

FIAP

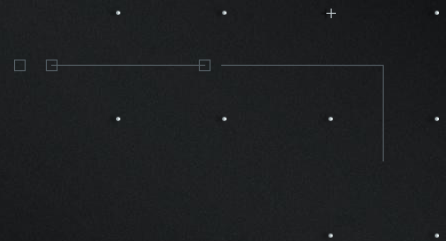
FIAP

SLIDER ▢■◀





# React



# O que é React

- O React é uma biblioteca Javascript criada pelo Facebook, utilizada para criar interfaces para usuários, com uma particularidade de renderizar somente a parte da tela que é necessária, aumentando assim em muito a performance da página.
- É de código aberto e quem mantém e evolui o React é você, ou seja, a comunidade.
- Seu foco principal é transformar a experiência do usuário mais eficiente, tornando a aplicação mais leve e performática, permitindo a reusabilidade de componentes.
- É uma biblioteca da linguagem JavaScript e que atua na camada View de um projeto com MVC.
- <https://react.dev/reference/react>

# O que é React

- No React tudo é Javascript, até os elementos HTML são criados por ele através do JSX, uma extensão de sintaxe que nos permite trazer a criação dos elementos HTML para dentro do Javascript.
- Para que as mudanças dos componentes da tela sejam harmoniosas, ele utiliza o Virtual DOM (VDOM) que gerencia os componentes em memória e sincroniza com o DOM real, utilizando a biblioteca do ReactDOM. Isso aumenta a performance, melhorando até a classificação nos motores de busca.
- Lembrando que não trabalharemos com Javascript e sim com **TypeScript**.

# O que é React

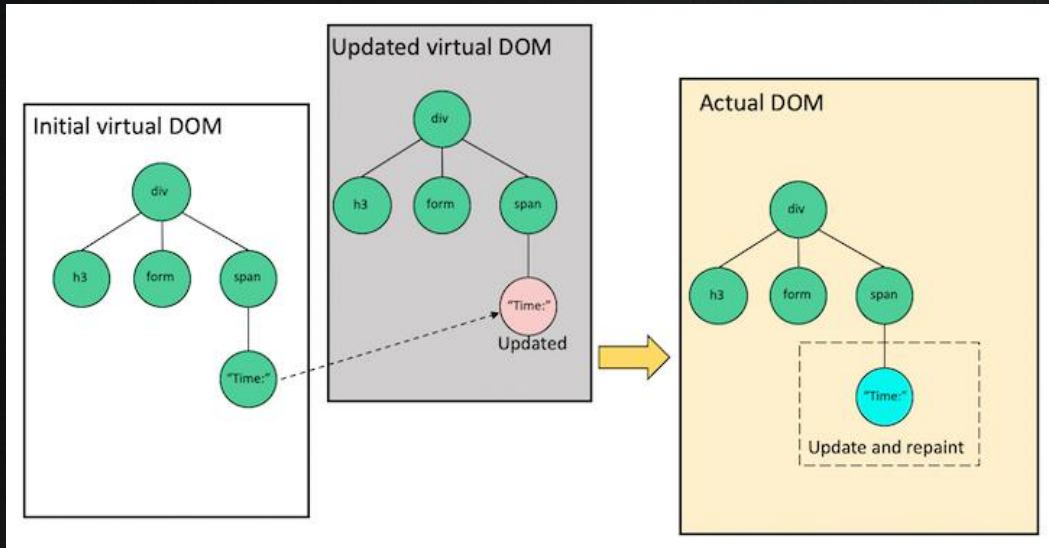
DOM – Manipulação dos dados HTML em tempo real com o React

- É possível alterar qualquer texto das tags do HTML.
- Os atributos do HTML, também podem ser alterados pelo React.
- Possibilidade de remover e adicionar as tags do HTML, através do React.
- Tem o mesmo nível de ação no HTML e CSS.
- O React é muito utilizado com o HTML, através dos eventos, um exemplo muito conhecido é o onclick do HTML.



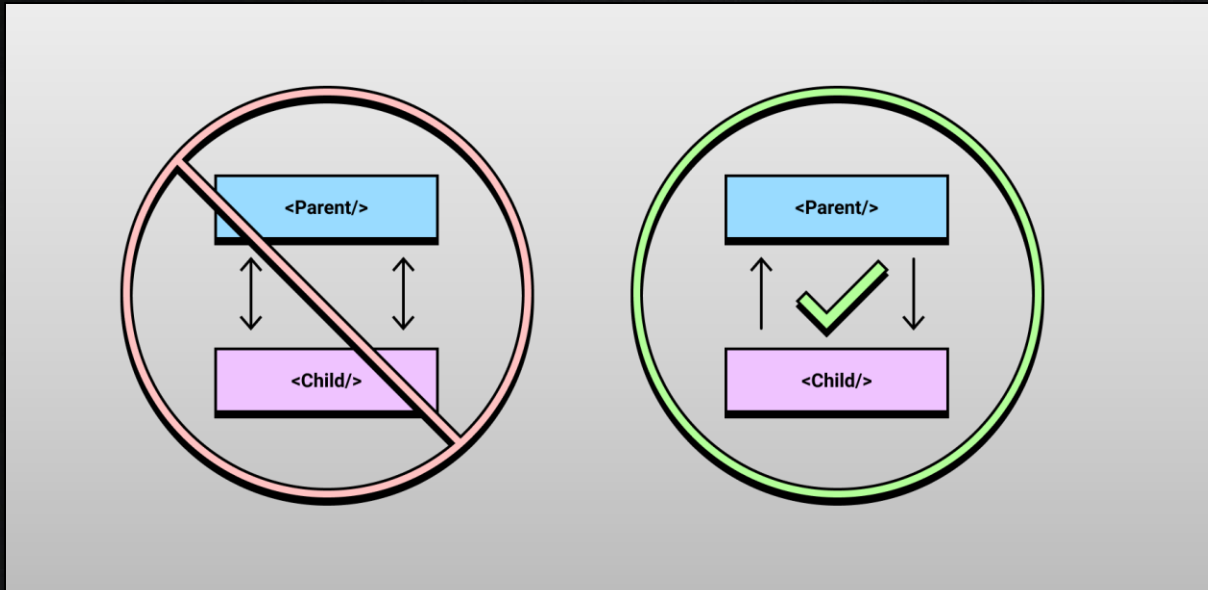
# O que é React

O React, por trabalhar com componentes, só precisa recarregar a parte da página que foi alterada, mantendo as demais partes, deixando o trabalho mais rápido.



# O que é React

Além disso o React trabalha com um fluxo unidirecional, em um único sentido (One-Way Data flow). As informações sempre vem do componente pai (Parent) para o filho (Child)



# O que é React

## Pontos positivos

- Muita eficiência na utilização dinâmica entre o JS e o HTML, refletindo no interface para o usuário (UI).
- Sabendo a sintaxe e lógica do JS, torna fácil a utilização do React.
- Mais entrega com menos esforço para o UI.

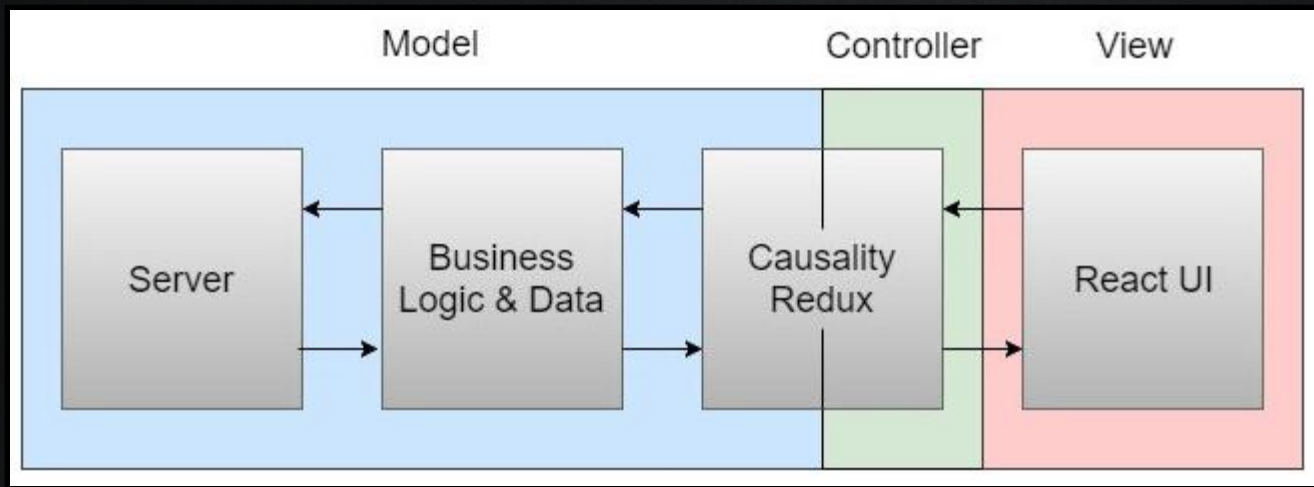
## Pontos negativos

- Trabalha apenas na camada da View.
- Para a utilização é necessário conhecimento prévio em JS, na sintaxe, lógica de programação, HTML e CSS.



# O que é React

Um exemplo é por exemplo na UI, se tem um botão que ao ser clicado envia para um controller (lembrando do conceito de JSP podemos lembrar do package servlet) que então aciona o código backend da aplicação.



# JSX/TSX

**JSX (JavaScript XML)** é uma extensão de sintaxe para JavaScript, popularizada pelo React, que permite escrever código que se assemelha a HTML dentro do JavaScript. Embora pareça HTML, JSX é convertido em chamadas de função JavaScript que criam elementos React. Isso permite que você construa interfaces de usuário de forma declarativa e intuitiva.

**TSX (TypeScript XML)** é uma extensão de sintaxe para TypeScript, semelhante ao JSX, mas adiciona suporte para tipos estáticos. Isso permite que você escreva componentes React com a segurança adicional da verificação de tipos do TypeScript, o que pode ajudar a evitar muitos erros comuns e melhorar a robustez do seu código

```
import React from 'react';

function App() {
  return (
    <div>
      <h1>Olá, Mundo!</h1>
      <p>Este é um parágrafo.</p>
    </div>
  );
}

export default App;
```

# Biblioteca x Framework

## O que é uma biblioteca?

Uma biblioteca é uma coleção de código reutilizável que fornece funcionalidades específicas para desenvolvedores. Ela funciona como um conjunto de ferramentas que podem ser usadas para realizar tarefas comuns em um programa

## O que é um Framework?

Um framework é uma estrutura completa que fornece um ponto de partida para o desenvolvimento de aplicações. Ele define a arquitetura geral da aplicação e oferece ferramentas e recursos para implementar as funcionalidades desejadas.

# Biblioteca x Framework

Característica	Biblioteca	Framework
Nível de controle	Alto	Baixo
Flexibilidade	Mais flexível	Menos flexível
Curva de aprendizado	Baixa	Mediana
Agilidade inicial	Média	Alta

# Frameworks - React

## NextJS

é uma estrutura React que oferece uma série de recursos que o tornam ideal para construir aplicativos da web renderizados em servidor e gerados estaticamente. Ele também oferece vários outros recursos, como divisão automática de código e otimização de imagem, que podem ajudar a melhorar o desempenho de seus aplicativos.



# Frameworks - React

## Vite

é uma ferramenta de construção e servidor de desenvolvimento projetado para ser rápido e eficiente. Ele pode ser usado com qualquer estrutura ou biblioteca JavaScript e oferece vários recursos que o tornam ideal para o desenvolvimento de aplicativos grandes e complexos.

# Frameworks - React

Recurso	Vite.js	Next.js
Ferramenta de construção	Sim	Sim
Servidor de desenvolvimento	Sim	Sim
Estrutura React	Não	Sim
Renderização do lado do servidor	Sim	Sim
Geração de site estático	Sim	Sim
Divisão automática de código	Sim	Sim
Otimização de imagem	Sim	Sim
Ecossistema	Menor	Maior

# Frameworks - React

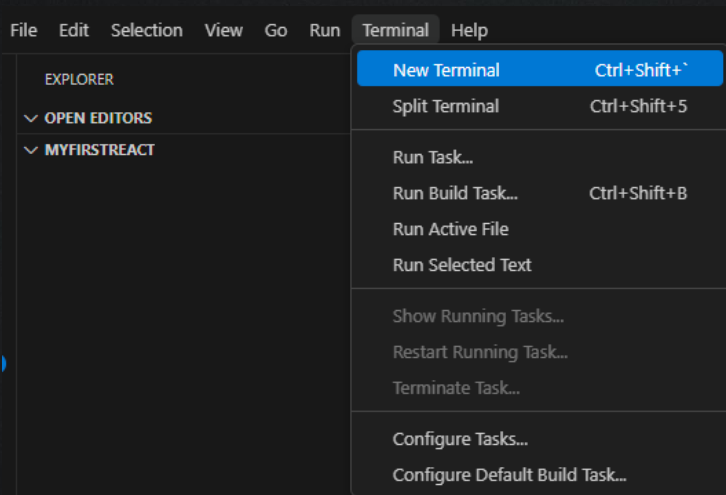
## Qual escolher?

- Se você está procurando uma ferramenta de construção e servidor de desenvolvimento rápido e eficiente, o Vite é uma ótima opção. Também é uma boa opção se você deseja ter mais controle sobre o processo de desenvolvimento.
- Se você está procurando uma estrutura React que ofereça vários recursos para construir aplicativos da web renderizados em servidor e gerados estaticamente, Next.js é uma boa escolha. Também é uma boa escolha se você deseja usar uma estrutura com um grande ecossistema e uma grande comunidade.
- Em última análise, a melhor maneira de decidir qual estrutura é a certa para você é considerar suas necessidades e requisitos específicos.
- Num primeiro momento trabalharemos com o Vite.js e posteriormente com o Next.js

# Criando uma aplicação React com Vite

Vamos usar o Vite para criar uma aplicação React para agilizar as coisas. Também poderíamos criar manualmente, instalando uma dependência de cada vez como vimos na aula introdutória de NodeJS e Typescript, mas o Vite fornece tudo pronto.

1. Crie uma pasta com o nome que desejar, só cuidado para não abrir o Desktop diretamente
2. Abra ela no seu VSCode
3. Abra o terminal integrado



# Criando uma aplicação React com Vite

4. No terminal vamos digitar o comando `npm create vite@latest` e dar Enter para prosseguir

```
(base) PS C:\Users\vinix\Desktop\MyFirstReact> npm create vite@latest
Need to install the following packages:
  create-vite@5.3.0
Ok to proceed? (y) ☐
```

5. Agora vamos escolher o nome do nosso projeto

```
> npx
> create-vite

? Project name: » my-first-react-app 
```

6. Na lista de Frameworks vamos escolher o React

```
? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
>  React
  Preact
  Lit
  Svelte
  Solid
  Quik
  Others
```



# Criando uma aplicação React com Vite

7. Na lista de variant, vamos escolher TypeScript + SWC. O SWC (Speedy Web Compiler) é um compilador super rápido escrito em Rust. A promessa é ele ser até 20x mais rápido que o compilador do webpack padrão

```
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
> TypeScript + SWC
  JavaScript
  JavaScript + SWC
  Remix ↗
```

8. Agora devemos receber uma mensagem de sucesso, com instruções iniciais

```
Scaffolding project in C:\Users\vinix\Desktop\MyFirstReact\my-first-react-app...
```

```
Done. Now run:
```

```
cd my-first-react-app
npm install
npm run dev
```

```
(base) PS C:\Users\vinix\Desktop\MyFirstReact> 
```

# Criando uma aplicação React com Vite

9. Vamos entrar na pasta, com o comando `cd nome-do-projeto`. No meu caso ficou `cd my-first-react-app`

10. Dentro da pasta vamos executar o comando `npm install` e aguardar a instalação dos pacotes. Se tudo der certo a saída do terminal deve ser semelhante a da imagem abaixo

```
• (base) PS C:\Users\vinix\Desktop\MyFirstReact> cd .\my-first-react-app\  
• (base) PS C:\Users\vinix\Desktop\MyFirstReact\my-first-react-app> npm install  
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.  
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported  
npm warn deprecated @humanwhocodes/config-array@0.11.14: Use @eslint/config-array instead  
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported  
npm warn deprecated @humanwhocodes/object-schema@2.0.3: Use @eslint/object-schema instead  
  
added 158 packages, and audited 159 packages in 19s  
  
38 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
○ (base) PS C:\Users\vinix\Desktop\MyFirstReact\my-first-react-app> []
```

# Criando uma aplicação React com Vite

11. Agora vamos digitar o comando `npm run dev`

```
VITE v5.3.3 ready in 247 ms
```

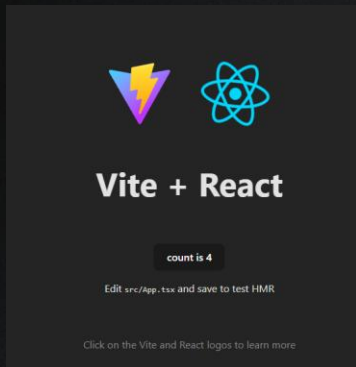
```
→ Local:   http://localhost:5173/
```

```
→ Network: use --host to expose
```

```
→ press h + enter to show help
```

```
█
```

12. Vamos abrir a URL em um navegador (ou segurar Control + click em cima da URL na saída do comando)



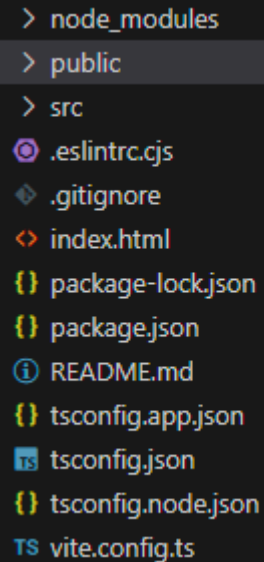
# Entendendo a estrutura do projeto

## node\_modules

Contém todas as bibliotecas instaladas, como já vimos na aula de Node e TypeScript. Raramente você precisará alterar qualquer coisa aqui.

## public

A pasta pública contém arquivos estáticos, arquivos de biblioteca javascript, imagens e outros ativos, etc. que você não deseja que sejam processados pelo comando de build. Os arquivos nesta pasta são copiados e colados conforme estão diretamente na pasta de construção. Apenas arquivos dentro da pasta **public** podem ser referenciados a partir do HTML.



```
node_modules
> public
src
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts
```

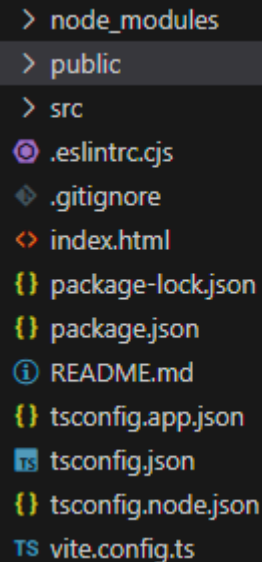
# Entendendo a estrutura do projeto

## src

A pasta src é onde nós iremos ficar a maior parte do nosso tempo, é onde o código fonte da nossa aplicação vive. A maior parte do trabalho que você fizer será aqui.

## .eslintrc.cjs

O arquivo `.eslintrc.cjs` é um arquivo de configuração do **ESLint**. O ESLint é uma ferramenta que ajuda a manter o código JavaScript/TypeScript limpo e consistente. O arquivo `.eslintrc.cjs` define as regras e configurações para a análise estática do código.



```
> node_modules
> public
> src
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts
```



# Entendendo a estrutura do projeto

## .gitignore

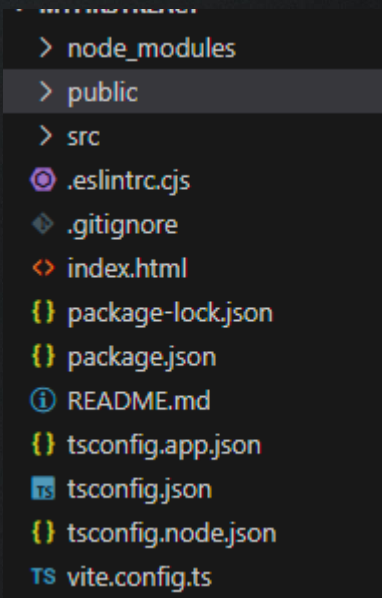
O arquivo `.gitignore` é um arquivo de texto que indica ao Git quais arquivos ou pastas ignorar num projeto. Um arquivo `.gitignore` local é geralmente colocado no diretório raiz de um projeto.

## index.html

O arquivo `index.html` desempenha um papel crucial como ponto de entrada e elemento central da aplicação. Ele vai além de ser um simples arquivo HTML estático, assumindo diversas responsabilidades importantes.

Se analisarmos ele, veremos que ele está importando o nosso `main.tsx` (vamos ver mais tarde do que se trata) e também tem uma div `<div id="root"></div>` que é onde o React vai inicializar a nossa aplicação.

Cuidado ao mexer nesse arquivo. Mas por hora já podemos mudar o nosso `<title>`



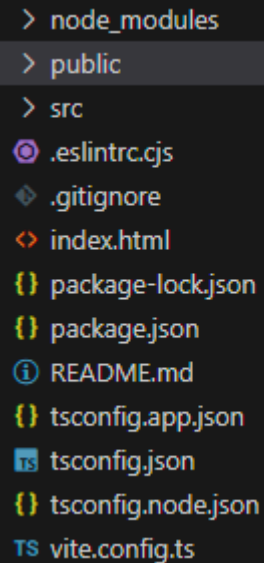
# Entendendo a estrutura do projeto

## README.md

É um arquivo markdown que contém muitas informações úteis sobre o projeto. O Vite já deixa algumas coisas importantes aqui, e é uma boa prática atualizar esse arquivo com passos básicos de documentação. Como rodar o projeto, o que é o projeto, etc.

## tsconfig.json/tsconfig.node.json/tsconfig.app.json

São os arquivos de configuração do TypeScript que definem as opções de compilação para o seu código, como vimos anteriormente na aula de TypeScript. A diferença aqui é que o nosso arquivo principal (tsconfig.json) está importando os outros dois.



```
> node_modules
> public
> src
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts
```

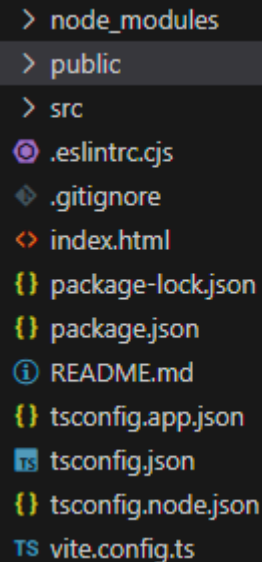
# Entendendo a estrutura do projeto

## package-lock.json

Arquivo importante do NPM para manter as versões das bibliotecas instaladas, como vimos anteriormente na aula de NodeJS. Ele funciona como um registro detalhado e imutável das versões exatas de cada dependência instalada em um projeto, garantindo reprodutibilidade e consistência ao longo do tempo.

## vite.config.ts

Esse arquivo é usado para configurar o Vite. Ele pode conter configurações relacionadas a plugins, roteamento, aliases de importação, entre outras coisas.



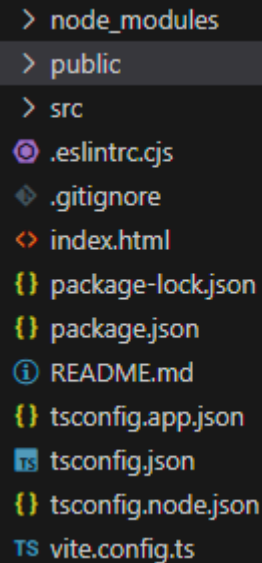
```
node_modules
public
src
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts
```

# Entendendo a estrutura do projeto

## package.json

é uma espécie de manifesto do seu projeto. Ele pode fazer uma variedade de coisas, completamente não relacionadas. É um repositório central de configurações para ferramentas, por exemplo. Também é onde o npm e o yarn armazenam os nomes e versões de todos os pacotes instalados.

Ver mais em <https://nodejs.reativa.dev/0019-package-json/index>



```
node_modules
public
src
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts
```



# Entendendo a estrutura do projeto – pasta src

## assets

é usada para armazenar arquivos estáticos que são importados diretamente no código da aplicação. Esses arquivos podem incluir imagens, fontes, ícones, arquivos de áudio e outros recursos que são necessários para a aplicação e são processados pelo bundler (no caso, o Vite).

```
▼ src
  ▼ assets
    react.svg
    # App.css
    ⚙ App.tsx
    # index.css
    ⚙ main.tsx
    TS vite-env.d.ts
```

Característica	src/assets	public
Importação	Importados diretamente no código	Acessados diretamente pela URL
Processamento	Processados pelo Vite (minificação, hash)	Não processados pelo Vite
Inclusão no Bundle	Incluídos no bundle final	Copiados diretamente para o dist
Uso Comum	Imagens, fontes, ícones usados em componentes React	Favicon, manifestos, arquivos acessíveis diretamente no HTML



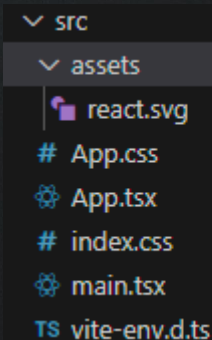
# Entendendo a estrutura do projeto – pasta src

## App.tsx

Responsável por criar e receber os dados para o arquivo .html para projetar no browser, utilizando a index.tsx para esse envio e recebimento dos dados. É o componente inicial do React, o ponto de partida

## App.css

A forma de estilização de componentes é muito parecida com a que utilizamos nos projetos sem o React. Podemos ter arquivos de estilização CSS dedicados a um ou vários componentes. Por boa prática, colocamos os nomes dos arquivos CSS iguais aos do componente.

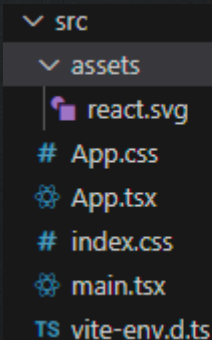


```
▼ src
  ▼ assets
    react.svg
  # App.css
  ⚙ App.tsx
  # index.css
  ⚙ main.tsx
  TS vite-env.d.ts
```

# Entendendo a estrutura do projeto – pasta src

## index.css

Responsável pela estilização global da nossa aplicação. Aqui os estilos vão ser aplicados a todos os componentes. É importado dentro do main.tsx



```
▼ src
  ▼ assets
    react.svg
  # App.css
  ⚙ App.tsx
  # index.css
  ⚙ main.tsx
  TS vite-env.d.ts
```

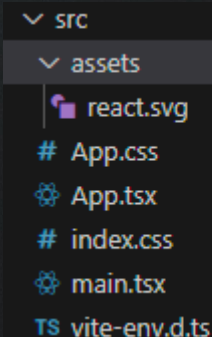
A screenshot of a file explorer interface showing the contents of the 'src' directory. The 'src' directory is expanded, showing a subdirectory 'assets' which contains 'react.svg'. Below 'assets', there are five files: 'App.css' (marked with a hash #), 'App.tsx' (marked with a gear icon), 'index.css' (marked with a hash #), 'main.tsx' (marked with a gear icon), and 'vite-env.d.ts' (marked with 'TS').

# Entendendo a estrutura do projeto – pasta src

## main.tsx

Arquivo de inicialização do React, importado no nosso index.html. Aqui inicializamos o React, importamos nosso componente principal (App.tsx) e os estilos globais (index.css)

```
1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import App from './App.tsx'
4  import './index.css'
5
6  ReactDOM.createRoot(document.getElementById('root')!).render(
7    <React.StrictMode>
8      <App />
9    </React.StrictMode>,
10  )
11
```

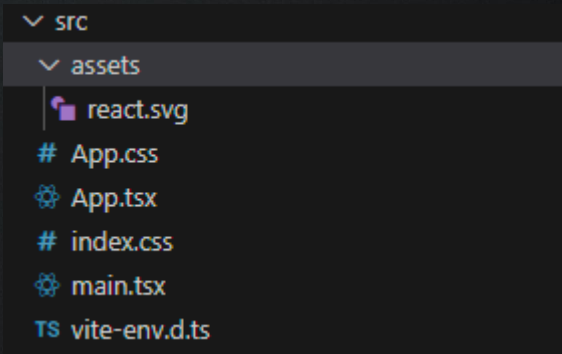


```
▼ src
  ▼ assets
    react.svg
    # App.css
    App.tsx
    # index.css
    main.tsx
    TS vite-env.d.ts
```

# Entendendo a estrutura do projeto – pasta src

## vite-env.d.ts

O arquivo vite-env.d.ts desempenha um papel crucial na definição e tipagem de variáveis de ambiente em aplicações React + TypeScript + Vite. Ele permite que você utilize IntelliSense do TypeScript para as variáveis de ambiente, facilitando o desenvolvimento e evitando erros de digitação.



# Componentização

- Componentização permitem você dividir a UI em partes independentes, reutilizáveis, ou seja, trata cada parte da aplicação como um bloco isolado, livre de outras dependências externas.
- Componentização são como funções JavaScript.
- A componentização é um conceito fundamental no desenvolvimento de software moderno, permitindo a criação de código modular e reutilizável. Quando combinada com o poderoso ecossistema do React e a segurança adicional proporcionada pelo TypeScript, essa abordagem torna-se ainda mais eficaz.
- A componentização é a prática de dividir uma aplicação em partes menores e independentes. Cada componente é responsável por uma parte específica da lógica ou da interface do usuário, facilitando a manutenção, o teste e a reutilização do código. No contexto do React, os Componentes são blocos de construção fundamentais



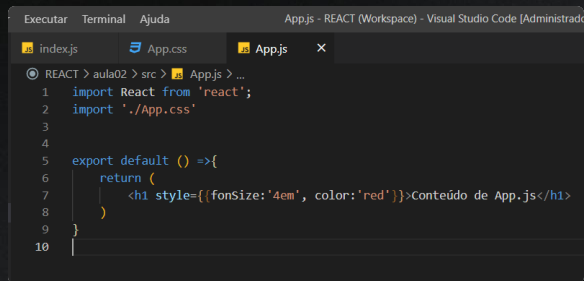
# Estilização

A estilização no REACT acontece de forma muito semelhante ao que estamos acostumados no HTML. Podemos utilizar os três tipos de estilização:

- Externo
- Incorporado e
- Inline

Temos duas grandes diferenças ao HTML padrão:

1. A propriedade **class** no React deve ser utilizada como **className**
2. Quando vamos utilizar a estilização **inline**, devemos usar um Objeto de Javascript e não uma string/texto como é no HTML



```
1 import React from 'react';
2 import './App.css'
3
4
5 export default () =>{
6   return (
7     <h1 style={{fontSize:'4em', color:'red'}}>Conteúdo de App.js</h1>
8   )
9 }
10
```

```
<p className='exemplo'>"A cachorra Baleia estava para morrer. Tinha emagrecido,
foseco, onde manchas escuras supuravam e sangravam, cobertas de moscas". Vidas Se
```

# Criando e estilizando nosso primeiro componente

# Exercícios

## 1. Criar um Componente de Boas-Vindas

Crie um componente React que exiba uma mensagem de boas-vindas.

## 2. Criar um Componente de Informações Básicas

Crie um componente React que exiba informações básicas de um usuário, usando um objeto.

Dados do usuário: Nome, Idade, Data de Nascimento, E-mail.

## 3. Criar um Componente de Lista Estática

Crie um componente React que exiba uma lista estática de itens.

## 4. Criar uma Lista estática de usuários.

Crie um componente que exibe informações a partir de um Array de Objetos.

Dados do usuário: Nome, Idade, Data de Nascimento, E-mail.

# Exercícios

## 5. Construindo um Card Interativo

Você foi designado para criar um componente React que represente um card interativo. Este card deve exibir informações fictícias sobre um produto e ter interações visuais quando o usuário passar o mouse sobre ele. No final crie uma lista estática de produtos e renderize o componente varias vezes com o uso do `.map`.

- Crie um componente funcional React chamado `InteractiveCard`.
- O card deve exibir as seguintes informações fictícias:
  - Nome do produto: "Smartphone XYZ"
  - Preço: R\$ 999,99
  - Descrição: "O smartphone XYZ é repleto de recursos incríveis para atender às suas necessidades diárias."
- Estilize o card de forma agradável e responsiva.
- Adicione uma animação de transição suave para as interações do mouse. Quando o usuário passar o mouse sobre o card, as cores de fundo e texto devem mudar gradualmente.
- Certifique-se de que o card seja visualmente atraente e que as informações sejam legíveis.



ABREU, Luis. Typescript - O Javascript moderno para criação de aplicações. Editora FCA, 2017.

ADRIANO, T. S. Guia prático de TypeScript. São Paulo: Casa do Código, 2021.

ANTONIO, C. Pro React: Build Complex Front-End Applications in a Composable Way With React. Apress, 2015.

BOSWELL, D; FOUCHER, T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. Estados Unidos: O'Reilly Media, 2012.

BRITO, Robin Cris. Android Com Android Studio - Passo A Passo. Editora Ciência Moderna.

BUNA, S. React Succinctly. Estados Unidos: [s.n], 2016. Disponível em: <[www.syncfusion.com/ebooks/reactjs\\_succinctly](http://www.syncfusion.com/ebooks/reactjs_succinctly)>. Acesso em: 12 de janeiro de 2023.

FACEBOOK (2019a). React: Getting Started. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-api.html](http://reactjs.org/docs/react-api.html)>. Acesso em: 13 de janeiro de 2023.

FACEBOOK (2019b). React Without ES6. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-without-es6.html](http://reactjs.org/docs/react-without-es6.html)>. Acesso em: 10 de janeiro de 2023.

FACEBOOK (2019c). React Without JSX. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-without-jsx.html](http://reactjs.org/docs/react-without-jsx.html)>. Acesso em: 10 de janeiro de 2023.

FREEMAN, Eric ROBSON, Elisabeth. Use a Cabeça! Programação em HTML5. Rio de Janeiro: Editora Alta Books, 2014

GACKENHEIMER, C. Introduction to React: Using React to Build scalable and efficient user interfaces.[s.i.]: Apress, 2015.

GOLDBERG, Josh. Aprendendo TypeScript: Melhore Suas Habilidades de Desenvolvimento web Usando JavaScript Type-Safe. São Paulo: Novatec. 2022

HUDSON, P. Hacking with React. 2016. Disponível em: <[www.hackingwithreact.com/read/1/3/introduction-to-jsx](http://www.hackingwithreact.com/read/1/3/introduction-to-jsx)>. Acesso em: 13 janeiro de 2023.



KOSTRZEWA, D. Is React.js the Best JavaScript Framework in 2018? 2018. Disponível em: <[hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8](https://hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8)>. Acesso em: janeiro de 2023.

MARTIN, R. Clean Code: A Handbook of Agile Software Craftsmanship. Estados Unidos: Prentice Hall, 2009.

MDN WEB DOCS. Guia JavaScript. Disponível em <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>>. Acessado em 29 de janeiro de 2023.

NELSON, J. Learn React's Fundamentals Without the Buzzwords? 2018. Disponível em: <[jamesknelson.com/learn-react-fundamentals-sans-buzzwords](https://jamesknelson.com/learn-react-fundamentals-sans-buzzwords)>. Acesso em: 12 janeiro de 2023.

NIELSEN, J. Response Times: The 3 Important Limits. 1993. Disponível em: <[www.nngroup.com/articles/response-times-3-important-limits](http://www.nngroup.com/articles/response-times-3-important-limits)>. Acesso em: 10 janeiro de 2023.

O'REILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <[www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap](http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap)>. Acesso em: 10 de janeiro de 2023.

PANDIT, N. What Is ReactJS and Why Should We Use It? 2018. Disponível em: <[www.c-sharpcorner.com/article/what-and-why-reactjs](http://www.c-sharpcorner.com/article/what-and-why-reactjs)>. Acesso em: 12 de janeiro de 2023.

RAUSCHMAYER, A. Speaking JavaScript: An In-Depth Guide for Programmers. Estados Unidos: O'Reilly Media, 2014.

REACTIVA. O arquivo package-lock.json. Disponível em: <<https://nodejs.reativa.dev/0020-package-lock-json/index>>. Acessado em 13 de janeiro de 2023.

\_\_\_\_\_. O guia do package.json. Disponível em: <<https://nodejs.reativa.dev/0019-package-json/index>>. Acessado em 13 de janeiro de 2023.

RICOY, L. Desmitificando React: Uma Reflexão para Iniciantes. 2018. Disponível em: <[medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114](https://medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114)>. Acesso em: 13 janeiro de 2023.

SILVA, Maurício Samy. Ajax com jQuery: requisições Ajax com a simplicidade de jQuery. São Paulo: Novatec Editora, 2009.

FIAP

\_\_\_\_\_. Construindo Sites com CSS e XHTML. Sites Controlados por Folhas de Estilo em Cascata. São Paulo: Novatec, 2010.

\_\_\_\_\_. CSS3 - Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS. São Paulo: Novatec Editora, 2010.

STACKOVERFLOW. Most Popular Technologies: Web Frameworks. Developer Survey Results, StackOverflow, 2019. Disponível em: <[insights.stackoverflow.com/survey/2019#technology](https://insights.stackoverflow.com/survey/2019#technology)>. Acesso em: 13 de janeiro de 2023.

SWC. Rust-based platform for the Web. 2024 Disponível em <<https://swc.rs/>>. Acessado em 23 de janeiro de 2024.

W3C. HTML5 - A linguagem de marcação que revolucionou a web. São Paulo: Novatec Editora, 2010.

\_\_\_\_\_. A vocabulary and associated APIs for HTML and XHTML. Disponível em <<https://www.w3.org/TR/2018/SPSD-html5-20180327/>>. Acessado em 28 de abril de 2020, às 20h53min.

\_\_\_\_\_. Cascading Style Sheets, level 1. Disponível em <<https://www.w3.org/TR/2018/SPSD-CSS1-20180913/>>. Acessado em 28 de abril de 2020, às 21h58min.

\_\_\_\_\_. Cascading Style Sheets, level 2 Revision 2. Disponível em <<https://www.w3.org/TR/2016/WD-CSS22-20160412/>>. Acessado em 28 de abril de 2020, às 22h17min.

\_\_\_\_\_. Cascading Style Sheets, level 2. Disponível em <<https://www.w3.org/TR/2008/REC-CSS2-20080411/>>. Acessado em 28 de abril de 2020, às 22h03min.

\_\_\_\_\_. Cascading Style Sheets, level 3. Disponível em <<https://www.w3.org/TR/css-syntax-3/>>. Acessado em 28 de abril de 2020, às 22h18min.

W3C. HTML 3.2 Reference Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html32-20180315/>>. Acessado em 28 de abril de 2020, às 19h37min

\_\_\_\_\_. HTML 4.0 Specification. Disponível em <<https://www.w3.org/TR/1998/REC-html40-19980424/>>. Acessado em 28 de abril de 2020, às 19h53min.

\_\_\_\_\_. HTML 4.01 Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html401-20180327/>>. Acessado em 28 de abril de 2020, às 20h04min.

\_\_\_\_\_.Cascading Style Sheets, level 2 Revision 1. Disponível em <<https://www.w3.org/TR/CSS2/>>. Acessado em 28 de abril de 2020, às 22h13min.

WIKIPEDIA. JavaScript. Disponível em <<https://pt.wikipedia.org/wiki/JavaScript>>. Acessado em 29 de abril de 2020, às 10h.

Dúvidas, críticas ou sugestões?



FIAP