**Aprendendo React – React Router Dom**

**React-router-dom:** O React-router-dom é uma extensão do react para manipulação de rotas. Ela nos permite altera de uma rota para

outra sem a fazer o reload da página.

**Principais coisas do React-router-dom:**

**BrowserRouter:** Ele é simplesmente um dos principais responsáveis na criação de rotas do react, ele será o “componente” que vai envolver o **Router** e que por sua vez irá envolver o **Route.**

**Router:** Ele vai ser envolvido pelo **BrowserRouter** e dentro dele será definidos as rotas da aplicação, através do **Route**.

**Route:** Ele será de fato a rota da aplicação, ele receberá dois argumentos, sendo eles: **Path** e **Element**. O **Path** será a **URL** da rota e o **Element** será o página que será renderizada quando aquela **URL** for acessada.

**Como fica a estrutura do BrowserRouter então?**

**<BrowserRouter>**

**<Router>**

**<Route path=’/sobre’ element={<Sobre />} >**

**</ Router>**

**</ BrowserRouter>**

**Observações:**

**1.** Para se utilizar o “**Element**” precisamos criar uma pasta dentro de **SRC** com o nome de **Pages,** Dentro de **Pages** devemos criar as página que serão renderizadas em cada **route.**

**Exemplo:** Dentro de **Pages** temos que criar um arquivo chamado de **Sobre.js** para podemos renderizar a página

**<Sobre />** quando a **URL** */sobre* for acessada.

**2.** Devemos importar a página que será utilizada dentro da **Route**, ela será importada no mesmo arquivo onde será usada. Geralmente as rotas são definidas no arquivo **App.js**

**3.** As páginas são praticamente **componentes**, para ser mais especifico, as páginas são **componentes** que serão acionados quando for chamado uma **URL** especifica.

**Criando rotas dinâmicas:**

**Rota dinâmica:** Uma rota dinâmica é praticamente uma rota comum, porém, ela pode receber **parâmetro** via **URL** e esses parâmetros podem ser utilizados para exibir coisas especificas, como por exemplo, renderizar uma coisa na página se o parâmetro for 1 ou renderizar outra coisa se o parâmetro for 2.

**Criando uma rota dinâmica:**

**1.** Devemos ir no local onde definimos a rota e criamos uma nova rota, a criação é praticamente igual a uma rota normal, exceto por um **:NameOfParams** que adicionamos no final da rota:

**Exemplo:** Aqui temos uma rota comum que renderiza uma

página chamada numero quando a URL é acessada, agora

vamos transformá-la em uma rota dinâmica:

**<Route path=’/numero’ element={<Numero />} />**

**Rota numero dinâmica:** Para essa rota ficar dinâmica vou adicionar no final do path dela um **/:num** e ela se tornará dinâmica:

**<Route path=’/numero/:num’ element={<Numero />} />**

**Certo, mas o que ganhamos com isso?**

Agora podemos acessar esse parâmetro que foi mandado via URL e exibir algo de acordo com o número:

Por exemplo, podemos acessar a seguinte **URL** e enviar um número dessa forma: **http//:localhost:3000/numero/21**

Agora na página número, podemos ter acesso a esse número que foi enviado via URL e com isso fazer uma renderização dinâmica:

**Como acessar esse número passado pela URL?**

Para acessá-lo, precisaremos importar do react-router-dom um hook chamado **useParams** e com ele vamos poder pegar o parâmetro que foi enviado via **URL**:

**import {useParams} from ‘react-router-dom’**

Dessa forma, podemos recuperar o valor recebido pela **URL** e usá-lo da forma que quisermos:

**const {num} = useParams();**

**Observação:** Esse processo de recuperação de parâmetro da **URL** deve ser feito na página em que o parâmetro foi enviado, no caso desse exemplo, página **Numero.js** que deve ser criada dentro da pasta **Pages**.

**Aprendendo Nested Routes:** Nested Routes é basicamente uma página após a **rota dinâmica**, como assim?

Aproveitando o **exemplo anterior**, temos uma rota **/numero** onde enviamos um número forma **dinâmica**, mas e se quisermos ter outra página dentro de **/numero** para exibir outra informação? E para isso que existe o **Nested Routes**.

**Exemplo:** Vamos supor que dentro da pagina Numero eu quero uma página chamada info, que exibe uma determina informação sobre o número, por exemplo se o número é PAR ou IMPAR:

Aqui temos a rota de número padrão: Agora vamos criar uma rota Nested Routes:

**<Route path=’/numero/:num’ element={<Numero />} />**

**Nested Routes:** Basta duplicar a rota que é dinâmica, no caso a rota **/numero** e no final dela, onde fica o **:id** adicionamos o **/info**, ficando assim a rota **nested**:

**<Route path=’/numero/:num/info’ element={<Info />} />**

Agora temos uma rota dentro da página /numero que podemos adicionar mais informações sobre o numero. A URL ficará dessa forma: **http://localhost:3000/numero/21/info**

**Observação:** Não podemos esquecer de criar a página chamada **Info.js**, que será responsável por renderizar a informações adicionais do número e importá-la no local onde definimos a rota.

**Criando rota de search:** Uma rota de search é uma rota de pesquisa, geralmente são usadas atrelada a um formulário, onde ao digitar algo no formulário e pesquisar, a rota vai exibir alguns itens ou informações de acordo com o que foi pesquisado:

**Como criar uma rota de search?**

É praticamente uma rota dinâmica, porém com algumas particularidades:

Criamos uma e definimos o elemento que será renderizado quando essa rota for acessada:

**<Route path=’/search’ element={<Search />} />**

**Como podemos atrelar a rota criada a um formulário?**

**1.** Criamos um formulário e atrelamos ao clique do botão de pesquisar uma função chamada **handleSubmit**.

**2.** Devemos criar um **useState** que vai armazenar o que foi digitado pelo usuário e chama ló de **query**.

3. Dentro da função **handleSubmit**, devemos prevenir o reload ao clicar no botão de pesquisar.

4. Para **navegar** até a página onde vai exibir os resultados da busca vamos precisar usar o **hook** ‘**navigate**’, ele será importado do **react-router-dom**.

**O código ficará assim:**

**const [query, setQuery] = useState();**

**const handleSubmit = (event) => {**

**event.preventDefault()**

**navigate(“/search?q=” + query);**

**}**

**<form>**

**<input type=”text”**

**onChange={({target}) => setQuery(target.vlaue)}**

**/>**

**<button type=”submit”**

**onClick={handleSubmit}**

**>Pesquisar</button>**

**</form>**

**Como fica a URL ao usuário pesquisar algo?**

Com isso, já obtemos na **URL** o termo digitado pelo usuário, por exemplo:

O usuário digita **camisa** e clica em pesquisar, a **URL**, a renderização da página **Search** irá acontecer, pelo fato da **URL** está sendo acessada e ficará dessa forma:

**http://localhost:3000/search?q=camisa**

**Como podemos acessar esse termo digitado pelo usuário na página Search?**

Agora podemos utilizar um **hook** chamado **useSearchParams** do react-router-dom para pegar o termo digitado pelo usuário e fazer uma **renderização dinâmica** de acordo com o termo digitado:

Com isso aqui já objetos o termo digitado pelo usuário, no caso do exemplo “**camisa**”:

const [searchParams] = useSearchParams();

Agora podemos utilizar a variável **searchParams** para renderizar algo da forma que quisermos.

**Página não encontrada**: É utilizada para informar ao usuário que a página que ele tentou acessar não existe.

**Como criar uma página de não encontrado?**

Precisaremos criar uma rota, que será uma rota que vai renderizar para o usuário uma página de error 404 “Não encontrado” quando o usuário digita uma URL inválida:

Primeiro vamos precisar ir onde definimos as rotas da aplicação e criar uma rota nova:

Usaremos o **path** \* para informar que queremos renderizar esse elemento para todas rotas que não foram definidas anteriormente.

**<Route path=”\*” element={<Error404 />} />**

Agora vamos criar uma página em **Pages**, com o nome **Error404** e **importa** lá o arquivo onde **definimos** a rota.

**E assim, nossa rota de error foi criada com sucesso!**