SUMÁRIO

1. O PROBLEMA: UMA SPA DE MÚLTIPLAS ROTAS

Como já sabemos, ou deveríamos saber, o React é um framework JS que serve para criar, a princípio, aplicações SPA, isto é, que possuem uma única página, sem reloads, totalmente assíncrono.

Certo, mas e se quisermos fazer uma aplicação com mais de uma página? Como fazer uma aplicação SPA com rotas? Imagine um site com um menu de navegação Home, Sobre e Contato. Como fazer para navegar entre as páginas, mantendo a aplicação como uma SPA? A solução para isso é a biblioteca React Router.

2. A BIBLIOTECA REACT ROUTER

O React Router é uma biblioteca padrão para roteamento no React. Ele permite a navegação entre visualizações de vários componentes em um aplicativo React, permite alterar a URL do navegador e mantém a IU sincronizada com a URL.

2.1. COMO INSTALAR

Para instalar essa biblioteca basta digitar na linha de comando, e no local do projeto, "**npm install react-router-dom –save**".

Veja mais na documentação.

3. GUIA DE USO

3.1. IMPORT DOS COMPONENTES

Antes da explicação de cada componente utilizado, eles devem ser importados do node modules, onde foram instalados. A importação pode ser no App.js, por enquanto.

import {BrowserRouter as Router, Route, Link, Switch} **from** 'react-router-dom'

3.2. EXPLICAÇÃO DOS COMPONENTES

As rotas de roteamento com React podem ser divididas em duas partes: links e rotas. Os links, **<Link ></>**, que são invocados como componentes, são responsáveis por alterar o caminho de URL do aplicativo como já faz a marca <a> da linguagem HTML. Eles são inclusive são interpretados como marcas de hiperlink <a>.

```
<ul>
        <li><Link to = "/">Home </Link></li>
        <li><Link to = "/sobre">Sobre </Link></li>
        <li><Link to = "/contato">Contato </Link></li>
</ul>
```

Em seguida, o aplicativo itera sobre um **conjunto de rotas pré-determinadas** dentro de um componente de nome usualmente igual a **<Switch>**, cada uma com um caminho exclusivo **<Route>** e um componente de página a ser renderizado.

O aplicativo sabe **qual página renderizar**, combinando o novo caminho com as rotas pré-determinadas. Por exemplo, clicando no <Link> que altera a URL para "/contato", o componente de página <Contato /> é renderizado.

```
<Router> //Elemento pai que implementa a sincronia da UI com a URL
        <Switch> //Verifica e renderiza a primeira rota correta (URL match)
            <Route exact path = "/"> //Rota/URL de Home
              <Home /> //Componente página Home
            </Route>
            <Route path = "/blog"> //Rota/URL de Blog
              <Blog /> //Componente página Blog
            </Route>
        </Switch>
</Router>
```

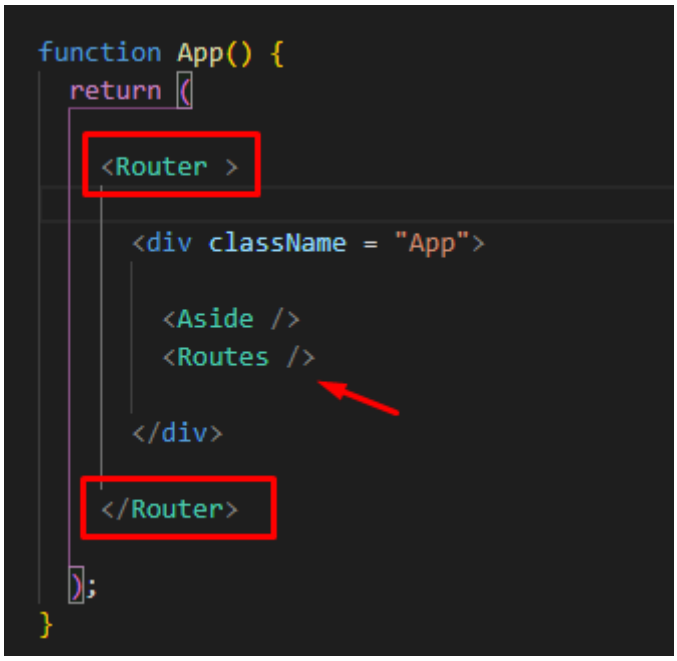
Note que na primeira rota, que levaria para a página principal, existe um **termo adicional "exact"** antes de "path", e assim deve ser porque o switch procura, por padrão, apenas pela primeira barra, isto é, se ela existir, na URL, ele verifica a primeira rota com a mesma barra, e carrega o componente dela. Ou seja, sem exact, em todos os casos apenas a primeira rota será carregada.

3.3. ORGANIZANDO O ROTEAMENTO

Como você provavelmente pode imaginar, gerenciar links e rotas para um aplicativo da web inteiro pode ficar muito confuso muito rapidamente.

Então é interessante que, ao invés de em um mesmo arquivo, a arquitetura das rotas seja quebrada sem partes acopláveis.

Pois bem: **antes de tudo**, o componente <Router>, pai de todos, deve sempre incluir todos os outros do react-router, e por isso, é interessante que englobe até mesmo a , ou o conteúdo do componente que tem seu conteúdo renderizado no container “root”. Por padrão existe o componente “App.js” para isso, mas a organização pode ser alterada, obviamente.



Se o <Router> deve incluir os outros componentes do react-router, e se vamos aplicar esses outros componentes em outros arquivos, e inclui-los, neste caso, na do “App.js”, faz sentido que <Router> seja pai da , porque por consequência também estará “acima” dos outros.

Os componentes de <Link> podem ser postos em qualquer outra estrutura. No caso da imagem, eles existem na programação do <Aside />, em cada item de um menu lateral, mas poderia também ser em um hipotético <Header />, ou em dois menus diferentes, enfim, não importa, desde que suas rotas sejam configuradas no <Switch>.

Quanto a configuração das rotas com <Switch>, pode existir uma pasta, dentro de src, chamada “routes”, e dentro dela um componente Routes.js. Nesse arquivo terá o <Switch> e as rotas <Route>. O componente será invocado na do “App.js”, e, claro, não irá imprimir conteúdo, pois será apenas a configuração dos links de rota.