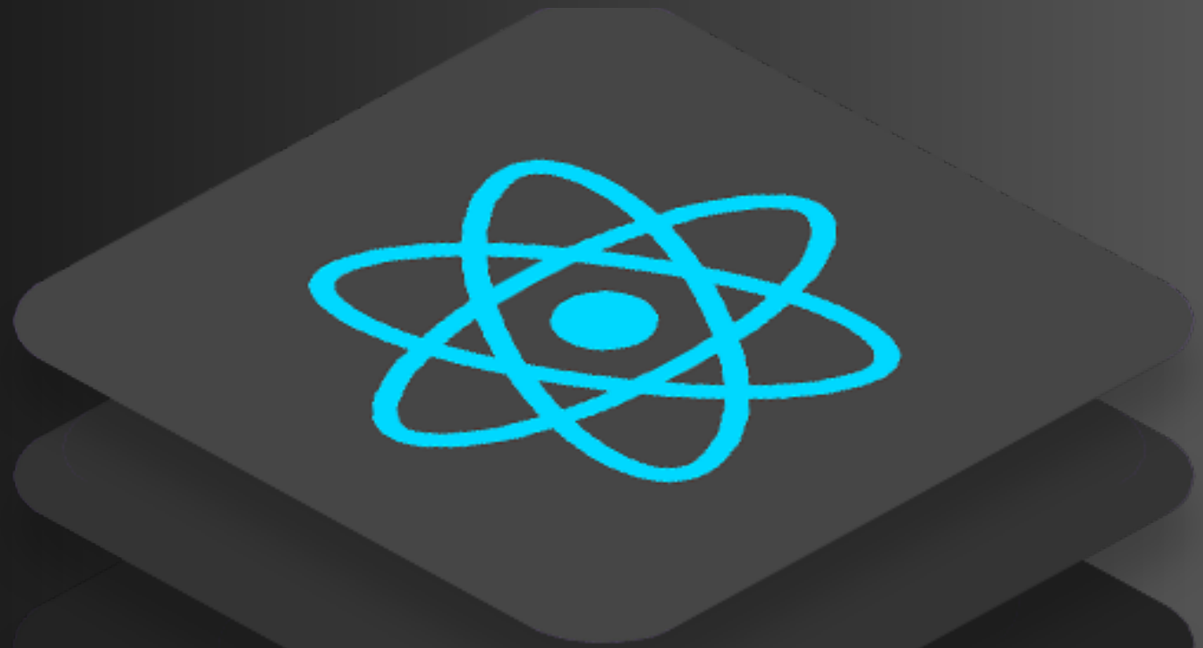


React

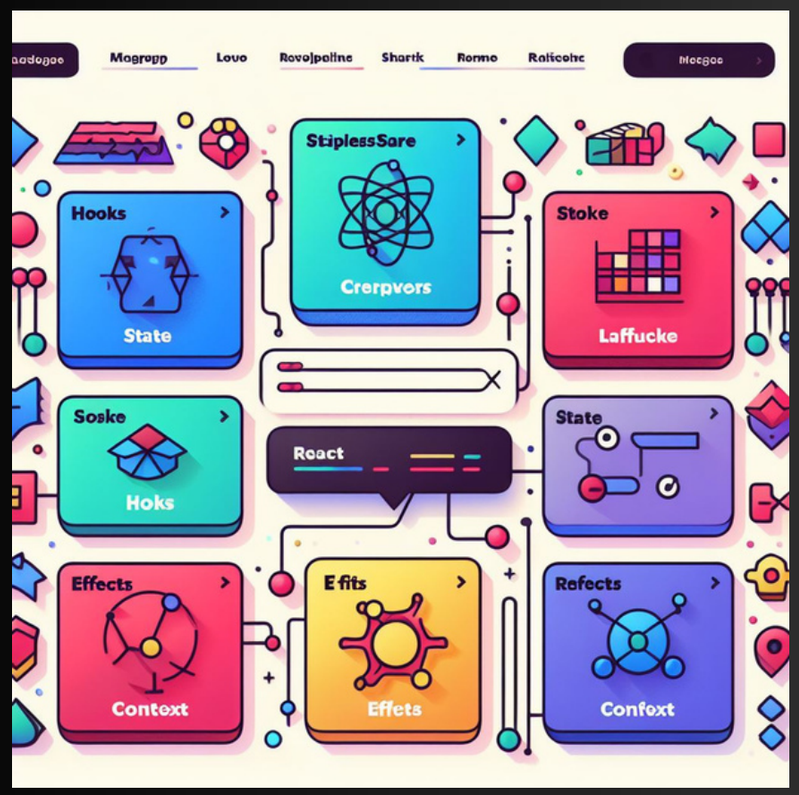
Frontend

O que é o React?

É um framework?

O React é:

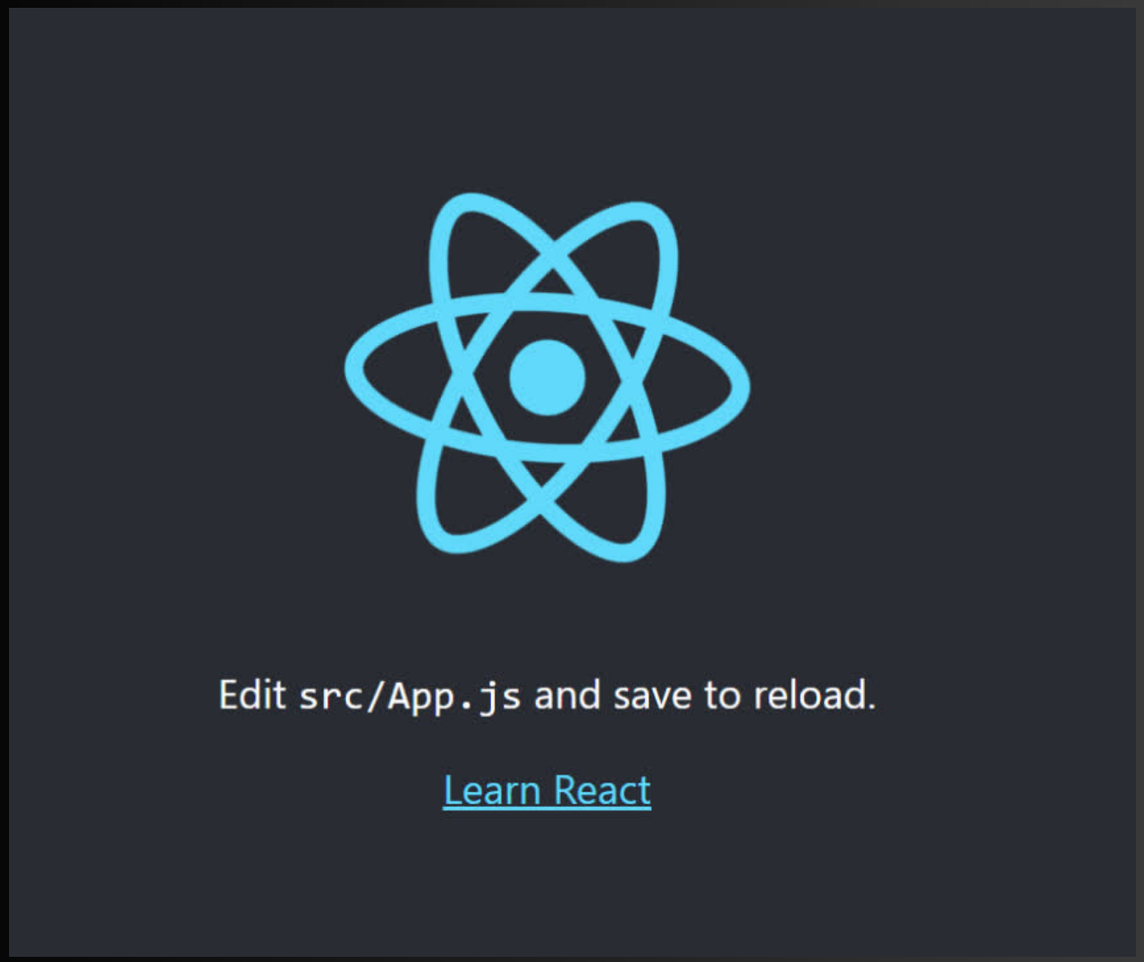
- **Baseado em Componente**

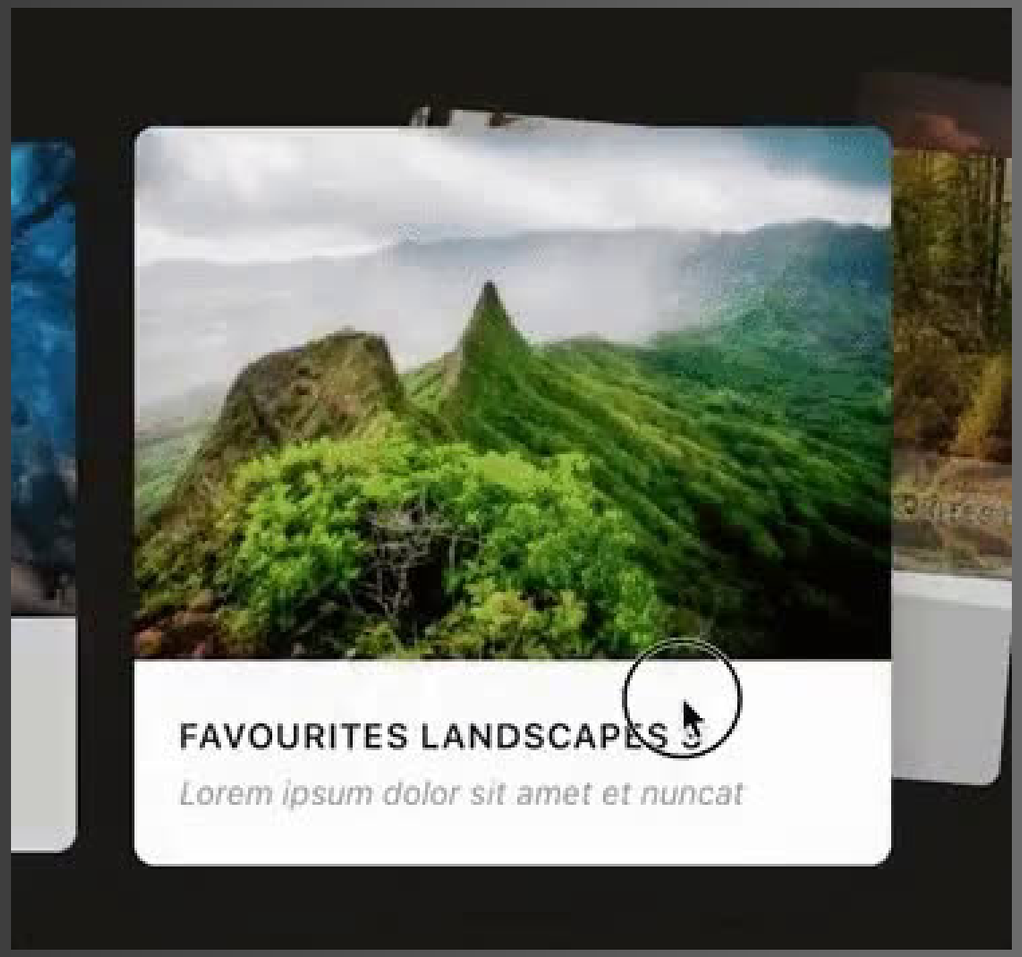


- **Multiplataforma**

O React é:

- **Declarativo - Interfaces Interativas**





Vamos começar?

Instalação e configuração

Pré-requisitos:

Pacote Node instalado com versão compatível

A documentação oficial recomenda utilizar um dos frameworks

O que utilizaremos?

Vite

Pronuncia-se /vit/ . É uma palavra francesa que significa rápido

Promete fornecer uma otimização da importação do ES Module. Através da ferramenta esbuild o servidor do vite verifica os arquivos common js importados e converte-os para ES module

HOT MODULE REPLACEMENT - Através dessa API o desenvolvedor pode ver a aplicação carregar as atualizações do script modificado.

O Vite oferece suporte nativo para importação de arquivos .ts (TypeScript). Entretanto, é importante notar que o Vite realiza apenas a transpilação dos arquivos .ts, sem realizar a verificação de tipos. A verificação de tipos é assumida como responsabilidade do seu ambiente de desenvolvimento (IDE) e do processo de compilação.

Vite

Agrupa seu código com o Rollup: O Rollup é uma ferramenta de empacotamento que combina diferentes módulos do seu código em um único arquivo (bundle). Esse processo de "empacotamento" ajuda a otimizar o desempenho do seu aplicativo.

Post CSS: **F**erramenta de processamento de CSS que permite transformar, estender e otimizar folhas de estilo CSS. No entanto, ao contrário de pré-processadores de CSS como Sass ou Less, o PostCSS não introduz uma nova sintaxe própria; em vez disso, ele utiliza a sintaxe padrão do CSS e aplica transformações por meio de plugins.

Usaremos o Vite

1- Passo: npm create vite@latest

2- Passo:Em project-name, escolha um nome para o seu projeto.(Será o título da pasta do projeto)

3- Passo:Em select a framework, selecione React.

4- Passo: Em select a variant, selecione Javascript+SWC

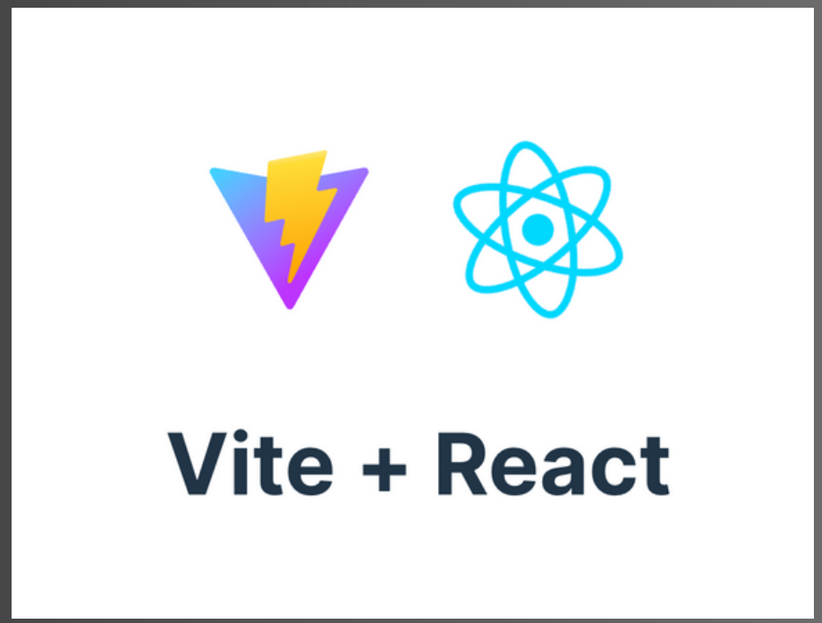
5 - Passo: Entre na pasta(cd Nome-projeto)

6 - Passo: Instale os módulos necessários (npm install)

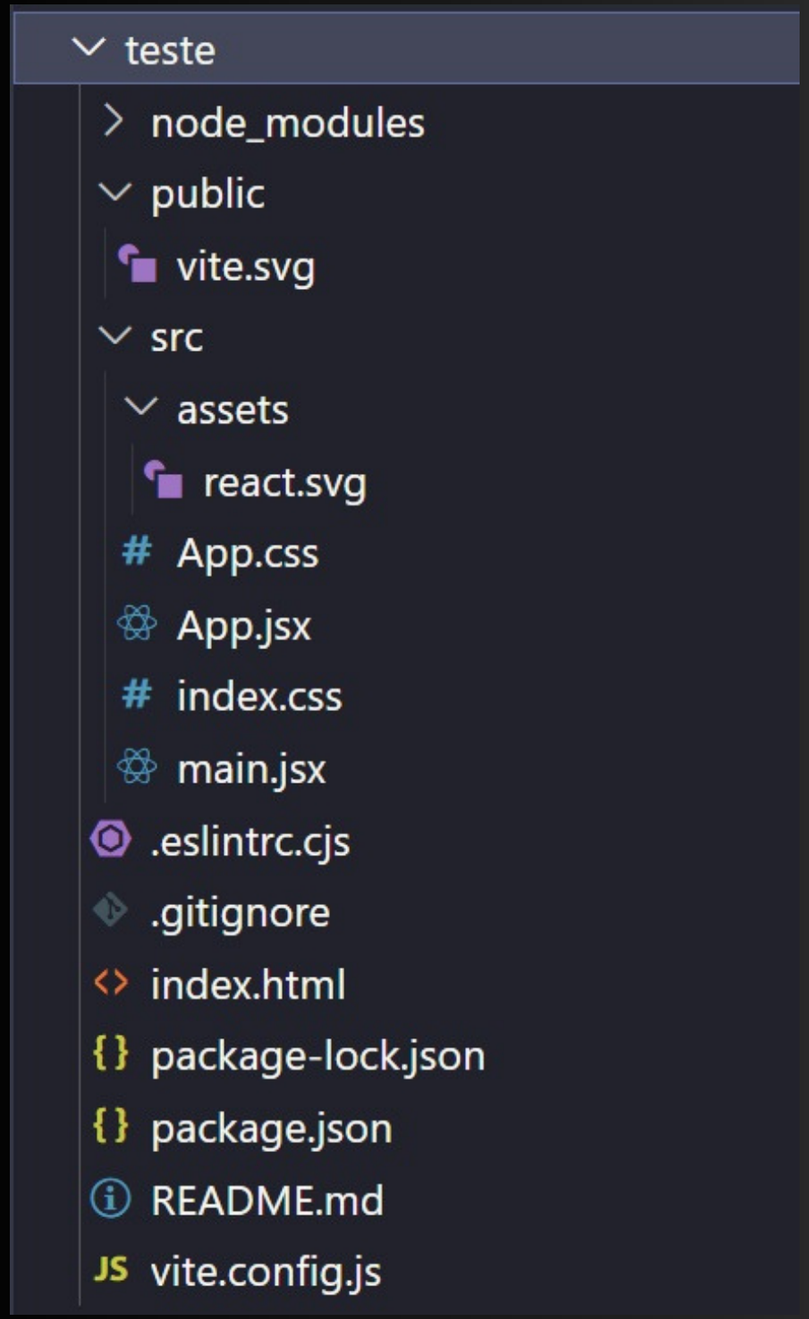
7- Passo: Inicie o projeto(npm run dev)

Site oficial: https://vitejs.dev/guide/

Para a instalação do Vite é necessária versão do 18+.20+ do Node.js



Estrutura das pasta e principais scripts



Pasta Src

Possui os arquivos que modificaremos para construir nosso projeto

Node Modules

Possui os arquivos das bibliotecas de terceiros que serão utilizadas no projeto (Por favor, não modifique nada aqui).

App.jsx

Componente principal onde todos os outros componentes serão aninhados

main.jsx

Script onde é montada a DOM virtual do React

Criando o primeiro componente estilizado

- **JSX**

  É uma extensão de sintaxe para Javascript que permite escrever uma marcação como se fosse HTML dentro de arquivo Javascript

```
const jsx = <h1>This is JSX</h1>
```

Browser não entende isso porque não é código Javascript válido. Esse código associa uma tag HTML e não uma string.

React converte isso a um código javascript que o browser possa entender. (Isso também pode ser feito com ferramentas como o Babel)

Criando o primeiro componente estilizado

- App componente

```
function App() {

    return (
        <div>
            <h1>Primeiro Componente</h1>
            <button>Botão</button>
        </div>
    )
}

export default App
```

App Component

```
function App() {
                                          1
    return (
        <div>
          <h1>Primeiro Componente</h1>
          <button>Botão</button>
        </div>                            2
    )
}

export default App
```

Fique Atento!!

O retorno da função do componente deve ser envolvido por uma marcação

```
function App() {

    return (
            <div>
              <h1>Primeiro Componente</h1>
              <button>Botão</button>
            </div>
        )
}

export default App
```

A tag (marcação) div envolve todos os elementos

```
function App() {

    return (
            <div>
              <h1>Primeiro Componente</h1>
              <button>Botão</button>
            </div>
            <div>
              <h1>Primeiro Componente</h1>
              <button>Botão</button>
            </div>
        )
}

export default App
```

Não possui uma tag única que envolva todas as tags

Fragment

```
function App() {


  return (
    <>
      <div>
        <h1>Primeiro Componente</h1>
        <button>Botão</button>
      </div>

      <div>
      <h1>Primeiro Componente</h1>
      <button>Botão</button>
      </div>
    </>
  )
}

export default App
```

Quando não tiver uma tag com boa semantica para envolver seu componente use um fragment

O fragment é uma tag vazia utilizada para envolver o grupo de elementos.

Aninhando Componentes

App.jsx | App.css | index.css

teste > src > App.jsx > App

```jsx

function NovoComponente(){
  return(
    <main>
      <h1>Aula de React</h1>
      <p>Meu novo component</p>
    </main>
  )
}

function App() {
  return (
    <>
      <NovoComponente/>
    </>
  )
}

export default App

```

0 0 0 Ln 11, Col 17 Spaces: 2 UTF-8 LF {} JavaScript JSX

localhost:5173

Aula de React

Meu novo component

Aninhando Componentes

```jsx
function Header(){
    return(
        <header className={style.header}>
            <img src="https://png.pngtree.com/png-vector/20190710/ourlarge/pngtree-user-vector-avatar-png-
            alt="" className={style.avatar}  />
            <h1>Fulano</h1>
        </header>
    )
}
function Photo(){
    return(
        <img src="https://fatocurioso.com.br/wp-content/uploads/2016/01/curiosidades-da-natureza-1024x768.
        alt=""  className={style.photo}  />
    )
}
function Buttons(){
    return(
        <div className={style.divButton}>
            <button>Like</button>
            <button>Comments</button>
        </div>
    )
}
export default function Card(){
    return(
     <div className={style.card}>
            <Header/>
            <Photo/>
            <Buttons/>
        </div>
    )
}
```

Importação de componentes


```jsx
export default function Card(){
    return(
        <div className={"card"}>
            <Header/>
            <Photo/>
            <Buttons/>
        </div>
    )
}
```

o import será: `import Card from "./components/Card"`

- Costuma-se criar uma pasta components dentro da pasta src
- Dentro da pasta do component o fica oarquivo jsx e oarquivo css
- Faz-se a importação do estilo

```jsx
export function Card(){
    return(
        <div className={"card"}>
            <Header/>
            <Photo/>
            <Buttons/>
        </div>
    )
}
```

o import será: `import {Card} from "./components/Card"`

Propriedades do componente

```jsx
//Componente Título
export  function Titulo(){
    return(
        <h1 className="titulo">
            Titulo do componente
        </h1>
    )
}
//Componente Card
export default function Card(){
    return(
        <section>
            <Titulo/>
        </section>
    )
}
```

```jsx
//Componente Titulo
{
    type:'h1',
    props:{
        className:'titulo',
        children:'Titulo do componente'
    }

}
//Componente Card
{
    type:'section',
    props:{
        children:{
            type:'h1',
            props:{
                className:'titulo',
                children:'Título do componente'
            }
        }
    }
}
```

Propriedades do componente

```jsx
//Componente Titulo
export function Titulo(props) {
    return (
        <h1>
            {props.titulo}
        </h1>
    )
}


//Componente Card
export default function Card(){
    return(
        <section>
            <Titulo titulo="Post Instagram"/>
        </section>
    )
}
```

```jsx
//Componente Titulo
{
    type:'h1',
    props:{
        className:'titulo',
        children:'Post instagram',
        titulo:'Post instagram'
    }

}
//Componente Card
{
    type:'section',
    props:{
        children:{
            type:'h1',
            props:{
                className:'titulo',
                children:'Título do componente',
                titulo:'Post Instagram'
            }
        }
    }
}
```

Vamos à prática!

Criando um componente estilizado

Framework para Estilização

Criar um estilo no frontend muitas vezes pode ser uma tarefa penosa e em projetos que exigem agilidade um framework de estilização de faz necessário

Frameworks populares

Tailwind CSS

Por que utilizar o Tailwind CSS?

- Permite estilizar diretamente no HTML sem ter que criar nome para as variáveis de estilo(Uma das tarefas chatas e difíceis da estilização)

- Diferente do Bootstrap, possui pouca opnião nos estilos e permite criar estílos bem específicos com certo grau de liberdade.

- Remove estilos não utilizados e promete construir um buddle CSS com CSS ao cliente em torno de 10 kB

- Possui boa documentação

- A estilização segue a filosofia de design Mobile First

Instalação do Tailwind CSS no Projeto Vite

Comandos para instalar a biblioteca:

- npm install -D tailwindcss postcss autoprefixer
- npx tailwindcss init -p

Link para documentação:
https://tailwindcss.com/docs/guides/vite

Ajuste no arquivo tailwind.config.js

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Adição das diretivas do Tailwind CSS no index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Padrões de tamanho de tela

sm - 640px

md - 768px

lg - 1024px

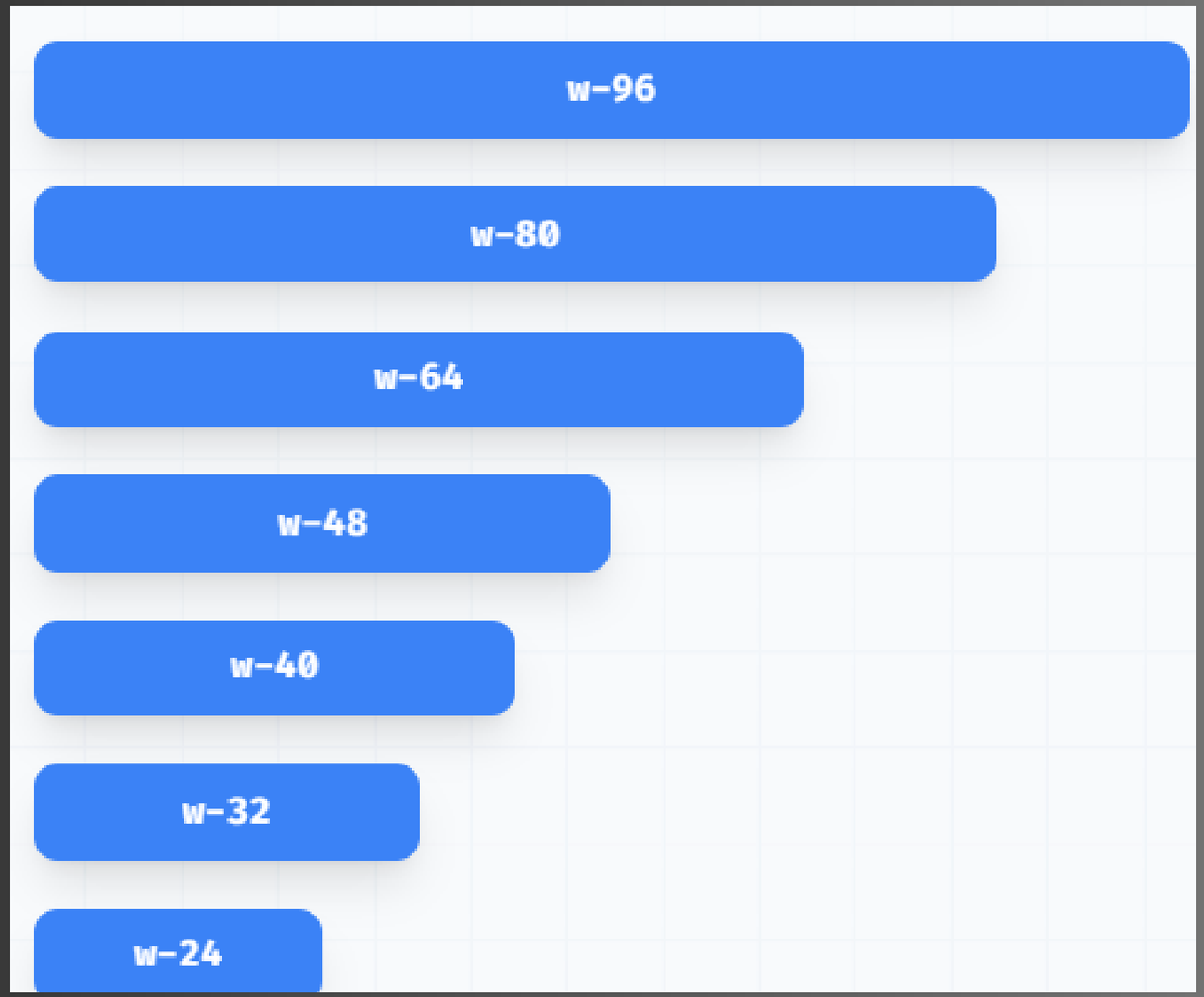
xl - 1280px

2xl - 1536px

Padrões

Tamanhos e Espaçamentos:

- Classes como w-, h-, p-, e m- são usadas para definir largura, altura, preenchimento e margens, respectivamente. Os valores podem ser especificados em pixels, porcentagens ou usando valores relativos.



Vamos estilizar nosso projeto