

## Objetivos:

- i. Criar um projeto React TypeScript;
- ii. Styled-components;
- iii. Temas.

### 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:

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

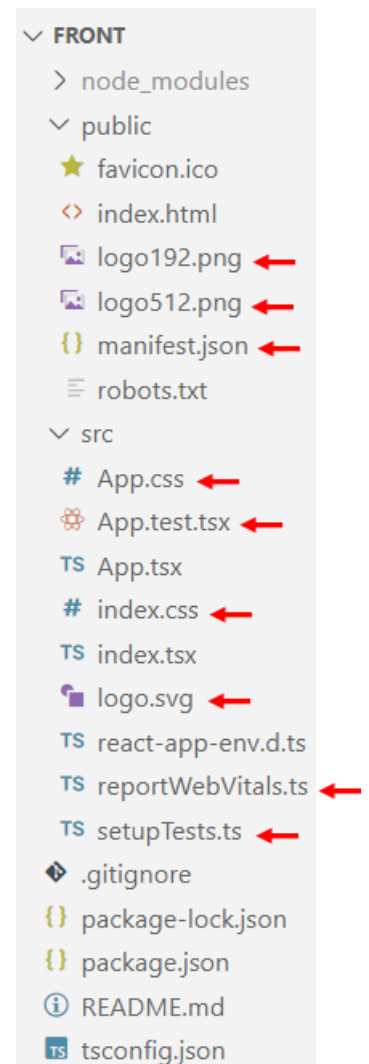
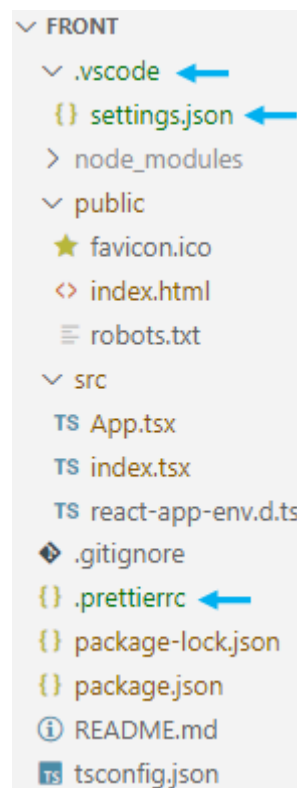
- b) O projeto foi criado na pasta **front**. No CMD acesse a pasta **front** e abra ela no VS Code;
- c) Ao lado tem-se a estrutura de pastas e arquivos da aplicação criada pelo Create React App. Para simplificar o projeto:
  - Delete os arquivos sinalizados pela seta vermelha;
  - Substitua os códigos dos arquivos index.html (Figura 1), index.tsx (Figura 2) e App.tsx (Figura 3).

- d) Crie a pasta **.vscode** e o arquivo **settings.json** (assim como é mostrado pelas setas azuis) e coloque o JSON da Figura 4. Essas propriedades são usadas pelo VS Code para formatar o código enquanto estamos digitando;

- e) Crie o arquivo **.prettierrc** (Figura 5) para sobrescrever as configurações padrões do Prettier

(<https://prettier.io/docs/en/configuration.html>);

- f) Adicione a dependência `npm i styled-components` (<https://www.npmjs.com/package/styled-components>);



- g) Adicione o pacote de tipos do styled-components como dependência de desenvolvimento npm i @types/styled-components -D;
- h) O comando `npx react-scripts start` é usado para subir o projeto React. Ele está na propriedade `scripts > start` do arquivo `package.json`. Desta forma, usamos `npm run start` ou `npm start`, no terminal do VS Code, para subir a aplicação na porta padrão 3000;

Porém, podemos definir a porta das seguintes formas:

- 1) Setar o número da porta no comando `react-scripts start` da propriedade `scripts > start`, assim como é mostrado a seguir:

```
"scripts": {
  "start": "set PORT=3100 && react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
```

- 2) Criar o arquivo `.env` na raiz do projeto e definir a variável de ambiente `PORT` assim como é mostrado ao lado.

```
1  PORT = 3008
```

Como a variável `PORT` foi definida como variável de ambiente, então não precisamos colocar ela no comando `react-scripts`:

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
```

```
<!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" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <title>React App</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 React from "react";
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.

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

export default App;
```

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

```
{
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "editor.bracketPairColorization.enabled": true,
  "editor.formatOnSave": true,
  "editor.formatOnPaste": true,
  "editor.wordWrap": "on"
}
```

Figura 4 – Código do arquivo .vscode/settings.json.

```
{
  "singleQuote": false,
  "tabWidth": 2,
  "useTabs": false,
  "semi": true
}
```

Figura 5 – Código do arquivo .prettierrc.

## ii. Styled-components

Styled-components é uma biblioteca CSS-in-JS que permite escrever CSS regular e anexá-lo a componentes JavaScript sem a necessidade de criar classes de estilos (<https://styled-components.com>).

No exemplo a seguir o componente Header tem o estilo atachado usando `styled.div`.

Módulo Headar.tsx	Módulo App.tsx	Resultado no navegador
<pre>import React from "react"; import styled from "styled-components";  export function Header() {   return &lt;Sld&gt;Título&lt;/Sld&gt;; }  const Sld = styled.div`   color: #0066b3;   font-weight: bold;   margin-bottom: 20px;   font-size: 1.6rem; `;</pre>	<pre>import React from "react"; import { Header } from "./components/Header";  function App() {   return (     &lt;&gt;       &lt;Header /&gt;       &lt;Header /&gt;     &lt;/&gt;   ); }  export default App;</pre>	<p><b>Título</b></p> <p><b>Título</b></p>

No exemplo a seguir o componente Header recebe uma string como conteúdo (o conteúdo de um elemento é aquilo que está entre as marcações de abertura e fechamento). O conteúdo do elemento precisa ser recebido na propriedade children, por este motivo `{children}` está entre colchetes (lembre-se que os colchetes são usados para criar/desestruturar objetos JSON). Observação:

- `{ children }`: Props Precisamos definir o tipo de dado que o componente Header recebe como parâmetro. A definição do tipo de dado pode ser usando `interface` ou `type`.

Módulo Headar.tsx	Módulo App.tsx	Resultado no navegador
<pre>import React from "react"; import styled from "styled-components";  /*interface Props {   children: string; }*/ type Props = {   children: string; };  export function Header({ children }: Props) {   return &lt;Sld&gt;{children}&lt;/Sld&gt;; }  const Sld = styled.div`   color: #0066b3;   font-weight: bold;</pre>	<pre>import React from "react"; import { Header } from "./components/Header";  function App() {   return (     &lt;&gt;       &lt;Header&gt;Bom dia&lt;/Header&gt;       &lt;Header&gt;Boa noite&lt;/Header&gt;     &lt;/&gt;   ); }  export default App;</pre>	<p><b>Bom dia</b></p> <p><b>Boa noite</b></p>

```
margin-bottom: 20px;
font-size: 1.6rem;
`;
```

No exemplo a seguir o componente Header recebe um elemento HTML como conteúdo. Desta forma, o objeto recebido como parâmetro precisa ser definido como pertencente ao tipo `HTMLAttributes<HTMLElement>`. Neste exemplo o `children` será `<p>Bom dia</p>` na primeira chamada e `<div>Boa noite</div>` na segunda chamada.

Módulo Headar.tsx	Módulo App.tsx	Resultado no navegador
<pre>import React, { HTMLAttributes } from "react"; import styled from "styled-components";  interface Props extends HTMLAttributes&lt;HTMLElement&gt; {} //type Props = HTMLAttributes&lt;HTMLElement&gt;;  export function Header({ children }: Props) {   return &lt;Sld&gt;{children}&lt;/Sld&gt;; }  const Sld = styled.div`   color: #0066b3;   font-weight: bold;   margin-bottom: 20px;   font-size: 1.6rem; `;</pre>	<pre>import React from "react"; import { Header } from "./components/Header";  function App() {   return (     &lt;&gt;       &lt;Header&gt;         &lt;p&gt;Bom dia&lt;/p&gt;       &lt;/Header&gt;       &lt;Header&gt;         &lt;div&gt;Boa noite&lt;/div&gt;       &lt;/Header&gt;     &lt;/&gt;   ); }  export default App;</pre>	<p><b>Bom dia</b></p> <p><b>Boa noite</b></p>

No exemplo a seguir o componente Header é um wrapper (envoltório) para os elementos `label` e `input`.

Observação:

- `styled.input.attrs({ type: "text" })` é usado para especificar que o elemento possui o atributo `type` como `text`.

Módulo Headar.tsx	Módulo App.tsx	Resultado no navegador
<pre>import React, { HTMLAttributes } from "react"; import styled from "styled-components";  interface Props extends HTMLAttributes&lt;HTMLElement&gt; {</pre>	<pre>import React from "react"; import { Header } from "./components/Header";  function App() {</pre>	<p>Nome <input type="text"/></p> <p>Idade <input type="text"/></p>

```

    label: string;
    itemID: string;
  }
  /*type Props = HTMLAttributes<HTMLElement>
  & {
    label: string;
    itemID: string;
  };*/

  export function Header({ label, itemID }:
  Props) {
    return (
      <WrapperSld>
        <LabelSld
htmlFor={itemID}>{label}</LabelSld>
        <InputSld id={itemID} />
      </WrapperSld>
    );
  }

  const WrapperSld = styled.div`
    margin: 5px;
    padding: 5px;
    background-color: #fff8dc;
  `;

  const LabelSld = styled.label`
    color: #0066b3;
    font-weight: bold;
    font-family: calibri;
    margin-bottom: 12px;
    font-size: 1.1rem;
  `;

  const InputSld = styled.input.attrs({
    type: "text" })`
    color: #0066b3;
    border: 1px solid #555;
    border-radius: 0.4rem;
    padding: 0.4rem;
    margin-left: 0.4rem;
    font-weight: bold;
    font-family: calibri;
    font-size: 1.1rem;
  `;

```

```

    return (
      <>
        <Header
          label="Nome"
          itemID="nome"
        />
        <Header
          label="Idade"
          itemID="idade"
        />
      </>
    );
  }

  export default App;

```

No exemplo a seguir os estados são definidos no componente App e passados como propriedade para o componente Header.

Módulo Header.tsx	Módulo App.tsx	Resultado no navegador
<pre>import React, { HTMLAttributes } from "react"; import styled from "styled- components";  /*interface Props extends HTMLAttributes&lt;HTMLElement&gt; {   label: string;   itemID: string;   value: string;   set: Function; }*/ type Props = HTMLAttributes&lt;HTMLElement&gt; &amp; {   label: string;   itemID: string;   value: string;   set: Function; };  export function Header({ label, itemID, ...rest }: Props) {   return (     &lt;WrapperSld&gt;       &lt;LabelSld htmlFor={itemID}&gt;{label}&lt;/LabelSld&gt;       &lt;InputSld         id={itemID}         value={rest.value}         onChange={e =&gt; rest.set(e.target.value)}       /&gt;     &lt;/WrapperSld&gt;   ); }  const WrapperSld = styled.div`   margin: 5px;   padding: 5px;   background-color: #fff8dc; `;  const LabelSld = styled.label`   color: #0066b3;</pre>	<pre>import React, { useState } from "react"; import { Header } from "./components/Header";  function App() {   const [nome, setNome] = useState("");   const [idade, setIdade] = useState("");   return (     &lt;&gt;       &lt;Header         label="Nome"         itemID="nome"         value={nome}         set={setNome}       /&gt;       &lt;Header         label="Idade"         itemID="idade"         value={idade}         set={setIdade}       /&gt;     &lt;/&gt;   ); }  export default App;</pre>	


```
font-weight: bold;
font-family: calibri;
margin-bottom: 12px;
font-size: 1.1rem;
`;

const InputSld =
styled.input.attrs({ type: "text"
})`
color: #0066b3;
border: 1px solid #555;
border-radius: 0.4rem;
padding: 0.4rem;
margin-left: 0.4rem;
font-weight: bold;
font-family: calibri;
font-size: 1.1rem;
`;
```

### iii. Temas

O tema é propagado na árvore de componentes usando o contexto `<ThemeProvider theme={tema}>`. A propriedade `theme` recebe o objeto JSON a ser propagado pela árvore de componentes.

As propriedades são acessadas nos objetos styled através do objeto props.

Módulo Headar.tsx	Módulo App.tsx	Resultado no navegador
<pre>import React, { HTMLAttributes } from "react"; import styled from "styled- components";  /*interface Props extends HTMLAttributes&lt;HTMLElement&gt; {   label: string;   itemID: string;   value: string;   set: Function; }*/ type Props = HTMLAttributes&lt;HTMLElement&gt; &amp; {   label: string;   itemID: string;   value: string;   set: Function; };  export function Header({ label,</pre>	<pre>import React, { useState } from "react"; import { Header } from "./components/Header"; import { ThemeProvider } from "styled-components";  const tema = {   fundo: "#e0ffff",   cor: "#444" };  function App() {   const [nome, setNome] = useState("");   const [idade, setIdade] = useState("");   return (     &lt;ThemeProvider       theme={tema}&gt;       &lt;Header</pre>	



<pre> itemID, ...rest }: Props) {   return (     &lt;WrapperSld&gt;       &lt;LabelSld htmlFor={itemID}&gt;{label}&lt;/LabelSld&gt;       &lt;InputSld         id={itemID}         value={rest.value}         onChange={e =&gt; rest.set(e.target.value)}       /&gt;     &lt;/WrapperSld&gt;   ); }  const WrapperSld = styled.div`   margin: 5px;   padding: 5px;   background-color: \${props =&gt; props.theme.fundo}; `;  const LabelSld = styled.label`   color: \${props =&gt; props.theme.cor};   font-weight: bold;   font-family: calibri;   margin-bottom: 12px;   font-size: 1.1rem; `;  const InputSld = styled.input.attrs({ type: "text" })`   color: \${props =&gt; props.theme.cor};   border: 1px solid #555;   border-radius: 0.4rem;   padding: 0.4rem;   margin-left: 0.4rem;   font-weight: bold;   font-family: calibri;   font-size: 1.1rem; `; </pre>	<pre> label="Nome" itemID="nome" value={nome} set={setNome} /&gt; &lt;Header   label="Idade"   itemID="idade"   value={idade}   set={setIdade} /&gt; &lt;/ThemeProvider&gt; ); }  export default App; </pre>
--	--

No exemplo a seguir existem dois temas que são trocados de acordo com o valor da propriedade idade. Podemos usar o mesmo mecanismo para criar páginas com temas black e light.

## Módulo App.tsx

```
import React, { useState } from "react";
import { Header } from "../components/Header";
import { ThemeProvider } from "styled-components";

const par = {
  fundo: "#e0ffff",
  cor: "#444"
};

const impar = {
  fundo: "#d8bfd8",
  cor: "#800000"
};

function App() {
  const [nome, setNome] = useState("");
  const [idade, setIdade] = useState("");

  return (
    <ThemeProvider
      theme={parseInt(idade) % 2 === 0 ? par : impar}>
      <Header
        label="Nome"
        itemID="nome"
        value={nome}
        set={setNome}
      />
      <Header
        label="Idade"
        itemID="idade"
        value={idade}
        set={setIdade}
      />
    </ThemeProvider>
  );
}

export default App;
```

## Resultado no navegador

Nome

Idade

Nome

Idade