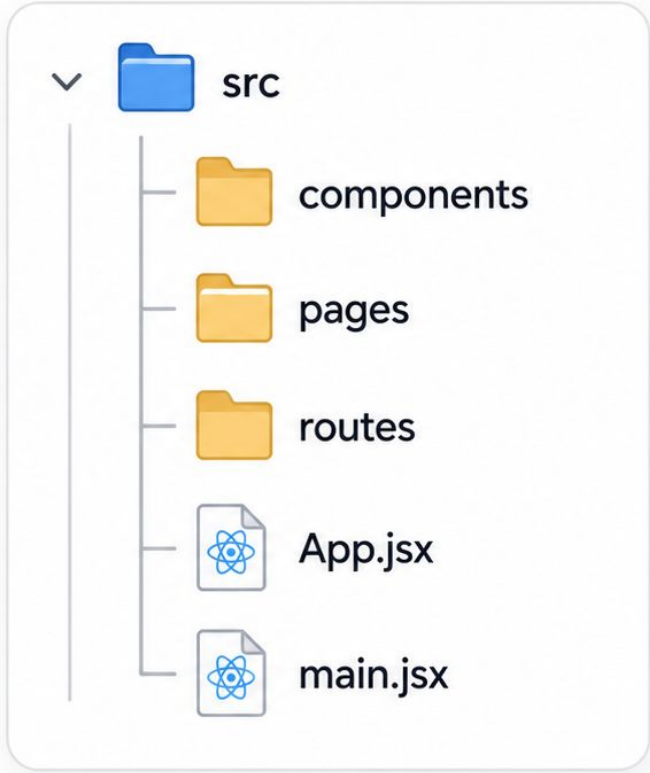
Como organizar páginas

- Cada rota deve apontar para uma página
- Em React, uma página também é um componente
- Mas tratamos como página quando ela representa uma tela inteira
- Exemplos:
  - → `HomePage`
  - → `LoginPage`
  - → `ProdutosPage`
  - → `ProdutoDetalhesPage`

Organização de arquivos

Uma organização simples:

```
src/
├── components/
├── pages/
├── routes/
├── App.jsx
└── main.jsx
```



Explicação:

components → componentes reutilizáveis

pages → telas principais

routes → configuração das rotas

Separando as rotas

- Podemos criar um componente só para as rotas
- Isso deixa o App mais limpo
- Nome comum: **AppRoutes**

App.jsx

```jsx
import AppRoutes from "./AppRoutes";

function App() {
  return (
    <BrowserRouter>
      <AppRoutes />
    </BrowserRouter>
  );
}

export default App;
```

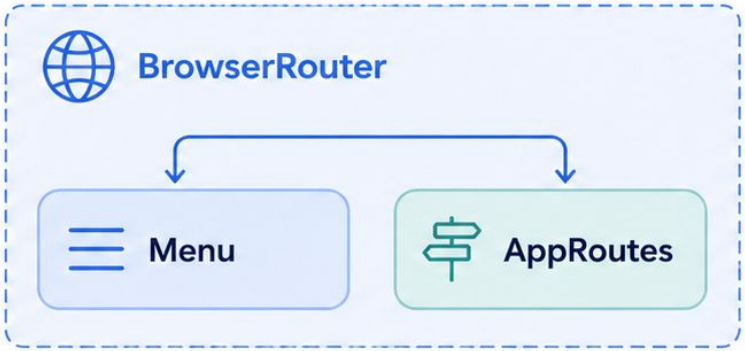
AppRoutes.jsx

Define todas as rotas da aplicação

```jsx
import { Routes, Route } from "react-router";

import HomePage from "../pages/HomePage";
import SobrePage from "../pages/SobrePage";
import NotFoundPage from "../pages/NotFoundPage";

function AppRoutes() {
  return (
    <Routes>
      <Route path="/" element={<HomePage />} />
      <Route path="/sobre" element={<SobrePage />} />
      <Route path="*" element={<NotFoundPage />} />
    </Routes>
  );
}

export default AppRoutes;
```

App mais limpo

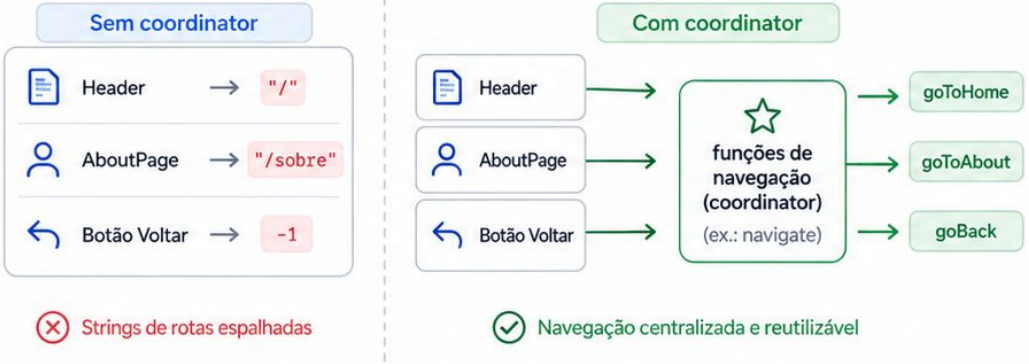
- O App pode ficar responsável pela estrutura geral
- Exemplo:
  - BrowserRouter
  - Menu
  - AppRoutes




```jsx
import { BrowserRouter } from "react-router";
import Menu from "./components/Menu";
import AppRoutes from "./routes/AppRoutes";

function App() {
  return (
    <BrowserRouter>
      <Menu />
      <AppRoutes />
    </BrowserRouter>
  );
}

export default App;
```

Como organizar rotas: coordinator

- Em alguns projetos, podemos centralizar funções de navegação
- Isso evita espalhar strings de rotas pelo sistema
- Exemplo:
  - → goToHome
  - → goToAbout
  - → goBack




```
export function goToHome(navigate) {
  navigate("/");
}

export function goToAbout(navigate) {
  navigate("/sobre");
}

export function goBack(navigate) {
  navigate(-1);
}
```

Quando usar coordinator?

- Pode ser útil quando várias partes do sistema navegam para as mesmas rotas
- Ajuda a evitar erros de digitação
- Facilita mudanças futuras
- Mas em projetos pequenos, usar navigate direto também é aceitável

Exemplo

Sem coordinator:

```
navigate("/produtos/10")
```

→ Navegação direta e dispersa

Com coordinator:

```
goToProductDetails(navigate,10)
```

→ Uso de helper centralizado e consistente

Coordinator centraliza a navegação e torna o código mais consistente e fácil de manter.

Deploy

Um cuidado no deploy

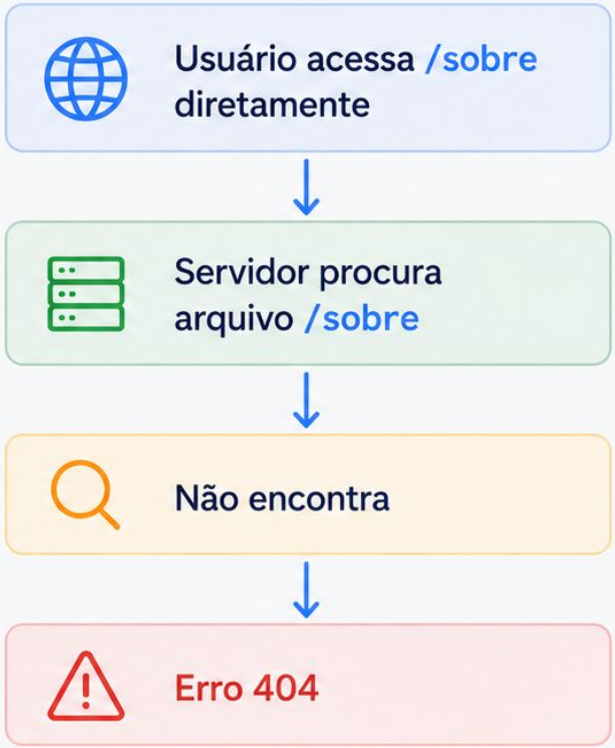
- Em uma SPA, o React controla as rotas no navegador
- Isso funciona bem enquanto navegamos dentro do app
- Mas pode dar problema ao atualizar a página em uma rota interna
- Exemplo:

`/sobre`

React controla as rotas no navegador.
A URL muda, mas tudo acontece dentro da SPA.

O problema do refresh

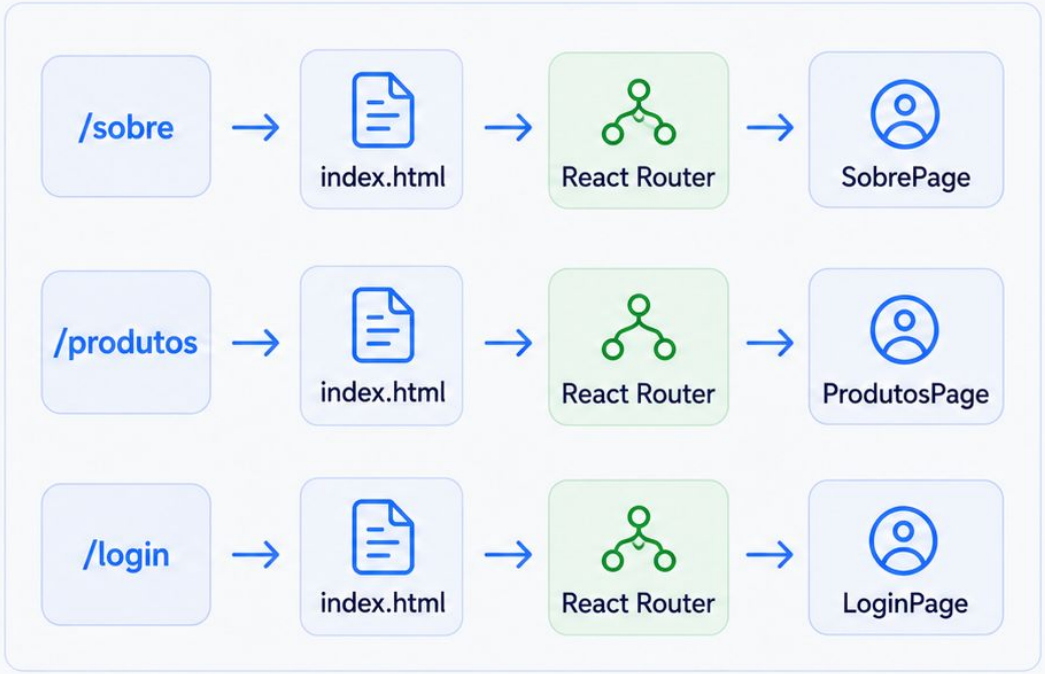
- Dentro do app:
  - clicar em Sobre funciona
- Mas se o usuário atualizar a página em /sobre:
  - o servidor pode procurar uma página /sobre real
- Se ela não existir, pode retornar erro 404



A solução

- O servidor precisa enviar o index.html para as rotas da SPA
- Depois disso, o React Router assume o controle
- Ideia:

qualquer rota do app → **index.html**



Configuração comum:
qualquer rota não encontrada deve retornar o `index.html`

Mapa mental

Exemplo:

/sobre

```
<Route path="/sobre" element={<SobrePage />} />
```

<SobrePage />

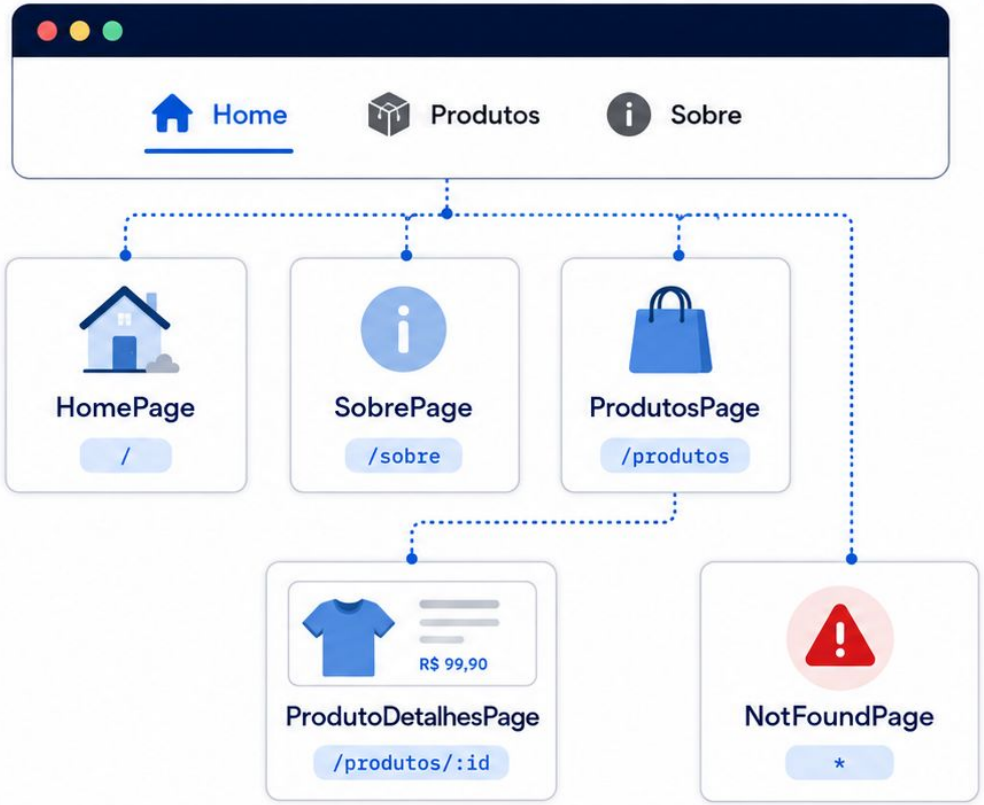
Mini-desafio final

Crie uma pequena aplicação com:

- Menu com Home, Produtos e Sobre
- Página Home
- Página Sobre
- Página Produtos
- Página de detalhes do produto
- Página de erro

Rotas obrigatórias:

```
</>  /
</>  /sobre
</>  /produtos
</>  /produtos/:id
</>  *
```



Requisitos do desafio

- Usar BrowserRouter
- Usar Routes
- Usar Route
- Usar Link ou NavLink no menu
- Usar useNavigate em pelo menos um botão
- Usar useParams na página de detalhes
- Usar useSearchParams na página de produtos