React – Router

React Router

Até agora vimos como criar componentes e gerenciar a arvore de estados no React, mas fizemos tudo isso em uma única página na nossa aplicação. E se quisermos ter mais de uma página com URLs diferentes? Ai entra o React Router

React Router é uma biblioteca padrão para gerenciar rotas em aplicações React. Ele permite a criação de navegações em Single Page Applications (SPAs), onde a transição entre as páginas é rápida e não requer um recarregamento completo da página. Em vez disso, ele manipula a troca de componentes de forma eficiente, proporcionando uma experiência de usuário fluida.

React Router

1. Vamos criar um novo projeto
2. Vamos instalar o React Router com o comando npm install react-router react-router-dom. As dependências devem ser adicionadas ao nosso package.json

```
},
"dependencies": {
  "react": "^18.3.1",
  "react-dom": "^18.3.1",
  "react-router": "^6.24.1",
  "react-router-dom": "^6.24.1"
},
"devDependencies": {
```

3. Vamos criar uma pasta chamada Pages, e dentro dessa pasta criar mais duas: Home e About
4. Dentro de cada pasta vamos criar os arquivos Home.tsx e Home.css, e About.tsx e Home.css respectivamente

React Router

5. Vamos criar uma estrutura simples para a nossa Home Page e fazer o mesmo para a About

```
import './Home.css'

const HomePage = () => {
  return (
    <div>
      <h1>Home Page</h1>
    </div>
  )
}

export default HomePage
```

```
import './About.css'

const AboutPage = () => {
  return (
    <div>
      <h1>About Page</h1>
    </div>
  )
}

export default AboutPage
```

React Router

6. Com as duas páginas criadas, vamos até o nosso App.tsx e começar a criar o nosso roteamento. Ao Abrir a aplicação devemos ver a nossa HomePage

```tsx
import { Route, BrowserRouter as Router, Routes } from 'react-router-dom'
import './App.css'
import HomePage from './pages/Home/Home'
import AboutPage from './pages/About/About'

function App() {
  return (
    // Componente de Instancia geral das rotas do BrowserRouter
    <Router>
      {/* Componente que indica as rotas, funciona como um switch */}
      <Routes>
        {/* Componente que indica a rota, funciona como um case */}
        <Route path="/" element={<HomePage />} />
        <Route path="/about" element={<AboutPage />} />
        <Route path="*" element={<HomePage />} />
      </Routes>
    </Router>
  )
}

export default App
```

React Router

7. Vamos criar um botão em ambas as páginas para criar a navegação. Para isso vamos usar o componente Link do React Router

router > src > pages > Home > Home.tsx > ...

```tsx
import { Link } from 'react-router-dom'
import './Home.css'

const HomePage = () => {
  return (
    <div>
      <h1>Home Page</h1>
      <Link to="/about">Go to About</Link>
    </div>
  )
}

export default HomePage
```

```tsx
import { Link } from 'react-router-dom'
import './About.css'

const AboutPage = () => {
  return (
    <div>
      <h1>About Page</h1>
      <Link to="/">Go to Home</Link>
    </div>
  )
}

export default AboutPage
```

React Router

8. Vamos criar mais uma página, a NotFoundPage na mesma estrutura

```
router > src > pages > NotFound > NotFound.tsx > ...
1   import { Link } from 'react-router-dom'
2   import './NotFound.css'
3
4   const NotFoundPage = () => {
5     return (
6       <div>
7         <h1>Not Found - 404</h1>
8         <Link to="/">Go to Home</Link>
9       </div>
10    )
11  }
12
13  export default NotFoundPage
14
```

React Router

8. No App.tsx vamos usar o path=‘*’ para receber a nossa página de 404. O asterisco significa qualquer coisa, é um coringa. Como o código é executado de cima para baixo, o React primeiro vai procurar pela rota raiz (/), depois pela rota about (/about) e se não for nenhum das duas, vai cair no nosso asterisco

router > src > App.tsx > ...

```tsx
import { Route, BrowserRouter as Router, Routes } from 'react-router-dom'
import './App.css'
import HomePage from './pages/Home/Home'
import AboutPage from './pages/About/About'
import NotFoundPage from './pages/NotFound/NotFound'

function App() {
  return (
    // Componente de Instancia geral das rotas do BrowserRouter
    <Router>
      {/* Componente que indica as rotas, funciona como um switch */}
      <Routes>
        {/* Componente que indica a rota, funciona como um case */}
        <Route path="/" element={<HomePage />} />
        <Route path="/about" element={<AboutPage />} />
        {/* Componente que indica a rota default, funciona como um default */}
        <Route path="*" element={<NotFoundPage />} />
      </Routes>
    </Router>
  )
}

export default App
```

React Router

8. No App.tsx vamos usar o path=‘*’ para receber a nossa página de 404. O asterisco significa qualquer coisa, é um coringa. Como o código é executado de cima para baixo, o React primeiro vai procurar pela rota raiz (/), depois pela rota about (/about) e se não for nenhum das duas, vai cair no nosso asterisco

router > src > App.tsx > ...

```tsx
import { Route, BrowserRouter as Router, Routes } from 'react-router-dom'
import './App.css'
import HomePage from './pages/Home/Home'
import AboutPage from './pages/About/About'
import NotFoundPage from './pages/NotFound/NotFound'

function App() {
  return (
    // Componente de Instancia geral das rotas do BrowserRouter
    <Router>
      {/* Componente que indica as rotas, funciona como um switch */}
      <Routes>
        {/* Componente que indica a rota, funciona como um case */}
        <Route path="/" element={<HomePage />} />
        <Route path="/about" element={<AboutPage />} />
        {/* Componente que indica a rota default, funciona como um default */}
        <Route path="*" element={<NotFoundPage />} />
      </Routes>
    </Router>
  )
}

export default App
```

React Router

9. Caso precisemos criar um redirecionamento com JavaScript sem usar o componente do Link, podemos usar o Hook useNavigate, do React Router. Vamos testar na nossa página de 404.

```jsx
import { useNavigate } from 'react-router-dom'
import './NotFound.css'

const NotFoundPage = () => {
  const navigate = useNavigate()

  const toHomePage = () => {
    console.error('Redirecting 404 to Home Page')
    navigate('/')
  }


  return (
    <div>
      <h1>Not Found - 404</h1>
      <button onClick={toHomePage}>Go to Home</button>
    </div>
  )
}

export default NotFoundPage

```

React Router – Rotas dinamicas

10. Também podemos usar o React Router para criar rotas dinâmicas, com parametros. Vamos criar uma nova página User e definir a mesma estrutura básica inicial das outras páginas

```jsx
import { Link } from 'react-router-dom'
import './User.css'

const UserPage = () => {
  return (
    <div>
      <h1>User Page</h1>
      <Link to="/">Go to Home</Link>
    </div>
  )
}

export default UserPage
```

React Router – Rotas dinamicas

11. Agora vamos supor que temos uma tabela em um banco de dados, onde temos usuários cadastrados por um id numérico. E queremos carregar a página do usuário de acordo com o ID numérico que vier da URL. Exemplo: /user/1 -> Vai carregar o usuário com o ID 1 e assim por diante. Vamos primeiro no App.tsx definir nossa rota como dinâmica. Os :(dois pontos) indicam que aquela parte da rota é um parâmetro de URL que pode ser usado posteriormente.

```tsx
import { Route, BrowserRouter as Router, Routes } from 'react-router-dom'
import './App.css'
import HomePage from './pages/Home/Home'
import AboutPage from './pages/About/About'
import NotFoundPage from './pages/NotFound/NotFound'
import UserPage from './pages/User/User'

function App() {
  return (
    // Componente de Instancia geral das rotas do BrowserRouter
    <Router>
      {/* Componente que indica as rotas, funciona como um switch */}
      <Routes>
        {/* Componente que indica a rota, funciona como um case */}
        <Route path="/" element={<HomePage />} />
        <Route path="/about" element={<AboutPage />} />
        {/* Componente que indica a rota com parametro */}
        <Route path='/user/:id' element={<UserPage />} />
        {/* Componente que indica a rota default, funciona como um default */}
        <Route path="*" element={<NotFoundPage />} />
      </Routes>
    </Router>
  )
}

export default App
```

React Router – Rotas dinamicas

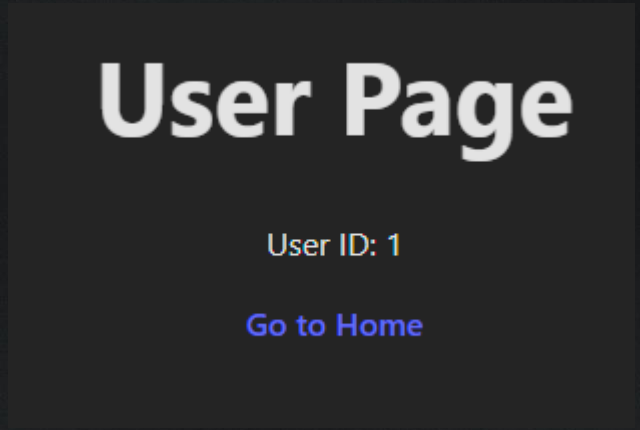
12. Agora vamos pegar o parâmetro da URL e colocar a informação na tela. Em uma aplicação real iriamos mandar o ID para uma API para pegar o Objeto do Usuário e exibir as informações na tela. Vamos usar o hook useParams

```tsx
import { Link, useParams } from 'react-router-dom'
import './User.css'

const UserPage = () => {
  // Hook que retorna os parametros da URL
  const { id } = useParams()

  return (
    <div>
      <h1>User Page</h1>
      <p>
        User ID: {id}
      </p>
      <Link to="/">Go to Home</Link>
    </div>
  )
}

export default UserPage
```



React Router – Rotas dinamicas

Agora vamos fazer um pequeno teste. Vamos supor que tenhamos apenas 5 usuários criados. Qualquer usuário com ID maior que 5 não existe, então o sistema deve redirecionar para a página de 404. Vamos combinar o useEffect com os useParams da nossa página de usuários

```
import { Link, useNavigate, useParams } from 'react-router-dom'
import './User.css'
import { useEffect } from 'react'

const UserPage = () => {
  // Hook que retorna os parametros da URL
  const { id } = useParams()
  const navigate = useNavigate()
```

```
const UserPage = () => {
  // Hook que retorna os parametros da URL
  const { id } = useParams()
  const navigate = useNavigate()


  useEffect(() => {
    if (id && parseInt(id) > 5) {
      console.error('Redirecting User Page to 404')
      navigate('/404')
    }
  }, [id, navigate])

  if (!id) {
    return null
  }

  return (
    <div>
      <h1>User Page</h1>
      <p>
        User ID: {parseInt(id)}
      </p>
      <Link to="/">Go to Home</Link>
    </div>
  )
}

export default UserPage
```

React Router

Documentação
https://reactrouter.com/en/main

React Router - Exercício

- Crie uma página de blog na HomePage, que deve ser como um Feed de postagens contendo pelo menos 5 artigos. A página deve ter footer e header e ser responsiva. Os dados dos artigos podem ser um Array estático no próprio arquivo, ou vir de um arquivo externo como um JSON

- Crie uma página para a leitura do artigo, a página deve ler o parâmetro da URL e encontrar o post correto da lista montada anteriormente. No caso de não encontrar o artigo, levar para a página de 404.

- Cada artigo deve ter pelo menos:
  - Titulo
  - ID
  - Conteúdo

- Lembre de usar os tipos do TypeScript

Dúvidas, críticas ou sugestões?


FIAP