Tutorial passo a passo — React.js com Vite (para iniciantes)

adaptado do chatgpt

Esse tutorial aborda de maneira prático como criar seu primeiro projeto React usando **Vite**, criar um componente simples e entender o fluxo básico.

O que é o Reactjs: https://pt-br.react.dev/

1) Pré-requisitos

Node.js (recomendado: versão LTS atual — por exemplo 18.x ou 20.x). Verifique com:

```
node -v
npm -v
```

- Um terminal (Windows: PowerShell / Terminal; macOS/Linux: Terminal)
- Editor: VS Code.

Se não tiver Node instalado, baixe do site oficial ou use um gerenciador de versões (nvm).

2) Criando o projeto com Vite

Abra o terminal e execute (usando `npm`, `yarn` ou `pnpm`):

Com npm:

```
# cria pasta "meu-app"
npm create vite@latest meu-app
```

O `create vite` perguntará:

- nome do projeto (ex: `meu-app`)
- framework → escolha `react`
- variante → `react` (JS) ou `react-ts` (TypeScript)

Alternativa rápida (não interativa) para React JS:

```
npm create vite@latest meu-app -- --template react
```

Para TypeScript:

```
npm create vite@latest meu-app -- --template react-ts
```

Depois entre na pasta e instale dependências:

```
cd meu-app
npm install
```

Inicie o servidor de desenvolvimento:

```
npm run dev
```

Abra o endereço mostrado no terminal (normalmente `http://localhost:5173`).

Estrutura inicial do projeto

```
None
meu-app/
├─ index.html
├─ package.json
├─ tsconfig.json
├─ vite.config.ts
├─ src/
│  ├─ main.tsx        # ponto de entrada da aplicação
│  ├─ App.tsx         # componente principal
│  ├─ assets/
│  └─ styles/
```

Falar da pasta node_modules e do arquivo .gitignore

- `main.tsx` → monta o React na DOM.
- `App.tsx` → componente raiz.
- `index.html` → arquivo base.
- `vite.config.ts` → configuração do Vite.

Entendendo o ponto de entrada (`main.tsx`)

```
None
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import './index.css'

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

- O `!` depois de `getElementById('root')!` indica ao TypeScript que o elemento **não é nulo**.

Criando o primeiro componente (`App.tsx`)

Abra `src/App.tsx` e substitua por:

```
None

export default function App() {
  return (
    <div>
      <h1>Meu Primeiro Projeto</h1>
    </div>
  )
}


/*********************************
Adicionando o primeiro componente: App.tsx
*********************************/
```

```
  export default function App() {
    return (
      <div>
        <h1>Meu Primeiro Projeto</h1>
        <Aluno />
      </div>
    )
  }

  function Aluno () {
    return (
      <h2>João Costa</h2>
    )
  }
```

```
/*******************************/
Trabalhando com parâmetros: App.tsx
/*******************************/
```

```
export default function App() {
  return (
    <div>
      <h1>Meu Primeiro Projeto</h1>
      <Aluno nome="João Carlos"/>
      <Aluno nome="Maria Silva"/>
      <Aluno nome="Pedro Santos"/>
    </div>
  )
}

interface AlunoProps {
  nome: string
}
```

```
function Aluno ({nome}: AlunoProps) {
  return (
    <h2>Aluno: {nome}</h2>
  )
}
```

```
/*******************************/
Trabalhando com parâmetros: App.tsx
/*******************************/
export default function App() {
  return (
    <div>
      <h1>Meu Primeiro Projeto</h1>
      <Aluno nome="João Carlos" matricula={123} />
      <Aluno nome="Maria Silva" matricula={223} />
      <Aluno nome="Pedro Santos" matricula={323}/>
    </div>
  )
}
interface AlunoProps {
  nome: string
  matricula: number
}

function Aluno ({nome, matricula}: AlunoProps) {
  return (
    <div>
      <h2>Aluno: {nome}</h2>
      <p>Matricula: {matricula}</p>
    </div>
  )
}
```

Criando um componente personalizado (`header.tsx`)

Crie a pasta `src/components` e adicione o arquivo:

`src/components/header.tsx`

```tsx
export function Header() {
  return (
    <header>
      <h1>Bem-vindo ao Sistema de Alunos</h1>
      <hr />
    </header>
  )
}
```

Adicione a seguinte linha no início do arquivo **App.tsx**

```tsx
import { Header } from './components/header.tsx'
```

Estilos do componente

Crie o arquivo `src/components/header.css`:

```css
.header{
    width: 100%;
    height: 100px;
    background-color: #0c3c6d   ;
    padding: 20px;
    text-align: center;
}

.title{
    color: #f7f8f9;
    text-align: center;
    padding-top: 10px;
    font-family: Arial, sans-serif;
}
```

Depois, no arquivo `header.tsx` **importe** esse arquivo css

```
import './header.css'

export function Header() {
  return (
    <header className="header"accordion-header>
      <h1 className="title">Bem-vindo ao Sistema de Alunos</h1>
      <hr />
    </header>
  )
}te recarrega automaticamente (HMR) assim que salvar.
```

Exercício: Separe o componente **aluno** num arquivo aluno.tsx

Crie o arquivo aluno.tsx

```
interface AlunoProps {
  nome: string
  matricula: number
}

export function Aluno ({nome, matricula}: AlunoProps) {
  return (
    <div>
      <h2>Aluno: {nome}</h2>
      <p>Matricula: {matricula}</p>
    </div>
  )
}
```

Despoi no arquivo App.tsx importe aluno.tsx

```
import { Header } from './components/header.tsx'
import { Aluno } from './components/aluno.tsx'

export default function App() {
  return (
    <div>
      <Header />
      <Aluno nome="João Carlos" matricula={123} />
      <Aluno nome="Maria Silva" matricula={223} />
```

```
      <Aluno nome="Pedro Santos" matricula={323}/>
    </div>
  )
}
```

Por fim, no arquivo header.tsx torne a propriedade **`title`** opcional

```
import './header.css'

interface HeaderProps {
    title?: string
}

export function Header({title = "Bem-vindo ao Sistema de Alunos"}:
HeaderProps) {
  return (
    <header className="header">
      <h1 className="title">{title}</h1>
      <hr />
    </header>
  )
}
```