

Objetivos:

- I. Criar um projeto React TypeScript;
- II. Routes;
- III. Rotas com restrição de acesso.

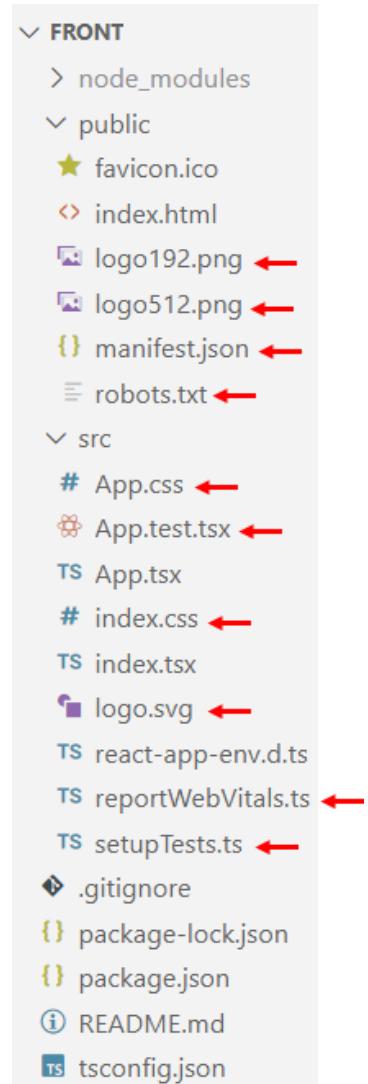
I. Criar um projeto React TypeScript

Siga os passos para criar uma aplicação React TS:

- a) Acesse pelo prompt do CMD o local que você deseja criar o projeto React e digite o comando a seguir para criar o projeto React usando o template para TS:

```
npx create-react-app front --template typescript
```

O projeto será criado na pasta `front`.
- b) No CMD acesse a pasta `front` e abra ela no VS Code;
- c) Ao lado tem-se a estrutura de pastas e arquivos da aplicação criada pelo CRA. Para simplificar o projeto:
 - Delete os arquivos sinalizados pela seta vermelha;
 - Substitua os códigos dos arquivos `index.html` (Figura 1), `index.tsx` (Figura 2) e `App.tsx` (Figura 3);
 - Para subir o projeto digite `npm run start` ou `npm start` no terminal do VS Code. A aplicação estará na porta padrão 3000.
- d) Adicione a dependência `npm i react-router-dom` (<https://www.npmjs.com/package/react-router-dom>).



```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Front</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
```

```
<div id="root"></div>
</body>
</html>
```

Figura 1 – Código do arquivo public/index.html.

```
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render( <App /> );
```

Figura 2 – Código do arquivo src/index.tsx.

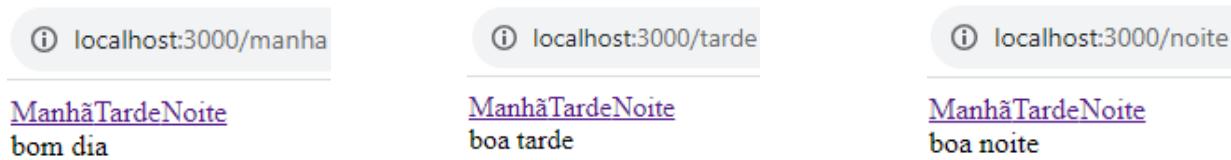
```
export default function App() {
  return <div>boa noite</div>;
}
```

Figura 3 – Código do arquivo src/App.tsx.

II. Routes

A aplicação React é uma SPA (Single-Page Application), desta forma, toda a renderização é feita em apenas uma página (documento web). Porém, é possível criar rotas, com base na URL, para componentes React, fazendo com que o aplicativo tenha várias páginas, como um portal.

O React Router (<https://reactrouter.com/en/main>) é a biblioteca mais usada para implementar roteamento em aplicativos React. O React Router fornece um conjunto de componentes e utilitários que permitem criar e gerenciar rotas no aplicativo. O roteamento do lado do cliente é criado usando os componentes `BrowserRouter`, `Routes` e `Route` dos pacotes `react-router` e `react-router-dom`. No exemplo da Figura 4 foram definidas as rotas `/manha`, `/tarde` e `/noite` para os componentes `Manha`, `Tarde` e `Noite`, respectivamente. A seguir tem-se o teste das rotas no navegador:



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

export default function App() {
  return (
    <BrowserRouter>
      <Menu />
      <Rotas />
    </BrowserRouter>
  );
}

function Menu() {
  return (
    <ul>
      <li><Link to="/">Manhã</Link></li>
      <li><Link to="/tarde">Tarde</Link></li>
      <li><Link to="/noite">Noite</Link></li>
    </ul>
  );
}

function Rotas() {
  return (
    <Routes>
      <Route path="/" element={<Manha />} />
      <Route path="/tarde" element={<Tarde />} />
      <Route path="/noite" element={<Noite />} />
    </Routes>
  );
}

function Manha() {
  return <p>ManhãTardeNoite</p>;
}

function Tarde() {
  return <p>boa tarde</p>;
}

function Noite() {
  return <p>boa noite</p>;
}
```

```

    );
}

function Rotas() {
  return (
    <Routes>
      <Route path="/" element={<Erro />} />
      <Route path="/manha" element={<Manha />} />
      <Route path="/tarde" element={<Tarde />} />
      <Route path="/noite" element={<Noite />} />
    </Routes>
  );
}

function Manha() {
  return <div>bom dia</div>;
}

function Tarde() {
  return <div>boa tarde</div>;
}

function Noite() {
  return <div>boa noite</div>;
}

function Erro() {
  return <div>Rota inexistente</div>;
}

function Menu() {
  return (
    <div>
      <Link to="manha">Manhã</Link>
      <Link to="tarde">Tarde</Link>
      <Link to="noite">Noite</Link>
    </div>
  );
}

```

Figura 4 – Exemplo de rotas - código do arquivo src/App.tsx.

Componentes do React Router necessários para criar o roteamento:

- BrowserRouter: envolve todo o aplicativo em um contexto de roteamento. Por isso colocamos o seguinte código no componente APP:

```

<BrowserRouter>
  <Menu />
  <Rotas />

```

```
</BrowserRouter>
```

- Routes: define um container de rotas aninhadas:

```
<Routes>
  <Route path="/" element={<Erro />} />
  <Route path="/manha" element={<Manha />} />
  <Route path="/tarde" element={<Tarde />} />
  <Route path="/noite" element={<Noite />} />
</Routes>
```

- Routes: define como um determinado componente deve ser renderizado com base na URL. No exemplo anterior são as rotas /manha, /tarde e /noite, além da rota para qualquer caminho;
- Link: permite criar links para navegar entre diferentes rotas sem recarregar a página. Na prática funciona como o hyperlink [do HTML](#):

```
<Link to="manha">Manhã</Link>
<Link to="tarde">Tarde</Link>
<Link to="noite">Noite</Link>
```

III. Rotas com restrição de acesso

O React Router é uma rota para um componente, então se a rota deixa de existir, o componente não será acessado por uma URL. No exemplo da Figura 5 existem os componentes **Manha**, **Tarde**, **Noite** e **Madrugada**. Esses componentes foram mapeados usando dois conjuntos de rotas retornados, respectivamente, pelas funções **DiaRotas** e **NoiteRotas**:

```
function DiaRotas() {
  return (
    <BrowserRouter>
      <Menu />
      <Routes>
        <Route path="/" element={<Erro />} />
        <Route path="/manha" element={<Manha />} />
        <Route path="/tarde" element={<Tarde />} />
      </Routes>
    </BrowserRouter>
  );
}

function NoiteRotas() {
  return (
    <BrowserRouter>
      <Menu />
      <Routes>
        <Route path="/" element={<Erro />} />
        <Route path="/noite" element={<Noite />} />
        <Route path="/madrugada" element={<Madrugada />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Como somente um conjunto de rotas pode estar disponível, então quando as rotas da DiaRotas forem retornadas pela função Rotas. O acesso aos componentes **Noite** e **Madrugada** não estarão disponíveis para acesso pela URL:

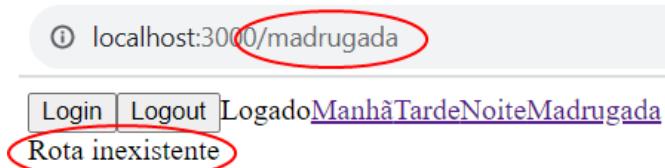
```
function Rotas() {
  const { logado } = useContext(Contexto);
  return logado ? <DiaRotas /> : <NoiteRotas />;
}
```

Na prática a restrição de acesso foi construída usando intercâmbio das rotas através do retorno da função Rotas. Para viabilizar a mudança de logado/desconectado usou-se o React Hooks para que o estado da propriedade **logado** fosse propagado por toda a árvore de componentes:

```
const [logado, setLogado] = useState(false);
```

Essa organização do código faz com que a definição das rotas seja trocada mudando a propriedade de estado **logado**.

A rota **/madrugada** não existe quando está **logado**, pois estão disponíveis apenas as rotas retornadas pela função **DiaRotas**:



A rota **/tarde** está disponível quando está **logado**, pois estão disponíveis as rotas retornadas pela função **DiaRotas**:



```
import { BrowserRouter, Link } from "react-router-dom";
import { Route, Routes } from "react-router";
import { createContext, useContext, useState } from "react";

export default function App() {
  return (
    <Provider>
      <Rotas />
    </Provider>
  );
}

function Rotas() {
  const { logado } = useContext(Contexto);
  return logado ? <DiaRotas /> : <NoiteRotas />;
}

function DiaRotas() {
  return (
    <BrowserRouter>
      <Menu />
      <Routes>
        <Route path="*" element={<Erro />} />
      </Routes>
    </BrowserRouter>
  );
}
```

```

        <Route path="/manha" element={<Manha />} />
        <Route path="/tarde" element={<Tarde />} />
    </Routes>
</BrowserRouter>
);
}

function NoiteRotas() {
    return (
        <BrowserRouter>
            <Menu />
            <Routes>
                <Route path="/" element={<Erro />} />
                <Route path="/noite" element={<Noite />} />
                <Route path="/madrugada" element={<Madrugada />} />
            </Routes>
        </BrowserRouter>
);
}

function Manha() { return <div>bom dia</div>; }

function Tarde() { return <div>boa tarde</div>; }

function Noite() { return <div>boa noite</div>; }

function Madrugada() { return <div>bom sono</div>; }

function Erro() { return <div>Rota inexistente</div>; }

function Menu() {
    const { logado, setLogado } = useLogado();
    return (
        <div>
            <button onClick={() => setLogado(true)}>Login</button>
            <button onClick={() => setLogado(false)}>Logout</button>
            <span>{logado ? "Logado" : "Desconectado"}</span>
            <Link to="manha">Manhã</Link>
            <Link to="tarde">Tarde</Link>
            <Link to="noite">Noite</Link>
            <Link to="madrugada">Madrugada</Link>
        </div>
    );
}

interface Props {
    logado: boolean;
    setLogado: Function;
}

```

```
const Contexto = createContext<Props>({} as Props);

function Provider({ children }: any) {
  const [logado, setLogado] = useState(false);
  return (
    <Contexto.Provider value={{ logado, setLogado }}>
      {children}
    </Contexto.Provider>
  );
}

// definição do Hook
function useLogado() {
  const context = useContext(Contexto);
  return context;
}
```

Figura 5 – Exemplo de rotas com restrição de acesso- código do arquivo src/App.tsx.

IV. Exercícios

Veja os vídeos se tiver dúvidas nos exercícios:

Exercício 1 - <https://youtu.be/Ofg-q8GERSI>

Exercício 2 - <https://youtu.be/0fQSWRnX8YA>

Exercício 3 - <https://youtu.be/F4T7Yo1bjy8>

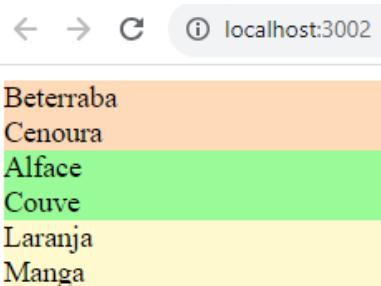
Exercício 4 - <https://youtu.be/zXB-wcZ3BQY>

Exercício 1 – Utilize o código a seguir para criar um aplicativo React TS com os componentes roteados para os caminhos /legume, /verdura e /fruta. Qualquer URL desconhecida deverá ser roteada para o componente Erro.

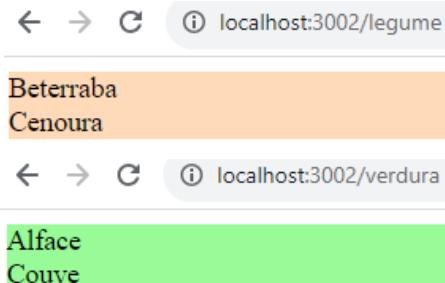
Dicas:

- A definição de rotas usa a seguinte estrutura de componentes:
- ```
<BrowserRouter>
 <Routes>
 <Route path="/caminho" element={<Componente />} />
 <Route path="/caminho" element={<Componente />} />
 </Routes>
</BrowserRouter>
```
- Precisará criar o elemento <Route> quatro vezes, sendo uma para cada rota para componente uma rota para caminhos desconhecidos.

Todos os componentes na mesma página:



Cada componente em uma URL:



localhost:3002/fruta

Laranja  
Manga

localhost:3002

## Rota inexistente

```
export default function App() {
 return (
 <div>
 <Legume />
 <Verdura />
 <Fruta />
 </div>
);
}

function Legume(){
 return (
 <div style={{backgroundColor:"PeachPuff"}>
 <div>Beterraba</div>
 <div>Cenoura</div>
 </div>
);
}

function Verdura(){
 return (
 <div style={{backgroundColor:"palegreen"}>
 <div>Alface</div>
 <div>Couve</div>
 </div>
);
}

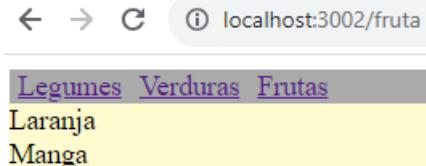
function Fruta(){
 return (
 <div style={{backgroundColor:"LemonChiffon"}>
 <div>Laranja</div>
 <div>Manga</div>
 </div>
);
}

function Erro() {
 return <h3>Rota inexistente</h3>;
}
```

**Exercício 2** – Alterar o aplicativo do Exercício 1 para ter um menu com os links para /legume, /verdura e /fruta.

Dicas:

- Use o elemento `<Link>` para criar um hyperlink para cada URL;
- Coloque os elementos `<Link>` no corpo do `<BrowserRouter>`.



**Exercício 3** – Alterar o aplicativo do Exercício 2 para ter rotas para os componentes Carro e Moto. Incluir também controle de rotas, as rotas para /legume, /verdura e /fruta estarão disponíveis quando estiver logado e as rotas /carro e /moto deverão estar disponíveis quando não estiver logado.

Dicas:

- Para facilitar a propagação de propriedades pela árvore de componentes é mais prático utilizar contexto. Crie um contexto com as propriedades `{logado, setLogado}`;
- Defina a propriedade de estado no componente App para criar o estado logado;
- Envolve o `<BrowserRouter>` pelo `<Context.Provider>` para toda a árvore ter acesso às propriedades `{logado, setLogado}`;
- Crie dois conjuntos de rotas, um para as rotas /legume, /verdura e /fruta e o outro para as rotas /carro e /moto. Faça o intercâmbio desses conjuntos de rotas usando o valor da propriedade logado.

Quando está logado:

<a href="#">Deslogar</a>	<a href="#">Legumes</a>	<a href="#">Verduras</a>	<a href="#">Frutas</a>	<a href="#">Carros</a>	<a href="#">Motos</a>
Laranja					
Manga					

Quando não está logado:

<a href="#">Logar</a>	<a href="#">Legumes</a>	<a href="#">Verduras</a>	<a href="#">Frutas</a>	<a href="#">Carros</a>	<a href="#">Motos</a>
Gol					
Uno					

```

function Carro() {
 return (
 <div style={{ backgroundColor: "LightSkyBlue" }}>
 <div>Gol</div>
 <div>Uno</div>
 </div>
);
}

function Moto() {
 return (
 <div style={{ backgroundColor: "Wheat" }}>
 <div>CG160</div>
 <div>DT180</div>
 </div>
);
}

```

**Exercício 4** – Criar um aplicativo React TS para o cadastro de clientes. O aplicativo deverá ter as páginas de cadastro e repositório.

Cadastro

Nome: Carla

Idade: 20

Salvar

- 
- Cadastro
- Repositório
- Cadastrados
1. Ana 21
  2. Maria 22
  3. Pedro 19
  4. Carla 20

Dicas:

- Crie o array de registros como parte do estado do componente App;
- Deixe a função para adicionar registros no array no componente App;
- Crie um contexto para propagar o array e a função pela árvore de componentes;
- Envolve o `<BrowserRouter>` pelo `<Contexto.Provider>` para toda a árvore ter acesso às propriedades `{lista, add}`.