Aula 06 — Rotas no React

React Router & Navegação entre páginas

Prof. Me. José Olinda

SPA - Single Page Application

Um aplicação **SPA (Single Page Application)** é uma aplicação web que carrega uma única página HTML e atualiza dinamicamente o conteúdo sem recarregar a página inteira.

O que é um roteador (Router)?

Um **roteador** gerencia a navegação entre diferentes páginas/componentes sem recarregar a página.

Sem React Router:

- Cada link recarrega a página inteira
- Você perde o estado da aplicação
- Experiência mais lenta

--

Com React Router:

- Navegação rápida e fluida
- Estado da aplicação preservado
- URLs amigáveis e bookmarkáveis

Instalando React Router

01 - Crie um novo projeto React

```
npm create vite@latest meu-app --template react
cd meu-app
npm install
```

02 - Instale o React Router

```
npm i react-router
```

Configurando BrowserRouter

No arquivo `main.jsx`, configure o `<BrowserRouter>` envolvendo sua aplicação para permitir usar o roteador.

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router";
import App from "./App";
const root = document.getElementById("root");
ReactDOM.createRoot(root).render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

Criando rotas básicas

No arquivo `App.jsx`, configure suas rotas:

```jsx
import { Routes, Route } from "react-router";
import Home from "./pages/Home";
import About from "./pages/About";
import Contact from "./pages/Contact";

export default function App() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/sobre" element={<About />} />
      <Route path="/contato" element={<Contact />} />
    </Routes>
  );
```

- `<Routes>` — container para todas as rotas
- `<Route>` — define uma rota com `path` e `element`

Estrutura de pastas recomendada

```
src/
├── App.jsx
├── main.jsx
├── App.css
├── pages/
│   ├── Home.jsx
│   ├── About.jsx
│   ├── Contact.jsx
│   └── NotFound.jsx
├── components/
│   └── Header.jsx
└── assets/
```

Página Home.jsx

```jsx
export default function Home() {
  return (
    <div className="page">
      <h1>Bem-vindo ao Home</h1>
      <p>Esta é a página inicial da aplicação.</p>
    </div>
  );
}
```

Página About.jsx

```jsx
export default function About() {
  return (
    <div className="page">
      <h1>Sobre</h1>
      <p>Saiba mais sobre nossa aplicação.</p>
    </div>
  );
}
```

Página Contact.jsx

```jsx
export default function Contact() {
  return (
    <div className="page">
      <h1>Contato</h1>
      <p>Entre em contato conosco.</p>
    </div>
  );
}
```

Navegando com Link

Use `<Link>` para navegação sem estilos ativos. É equivalente a `<a>` mas sem recarregar a página.

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

export default function Header() {
  return (
    <header>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/sobre">Sobre</Link>
        <Link to="/contato">Contato</Link>
      </nav>
    </header>
```

Navegando com NavLink

Use `<NavLink>` quando você precisa de **estilos ativos**:

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

export default function Header() {
  return (
    <header>
      <nav>
        <NavLink to="/" className={({ isActive }) => isActive ? "active" : ""}>
          Home
        </NavLink>
        <NavLink to="/sobre" className={({ isActive }) => isActive ? "active" : ""}>
          Sobre
        </NavLink>
        <NavLink to="/contato" className={({ isActive }) => isActive ? "active" : ""}>
          Contato
        </NavLink>
```

CSS para NavLink ativo

```css
nav {
  display: flex; gap: 20px; padding: 15px; background-color: #44475a;
}

nav a {
  color: #f8f8f2; text-decoration: none;
  padding: 8px 16px; border-radius: 4px; transition: 0.3s;
}

nav a:hover {
  background-color: #6272a4;
}

nav a.active {
  background-color: #bd93f9; color: #282a36; font-weight: bold;
```

Parâmetros dinâmicos (Dynamic Segments)

Crie rotas com parâmetros usando `:` :

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/produtos/:id" element={<Produto />} />
</Routes>
```

A rota `/produtos/123` passará `id: "123"` como parâmetro.

Acessando parâmetros com useParams

`useParams()` retorna um objeto com todos os parâmetros da rota.

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

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

  return (
    <div className="page">
      <h1>Produto #{id}</h1>
      <p>Detalhes do produto com ID: {id}</p>
    </div>
  );
}
```

Navegando com useNavigate

Use `useNavigate()` para navegar **programaticamente** (sem clique):

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

export default function LoginPage() {
  const navigate = useNavigate();

  const handleLogin = () => {
    // Simula login
    console.log("Usuário logado!");
    // Navega após sucesso
    navigate("/dashboard");
  };
```

Use `useNavigate` para:

- Redirecionamento após formulário
- Logout automático
- Navegação temporizada

Rotas aninhadas (Nested Routes)

Crie rotas dentro de rotas para compartilhar layouts, podendo renderizar uma página dentro da outra:

```
<Routes>
  <Route path="/" element={<Layout />}>
    <Route index element={<Home />} />
    <Route path="sobre" element={<About />} />
    <Route path="contato" element={<Contact />} />
  </Route>
</Routes>
```

Layout com Outlet

No componente `Layout`, use `<Outlet>` para renderizar a página filha:

```
import { Outlet } from "react-router";
import Header from "../components/Header";
import Footer from "../components/Footer";

export default function Layout() {
  return (
    <div className="container">
      <Header />
      {/* Aqui a página filha será renderizada */}
      <Outlet />
      <Footer />
    </div>
```

Rota 404 - Página não encontrada

O `path="*"` funciona como "catchall" para rotas não encontradas.

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

export default function App() {
  return (
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route index element={<Home />} />
        <Route path="sobre" element={<About />} />
        <Route path="*" element={<NotFound />} />
      </Route>
    </Routes>
  );
```

Página NotFound.jsx

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

export default function NotFound() {
  return (
    <div className="page">
      <h1>404 – Página não encontrada</h1>
      <p>Desculpe, a página que você procura não existe.</p>
      <Link to="/">Voltar ao Home</Link>
    </div>
  );
}
```

Atividade: criar um blog

Vamos criar um **blog simples** com React Router

Funcionalidades:

- Página Home com lista de posts
- Página individual de cada post
- Navegação entre posts
- Menu com NavLink

Estrutura do Blog

```
src/
├── App.jsx
├── main.jsx
├── App.css
├── pages/
│   ├── Home.jsx
│   ├── Post.jsx
│   └── NotFound.jsx
├── components/
│   └── Header.jsx
└── data/
    └── posts.js
```

data/posts.js — Dados dos posts

```
export const posts = [
  {
    id: 1,
    title: "Introdução ao React",
    excerpt: "Aprenda os fundamentos do React...",
    content: "React é uma biblioteca JavaScript para criar interfaces..."
  },
  {
    id: 2,
    title: "Hooks no React",
    excerpt: "Entenda useState, useEffect e mais...",
    content: "Hooks são funções que permitem usar estado em componentes funcionais..."
  },
  {
    id: 3,
    title: "React Router",
    excerpt: "Navegação em aplicações React...",
```

pages/Home.jsx — Lista de posts

```jsx
import { Link } from "react-router";
import { posts } from "../data/posts";

export default function Home() {
  return (
    <div className="page">
      <h1>Blog</h1>
      <div className="posts-list">
        {posts.map(post => (
          <article key={post.id} className="post-card">
            <h2>{post.title}</h2>
            <p>{post.excerpt}</p>
            <Link to={`/posts/${post.id}`} className="read-more">
              Ler mais →
            </Link>
```

pages/Post.jsx — Detalhes do post

```jsx
import { useParams, Link } from "react-router";
import { posts } from "../data/posts";

export default function Post() {
  const { id } = useParams();
  const post = posts.find(p => p.id === parseInt(id));

  if (!post) {
    return (
      <div className="page">
        <h1>Post não encontrado</h1>
        <Link to="/">Voltar ao Blog</Link>
      </div>
    );
  }
```

App.jsx — Rotas do Blog

```jsx
import { Routes, Route } from "react-router";
import Header from "./components/Header";
import Home from "./pages/Home";
import Post from "./pages/Post";
import NotFound from "./pages/NotFound";
import "./App.css";

export default function App() {
  return (
    <div className="container">
      <Header />
      <Routes>
        <Route index element={<Home />} />
        <Route path="/posts/:id" element={<Post />} />
        <Route path="*" element={<NotFound />} />
```

components/Header.jsx

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

export default function Header() {
  return (
    <header className="header">
      <div className="header-content">
        <h1 className="logo">📝 Meu Blog</h1>
        <nav>
          <NavLink
            to="/"
            className={({ isActive }) => isActive ? "active" : ""}
            end
          >
            Home
```

App.css — Estilos do Blog

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #1e1e2e;
  color: #f8f8f2;
}

.container {
  display: flex;
  flex-direction: column;
```

App.css — Posts (continuação)

```css
.page {
  flex: 1;
  padding: 40px 20px;
  max-width: 1200px;
  margin: 0 auto;
  width: 100%;
}

.posts-list {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
  margin-top: 30px;
}
```

App.css — Detalhes do Post

```
.back-link {
  color: #bd93f9;
  text-decoration: none;
  font-size: 1.1em;
  margin-bottom: 30px;
  display: inline-block;
  transition: 0.3s;
}

.back-link:hover {
  color: #8be9fd;
}

.post-detail {
  background-color: #44475a;
```

Hooks úteis do React Router

Hook	Uso
`useParams()`	Acessar parâmetros da rota
`useNavigate()`	Navegar programaticamente
`useLocation()`	Acessar informações da URL atual
`useSearchParams()`	Manipular query strings

Todos são importados de `"react-router"` e usados em componentes.

Referências

 **React Router Documentation:**

- Instalação: https://reactrouter.com/start/declarative/installation
- Routing: https://reactrouter.com/start/declarative/routing
- Navegação: https://reactrouter.com/start/declarative/navigating