



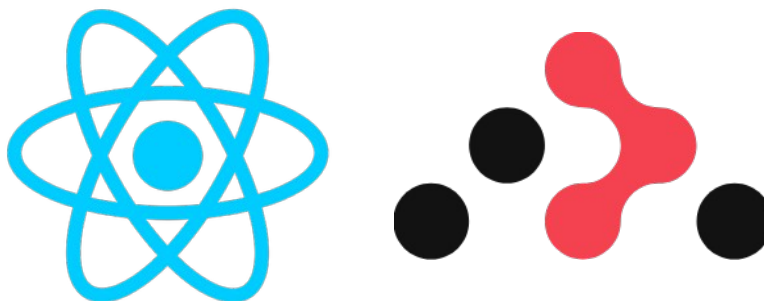
INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Pelotas



DESENVOLVIMENTO FRONT-END II

Configuração de Rotas e Controle de Navegação no ReactJS
com React Router



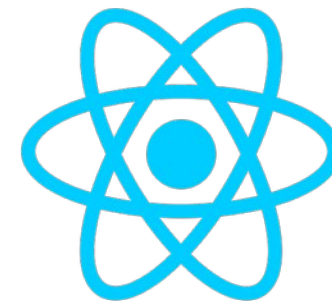
Prof. Gill Velleda Gonzales | gillgonzales@ifsul.edu.br | @g1ll
CSTSI - Novembro - 2024

Desenvolvimento Front-End II

Tópicos

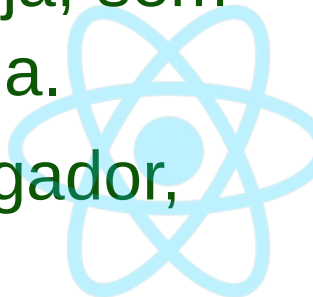


- **Rotas no Lado do Cliente:**
- **Introdução ao React Router**
 - Instalação
 - Configuração de rotas:
 - **Objetos e Elementos JSX**
- **Criação de Rotas Aninhadas**
 - Children, Outlet e Route
- **Cliente Side Routing:**
 - Link
- **Hooks**
 - useParams



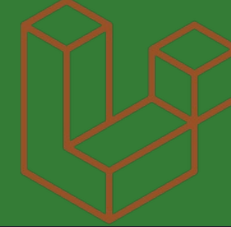


- **Rotas** no contexto de **Web**, significa um caminho na **URL** para uma **página**.
- Geralmente **caminho** (*path*) **“/”** leva à página inicial, enquanto **“/about”**, ou sobre, redireciona para a página com o conteúdo sobre o próprio site.
- Existe dois grandes tipos de configurações de Rotas:
 - ▶ **Server-side routing**: rotas gerenciadas do lado do servidor, como o framework php *laravel* por exemplo.
 - Neste caso é necessário o cliente requisitar a rota pretendida ao servidor que responderá com a página adequada para a rota solicitada.
 - ▶ **Client-side routing**: rotas gerenciadas **localmente no cliente**, ou seja, sem enviar ao **servidor** (*backend*) outra requisição solicitando uma página.
 - A navegação é gerenciada pelo próprio **javascript** no navegador, atualizando o conteúdo de acordo com a rota solicitada.



Rotas no Lado do Servidor

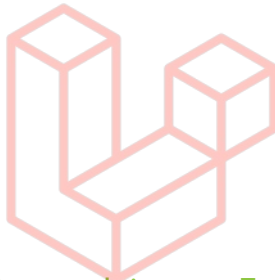
INTRODUÇÃO AO CONCEITO



- 1) O cliente (ou seja, navegador) faz uma solicitação ao servidor para uma página específica.
- 2) O servidor usa os identificadores no nome do **caminho da URL (/path)** para recuperar os dados relevantes de seu banco de dados.
- 3) O servidor preenche um modelo, documento HTML ou template, com esses dados.
- 4) O servidor retorna o modelo junto com outros ativos (**assets**) como **CSS/imagens** para o cliente.
- 5) O cliente renderiza esses ativos.
- 6) Para alterações de rota subsequentes, o cliente envia novamente uma solicitação ao servidor (**REQUEST**), repetindo o processo.



Veja este exemplo  **php**

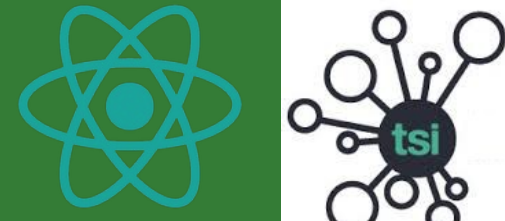


routes >  web.php >  Closure

```
1  <?php
2  use App\Http\Controllers\ProdutoController;
3  use App\Http\Controllers\ProfileController;
4  use App\Models\Produto;
5  use App\Models\User;
6  use Illuminate\Support\Facades\Storage;
7  use Illuminate\Support\Facades\Route;
8
9  Route::get('/', function () {
10     return view('playground', [
11         'title' => 'Server Side Routing'
12     ]);
13 });
14
15 Route::get('/ola', function () {
16     return view('ola', [
17         'title' => "Mundo"
18     ]);
19 });
20
21 Route::get('/ola/{name}', function ($name) {
22     return view('ola', [
23         'title' => $name ? $name : "Mundo"
24     ]);
25 });
```


Rotas no Lado do Cliente

INTRODUÇÃO AO CONCEITO



- 1) O cliente faz uma solicitação inicial ao servidor.
- 2) O servidor responde com um documento HTML primário (a página única do **SPA**) e ativos associados (**assets**).
- 3) O cliente interpreta o **JavaScript**, e a lógica do aplicativo determina qual conteúdo exibir com base no **caminho da URL (/path)**.
- 4) Para alterações de rota subsequentes, o **JavaScript** atualiza o **histórico do navegador** e o conteúdo exibido sem uma recarga completa da página (**SEM REQUEST**).

   [Veja este exemplo](#)   

```
1 import { StrictMode } from 'react';
2 import { createRoot } from 'react-dom/client';
3 import { createBrowserRouter,
4   RouterProvider } from 'react-router-dom';
5 import Ola from './components/Ola/Ola.jsx';
6 import './index.css';
7
8 const router = createBrowserRouter([
9   {
10     path: '/',
11     element: <div>Rota Gerenciada no lado do Cliente</div>,
12   },
13   {
14     path: '/ola',
15     element: <Ola />,
16   },
17   {
18     path: '/ola/:name',
19     element: <Ola />,
20   },
21 ]);
22
23 createRoot(document.getElementById('root')).render(
24   <StrictMode>
25     <RouterProvider router={router} />
26   </StrictMode>
27 );
```

Instalação da React Router

A BIBLIOTECA REACT ROUTER



- **Instalação da biblioteca:**

- ▶ Para instalar a biblioteca **React Router**, usamos os comandos do gerenciador de pacotes para adicionar a nova dependência ao projeto: (**react-router-dom**).

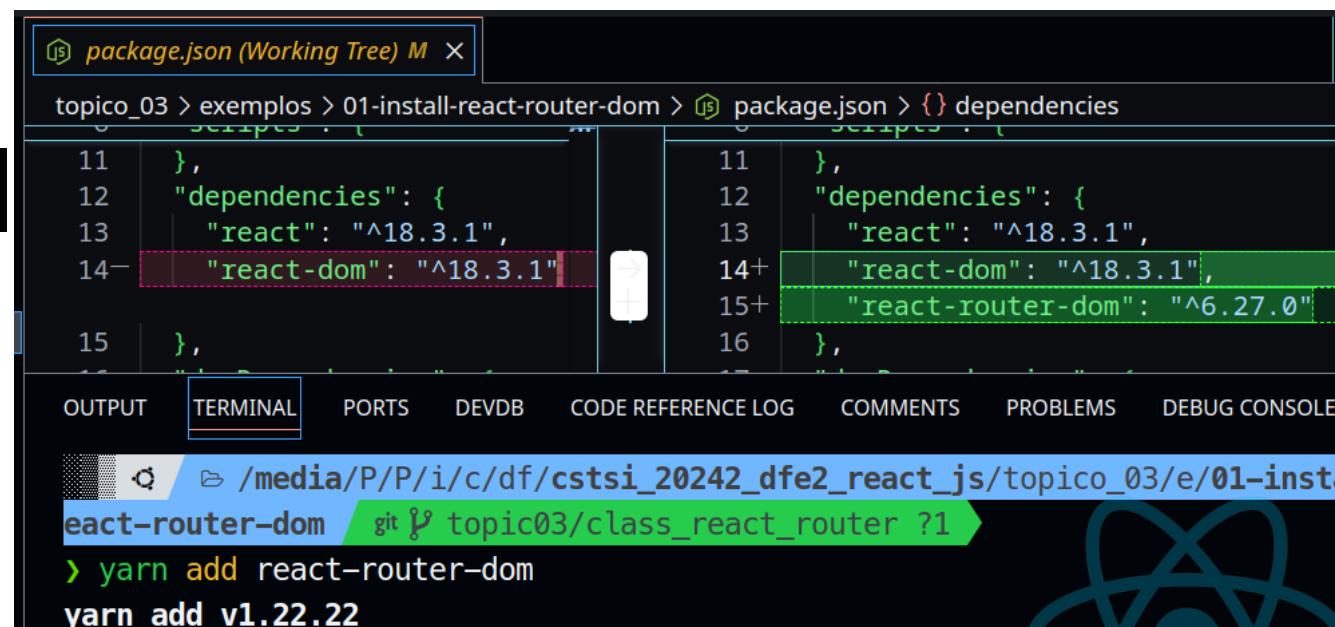
- ▶ Com **NPM**:

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

- ▶ Ou com **YARN**

```
$>yarn add react-router-dom
```

 [Veja a documentação](#)



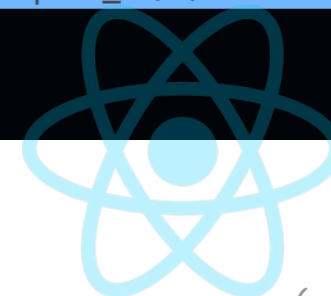
The screenshot shows a VS Code editor with two files open: `package.json (Working Tree) M` and `package.json`. The left pane shows the `dependencies` section of the `package.json` file, with the following content:

```
11 },
12 "dependencies": {
13   "react": "^18.3.1",
14   "react-dom": "^18.3.1"
15 },
```

The right pane shows the `package.json` file, with the following content:

```
11 },
12 "dependencies": {
13   "react": "^18.3.1",
14+  "react-dom": "^18.3.1",
15+  "react-router-dom": "^6.27.0"
16 },
```

The terminal output at the bottom shows the command `yarn add react-router-dom` being executed, resulting in `yarn add v1.22.22`.



Configuração das Rotas

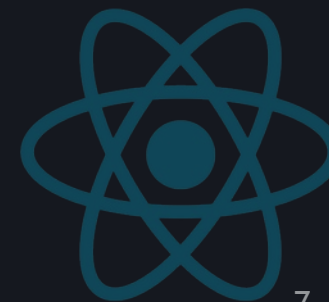
A BIBLIOTECA REACT ROUTER



- Criação de Rotas:

- ▶ A criação da Rota é feita através da função **createBrowserRouter** e do componente **RouteProvider** ambos importados da biblioteca **react-router-dom**.

```
1  import { StrictMode } from 'react';
2  import { createRoot } from 'react-dom/client';
3  import { createBrowserRouter, RouterProvider } from 'react-router-dom';
4  import Ola from './components/Ola/Ola.jsx';
5  import './index.css';
6
7  > const router = createBrowserRouter([ ...
20  ]);
21
22  ∨ createRoot(document.getElementById('root')).render(
23  ∨   <StrictMode>
24    |   <RouterProvider router={router} />
25    | </StrictMode>
26  );
```



Configuração das Rotas

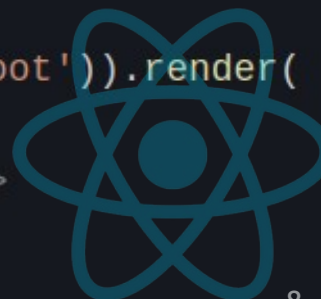
A BIBLIOTECA REACT ROUTER



- **Criação de Rotas:**

- ▶ A função ***createBrowserRouter*** é responsável por retornar um objeto que representa a configuração de rotas. Ela recebe como parâmetro esta definição.
- ▶ O objeto ***router*** é então repassado como propriedade para o componente ***RouterProvider***.

```
7  const router = createBrowserRouter([
8    {
9      path: '/',
10     element: <div>Rota Gerenciada no
11       | lado do Cliente com React Router!</div>,
12   },
13   {
14     path: 'ola',
15     element: <Ola />,
16   },
17   {
18     path: 'ola/:name',
19     element: <Ola />,
20   },
21 ]);
22
23 createRoot(document.getElementById('root')).render(
24   <StrictMode>
25     <RouterProvider router={router} />
26   </StrictMode>
27 );
```



Configuração das Rotas

A BIBLIOTECA REACT ROUTER

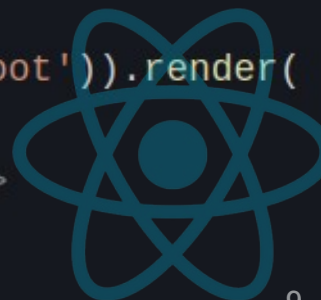


- **O Objeto de Rotas:**

- ▶ **path:** significa o **caminho informado na URL**, após o domínio do site.
- ▶ **element:** tratado de qual elemento deverá ser retornado de acordo com a **URL** informada.
- ▶ **Parâmetros:** colocamos **(:)** antes do nome do parâmetro. Veremos mais adiante como ler estes parâmetros.

 ***Veja este exemplo***

```
7  ✓ const router = createBrowserRouter([
8  ✓    {
9      path: '/',
10     ✓ element: <div>Rota Gerenciada no
11       | lado do Cliente com React Router!</div>,
12     },
13     ✓ {
14       path: 'ola',
15       element: <Ola />,
16     },
17     ✓ {
18       path: 'ola/:name',
19       element: <Ola />,
20     },
21   ]);
22
23  ✓ createRoot(document.getElementById('root')).render(
24  ✓    <StrictMode>
25      | <RouterProvider router={router} />
26    </StrictMode>
27  );
```



Configuração das Rotas

A BIBLIOTECA REACT ROUTER

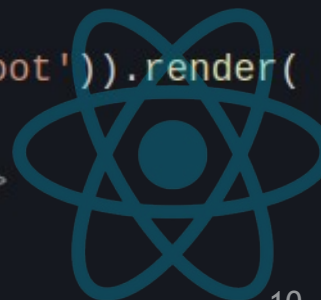


- **O Objeto de Rotas:**

- ▶ **path:** significa o *caminho informado* na *URL*, após o domínio do site.
- ▶ **element:** tratado de qual elemento deverá ser retornado de acordo com a *URL* informada.
- ▶ **Parâmetros:** colocamos *(:)* antes do nome do parâmetro. Veremos mais adiante como ler estes parâmetros.

 ***Veja este exemplo***

```
7  ✓ const router = createBrowserRouter([
8  ✓    {
9      path: '/',
10     ✓ element: <div>Rota Gerenciada no
11       | lado do Cliente com React Router!</div>,
12     },
13     ✓ {
14       path: 'ola',
15       element: <Ola />,
16     },
17     ✓ {
18       path: 'ola/:name',
19       element: <Ola />,
20     },
21   ]);
22
23  ✓ createRoot(document.getElementById('root')).render(
24  ✓    <StrictMode>
25      | <RouterProvider router={router} />
26    </StrictMode>
27  );
```



Configuração das Rotas

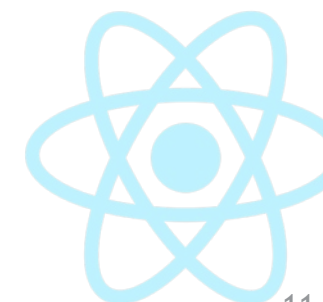
A BIBLIOTECA REACT ROUTER



- **Rotas com Componentes:**

- ▶ Outra forma de configurar rotas é através do componente **Route**, para isso usamos a função **createRoutesFromElements** e a passamos como **argumento** da função **createBrowserRouter**.

```
3  ∨ import {  
4    BrowserRouter,  
5    createRoutesFromElements,  
6    Route,  
7    RouterProvider,  
8  } from 'react-router-dom';  
9  import Ola from './components/Ola/Ola.jsx';  
10 import './index.css';  
11  
12 ∨ const router = createBrowserRouter(  
13 >   createRoutesFromElements(...  
24   )  
25 );
```



Configuração das Rotas

A BIBLIOTECA REACT ROUTER

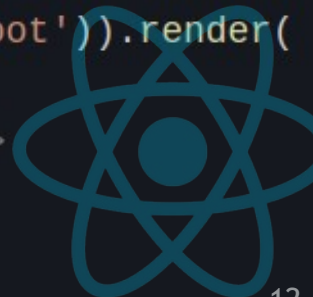


- **O Componente de Rotas:**
 - ▶ A configuração das rotas agora pode ser feita com o componente **Route**.
 - ▶ Possui as mesmas propriedades:
 - ▶ **path**: o **caminho informado** na **URL**.
 - ▶ **element**: o elemento a ser montado de acordo com a **URL** informada.



Veja este exemplo

```
12 const router = createBrowserRouter(  
13   createRoutesFromElements(  
14     <>  
15       <Route  
16         path="/"  
17         element={  
18           <div>Rota Gerenciada no lado do  
19             | Cliente com React Router!</div>  
20         }  
21       />  
22       <Route path="ola" element={<Ola />} />  
23       <Route path="ola/:name" element={<Ola />} />  
24     </>  
25   )  
26 );  
  
27  
28 createRoot(document.getElementById('root')).render(  
29   <StrictMode>  
30     <RouterProvider router={router} />  
31   </StrictMode>  
32 );
```



Configuração das Rotas

A BIBLIOTECA REACT ROUTER



- **Provedor de Rotas:**

- ▶ **INDEPENDENTE** da forma que configuramos nosso **objeto de rotas**, via **objetos** ou **componente Route**, **SEMPRE** devemos **ENVOLVER** nossa aplicação com o **provedor de rotas RouterProvider**.

```
1  import { StrictMode } from 'react';
2  import { createRoot } from 'react-dom/client';
3  import { createBrowserRouter, RouterProvider } from
   'react-router-dom';
4  import Ola from './components/Ola/Ola.jsx';
5  import './index.css';
6
7  > const router = createBrowserRouter([ ...
21  ]]);
22
23  < createRoot(document.getElementById('root')).render(
24  < <StrictMode>
25  < | <RouterProvider router={router} />
26  < </StrictMode>
27  < >);
28
```



Configuração das Rotas

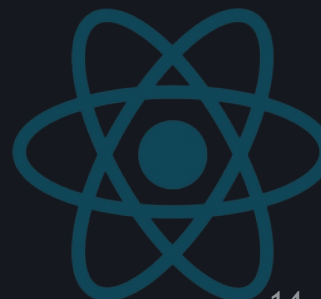
A BIBLIOTECA REACT ROUTER



- **Provedor de Rotas:**

- ▶ O **RouteProvider** é mais um componente, mas baseado na **API de Contexto do React**, um **Context**. Ele mantém um **estado global** que pode ser acessado por toda a aplicação, ou seja, um **contexto**. Sem ele, não é possível gerenciar as rotas.
- ▶ Veremos como criar contextos com a **Context API** nas próximas aulas.

```
5 import {
6   BrowserRouter,
7   createRoutesFromElements,
8   Route,
9   RouterProvider,
10 } from 'react-router-dom';
11
12 const router = createBrowserRouter(
13   > createRoutesFromElements( ...
24   )
25 );
26
27 createRoot(document.getElementById('root')).render(
28   <StrictMode>
29     <RouterProvider router={router} />
30   </StrictMode>
31 );
32
```



Rotas Aninhadas

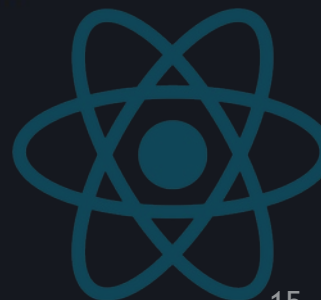
O USO DE CHILDREN E OUTLET

- As vezes pode ser interessante criar rotas aninhadas para manter um mesmo layout. Para isso usamos **Children** e **Outlet**. — — — — —

- ▶ **Children**: é o atributo do objeto de configuração a rota principal que possui as rotas filhas, definidas em um **array de objetos**. — — — — —
- ▶ **Outlet**: é o elemento usada no elemento **pai** para determinar onde serão renderizados os elementos **filhos**, é definido como um **elemento JSX**.

 [Veja este exemplo](#)

```
12 const router = createBrowserRouter([
13   {
14     path: '/',
15     element:
16       <div>
17         Exemplo de rotas aninhadas!
18         <hr />
19         <Outlet />
20       </div>,
21     children: [
22       {
23         path: '/',
24         element: <a href="/ola">Olá</a>,
25       },
26       {
27         path: '/ola',
28         element: <ola />,
29       },
30       {
31         path: '/ola/:name',
32         element: <ola />,
33       },
34     ],
35   },
36 ]);
```



Rotas Aninhadas

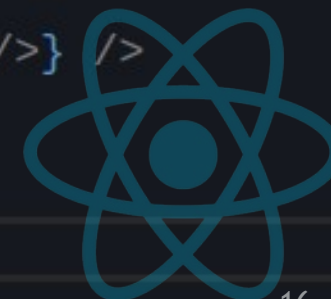
O USO DE CHILDREN E OUTLET



- As rotas aninhadas também podem ser configuradas com o elemento **Route**. Neste caso os elementos filhos serão outros elementos do tipo **Route**, criados como **filhos do primeiro**. Não será necessário o **children**.
- O elemento **Outlet** segue sendo utilizado da mesma maneira.

 [Veja este exemplo](#)

```
14  const router = createBrowserRouter(  
15    createRoutesFromElements(  
16      <Route  
17        path="/"   
18        element={  
19          <div>  
20            Exemplo de rotas aninhadas!  
21            <hr />  
22            <Outlet />  
23          </div>  
24        }  
25      >  
26        <Route path="/" element={<a href="/ola">Olá</a>} />  
27        <Route path="ola" element={<Ola />} />  
28        <Route path="ola/:name" element={<Ola />} />  
29      </Route>  
30    )  
31  );
```



Componente Link

CLIENT SIDE ROUTING COM O COMPONENTE LINK

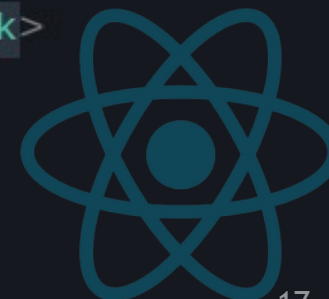


- Quando usamos simplesmente as tags `<a>` normais do *HTML*, o navegador acaba gerando uma **nova requisição ao documento inteiro**.
- Podemos deixar este comportamento **mais rápido** com o componente **Link**, ao em vez do uso da tag `<a>`.

 Exemplo Link com Children

 Exemplo Link com Route

```
1  import { Link, useParams } from 'react-router-dom';
2
3  export default function Ola() {
4    let { name } = useParams();
5
6    return (
7      <>
8        <div>Ola {name || 'Mundo'} !!!</div>
9        <div>
10         <Link to="/">{'< voltar'}</Link>
11       </div>
12      </>
13    );
14  }
```



Hooks: useParams

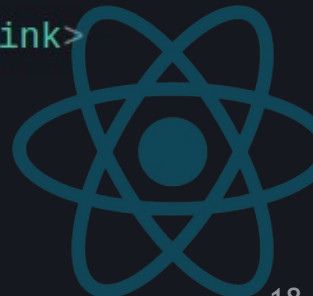
ALGUNS HOOKS DO REACT ROUTER



- Já sabemos que podemos passar parâmetros na rota utilizando o caracter **(:)** antes do nome do parâmetro.
- Mas para recuperar este parâmetro, devemos utilizar um **Hook** próprio do **React Router**, entre tantos outros que a biblioteca possui.
- Veja no componente **Ola** como é simples recuperar o parâmetro name, com o uso do Hook **useParams()**.

Exemplo de useParams

```
1  import { Link, useParams } from 'react-router-dom';
2
3  export default function Ola() {
4    let { name } = useParams();
5
6    return (
7      <>
8        <div>Ola {name || 'Mundo'} !!!</div>
9        <div>
10         <Link to="/">{'< voltar'}</Link>
11       </div>
12     </>
13   );
14 }
```





- **Atividades**

- **1)** Atualizar o leiaute do projeto escolhido na atividade anterior usando **React Router**, configurar rotas para outra quatro páginas do projeto.
- **Requisitos:**
 - Usar git e um repositório privado no github
 - Adicionar ***gillgonzales*** como colaborador.
 - Usar a biblioteca **React Router**;
 - Utilizar rotas aninhadas para manter um leiaute padrão;
 - Configurar as páginas de login e registro.
 - Configurar rotas para outras **DUAS** páginas páginas do projeto
 - Usar parâmetros (***useParams***) em pelo menos uma rota.
 - Manter dados que seriam carregado do backend “mockados”, **arquivos JSON**.



• REFERÊNCIAS

SILVA, Maurício Samy. React Aprenda Praticando. São Paulo: Novatec, 2021.

META INC. Quick Start – React: 2022 Meta Platforms, Inc. Disponível em: <https://react.dev/learn>. Acesso em: 17/10/2024.

SILVA, Maurício Samy. CSS3 – Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS3. São Paulo: Novatec, 2011.

SILVA, Maurício Samy. HTML5 – A linguagem de marcação que revolucionou a web. São Paulo: Novatec, 2011.

SILVA, Maurício Samy. JavaScript: guia do programador. São Paulo: Novatec, 2010.

Tutorial | React Router - reactrouter.com. Disponível em: <https://reactrouter.com/en/main/start/tutorial>. Acesso em: 28/10/2024

DJIRDEH, Hassan. Server-Side Routing vs. Client-Side Routing. Disponível em: <https://www.telerik.com/blogs/server-side-routing-vs-client-side-routing>, Acesso em: 28/10/2024