

# Navegação com React Router

Labenu\_



# O que vamos ver hoje?

- Revisão de Rotas
- React Router
  - useNavigate
  - useParams
- Boas práticas
- Versão legado



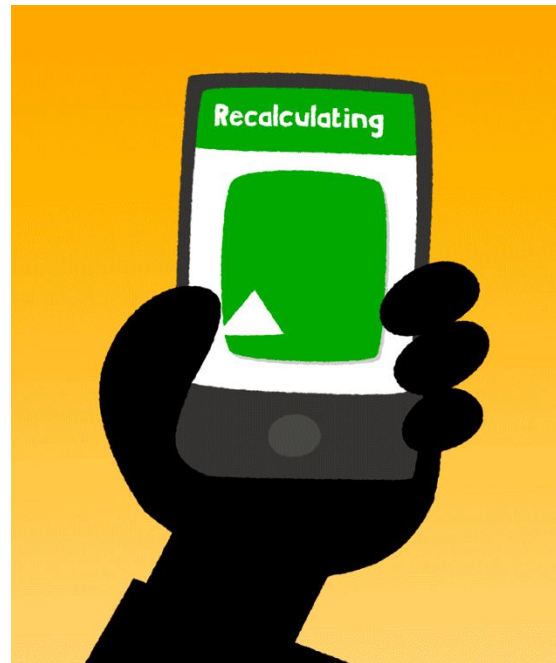
# Revisão - Rotas

Labenu\_



# Rotas

- Rota é um termo utilizado para se referir a um conjunto de **endereço** e **valor**. Por exemplo:
  - **"Vá até a Rua Nascimento Silva, 107, Ipanema"**  
**"Você encontrará a casa do Tom Jobim"**
- Trazendo para o nosso contexto, pensando na URL:
  - **"Acesse site.com.br/login"**  
**"Você encontrará a página de login"**



# Rotas

- No começo do curso com HTML + JS, nós mudávamos de página usando a tag `<a>` e passando um atributo `href` para ela
- Esse atributo `href` correspondia a um arquivo HTML
- Era possível trocar entre as páginas pela rota na **barra de endereços** do navegador



# Rotas

- No React, a solução que estávamos usando até agora para trocar de página era a **renderização condicional**
- Decidimos qual página renderizar com base em algum parâmetro do **estado**
- Mas gerenciar componentes com estado desta forma é chato e trabalhoso 😞 => **Solução: React Router** ✨



# React Router

Labenu\_



# React Router

- Biblioteca que simula comportamento de rotas
- Ela fará o gerenciamento das rotas para a gente, mapeando cada **rota** para um **componente**
- Exemplo: **"/login"**  $\Rightarrow$  **LoginPage**  $\longrightarrow$  valor

  
**endereço**





# React Router - Vantagens

- Gestão de múltiplas páginas simplificada, sem gerenciamento direto de estado
- Possibilidade de acessar **páginas específicas** diretamente através da URL
- Acesso ao **histórico do navegador**



# React Router - Instalação

- Como é uma biblioteca, precisamos instalá-la. Fazemos isso executando o seguinte comando na raiz do projeto:

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

- [Documentação da Lib](#)



# Componentes do Router

Labenu\_




# Componentes - Visão Geral

- A biblioteca nos disponibiliza componentes para facilitar o uso da lógica de roteamento
- 3 componentes principais:
  - **BrowserRouter**
  - **Routes**
  - **Route**



# Componentes - **BrowserRouter**

- Componente responsável por "abrigar" e gerenciar todas as rotas
- Funciona em conjunto com outros componentes react-router

```
  
<BrowserRouter>  
  
  { /*Aqui dentro vai tudo que for usar da biblioteca */ }  
  
</BrowserRouter>
```



# Componentes - Routes

- Componente responsável por permitir que **apenas uma rota** seja renderizada por vez
- Todas as rotas ficam dentro desse componente.
- Comportamento semelhante ao switch/case



# Componentes - Route

- Deve estar **dentro** do **Routes** dentro **BrowserRouter**
- Componente responsável por **definir uma rota**
- Recebe uma prop ***path***, que tem a URL à qual aquela rota corresponde
- Recebe o componente através da props ***element***
- O componente é renderizado quando o ***path*** coincidir com a **URL**



# Componentes - Route



```
<BrowserRouter>
  <Routes>
    <Route index element = {<Home />} />
    <Route path = "teams" element = {<Teams />} />
  </Routes>
</BrowserRouter>
```



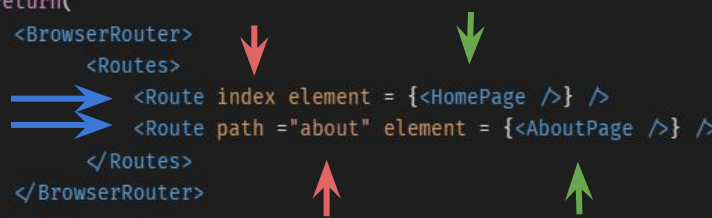


# Juntando as Peças

- Cada **<Route />** recebe como props **element**: a página que deve ser aparecer
- Também recebe **path**: a URL da página
- **<BrowserRouter>** e **<Routes>** envolvem todas as rotas

```
import { BrowserRouter, Routes, Route } from 'react-router-dom'
import HomePage from './HomePage/HomePage'
import AboutPage from './AboutPage/AboutPage'

const App = () => {
  return(
    <BrowserRouter>
      <Routes>
        <Route index element = {<HomePage />} />
        <Route path = "about" element = {<AboutPage />} />
      </Routes>
    </BrowserRouter>
  )
}
```





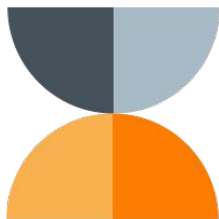
# Exercício 1

- Criar um site usando React Router que tenha 3 Páginas:
  - Home
  - Sobre este site
  - Página de Erro
- Fazer o roteamento correto dessas 3 páginas para que fiquem acessíveis por URLs
- Criar um componente Router.js e organizar arquivos



# Pausa para relaxar 🤪

10 min



- `<BrowserRouter>` envolve tudo relativo à roteamento
- `<Routes>` troca entre as opções de tela e garante que apenas 1 será renderizada
- `<Route>` relaciona uma rota (path) ao componente (tela) que será mostrado



# Hooks do Router

Labenu\_



# Hooks do Router

- Como vimos na última aula, é possível criar **hooks customizados** para atender às nossas necessidades
- Bibliotecas também podem ter hooks customizados!
- Vamos ver hoje 2 custom hooks do React Router:
  - **useNavigate**
  - **useParams**



# Navigate e useNavigate

Labenu\_



# useNavigate

- A função **navigate** no react-router nos permite **trocar a página atual**
- Infelizmente não temos acesso fácil a ela dentro da nossa página **HomePage**
- Para facilitar, foi criado o hook **useNavigate** que nos permite acessar o histórico dentro de qualquer **componente funcional**



# useNavigate - Exemplo

- O exemplo abaixo está sendo usado no nosso componente **HomePage**

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

const HomePage = () => {
  const navigate = useNavigate()
  return <div> Home</div>
}
```





# Navigate

A função **navigate** pode receber alguns parâmetros:

- `navigate("/rota");`  $\Rightarrow$  vai para página indicada
- `navigate("/rota", {replace:true})`  $\Rightarrow$  troca de página impedindo que o usuário volte
- `navigate(-1)`  $\Rightarrow$  retorna para a última página



# history - Métodos - Exemplo

- O exemplo abaixo usa a função `navigate("/rota")`

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

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

  const goToAboutPage = () => {
    navigate("/sobre")
  }

  return (
    <div>
      <p>Home</p>
      <button onClick={goToAboutPage}>Sobre</button>
    </div>
  )
}
```



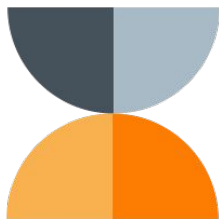


## Exercício 2

Nesse exercício faremos a navegação entre as telas!

- 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 volta para a última tela





# Pausa para relaxar 🧘

5 min

- A lib React Router possui alguns hooks que podemos usar:
  - **useNavigate()**
  - **useParams()**
- Com o useNavigate() temos acesso ao histórico de páginas e assim podemos trocar entre as nossas telas



# Path Params e useParams()

Labenu\_



# Path Params

- Antes de vermos como o hook useParams() funciona, precisamos entender qual **problema** ele resolve
- Vocês já devem ter observado que em vários sites, a URL é utilizada para **guardar informações sobre a página**
- `https://facebook.com/post/ID_DO_POST`




# Path Params

- A grande vantagem de termos dados "salvos" na URL é que se o usuário **atualiza** a página, a **URL vai se manter a mesma** então podemos usar essas infos
- **Exemplo:** se tivermos uma rota **"/postDetails"**, perderemos o id ao atualizar
- Se tivermos uma rota: **"/postDetails/ID\_DO\_POST"**, o id permanecerá na URL mesmo que a página atualize



# Path Params

- Vocês já usaram Path Params antes sem saber! 
- Lembrem das APIs no Postman, muitas tinham uma parte da URL que vocês tinham que substituir por **":turma-nome-aluno"**? Os **:** indicam que é um **pathParam**
- Ao substituir o nome na URL, você estão passando uma **variável** para o backend





# Path Params

- Utilizamos **path params** nas nossas rotas!
- Lembrando, é no componente `<Route>` que definimos uma rota



```
<Route path="sobre/:linguagem" element={<AboutPage />} />
```



# Path Params

- Com a rota alterada para receber path params, o próximo passo é alterar a lógica que fizemos mais cedo



```
const navigate = useNavigate()

const goToAboutPage = () => {
  navigate("/sobre/portugues")
}
```



# Path Params

- Agora, com as informações sendo passadas, podemos fazer o uso do hook **useParams()**

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

export const AboutPage = () => {
  const pathParams = useParams();

  return (
    <div>
      <p>Sobre</p>
      <p>Linguagem: {pathParams.linguagem}</p>
    </div>
  );
};
```





## Exercício 3

- Na tela de Home, crie 2 botões:
  - O primeiro vai para a tela Sobre em **inglês**
  - O segundo vai para a tela Sobre em **português**
- Na tela Sobre, faça uma renderização condicional do texto baseando-se no parâmetro recebido de linguagem



# Observação sobre o surge ⚠

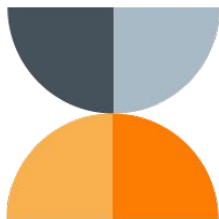
- Quando usamos o Router, para subir o site usando o surge, precisamos dar os 4 comandos abaixo:

```
npm run build  
cd build  
cp index.html 200.html  
surge
```



# Pausa para relaxar 🧘

5 min



- **Path Params** são informações que passamos pela URL
  - No Router: "posts/:postId"
  - Na função de mudar tela: "posts/123"
- Com o hook **useParams()** podemos pegar esse valor do parâmetro na nossa tela



# Boas Práticas

Labenu\_



# Como organizar rotas

- Cada rota deve mapear para uma **página**
- Como não temos conceito de página em React, trataremos ele como um componente grandão que ocupa a **tela inteira**
- Esses componentes são nomeados de forma a deixar claro que são páginas: **LoginPage, FeedPage**





# Como organizar rotas

- Podemos ter **um componente de Router próprio** que contém a lógica de roteamento (componentes do React Router)
- Dessa forma, temos uma noção rápida de onde os dados vêm e como a nossa **árvore de componentes** está organizada



# Como organizar rotas: coordinator

```
export const goToHomePage = (navigate) => {  
  navigate("/");  
};  
  
export const goToAboutPage = (navigate, language) => {  
  navigate(`/about/${language}`);  
};  
  
export const goBack = (navigate) => {  
  navigate(-1);  
};
```

Crie um **um componente que coordene as mudanças de páginas**, centralizando as funções



# Versão legado

Labenu\_



# Versão legado



```
return(  
  <BrowserRouter>  
  
    <Switch>  
  
      <Route path={"/login"}>  
        <LoginPage/>  
      </Route>  
  
    </Switch>  
  
  </BrowserRouter>  
)
```

```
1 import { Switch, Route, BrowserRouter } from 'react-router-dom'  
2 import HomePage from './HomePage/HomePage'  
3 import AboutPage from './AboutPage/AboutPage'  
4  
5 const App = () => {  
6   return(  
7     <BrowserRouter>  
8       <Switch>  
9  
10         <Route exact path="/sobre">  
11           <AboutPage />  
12         </Route>  
13  
14         <Route exact path="/">  
15           <HomePage />  
16         </Route>  
17  
18       </Switch>  
19     </BrowserRouter>  
20   )  
21 }
```

# Versão legado

`npm install react-router-dom@5.2.0`

Feature	Versão 5.2.0	Versão atual
Escolha das rotas	<Switch>	<Routes>
Passagem do componente	Componentes filhos	Props element
Hooks	useHistory	useNavigate
Histórico	Objeto	Função
Navegação	Através de métodos	Parâmetros na função

- Principais mudanças

# Resumo

Labenu\_



# Resumo

- Para reproduzir o comportamento natural de navegação no browser, usamos a lib react-router-dom
- Ela gerencia todas as páginas da nossa aplicação para nós
- Possui 3 componentes principais:
  - `<BrowserRouter>` envolve tudo relativo à roteamento
  - `<Routes>` troca entre as opções de tela
  - `<Route>` relaciona uma rota ao componente



# Resumo

- A lib também possui alguns custom hooks úteis:
  - `useNavigate()` nos dá acesso ao histórico que utilizamos para trocar entre telas
  - `useParams()` que nos permite pegar dados através da URL do nosso site





# Resumo

- Para criar o link do surge, usamos:

```
npm run build  
cd build  
cp index.html 200.html  
surge
```



# Dúvidas? 🧐

Labenu\_





Obrigado(a)!