1. Atividades Práticas - Styled Components com React

Nesta atividade prática, vamos criar um exemplo simples passo a passo para ilustrar como podemos usar Styled Components em uma aplicação React.

1.1 Criando o projeto, instalando dependências e iniciando o servidor local

Crie um projeto chamado **compras**, digitando o seguinte comando no terminal:

```
npm create vite@latest my-styled-components -- --template react
```

Este processo de configuração inicial do projeto leva alguns segundos. Ao terminar, o Vite repassa instruções para que você termine de instalar as dependências do seu projeto. Desta forma, digite o seguinte comando para **entrar no diretório** recém criado do nosso projeto:

```
cd my-styled-components
```

Em seguida, **instale as dependências** necessárias do projeto executando no terminal o seguinte comando:

```
npm install
```

Antes de começar a usar **Styled Components** em seus projetos, você precisa instalar a biblioteca no seu projeto React:

```
npm install styled-components
```

Você pode, agora, **abrir a aplicação no editor Visual Studio Code**. Para isso, dentro do diretório do projeto digite o seguinte comando no terminal:

```
code .
```

Em seguida, você inicializará um servidor local e executará o projeto em seu navegador:

```
npm run dev
```

Ao executar esse script, você iniciará um servidor local de desenvolvimento, executará o código do projeto, iniciará um observador que detecta alterações no código e abrirá o projeto em um navegador web. Ele irá rodar a aplicação em modo desenvolvimento em http://localhost:5173/.

1.2 Passo 1: Criar Componentes Estilizados

Agora, no arquivo **App.jsx**, vamos criar alguns componentes estilizados usando a biblioteca Styled Components. Vamos construir uma interface simples com um título, um parágrafo e um botão que muda de cor ao ser clicado.

```
import { useState } from "react";
import styled from "styled-components";
// Definindo o componente estilizado para o container
const Container = styled.div`
  background-color: lightblue;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
  min-height: 100vh;
`;
// Definindo o componente estilizado para o título
const Titulo = styled.h1`
  color: darkblue;
  font-size: 24px;
  margin-bottom: 10px;
`;
// Definindo o componente estilizado para o parágrafo
const Paragrafo = styled.p`
  font-size: 18px;
  color: darkgray;
`;
// Definindo o componente estilizado para o botão com estilo dinâmico baseado
em props
const Botao = styled.button`
  background-color: ${(props) => (props.ativo ? "green" : "gray")};
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
```

```
  &:hover {
    background-color: ${(props) => (props.ativo ? "darkgreen" : "darkgray")};
  }
`;
function App() {
  // Definindo o estado para controlar o estilo dinâmico
  const [ativo, setAtivo] = useState(false);

  // Função para alternar o estado do botão
  const alternarEstado = () => {
    setAtivo(!ativo);
  };
  return (
    <Container>
      <Titulo>Bem-vindo ao Meu App</Titulo>
      <Paragrafo>
        Este é um exemplo simples de uso de Styled Components com React.
      </Paragrafo>
      <Botao ativo={ativo} onClick={alternarEstado}>
        {ativo ? "Ativo" : "Inativo"}
      </Botao>
    </Container>
  );
}
export default App;
```

Observe neste exemplo:

- **Componente Container:** Um componente estilizado que serve como o container da interface, com fundo azul-claro, padding, bordas arredondadas e altura mínima da tela definida como 100vh (altura total da tela).
- **Componente Titulo:** Um título estilizado com cor azul escuro e tamanho de fonte maior.
- **Componente Paragrafo:** Um parágrafo estilizado com tamanho de fonte moderado e cor cinza escuro.

- **Componente Botao:** Um botão estilizado que muda a cor de fundo com base no valor da prop ativo. Se ativo for **true**, o fundo será verde; se for **false**, será cinza. O botão também tem um efeito hover que altera a cor quando o mouse passa por cima.
- **Estado ativo:** Controla a aparência do botão. Quando o botão é clicado, o estado ativo alterna entre **true** e **false**.

1.3 Passo 2: Testar a aplicação

Agora, você pode rodar o projeto para ver os estilos aplicados.

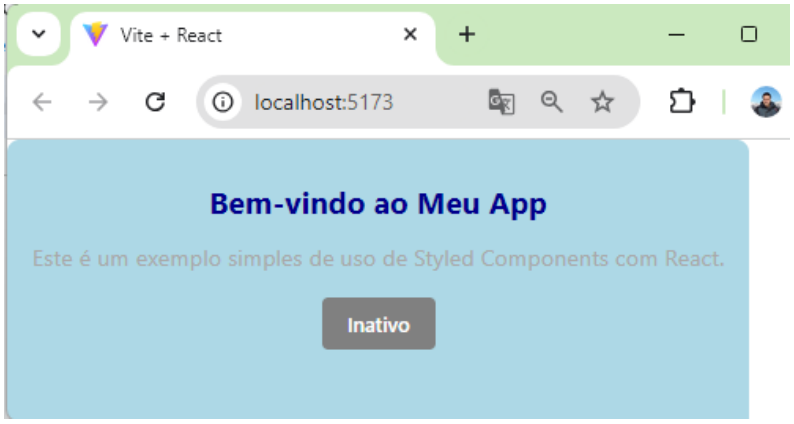

Figura 1: Exemplo de uso de Styled Components com React

Quando a página carregar o fundo será azul-claro e o título será azul-escuro. O parágrafo estará cinza-escuro e o botão começará no estado "Inativo" com fundo cinza. Porém, ao clicar no botão, o estado mudará para "Ativo" e o fundo do botão mudará para verde. Ao passar o mouse sobre o botão, ele ficará em um tom mais escuro.

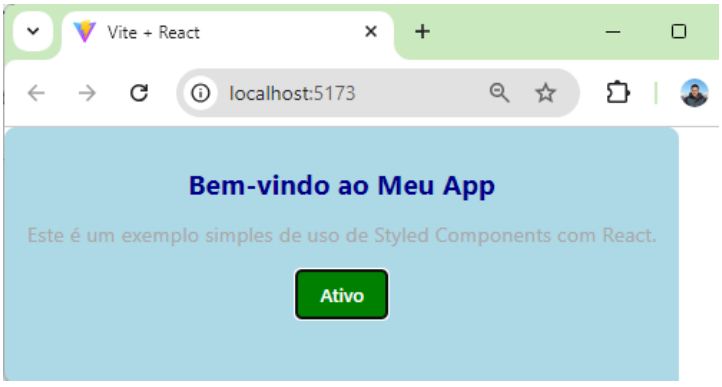

Figura 2: Exemplo de uso de Styled Components com React após clicar no botão

1.4 Passo 3: Usando Temas com Styled Components

Uma das funcionalidades mais poderosas dos Styled Components é o suporte a temas. Vamos adicionar a capacidade de alternar entre um tema claro e um tema escuro usando o **ThemeProvider**.

Dessa forma, crie um arquivo chamado **theme.jsx** para definir os temas claro e escuro:

```
export const temaClaro = {
  cores: {
    fundo: "#f0f0f0",
    texto: "#333",
    botaoFundo: "#4CAF50",
    botaoTexto: "#fff",
  },
};
export const temaEscuro = {
  cores: {
    fundo: "#333",
    texto: "#f0f0f0",
    botaoFundo: "#FF5722",
    botaoTexto: "#fff",
  },
};
```

Agora, vamos modificar o arquivo **App.jsx** para alternar entre os temas claro e escuro usando o **ThemeProvider**.

```
import { useState } from "react";
import styled, { ThemeProvider } from "styled-components";
import { temaClaro, temaEscuro } from "./theme";

// Componentes estilizados
const Container = styled.div`
  background-color: ${(props) => props.theme.cores.fundo};
  color: ${(props) => props.theme.cores.texto};
  padding: 20px;
  border-radius: 10px;
  text-align: center;
```

```
  min-height: 100vh;
  transition: background-color 0.3s ease, color 0.3s ease;
`;
const Titulo = styled.h1`
  font-size: 24px;
  margin-bottom: 10px;
`;
const Paragrafo = styled.p`
  font-size: 18px;
`;
const Botao = styled.button`
  background-color: ${(props) => props.theme.cores.botaoFundo};
  color: ${(props) => props.theme.cores.botaoTexto};
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  margin-top: 20px;
  transition: background-color 0.3s ease;
`;
function App() {
  const [temaAtual, setTemaAtual] = useState(temaClaro);

  // Alterna entre tema claro e escuro
  const alternarTema = () => {
    setTemaAtual(temaAtual === temaClaro ? temaEscuro : temaClaro);
  };
  return (
    <ThemeProvider theme={temaAtual}>
      <Container>
        <Titulo>Bem-vindo ao Meu App</Titulo>
        <Paragrafo>
          Este é um exemplo de alternância de temas com Styled Components.
        </Paragrafo>
        <Botao onClick={alternarTema}>
         Alternar para {temaAtual === temaClaro ? "Tema Escuro" : "Tema Claro"}
        </Botao>
      </Container>
    </ThemeProvider>
  );
}
export default App;
```

Observe neste código:

- **ThemeProvider:** Este componente envolve a aplicação e injeta o tema selecionado em todos os componentes estilizados.
- **Acessando o Tema:** Dentro dos componentes estilizados, como **Container**, **Titulo**, **Paragrafo** e **Botao**, acessamos as cores e estilos do tema usando **props.theme.cores**.
- **Alternância de Tema:** A função **alternarTema** altera o tema entre claro e escuro, e o botão exibe o tema atual, permitindo alternar com um clique.

Agora, ao rodar o projeto, você verá a interface inicial no tema claro.

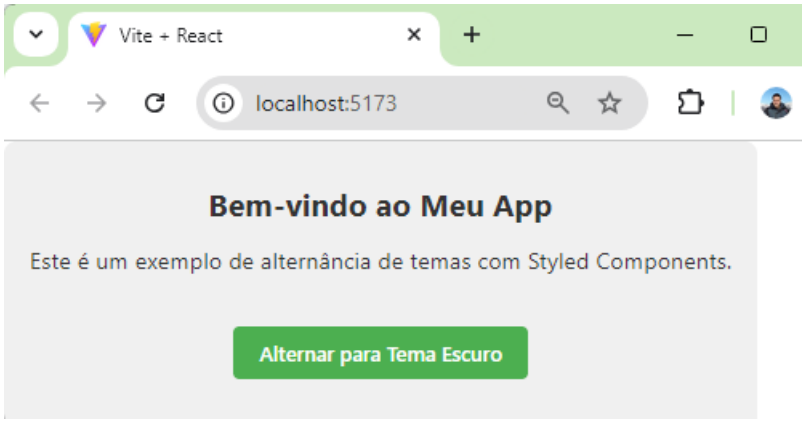

Figura 3: Exemplo de uso de ”tema claro” com Styled Components

Porém, ao clicar no botão, o tema mudará para escuro, alterando as cores de fundo, texto e do botão.

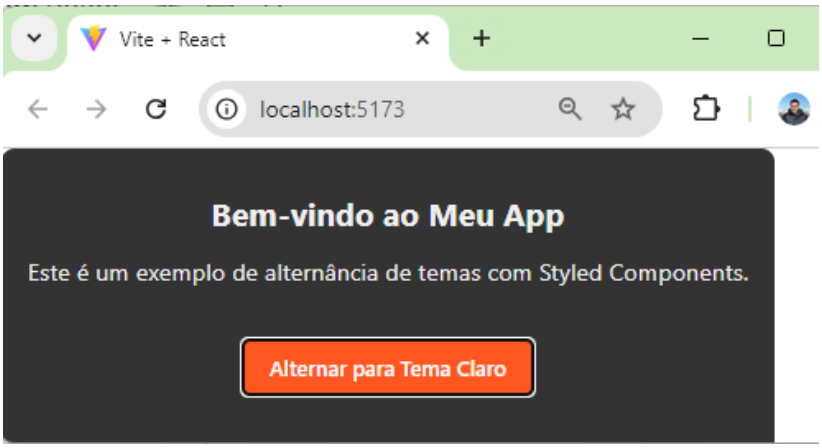

Figura 4: Exemplo de uso de ”tema escuro” com Styled Components