

# React

Site: [Geração Tech](#)

Curso: Formação em Desenvolvedor Web - Online

Livro: React

Impresso por: JOÃO VITOR DE MELO FREITAS

Data: quinta-feira, 18 jul. 2024, 22:43

# Índice

## **1. Trabalhando com CSS**

1.1. Vídeo da Aula

## **2. PrimeReact para Estilização**

2.1. Vídeo Aula

## **3. PrimeReact: Criando Tela de Login**

3.1. Vídeo da aula

## **4. Estilizando a Tela de Login**

4.1. Vídeo Aula

## **5. Interatividade com Input de Senha**

5.1. Vídeo Aula

# 1. Trabalhando com CSS

## Método 1: Arquivo CSS Tradicional

Você pode criar um arquivo CSS tradicional e importá-lo em seus componentes [React](#), semelhante ao que faria em um projeto HTML/CSS puro.

### Passo 1: Criar o Arquivo CSS

Crie um arquivo `index.css` na raiz do seu projeto. Adicione algumas regras CSS básicas.

CSS

 Copiar código

```
/* index.css */ body { margin: 0; font-family: Arial, sans-serif; } header { background-color: #004aad; color: white; padding: 20px; text-align: center; }
```

### Passo 2: Importar o CSS no Componente

Importe o arquivo CSS no componente principal `App.jsx`.

jsx

 Copiar código

```
import React from 'react'; import './index.css'; const App = () => { return ( <header> <h1>Meu Site</h1> </header> ); }; export default App;
```

## Método 2: Styled Components

Styled Components é uma biblioteca que nos permite escrever CSS dentro dos componentes [React](#), criando componentes de estilo.

### Passo 1: Instalar a Biblioteca

Abra o terminal e instale a biblioteca Styled Components.

bash

 Copiar código

```
npm install styled-components
```

### Passo 2: Criar Componentes Estilizados

Vamos criar um componente estilizado para o `Header`.

jsx

 Copiar código

```
import React from 'react'; import styled from 'styled-components'; const HeaderComponent = styled.header` background-color: #004aad; color: white; padding: 20px; text-align: center; `; const Header = () => { return ( <HeaderContainer> <h1>Meu Site</h1> </HeaderContainer> ); }; export default Header;
```

## Utilizando Styled Components em Outros Componentes

Vamos adicionar mais estilos e componentes para mostrar o poder dos Styled Components.

### Header.jsx

jsx

 Copiar código

```
import React from 'react'; import { NavLink } from 'react-router-dom'; import styled from 'styled-components'; const HeaderContainer = styled.header` background-color: #004aad; color: white; padding: 20px; text-align: center; `; const Nav = styled.nav` display: flex; justify-content: center; gap: 20px; `; const StyledLink = styled(NavLink)` color: rgba(255, 255, 255, 0.8); text-decoration: none; &.active, &:hover { color: white; text-decoration: underline; } `; const Header = () => { return ( <HeaderContainer> <h1>Meu Site</h1> <Nav> <StyledLink to="/" exact activeClassName="active">Home</StyledLink> <StyledLink to="/products" activeClassName="active">Produtos</StyledLink> </Nav> </HeaderContainer> ); }; export default Header;
```

## Explicação

- **Styled Components:** Criamos componentes estilizados utilizando a função `styled`.
- **NavLink:** Substituímos o `Link` pelo `NavLink` para aproveitar a classe `active` automática do [React Router](#).
- **Estilos Dinâmicos:** Aplicamos estilos dinâmicos baseados no estado dos links (`hover` e `active`).

## Testando os Estilos

Inicie sua aplicação (`npm start`). Você verá que os estilos são aplicados corretamente e que a navegação entre as páginas funciona, com os links sendo destacados conforme esperado.

## Conclusão

Nesta aula, aprendemos diferentes maneiras de aplicar CSS em uma aplicação [React](#). Exploramos o uso de arquivos CSS tradicionais e a poderosa biblioteca `Styled Components`. Com isso, você tem flexibilidade para escolher a melhor abordagem para seus projetos.

Vejo vocês na próxima aula, onde continuaremos explorando mais funcionalidades do [React](#)!

## 1.1. Vídeo da Aula

dia 21 video 1



## 2. PrimeReact para Estilização

### Introdução

Olá, pessoal! Tudo bem com vocês? Continuando nosso estudo de [React](#), vamos explorar uma nova maneira de lidar com CSS utilizando uma biblioteca chamada PrimeReact. Esta biblioteca é excelente para quem busca componentes pré-estilizados e classes utilitárias para CSS.

### O que é o PrimeReact?

PrimeReact é uma biblioteca que fornece uma ampla gama de componentes já estilizados e prontos para uso. Além disso, ela tem um "irmãozinho" chamado PrimeFlex, que fornece classes utilitárias para estilização, semelhante ao Bootstrap.

### Configuração Inicial do Projeto

Antes de começarmos a usar o PrimeReact, vamos configurar nosso projeto.

#### Passo 1: Limpeza Inicial

Vamos começar limpando o projeto [React](#). Remova os arquivos e importações CSS desnecessários.

1. Remova os arquivos `index.css` e `App.css`:

plaintext



```
src/index.css src/App.css
```

2. Remova as importações de CSS no `index.js` e `App.js`:

jsx



```
// index.js import './index.css'; // Remova esta linha // App.js import './App.css'; // Remova esta linha
```

3. Reinicie o componente App:

jsx



```
// App.js import React from 'react'; const App = () => { return ( <div> <h1>Meu Projeto com PrimeReact</h1> </div> ); }; export default App;
```

#### Passo 2: Instalar as Bibliotecas Necessárias

Vamos instalar as bibliotecas PrimeReact, PrimeFlex, PrimeIcons e Styled Components.

bash



```
npm install primereact primeflex primeicons styled-components
```

#### Passo 3: Importar os Arquivos CSS

Importe os arquivos CSS necessários no `index.js`.

jsx



```
import 'primereact/resources/themes/lara-light-blue/theme.css'; // Tema PrimeReact import 'primereact/resources/primereact.min.css'; // Estilos PrimeReact import 'primeicons/primeicons.css'; // Ícones PrimeReact
```

```
import 'primeflex/primeflex.css'; // PrimeFlex para utilitários CSS
```

## Utilizando o PrimeReact

Vamos adicionar um componente simples, como um botão, para testar se está tudo funcionando corretamente.

### Passo 1: Importar e Usar o Componente de Botão

Adicione um botão do PrimeReact no `App.js`.

jsx

Copiar código

```
import React from 'react'; import { Button } from 'primereact/button'; const App = () => { return ( <div className="p-d-flex p-jc-center p-ai-center" style={{ height: '100vh' }}> <Button label="Clique Aqui" icon="pi pi-check" /> </div> ); }; export default App;
```

### Explicação:

- **Button:** Importamos e usamos o componente de botão do PrimeReact.
- **PrimeFlex Classes:** Utilizamos classes utilitárias do PrimeFlex (`p-d-flex`, `p-jc-center`, `p-ai-center`) para centralizar o botão na tela.

## Testando a Aplicação

Execute sua aplicação com `npm start` e verifique se o botão está centralizado na tela e estilizado conforme o tema escolhido.

## Explorando o PrimeReact

O PrimeReact oferece uma vasta gama de componentes prontos para uso. A seguir, vamos explorar alguns dos componentes mais úteis e como utilizá-los.

### Exemplo 1: Card

Vamos adicionar um componente de cartão para exibir informações.

jsx

Copiar código

```
import React from 'react'; import { Card } from 'primereact/card'; import { Button } from 'primereact/button'; const App = () => { const header = ; const footer = ( <span> <Button label="Salvar" icon="pi pi-check" /> <Button label="Cancelar" icon="pi pi-times" className="p-button-secondary" /> </span> ); return ( <div className="p-d-flex p-jc-center p-ai-center" style={{ height: '100vh' }}> <Card title="Título do Card" subTitle="Subtítulo" style={{ width: '25em' }} header={header} footer={footer}> <p className="p-m-0" style={{ lineHeight: '1.5' }}> Este é um exemplo de conteúdo dentro de um card usando PrimeReact. </p> </Card> </div> ); }; export default App;
```

### Exemplo 2: DataTable

Vamos adicionar um componente de tabela para exibir uma lista de dados.

jsx

Copiar código

```
import React, { useState, useEffect } from 'react'; import { DataTable } from 'primereact/datatable'; import { Column } from 'primereact/column'; const App = () => { const [products, setProducts] = useState([]); useEffect(() => { fetch('https://fakestoreapi.com/products') .then((res) => res.json()) .then((data) => setProducts(data)); }, []); return (
```

```
<div className="p-d-flex p-jc-center p-ai-center" style={{ height: '100vh' }}> <DataTable value={products} paginator rows={5} className="p-datatable-sm"> <Column field="title" header="Produto" /> <Column field="price" header="Preço" /> <Column field="category" header="Categoria" /> </DataTable> </div> ); }; export default App;
```

## Conclusão

Nesta aula, configuramos e utilizamos a biblioteca PrimeReact para estilização e criação de componentes pré-estilizados em nossa aplicação [React](#). No próximo encontro, exploraremos mais componentes e funcionalidades do PrimeReact para melhorar ainda mais nossas aplicações.

Vejo vocês na próxima aula! Até lá!



## 2.1. Vídeo Aula

dia 21 video 2



## 3. PrimeReact: Criando Tela de Login


### Introdução

Olá, pessoal! Tudo bem com vocês? No último encontro, vimos a instalação e a configuração básica das bibliotecas PrimeReact, PrimeIcons e PrimeFlex. Agora, vamos criar uma tela de login e preparar a estrutura para uma navegação condicional, onde o usuário será redirecionado para um dashboard somente após passar pela tela de login.

### Passo 1: Configurando Rotas

Primeiro, precisamos configurar as rotas da aplicação. Vamos instalar o [React Router DOM](#), caso ainda não tenhamos feito isso.

```
bash
```

 Copiar código

```
npm install react-router-dom
```

### Estrutura de Pastas

Crie a seguinte estrutura de pastas dentro do diretório `src`:

```
plaintext
```

 Copiar código


```
src └─ pages └─ Login.jsx └─ Dashboard.jsx └─ routes └─ Paths.jsx └─ App.jsx
```

### Passo 2: Configurando as Rotas em Paths.jsx

Vamos configurar o arquivo `Paths.jsx` para gerenciar as rotas.

#### Paths.jsx

```
jsx
```

 Copiar código


```
import React from 'react'; import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'; import Login from '../pages/Login'; import Dashboard from '../pages/Dashboard'; const Paths = () => { return ( <Router> <Routes> <Route path="/" element={<Login />} /> <Route path="/dashboard" element={<Dashboard />} /> </Routes> </Router> ); }; export default Paths;
```

### Passo 3: Configurando o Componente App.jsx

Atualize o `App.jsx` para incluir o componente `Paths`.

#### App.jsx

```
jsx
```

 Copiar código


```
import React from 'react'; import Paths from '../routes/Paths'; const App = () => { return <Paths />; }; export default App;
```

### Passo 4: Criando a Tela de Login

Agora, vamos criar o componente de login utilizando os componentes do PrimeReact.

#### Login.jsx

```
jsx
```

 Copiar código

```
import React from 'react'; import { InputText } from 'primereact/inputtext'; import { Password } from 'primereact/password';
import { Button } from 'primereact/button'; import { useNavigate } from 'react-router-dom'; import
'primeflex/primeflex.css'; const Login = () => { const navigate = useNavigate(); const handleSubmit = (event) => {
event.preventDefault(); navigate('/dashboard'); }; return ( <div className="p-d-flex p-jc-center p-ai-center" style={{
height: '100vh' }}> <div className="p-card p-shadow-5 p-p-4" style={{ width: '300px' }}> <h3>Seja Bem-vindo</h3> <form
onSubmit={handleSubmit}> <div className="p-field p-mb-3"> <label htmlFor="email">Email</label> <InputText id="email"
type="email" placeholder="email@example.com" className="p-inputtext-sm" /> </div> <div className="p-field p-mb-3"> <label
htmlFor="password">Senha</label> <Password id="password" placeholder="*****" toggleMask feedback={false} /> </div>
<Button label="Entrar" type="submit" className="p-button-sm p-button-primary" /> </form> </div> </div> ); }; export default
Login;
```

## Passo 5: Criando a Tela de Dashboard

Vamos criar um componente simples para o dashboard.

### Dashboard.jsx

jsx

 Copiar código

```
import React from 'react'; const Dashboard = () => { return ( <div className="p-d-flex p-jc-center p-ai-center" style={{
height: '100vh' }}> <h1>Dashboard</h1> </div> ); }; export default Dashboard;
```

## Explicação dos Componentes

- **Login.jsx:** Cria uma tela de login com um formulário que inclui campos de email e senha e um botão de envio. O `handleSubmit` é chamado quando o formulário é enviado, redirecionando o usuário para o dashboard.
- **Dashboard.jsx:** Componente simples que exibe uma mensagem "Dashboard".

## Conclusão

Nesta aula, configuramos as rotas e criamos uma tela de login funcional utilizando os componentes do PrimeReact. No próximo encontro, vamos estilizar nossa tela de login e adicionar mais funcionalidades ao nosso projeto. Até a próxima!

### 3.1. Vídeo da aula

dia 21 video 3



## 4. Estilizando a Tela de Login

### Introdução

Olá, pessoal! Tudo bem com vocês? Continuando nosso estudo de PrimeReact, vamos estilizar a tela de login que criamos anteriormente. Vamos utilizar o PrimeFlex para ajudar na estilização e garantir que nossa aplicação fique bonita e responsiva.

### Estrutura Atual da Tela de Login

Vamos lembrar a estrutura que criamos para a tela de login.

#### Login.jsx

jsx



Copiar código

```
import React from 'react'; import { InputText } from 'primereact/inputtext'; import { Password } from 'primereact/password';
import { Button } from 'primereact/button'; import { useNavigate } from 'react-router-dom'; import
'primeflex/primeflex.css'; const Login = () => { const navigate = useNavigate(); const handleSubmit = (event) => {
event.preventDefault(); navigate('/dashboard'); }; return ( <div className="p-d-flex p-jc-center p-ai-center" style={{
height: '100vh' }}> <div className="p-card p-shadow-5 p-p-4" style={{ width: '300px' }}> <h3>Seja Bem-vindo</h3> <form
onSubmit={handleSubmit}> <div className="p-field p-mb-3"> <label htmlFor="email">Email</label> <InputText id="email"
type="email" placeholder="email@example.com" className="p-inputtext-sm" /> </div> <div className="p-field p-mb-3"> <label
htmlFor="password">Senha</label> <Password id="password" placeholder="*****" toggleMask feedback={false} /> </div>
<Button label="Entrar" type="submit" className="p-button-sm p-button-primary" /> </form> </div> </div> ); }; export default
Login;
```

### Passo 1: Utilizando o PrimeFlex para Estilização

Vamos adicionar classes do PrimeFlex para estilizar melhor nossa tela de login.

#### Estrutura de Login Atualizada

jsx



Copiar código

```
import React from 'react'; import { InputText } from 'primereact/inputtext'; import { Password } from 'primereact/password';
import { Button } from 'primereact/button'; import { useNavigate } from 'react-router-dom'; import
'primeflex/primeflex.css'; const Login = () => { const navigate = useNavigate(); const handleSubmit = (event) => {
event.preventDefault(); navigate('/dashboard'); }; return ( <div className="p-d-flex p-jc-center p-ai-center p-h-screen p-
bg-primary"> <div className="p-card p-shadow-5 p-p-4 p-text-center p-col-12 p-md-4"> <h3 className="p-text-uppercase p-text-
bold p-text-lg">Seja Bem-vindo</h3> <form onSubmit={handleSubmit} className="p-fluid"> <div className="p-field p-mb-3">
<label htmlFor="email" className="p-text-sm p-text-bold p-mb-2">Email</label> <InputText id="email" type="email"
placeholder="email@example.com" className="p-inputtext-sm" /> </div> <div className="p-field p-mb-3"> <label
htmlFor="password" className="p-text-sm p-text-bold p-mb-2">Senha</label> <Password id="password" placeholder="*****"
toggleMask feedback={false} className="p-inputtext-sm" /> </div> <Button label="Entrar" type="submit" className="p-button-sm
p-button-primary p-mt-3" /> </form> </div> </div> ); }; export default Login;
```

### Explicação das Classes Utilizadas

- **p-d-flex**: Aplica display flex.
- **p-jc-center**: Justifica o conteúdo ao centro.
- **p-ai-center**: Alinha itens ao centro verticalmente.
- **p-h-screen**: Define a altura como a altura total da tela.
- **p-bg-primary**: Aplica um fundo de cor primária (a ser definido no tema do PrimeReact).
- **p-card**: Aplica estilos de cartão.
- **p-shadow-5**: Aplica sombra ao elemento.
- **p-p-4**: Aplica padding ao redor do elemento.
- **p-text-center**: Centraliza o texto.
- **p-col-12**: Define a largura como 100% em telas pequenas.
- **p-md-4**: Define a largura como 4 colunas (33%) em telas médias.
- **p-text-uppercase**: Transforma o texto para letras maiúsculas.
- **p-text-bold**: Aplica negrito ao texto.

- **p-text-lg**: Define o tamanho do texto como grande.
- **p-fluid**: Garante que o formulário ocupe 100% da largura do contêiner.
- **p-field**: Aplica estilos de campo de formulário.
- **p-mb-3**: Aplica margem inferior ao elemento.
- **p-text-sm**: Define o tamanho do texto como pequeno.
- **p-mb-2**: Aplica margem inferior pequena ao elemento.
- **p-mt-3**: Aplica margem superior ao botão.

## Verificação da Responsividade

Certifique-se de que a aplicação está responsiva testando em diferentes tamanhos de tela.

## Conclusão

Nesta aula, estilizamos nossa tela de login utilizando classes do PrimeFlex. Nossa tela de login está agora centralizada, estilizada e responsiva. No próximo encontro, continuaremos aprimorando nossa aplicação e explorando mais componentes do PrimeReact.

Até a próxima!

## 4.1. Vídeo Aula

dia 21 video 4



## 5. Interatividade com Input de Senha

### Introdução

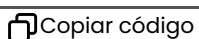
Olá, pessoal! Tudo bem com vocês? Vamos continuar nosso estudo de PrimeReact, adicionando interatividade à nossa tela de login. Hoje, vamos fazer com que o campo de senha mostre e oculte a senha ao clicar no ícone de olho.

### Relembrando a Estrutura Atual do Componente Login

Nossa estrutura de login atual está assim:

#### Login.jsx

jsx



```
import React from 'react'; import { InputText } from 'primereact/inputtext'; import { Password } from 'primereact/password';
import { Button } from 'primereact/button'; import { useNavigate } from 'react-router-dom'; import
'primeflex/primeflex.css'; const Login = () => { const navigate = useNavigate(); const handleSubmit = (event) => {
event.preventDefault(); navigate('/dashboard'); }; return ( <div className="p-d-flex p-jc-center p-ai-center p-h-screen p-
bg-primary"> <div className="p-card p-shadow-5 p-p-4 p-text-center p-col-12 p-md-4"> <h3 className="p-text-uppercase p-text-
bold p-text-lg">Seja Bem-vindo</h3> <form onSubmit={handleSubmit} className="p-fluid"> <div className="p-field p-mb-3">
<label htmlFor="email" className="p-text-sm p-text-bold p-mb-2">Email</label> <InputText id="email" type="email"
placeholder="email@example.com" className="p-inputtext-sm" /> </div> <div className="p-field p-mb-3"> <label
htmlFor="password" className="p-text-sm p-text-bold p-mb-2">Senha</label> <Password id="password" placeholder="*****"
toggleMask feedback={false} className="p-inputtext-sm" /> </div> <Button label="Entrar" type="submit" className="p-button-sm
p-button-primary p-mt-3" /> </form> </div> </div> ); }; export default Login;
```

### Passo 1: Adicionando Interatividade ao Campo de Senha

Vamos adicionar a funcionalidade de mostrar e esconder a senha ao clicar no ícone de olho.

#### Atualização do Componente Login

jsx



```
import React, { useState } from 'react'; import { InputText } from 'primereact/inputtext'; import { Password } from
'primereact/password'; import { Button } from 'primereact/button'; import { useNavigate } from 'react-router-dom'; import
'primeflex/primeflex.css'; const Login = () => { const navigate = useNavigate(); const [mostrarSenha, setMostrarSenha] =
useState(false); const handleSubmit = (event) => { event.preventDefault(); navigate('/dashboard'); }; return ( <div
className="p-d-flex p-jc-center p-ai-center p-h-screen p-bg-primary"> <div className="p-card p-shadow-5 p-p-4 p-text-center
p-col-12 p-md-4"> <h3 className="p-text-uppercase p-text-bold p-text-lg">Seja Bem-vindo</h3> <form onSubmit={handleSubmit}
className="p-fluid"> <div className="p-field p-mb-3"> <label htmlFor="email" className="p-text-sm p-text-bold p-mb-
2">Email</label> <InputText id="email" type="email" placeholder="email@example.com" className="p-inputtext-sm" /> </div>
<div className="p-field p-mb-3"> <label htmlFor="password" className="p-text-sm p-text-bold p-mb-2">Senha</label> <span
className="p-input-icon-right"> <i className={mostrarSenha ? 'pi-eye-slash' : 'pi-eye'} /> </i> </span> <InputText id="password" type={mostrarSenha ? 'text' :
'password'} placeholder="*****" className="p-inputtext-sm" /> </div> <Button label="Entrar" type="submit"
className="p-button-sm p-button-primary p-mt-3" /> </form> </div> </div> ); }; export default Login;
```

### Explicação do Código

- **useState:** Importamos e utilizamos o hook `useState` para gerenciar o estado `mostrarSenha`, que controla se a senha deve ser exibida como texto ou como pontos.
- **Ícone de Olho:** Adicionamos um ícone de olho (ou olho cortado) dentro de um `span` com a classe `p-input-icon-right` para posicionar o ícone à direita do campo de senha.
- **onClick:** Adicionamos um evento `onClick` ao ícone, que alterna o valor de `mostrarSenha` entre `true` e `false`.
- **Tipo do Input:** O tipo do campo de senha é definido condicionalmente como `text` ou `password` com base no estado `mostrarSenha`.



## Testando a Interatividade

Execute sua aplicação (`npm start`) e verifique se o campo de senha alterna corretamente entre mostrar e esconder a senha ao clicar no ícone de olho.

## Conclusão

Nesta aula, adicionamos interatividade ao campo de senha utilizando o PrimeReact e [React](#) hooks. Agora, o usuário pode optar por mostrar ou esconder a senha ao clicar no ícone de olho. No próximo encontro, vamos continuar aprimorando nossa aplicação.

Até a próxima!

## 5.1. Vídeo Aula

dia 21 video 5

