

React Router

É uma biblioteca do react que permite implementar navegação entre paginas sem recarregar a pagina inteira. Permite alterar o conteúdo da tela de forma dinâmica.

Roteamento baseado em URL: Define rotas que renderizam diferentes componentes com base na URL atual.

Navegação Declarativa: Usa componentes como `<Link>` em vez de tags `<a>` para navegação sem recarregar a pagina.

Rotas Aninhadas: Permite definir rotas dentro de outras rotas.

Rotas Dinâmicas: Suporta parametros nas rotas, como `/users/:id`.

Proteção de Rotas (route guarding): Pode ser usado com autenticação para proteger paginas.

Ex:

```
import { BrowserRouter, Routes, Route, Link } from 'react-router-dom'
```

```
function App() {  
  return (  
    <BrowserRouter>  
      <nav>  
        <Link to="/">Home</Link>  
        <Link to="/sobre">Sobre</Link>  
      </nav>  
  
      <Routes>  
        <Route path="/" element={ <Home /> } />  
        <Route path="/sobre" element={ <Sobre /> } />  
      </Routes>  
    </BrowserRouter>  
  )  
}
```

```
)  
}  
  
function Home() {  
  return <h1>Bem-vindo a Home</h1>  
}  
  
function Sobre() {  
  return <h1>Sobre Mim</h1>  
}
```

Hora de Codar

É o pacote mais utilizado para criar rotas em uma aplicação react.

Cada rota é uma pagina, rota é a nomenclatura utilizada, ou seja, permite o app SPA ter múltiplas paginas.

É preciso instalar e configurar no projeto.

Há diversas funcionalidades no react router: redirecionamento, rotas aninhadas , 404 e etc.

Instalação

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

Configuração

É configurado no `main.jsx`

Importando as funcionalidades:

```
import { BrowserRouter, RouterProvider } from "react-router-dom"
```

Importar as paginas: Essa é a primeira coisa a se fazer

```
import Home from “./caminho/Home”
```

```
import Pagina2 from “./caminho2/Pagina2”
```

Depois é preciso criar uma constante colocando a função
createBrowserRouter:

```
const router = createBrowserRouter()
```

Dentro da função será passado um array de objetos com as rotas
(paginas) configuradas, onde cada rota vai ser um objeto:

```
const router = createBrowserRouter([  
  {path: “/”, element: <Home />},  
  {path: “pagina2”, element: <Pagina2/>}  
])
```

path: É o caminho até a pagina

element: Vai ser o componente principal da pagina, ou seja, o nome da
pagina

Para fazer tudo funcionar, ainda no main.jsx, é preciso tirar o <App /> e
colocar o RouterProvider contendo uma propriedade que receberá a
constante router:

```
<React.StrictMode>  
  <RouterProvider router={router} />  
</React.StrictMode>
```

Com isso a aplicação estará ciente das paginas. Da estrutura do react-
router.

Reaproveitamento de Estrutura

No App.jsx será colocado o Outlet, ele permite reaproveitar a pagina e fazer com que apenas uma área mude e não a pagina inteira. Onde ele estiver é onde o conteúdo vai ser alterado.

Importá-lo: `import { Outlet } from "react-router-dom"`

<Outlet />: É onde o conteúdo do router será colocado, ou seja, onde o conteúdo vai ser dinâmico, o que vai mudar quando clicar para mudar de paginas / seção.

Ex: O Titulo e o Footer se manterão estático, enquanto que apenas onde o Outlet esta vai ser dinâmico, vai mudar.

```
<h1>Titulo</h1>
```

```
<Outlet />
```

```
<p>Footer</p>
```

Para fazer isso funcionar, a configuração no main.jsx precisará mudar, já que as paginas serão renderizadas como filhas. Agora as paginas ficam no array da propriedade children e o element é o <App />. É preciso fazer isso para que a hierarquia seja respeitada:

```
const router = createBrowserRouter([
  {path: "/", element: <App />, children: [
    {path: "/", element: <Home />},
    {path: "pagina2", element: <Pagina2/>}
  ]}
])
```

Pagina de Erro

Para aparecer uma pagina especifica toda vez que acontecer algum erro, é preciso colocar a propriedade `errorElement` no `main.jsx`:

```
errorElement: <ErrorPage />
```

```
const router = createBrowserRouter([
  {path: "/", element: <App />, errorElement: <ErrorPage />, children: [
    {path: "/", element: <Home />},
    {path: "pagina2", element: <Pagina2/>}
  ]}
])
```

Navegação Entre Páginas

Para alterar as paginas é utilizado o `<Link>`, ou seja, para navegar / mudar / se conectar ou ser redirecionado para outra pagina é utilizado ele. É um componente do `react-router`, como se fosse uma tag `<a>`.

Importá-lo: `import { Link } from "react-router-dom"`

Igual a tag `<a>` é preciso definir o atributo que determina qual pagina o usuário vai acessar ao clicá-lo:

```
<Link to="/">Home</Link>
```

```
<Link to="/pagina2">Pagina2</Link>
```

Rotas Aninhadas / Rotas Dinamicas

Algumas vezes é necessario ter identificadores únicos para cada pagina. Para fazer isso é utilizado um id (`/:id`) no path da pagina. O `:id` indica que o path vai receber um valor dinâmico no lugar dele, ou seja, que algum valor vai substituir o `:id`:

```
path: "/pagina2/:id"
```

Ex:

```
const router = createBrowserRouter([
  {path: "/", element: <App />, errorElement: <ErrorPage />, children: [
    {path: "/", element: <Home />},
    {path: "/pagina/:id", element: <Pagina/>}
  ]}
])
```

Isso vai permitir ter uma pagina / resposta diferente para cada informação única.

Para utilizar essa funcionalidade será usado o <Link> e no atributo to="" ao invés do :id será colocado um valor diferente, estático ou dinâmico:

```
<Link to="/pagina/2">Pagina 2</Link>
<Link to="/pagina/${valorDinâmico}`>Pagina Dinâmica</Link>
```

Esse id também é útil para carregar / pegar dados específicos do banco.

Ex: Pegar os dados dos usuários da pagina 2

```
<Link to="/pagina/2/users">Pagina 2</Link>
```

Para capturar esses dados é utilizado o useParams, que extrai a URL em forma de objeto, onde nele tem a propriedade id, que armazena o valor dinâmico da URL, ou seja, o valor que substitui o :id.

Em um componente diferente:

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

`const { id } = useParams()` ← Pega o id da URL (seja ele qual for) via destructuring

2° forma de Navegação (redirect)

O `useNavigate` permite a navegação entre paginas na parte da logica.

Ex: Quando um formulário é enviado e é preciso redirecionar o usuário para a home:

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

```
const navigate = useNavigate()
```

```
function handleContact() {  
  console.log('Mensagem Enviada')  
  return navigate("/")  
}
```

```
<button onClick={handleContact}>Enviar Mensagem</button>
```

No argumento do `navigate` é passado o path da pagina para qual o usuário será mandado.

3° forma de Navegação (redirect)

Redirecionamento por componente é mais utilizada para quando uma pagina deixa de existir no site e é preciso redirecionar o usuário para a pagina certa / que esta ativa:

```
import { Navigate } from "react-router-dom"
```

Na configuração da pagina desativada, a propriedade `element` terá o `Navigate` com o `to=""` para a pagina certa:

```
{path: "paginaAntiga", element: <Navigate to="/paginaNova" />}
```

Estrutura Principal

Para projetos mais simples:

main.jsx

```
import { BrowserRouter } from 'react-router-dom'
```

```
import App from './App'
```

```
ReactDOM.createRoot(document.getElementById('root')).render(  
  <React.StrictMode>  
    <BrowserRouter>  
      <App />  
    </BrowserRouter>  
  </React.StrictMode>  
)
```

App.jsx

```
import { Routes, Route, Link } from 'react-router-dom'
```

```
import Home from './pages/Home'
```

```
import About from './pages/About'
```

```
function App() {
```

```
  return (  
    <div>
```

```
      <nav>
```

```
        <Link to="/">Home</Link> <Link to="/about">About</Link>
```

```
      </nav>
```

```
      <Routes>
```

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

```
        <Route path="/about" element={ <About /> } />
```



```
    </Routes>
  </div>
}) export default App
```

Para projetos mais complexos:

main.jsx

```
import { RouterProvider, createBrowserRouter } from 'react-router-dom'
import Layout from './components/Layout'
import Home from './pages/Home'
import About from './pages/About'
import ErrorPage from './pages/ErrorPage' // opcional, mas recomendado
```

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <Layout />,
    errorElement: <ErrorPage />,
    children: [
      { index: true, element: <Home /> },
      { path: "about", element: <About /> }
    ]
  }
])
```

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

Layout.jsx

```
import { Outlet, Link } from 'react-router-dom'
```

```
export default function Layout() {
```

```
  return (
```

```
    <div>
```

```
      <nav>
```

```
        <Link to="/">Home</Link> |
```

```
        <Link to="/about">About</Link>
```

```
      </nav>
```

```
      <Outlet /> // Onde as paginas irão aparecer
```

```
    </div>
```

```
  )
```

```
}
```