



# FRONT-END DESIGN ENGINEERING





[aula 02]

[Atualizado em: 23/01/2024]



[React – Introdução]



Material cedido gentilmente pelo professor Alexandre Carlos ([profalexandre.jesus@fiap.com.br](mailto:profalexandre.jesus@fiap.com.br)) e professor Luís Carlos ([lsilva@fiap.com.br](mailto:lsilva@fiap.com.br)) e atualizado pelo professor Adriano Milanez ([profadriano.milanez@fiap.com.br](mailto:profadriano.milanez@fiap.com.br))



## REACT – INTRODUÇÃO

REACT – INTRODUÇÃO

# REACT – INTRODUÇÃO

## ✓ O que é o 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.

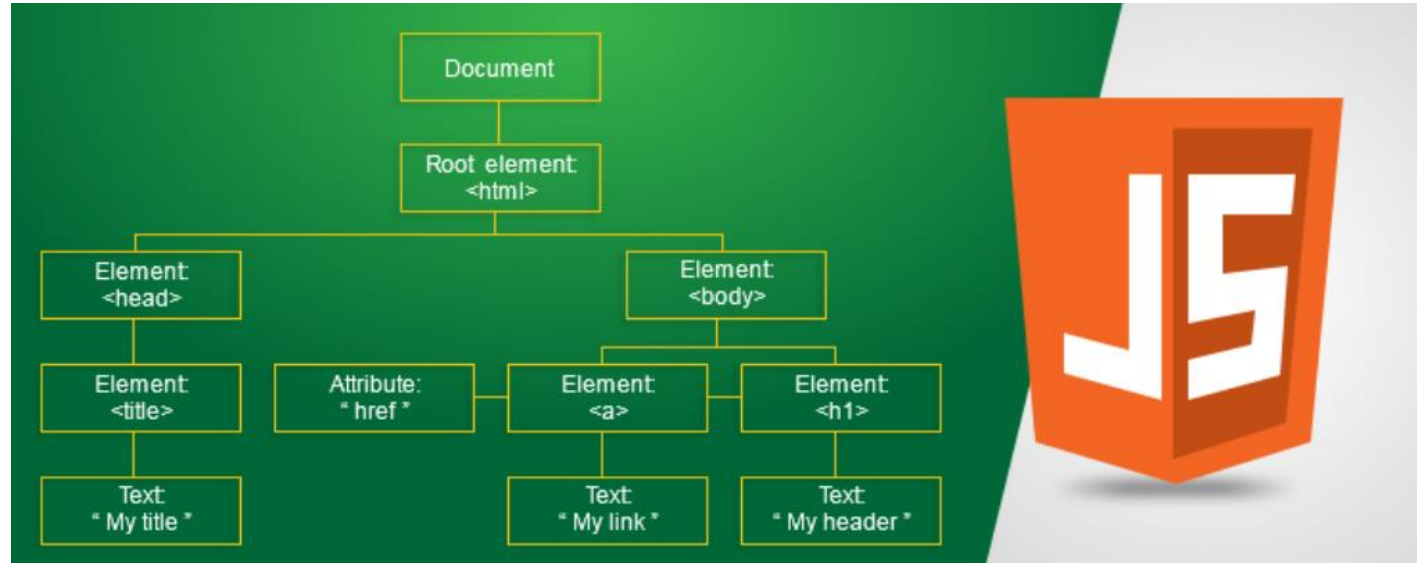
# REACT – INTRODUÇÃO

## ✓ O que é o 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**.

# REACT – INTRODUÇÃO

✓ O que é o REACT?



# REACT – INTRODUÇÃO

## ✓ O que é o REACT?

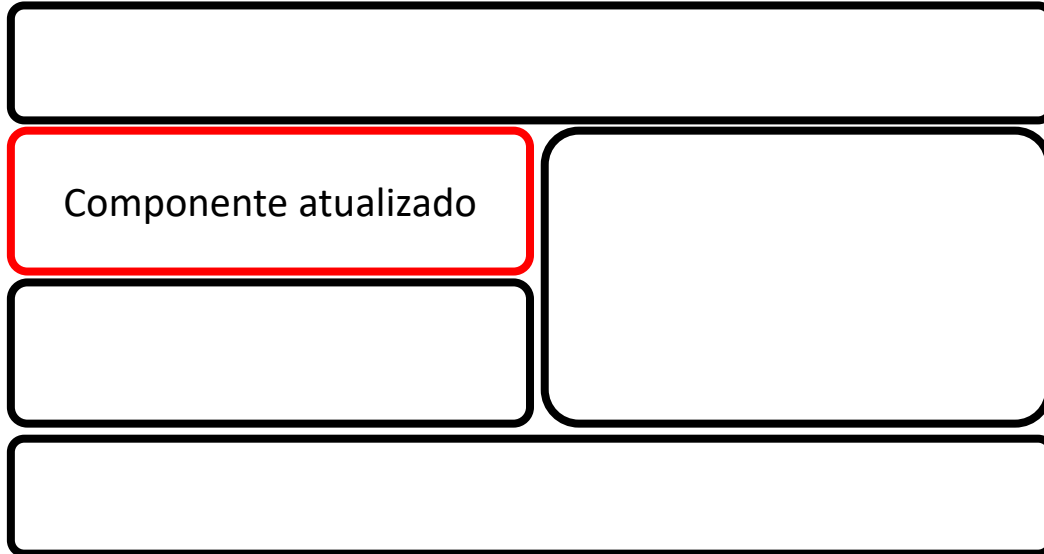
### ✓ DOM – Manipulação dos dados HTML em tempo real com o React

- ✓ 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.

# REACT – INTRODUÇÃO

## ✓ O que é o REACT?

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

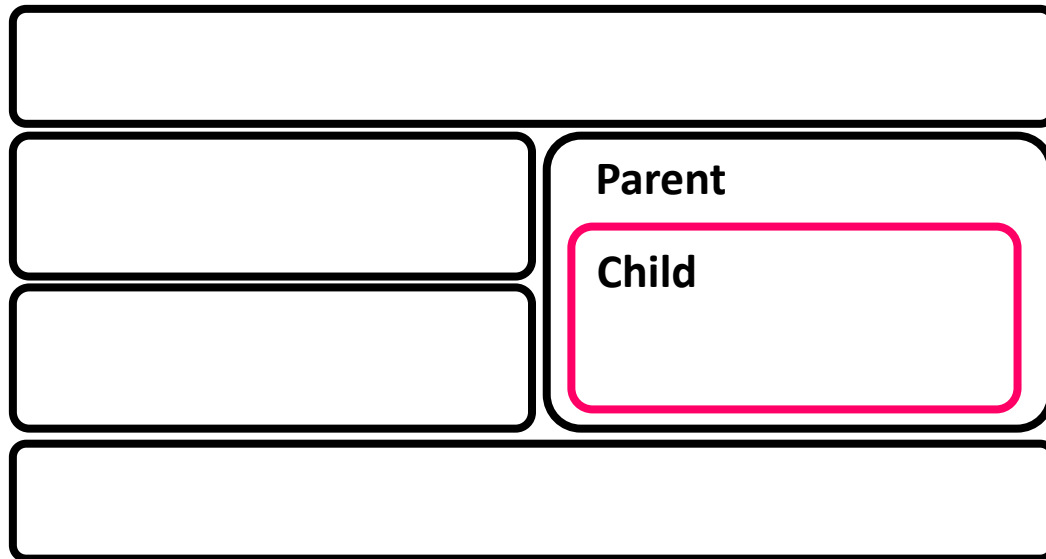




# REACT – INTRODUÇÃO

## ✓ O que é o REACT?

- ✓ Ele trabalha com um fluxo unidirecional, em um único sentido, ou “One-way data flow”. As informações devem sempre vir do elemento pai (Parent) para o elemento filho (Child).



# REACT – INTRODUÇÃO

## ✓ O que é o 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.

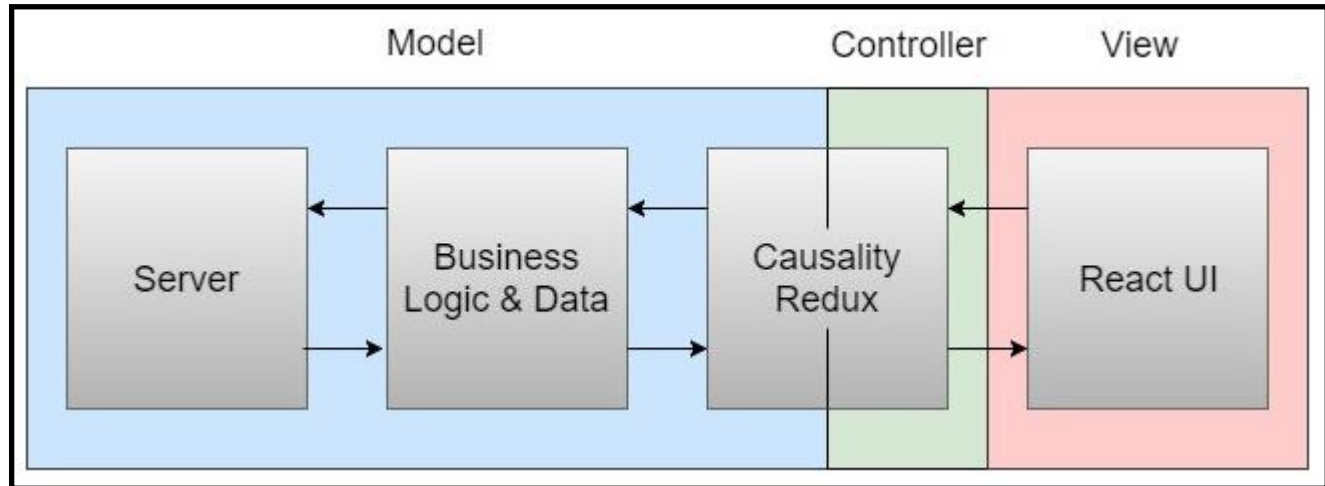
### ✓ Pontos negativos

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

# REACT – INTRODUÇÃO

## ✓ O que é o 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.



# REACT – INTRODUÇÃO

## ✓ O que é o REACT?

### ✓ Frameworks

✓ Next.JS

✓ Vite.JS

✓ Vite e Next.js são duas das estruturas de front-end mais populares da atualidade. Ambos oferecem uma variedade de recursos e benefícios, mas também apresentam algumas diferenças importantes.

# REACT – INTRODUÇÃO

## ✓ O que é o REACT?

### ✓ Frameworks – Diferenças

#### ✓ Next.JS

- ✓ 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.

#### ✓ Vite.JS

- ✓ 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.

# REACT – INTRODUÇÃO

## ✓ O que é o REACT?

### ✓ Frameworks – Diferenças

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

# REACT – INTRODUÇÃO

## ✓ O que é o REACT?

### ✓ Frameworks – Qual escolher?

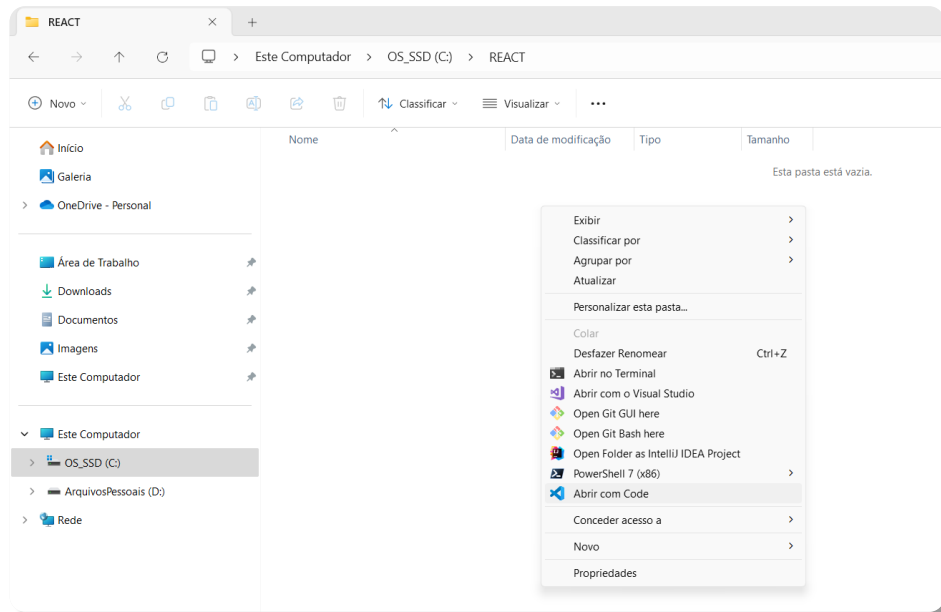
- ✓ Se você está procurando uma ferramenta de construção e servidor de desenvolvimento rápido e eficiente, o Vite é uma ótima opção. Também é uma boa opção se você deseja ter mais controle sobre o processo de desenvolvimento.
- ✓ Se você está procurando uma estrutura React que ofereça vários recursos 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

# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ Em seu Gerenciador de Arquivos crie uma pasta chamada “REACT” em seguida, clique com o botão direito e selecione “Abrir com o Code”.

Criaremos uma pasta chamada REACT na raiz e apontaremos todas as aulas/projetos para esse diretório.

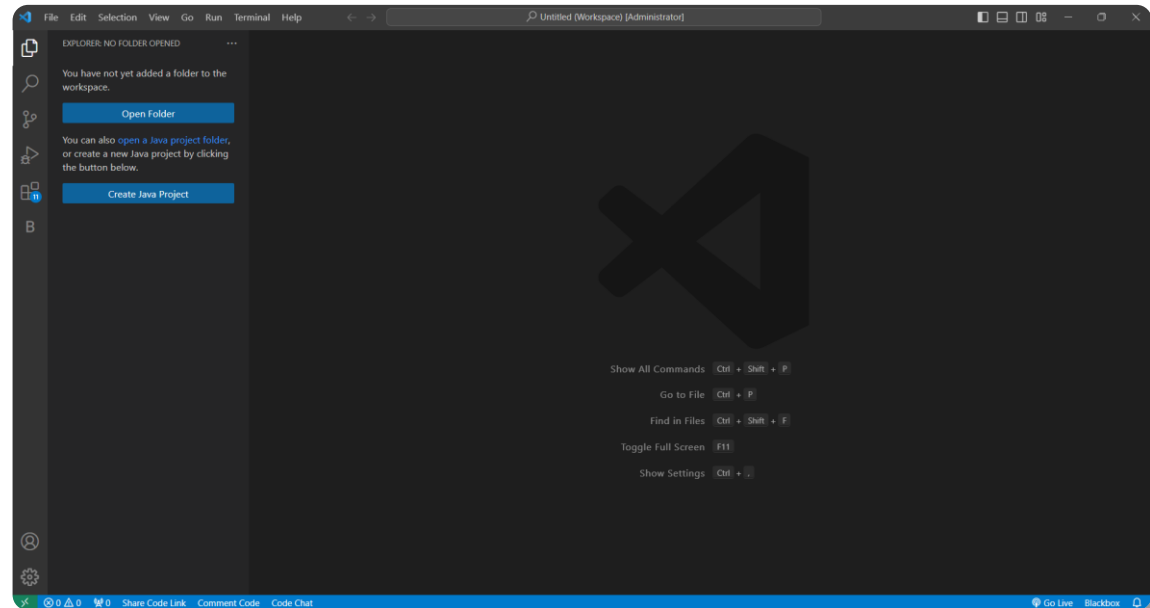
Para cada aula/projeto manteremos o padrão: aula01, aula02, aula02a, aula03b e assim sucessivamente.





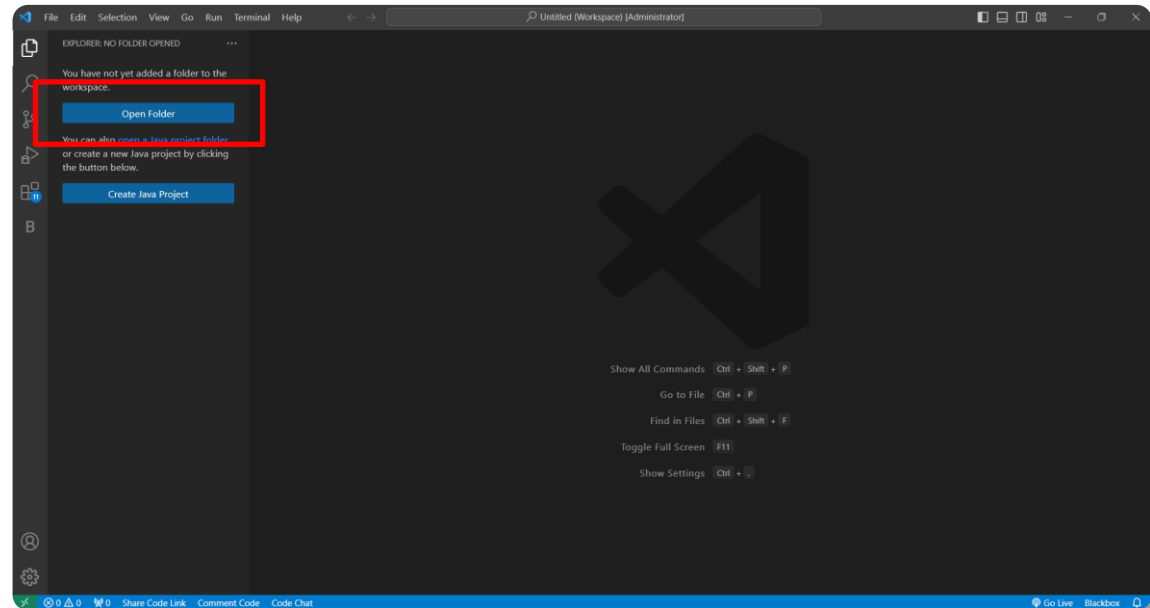
# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ Possivelmente seu VSCode deve ter aberto desta forma



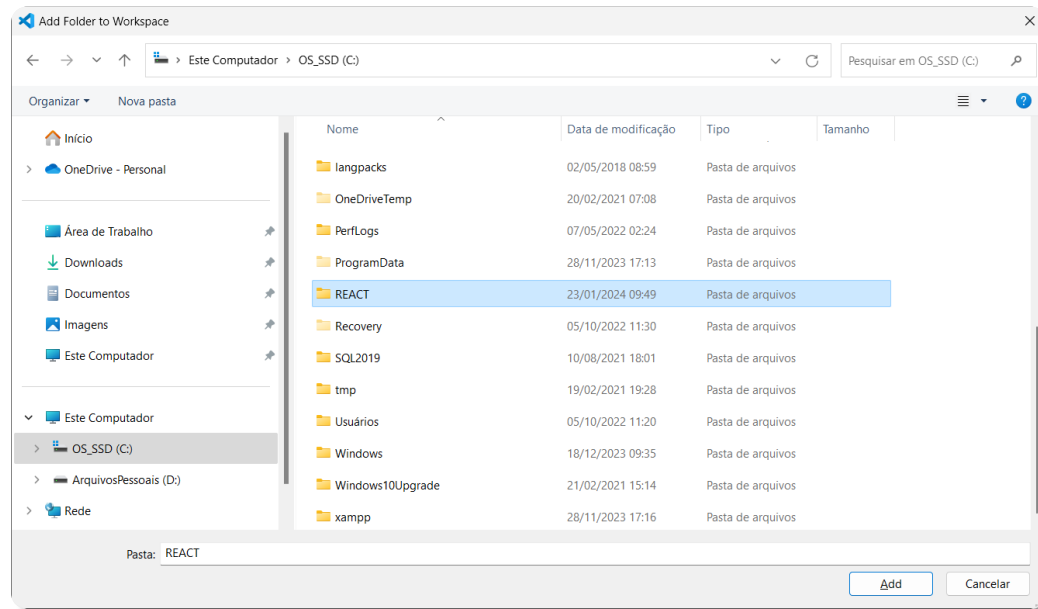
# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ Possivelmente seu VSCode deve ter aberto desta forma
- ✓ Se isso aconteceu selecione “Open Folder”



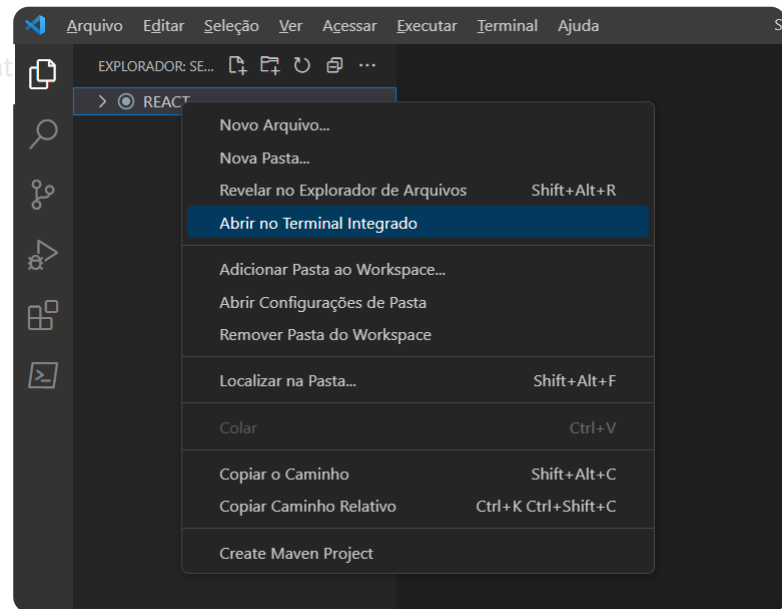
# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ Possivelmente seu VSCode deve ter aberto desta forma
- ✓ Se isso aconteceu selecione “Open Folder”
- ✓ Selecione a pasta “REACT” que criamos anteriormente



# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ Possivelmente seu VSCode deve ter aberto desta forma
- ✓ Se isso aconteceu selecione “Open Folder”
- ✓ Selecione a pasta “REACT” que criamos anteriormente
- ✓ Na sequência clique o botão direito do mouse e selecione “Abrir no Terminal Integrado”

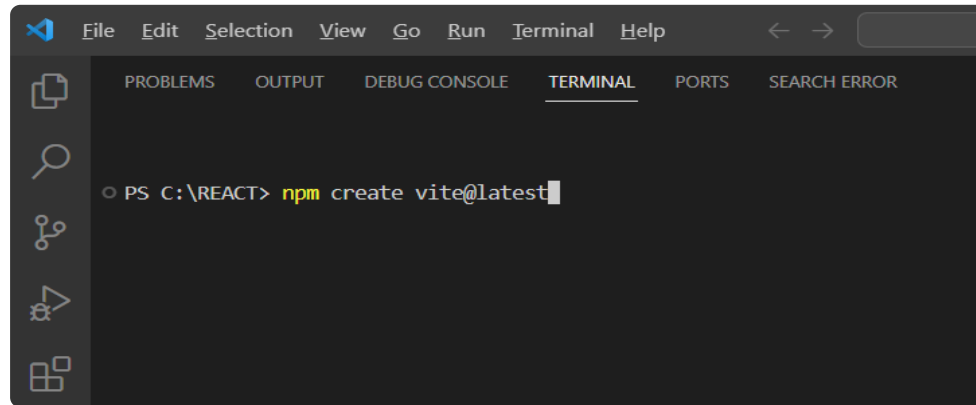


# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**

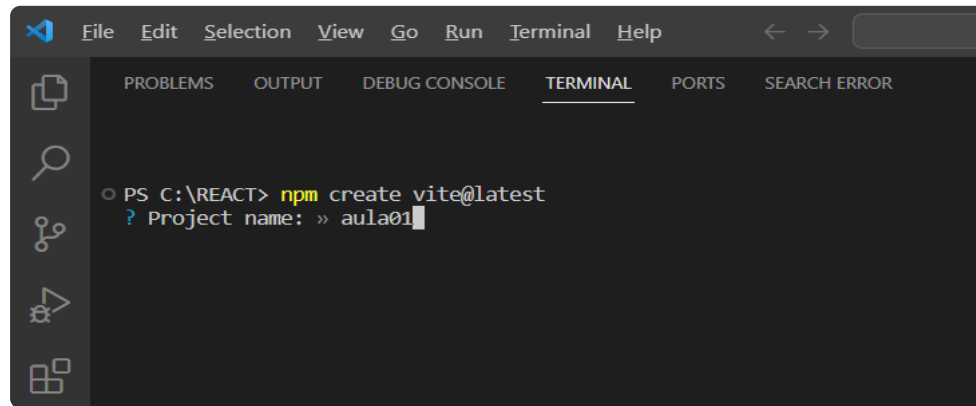
- ✓ No terminal aberto vamos digitar o seguinte:

- ✓ `npm create vite@latest`



# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ No terminal aberto vamos digitar o seguinte:
  - ✓ `npm create vite@latest`
  - ✓ Na sequência daremos um nome ao nosso projeto



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS C:\REACT> npm create vite@latest
? Project name: > aula01
```

# REACT – INTRODUÇÃO

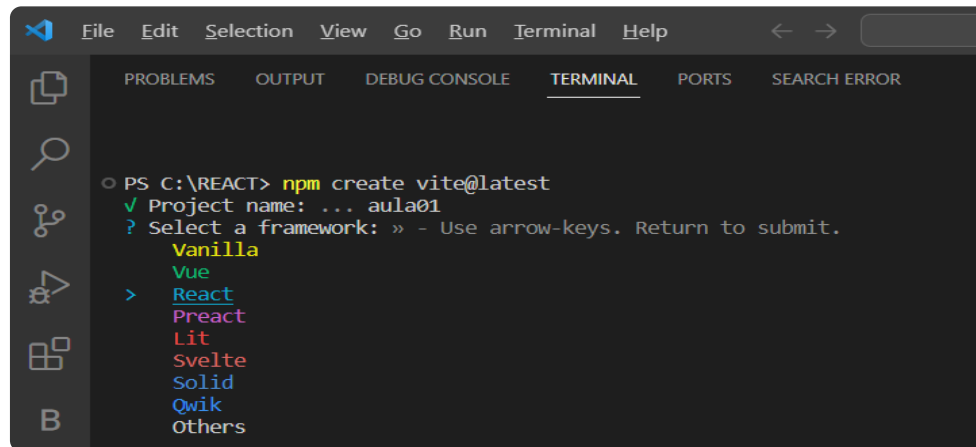
## ✓ Vamos criar nossa primeira aplicação REACT

✓ No terminal aberto vamos digitar o seguinte:

✓ `npm create vite@latest`

✓ Na sequência daremos um nome ao nosso projeto

✓ Descendo com a seta de direção no teclado, selecionamos o framework (REACT) que vamos criar nosso projeto.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS C:\REACT> npm create vite@latest
✓ Project name: ... aula01
? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
  > React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Others
```

# REACT – INTRODUÇÃO

## ✓ Vamos criar nossa primeira aplicação REACT

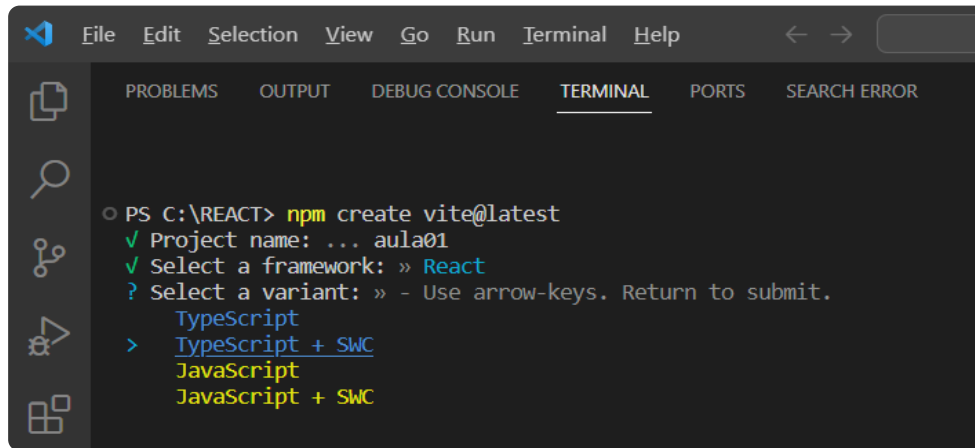
✓ No terminal aberto vamos digitar o seguinte:

✓ `npm create vite@latest`

✓ Na sequência daremos um nome ao nosso projeto

✓ Descendo com a seta de direção no teclado, selecionamos o framework (REACT) que vamos criar nosso projeto

✓ E na sequência o “linguagem” (TYPESCRIPT + SWC) que vamos trabalhar



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS C:\REACT> npm create vite@latest
✓ Project name: ... aula01
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
> TypeScript + SWC
  JavaScript
  JavaScript + SWC
```



# REACT – INTRODUÇÃO

## ✓ Vamos criar nossa primeira aplicação REACT

✓ No terminal aberto vamos digitar o seguinte:

✓ `npm create vite@latest`

✓ Na sequência daremos um nome ao nosso projeto

✓ Descendo com a seta de direção no teclado, selecionamos o framework (REACT) que vamos criar nosso projeto

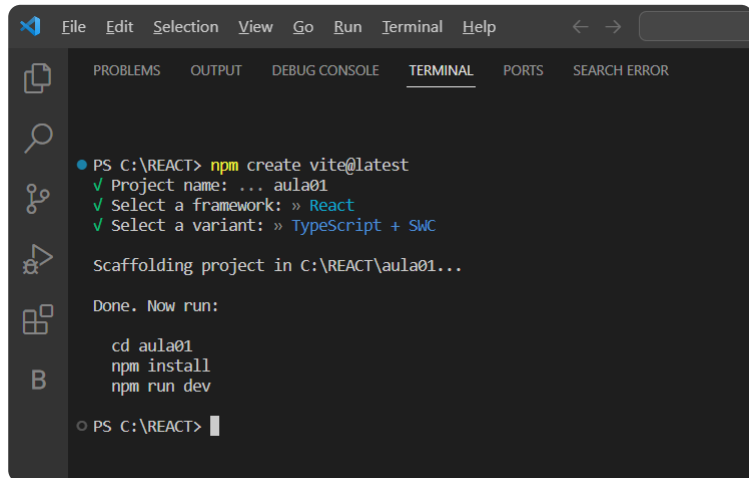
✓ E na sequência o “linguagem” (TYPESCRIPT + SWC) que vamos trabalhar

✓ SWC (sigla para Speedy Web Compiler) é um compilador TypeScript/JavaScript super-rápido escrito em Rust. Segundo a documentação Eles afirmam ser “20x mais rápido que o Babel em um único thread e 70x mais rápido em quatro núcleos”.

# REACT – INTRODUÇÃO

## ✓ Vamos criar nossa primeira aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, teremos a seguinte resposta.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

• PS C:\REACT> npm create vite@latest
✓ Project name: ... aula01
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in C:\REACT\aula01...

Done. Now run:

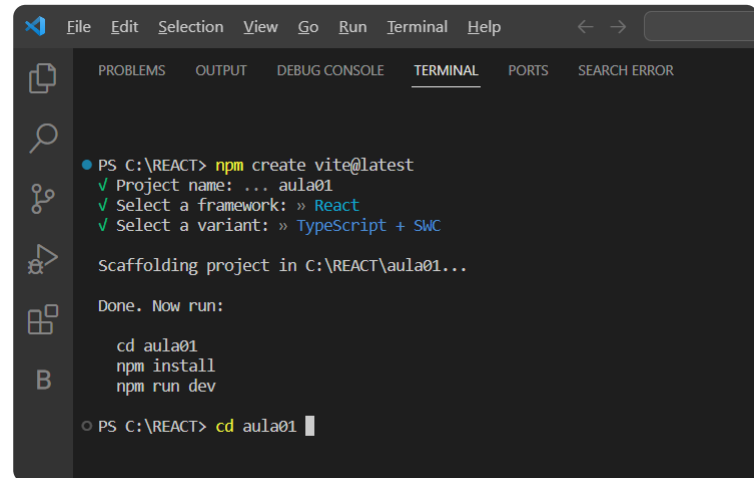
  cd aula01
  npm install
  npm run dev

○ PS C:\REACT> 
```

# REACT – INTRODUÇÃO

## ✓ Vamos criar nossa primeira aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, teremos a seguinte resposta.
- ✓ Como na imagem, vamos digitar o que se pede



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS C:\REACT> npm create vite@latest
✓ Project name: ... aula01
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in C:\REACT\aula01...

Done. Now run:

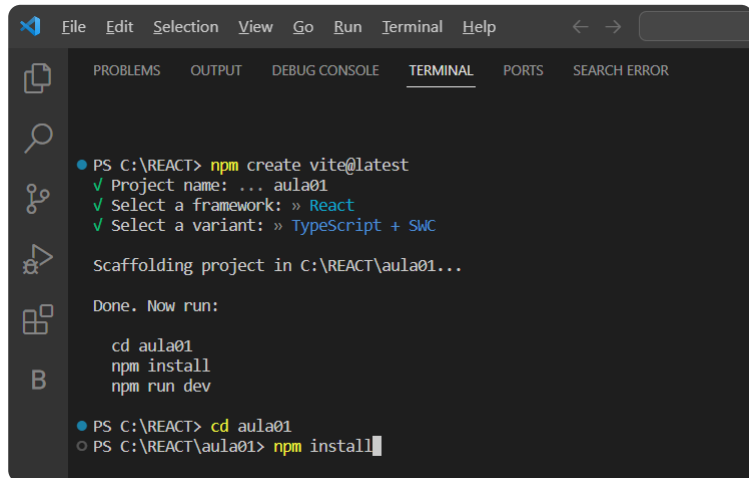
  cd aula01
  npm install
  npm run dev

PS C:\REACT> cd aula01
```

# REACT – INTRODUÇÃO

## ✓ Vamos criar nossa primeira aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, termos a seguinte resposta.
- ✓ Como na imagem, vamos digitar o que se pede



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

• PS C:\REACT> npm create vite@latest
✓ Project name: ... aula01
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in C:\REACT\aula01...

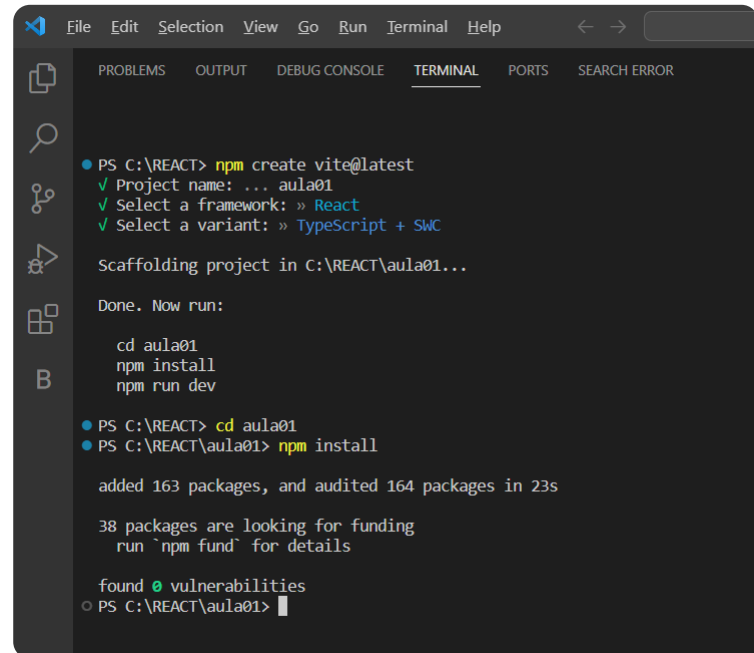
Done. Now run:

  cd aula01
  npm install
  npm run dev

• PS C:\REACT> cd aula01
○ PS C:\REACT\aula01> npm install
```

# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
  - ✓ No terminal aberto vamos digitar o seguinte:
  - ✓ Após o término da criação, teremos a seguinte resposta.
  - ✓ Como na imagem, vamos digitar o que se pede, veremos que foram adicionados em nosso projeto alguns pacotes
  - ✓ Na sequência, digitamos à última instrução solicitada



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

• PS C:\REACT> npm create vite@latest
✓ Project name: ... aula01
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in C:\REACT\aula01...

Done. Now run:

  cd aula01
  npm install
  npm run dev

• PS C:\REACT> cd aula01
• PS C:\REACT\aula01> npm install

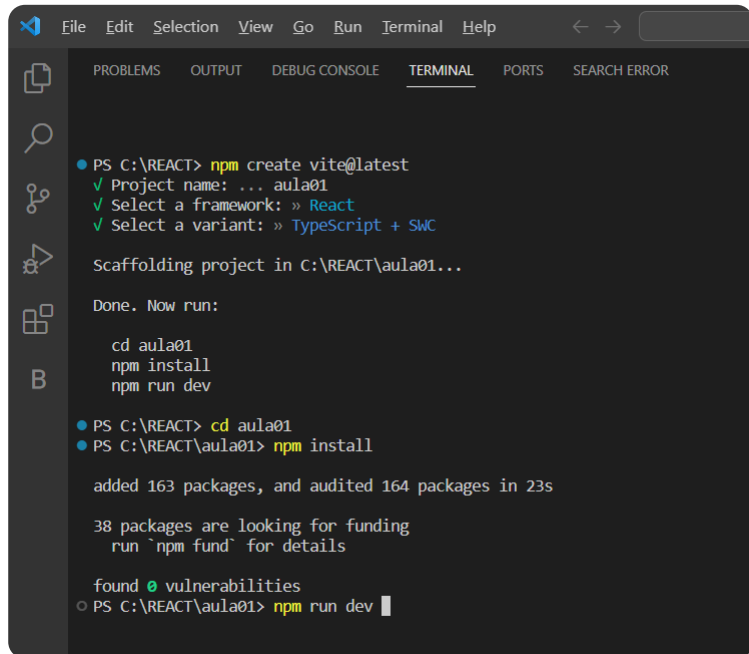
added 163 packages, and audited 164 packages in 23s

38 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
○ PS C:\REACT\aula01> 
```

# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
  - ✓ No terminal aberto vamos digitar o seguinte:
  - ✓ Após o término da criação, teremos a seguinte resposta.
  - ✓ Como na imagem, vamos digitar o que se pede, veremos que foram adicionados em nosso projeto alguns pacotes
  - ✓ Na sequência, digitamos à última instrução solicitada



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

• PS C:\REACT> npm create vite@latest
✓ Project name: ... aula01
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in C:\REACT\aula01...

Done. Now run:

  cd aula01
  npm install
  npm run dev

• PS C:\REACT> cd aula01
• PS C:\REACT\aula01> npm install

added 163 packages, and audited 164 packages in 23s

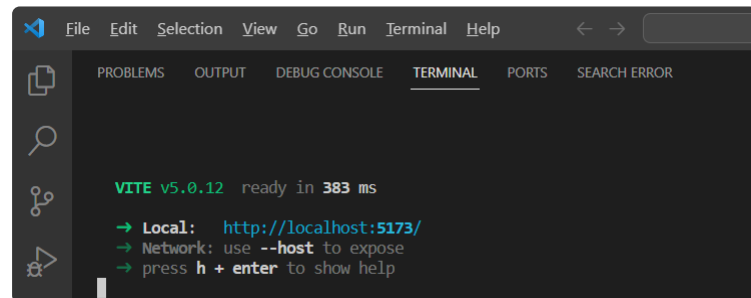
38 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
• PS C:\REACT\aula01> npm run dev
```

# REACT – INTRODUÇÃO

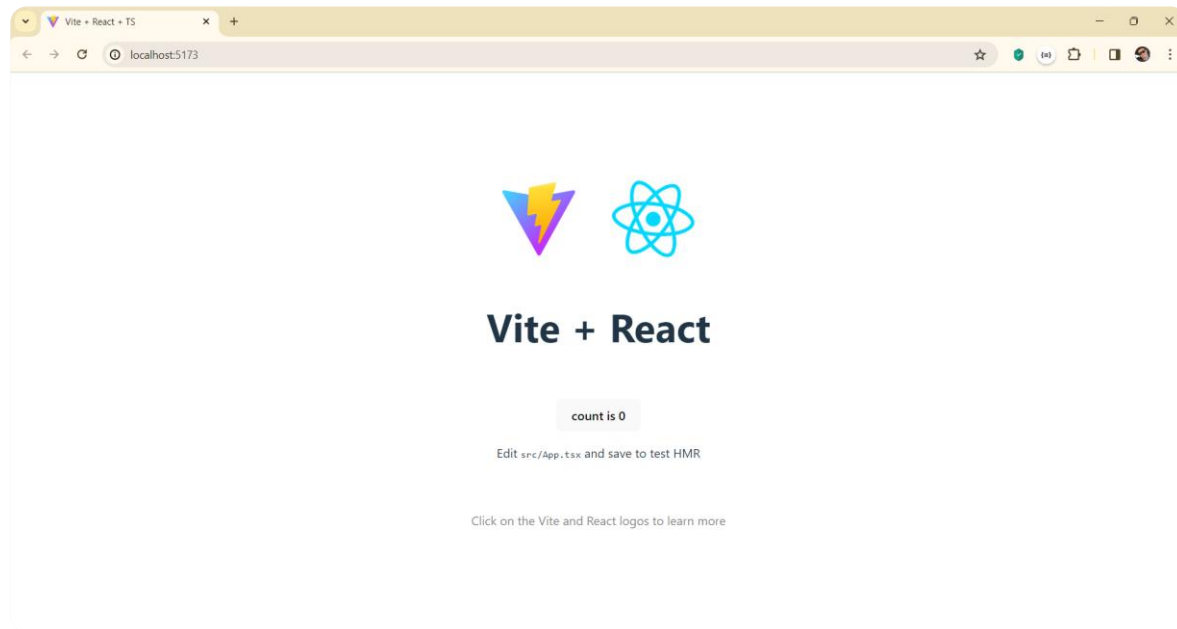
## ✓ Vamos criar nossa primeira aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, teremos a seguinte resposta.
- ✓ Como na imagem, vamos digitar o que se pede, veremos que foram adicionados em nosso projeto alguns pacotes
- ✓ Na sequência, digitamos à última instrução solicitada
- ✓ Pressionamos a tecla CTRL e clicamos na primeira opção.



# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.





# REACT – INTRODUÇÃO

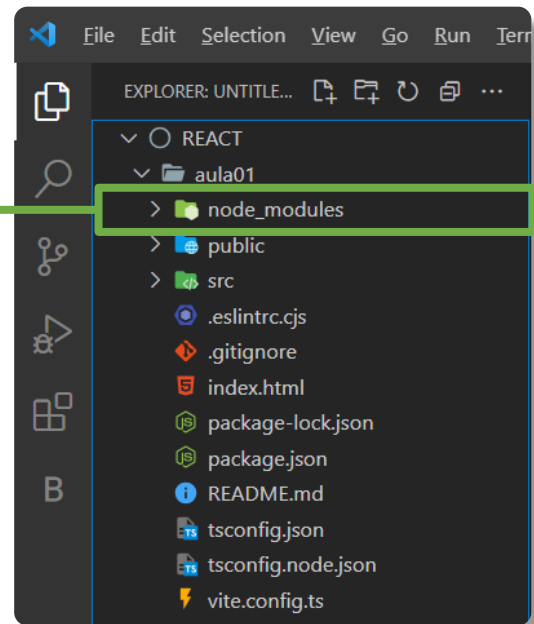
## ✓ Vamos criar nossa primeira aplicação REACT

- ✓ Analisando a estrutura de pastas criadas em nosso projeto temos a seguinte estrutura.

### node\_modules

São bibliotecas do Node que estão disponível para uso na aplicação.

Essa pasta **NÃO** deverá ser copiada no transporte da sua aplicação entre computadores diferentes.

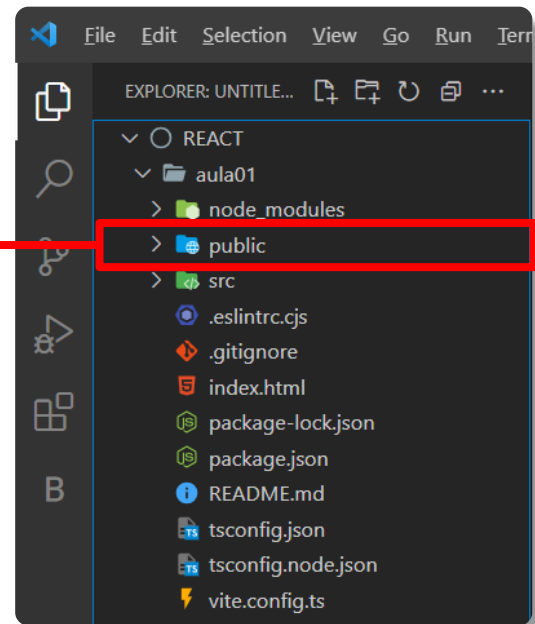


# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ Analisando a estrutura de pastas criadas em nosso projeto temos a seguinte estrutura.

Contém os arquivos que são carregados no lado do cliente

**public**

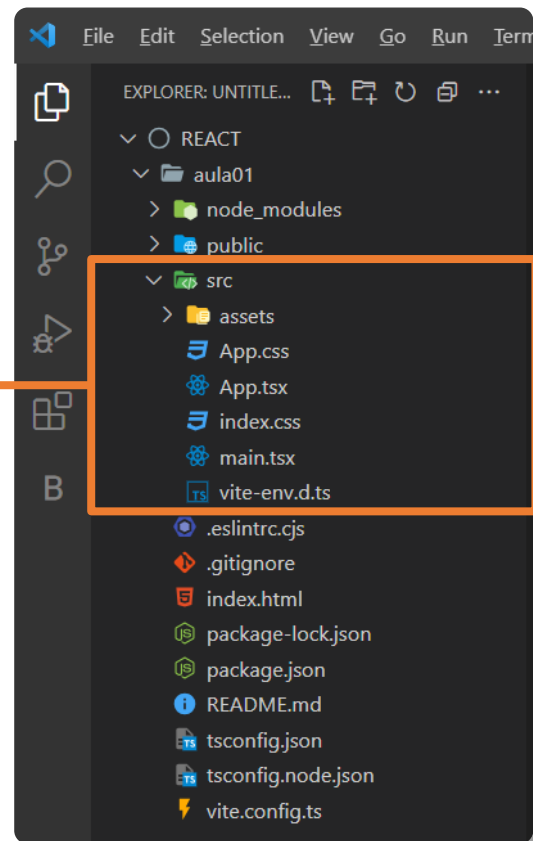


# REACT – INTRODUÇÃO

- ✓ **Vamos criar nossa primeira aplicação REACT**
- ✓ Analisando a estrutura de pastas criadas em nosso projeto temos a seguinte estrutura.

arquivos que estão do lado do servidor

src



# REACT – INTRODUÇÃO

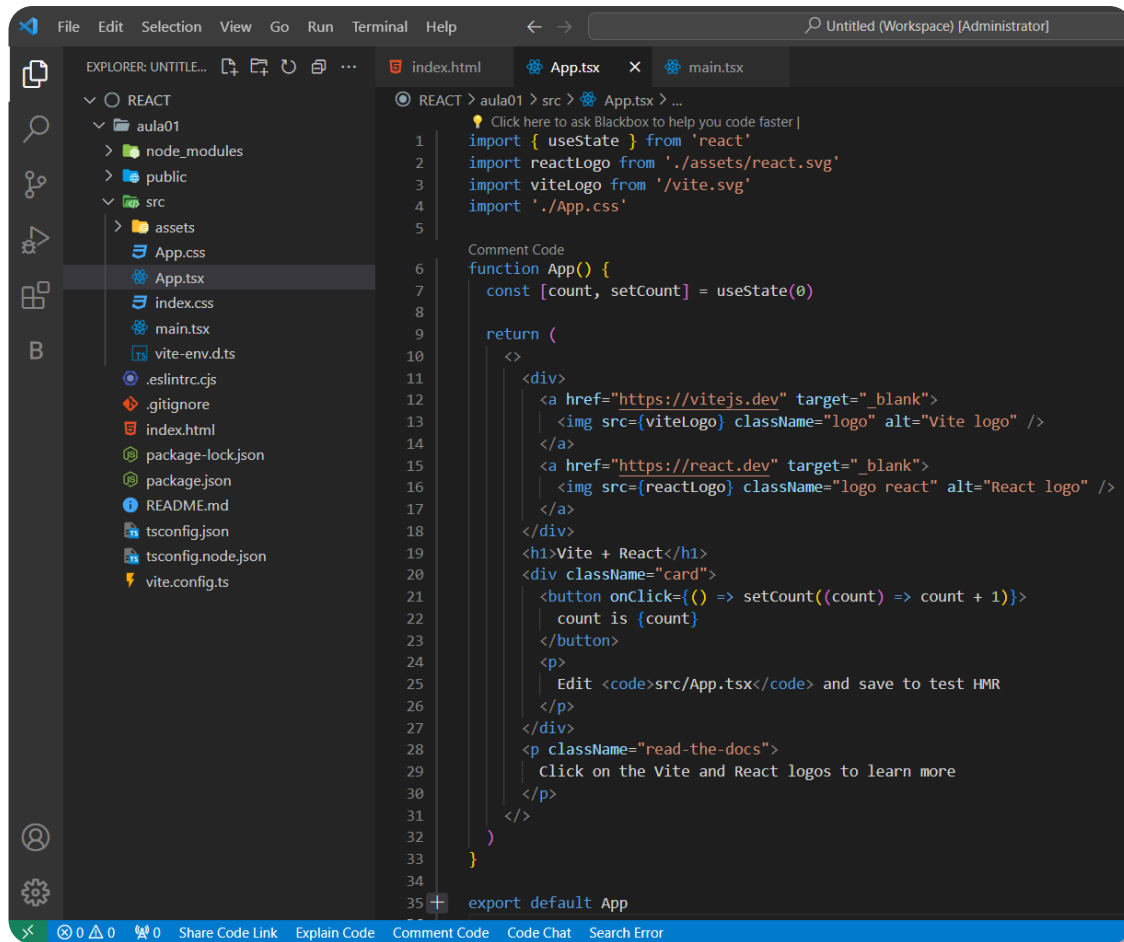
- ✓ Já na pasta SRC, o main.tsx é o responsável por carregar toda a nossa aplicação na div de id “root” que está no index.html. Repare que nele importamos o App.tsx.

```
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
```

# REACT – INTRODUÇÃO

- ✓ O arquivo APP.TSX é nosso arquivo que está dando origem ao **componente** principal da aplicação, repare que todos os componentes da tela estão sendo criados nele dentro de uma função que recebe o seu nome. Esta função retorna todos os elementos da tela que serão criados no html.
- ✓ Isso é possível por causa do JSX.

# REACT – INTRODUÇÃO



# REACT – INTRODUÇÃO

## ✓ Componentes

- ✓ Componentes são como funções Javascript.
- ✓ Eles aceitam entradas arbitrárias e retornam elementos React que descrevem o que deve aparecer na tela.
- ✓ Componentes podem possuir diversos formatos e ter diferentes responsabilidades.
- ✓ Um componente pode ser desde um botão, um formulário ou até uma página completa.

# REACT – INTRODUÇÃO

## ✓ Componentes

- ✓ Podemos ter tantos componentes quanto necessário, mas todos eles devem estar dentro de um componente principal, senão teremos erro no código. Este elemento principal pode ser uma tag como uma DIV, por exemplo, ou uma tag vazia simplesmente para marcar o conteúdo, como temos abaixo:

```
App.tsx X
REACT > aula01 > src > App.tsx > ...
1 import reactLogo from "../assets/react.svg";
2 import viteLogo from "/vite.svg";
3 import "./App.css";
4
5 Comment Code
6 function App() {
7   return (
8     <>
9       <div>
10         <p className="read-the-docs">
11           <a href="https://vitejs.dev" target="_blank">
12             <img src={viteLogo} className="logo" alt="Vite logo" />
13           </a>
14           <a href="https://react.dev" target="_blank">
15             <img src={reactLogo} className="logo react" alt="React logo" />
16           </a>
17         </p>
18       </div>
19     </>
20   );
21 }
22 export default App;
23
```

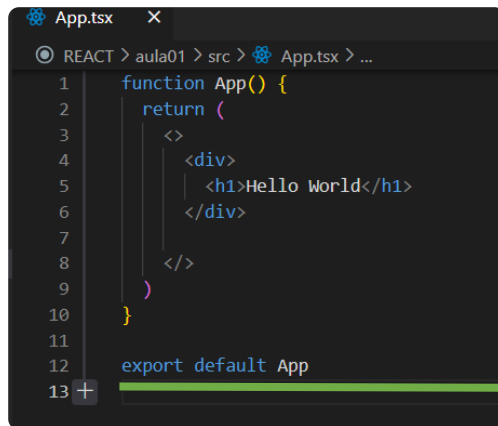
```
App.tsx X
REACT > aula01 > src > App.tsx > ...
1 +
2 Comment Code
3 function App() {
4   return (
5     <>
6       <div>
7         <p>Hello World!</p>
8       </div>
9     </>
10   );
11 }
12 export default App;
13
```



# REACT – INTRODUÇÃO

## ✓ Componentes

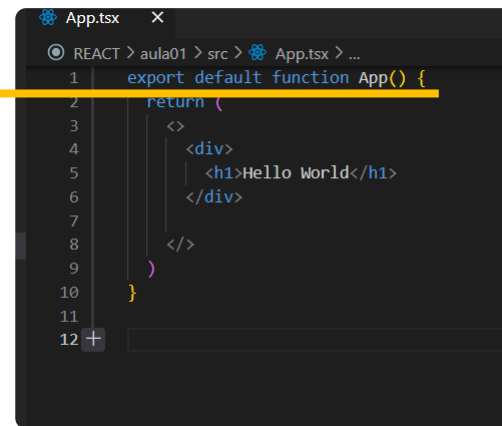
- ✓ Podemos, ainda, fazer a exportação dos componentes na mesma linha da definição deles ou no final.



```
App.tsx
REACT > aula01 > src > App.tsx > ...
1 function App() {
2   return (
3     <>
4       <div>
5         <h1>Hello World</h1>
6       </div>
7     </>
8   )
9 }
10
11
12 export default App
13 +
```

Podemos fazer a exportação da função após a criação e definição da mesma

Podemos fazer a exportação da função na mesma linha da criação

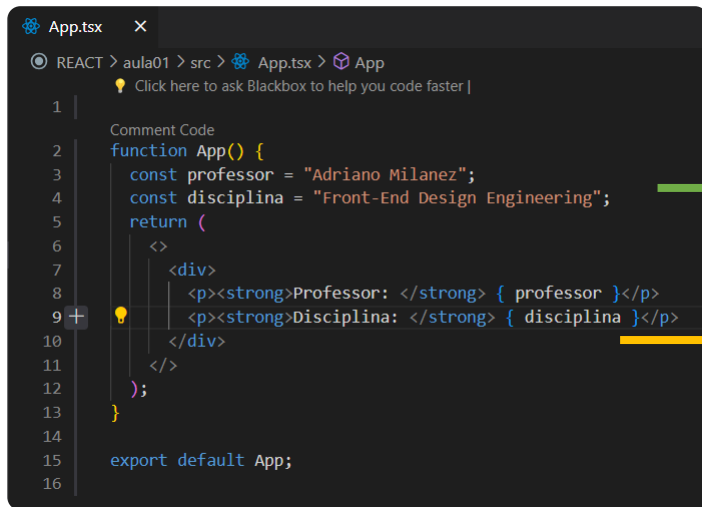


```
App.tsx
REACT > aula01 > src > App.tsx > ...
1 export default function App() {
2   return (
3     <>
4       <div>
5         <h1>Hello World</h1>
6       </div>
7     </>
8   )
9 }
10
11
12 +
```

# REACT – INTRODUÇÃO

## ✓ Componentes

- ✓ Todas as expressões javascript realizadas dentro do tsx devem ser feitas dentro de chaves “{ }”, para serem aceitas



```
1 |  
2 | function App() {  
3 |   const professor = "Adriano Milanez";  
4 |   const disciplina = "Front-End Design Engineering";  
5 |   return (  
6 |     <>  
7 |       <div>  
8 |         <p><strong>Professor: </strong> { professor }</p>  
9 |         <p><strong>Disciplina: </strong> { disciplina }</p>  
10 |       </div>  
11 |     </>  
12 |   );  
13 | }  
14 |  
15 | export default App;  
16 |
```

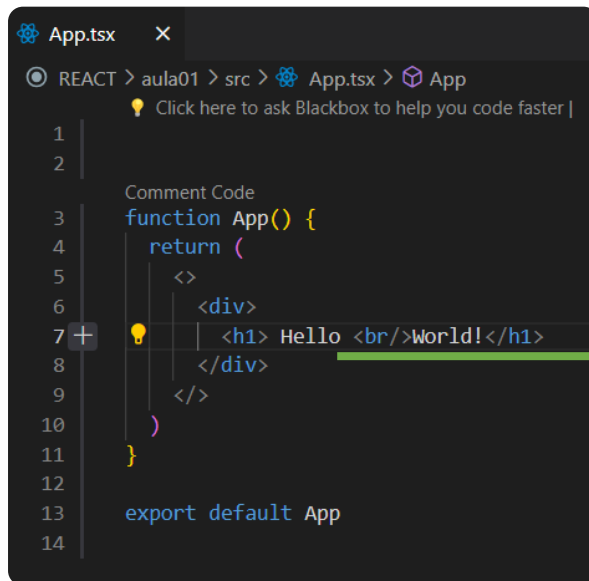
Código Javascript da função antes do return

Valores expressos pelas constantes entre chaves

# REACT – INTRODUÇÃO

## ✓ Javascript

- ✓ Tags com conteúdo vazio devem ter seu fechamento obrigatório.



```
1 |  
2 |  
3 | function App() {  
4 |   return (  
5 |     <>  
6 |       <div>  
7 |         <h1> Hello <br/>World!</h1>  
8 |       </div>  
9 |     </>  
10 |   )  
11 | }  
12 |  
13 | export default App  
14 |
```

**<tag/>**

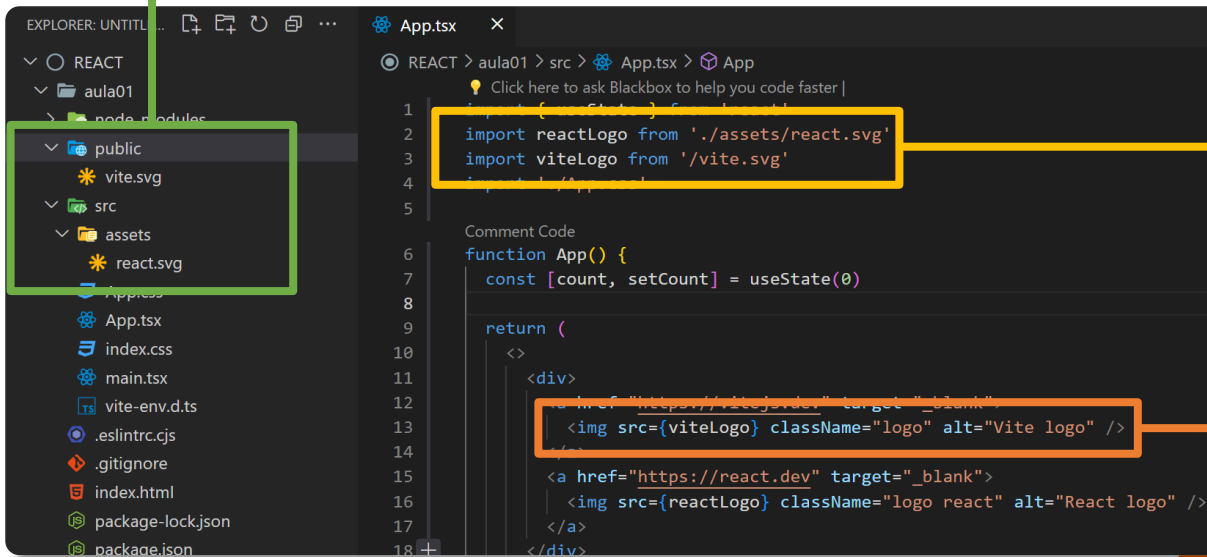
Tags self close precisam de barra de fechamento “obrigatório”

# REACT – INTRODUÇÃO

## ✓ Imagens

- ✓ Para inserir elementos como imagens devemos importar antes de inserir no componente, lembrando sempre se usar as chaves:

Imagem salva na pasta src/assets ou na pasta public

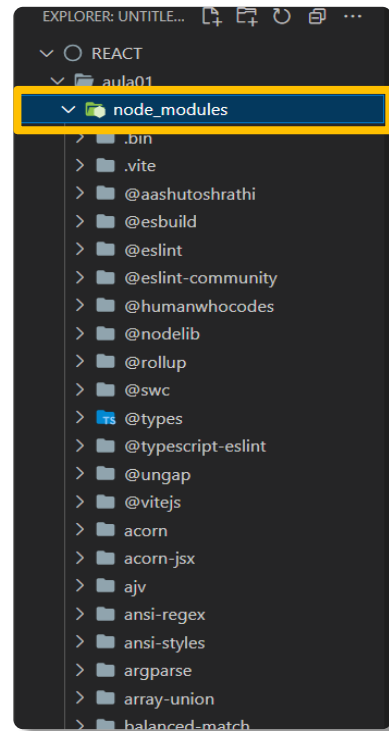


Importação da imagem

Inserção da imagem no elemento <img>

# REACT – INTRODUÇÃO

- ✓ **Entendendo a organização e o sistema de arquivos do projeto**
- ✓ **Pasta NODE\_MODULES**
  - ✓ Contém todas as bibliotecas externas do JavaScript usadas pelo aplicativo. Você raramente precisará abri-lo.

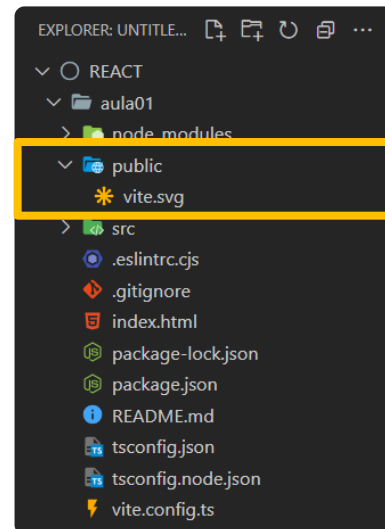


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Pasta PUBLIC

- ✓ A pasta pública contém arquivos estáticos, como index.html, arquivos de biblioteca javascript, imagens e outros ativos, etc. que você não deseja que sejam processados pelo webpack. 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.



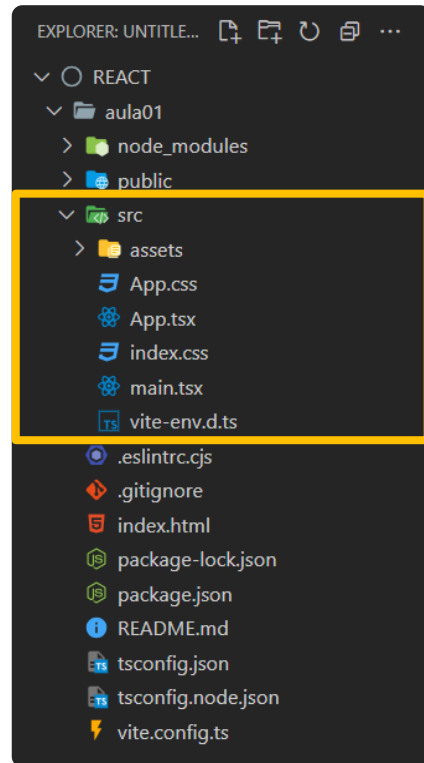
# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Pasta 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 NESTE diretório.



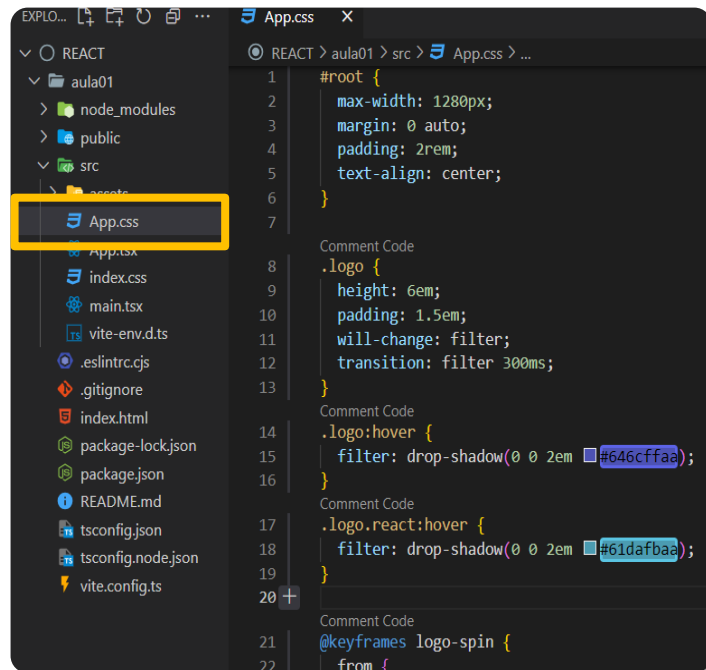
# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ 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.

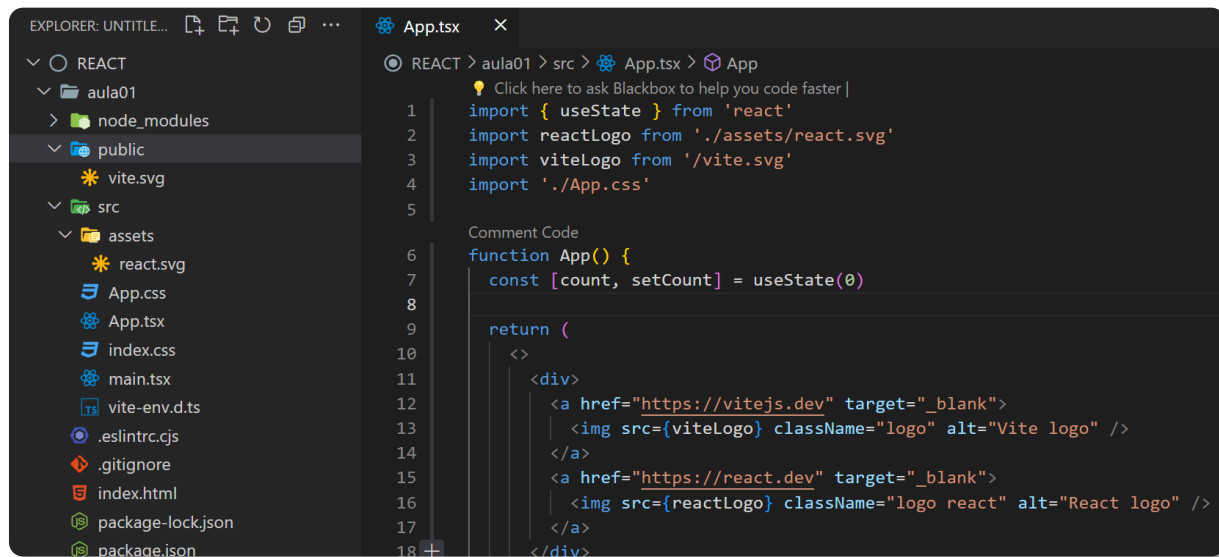
Por boa prática, colocamos os nomes dos arquivos CSS iguais aos do componente.





# REACT – INTRODUÇÃO

- ✓ Entendendo a organização e o sistema de arquivos do projeto
- ✓ 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.



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'REACT'. The tree structure is as follows:

- REACT
  - aula01
    - node\_modules
    - public
      - vite.svg
    - src
      - assets
        - react.svg
      - App.css
      - App.tsx
      - index.css
      - main.tsx
      - vite-env.d.ts
    - .eslintrc.cjs
    - .gitignore
    - index.html
    - package-lock.json
    - package.json

The main editor area shows the 'App.tsx' file. The code is as follows:

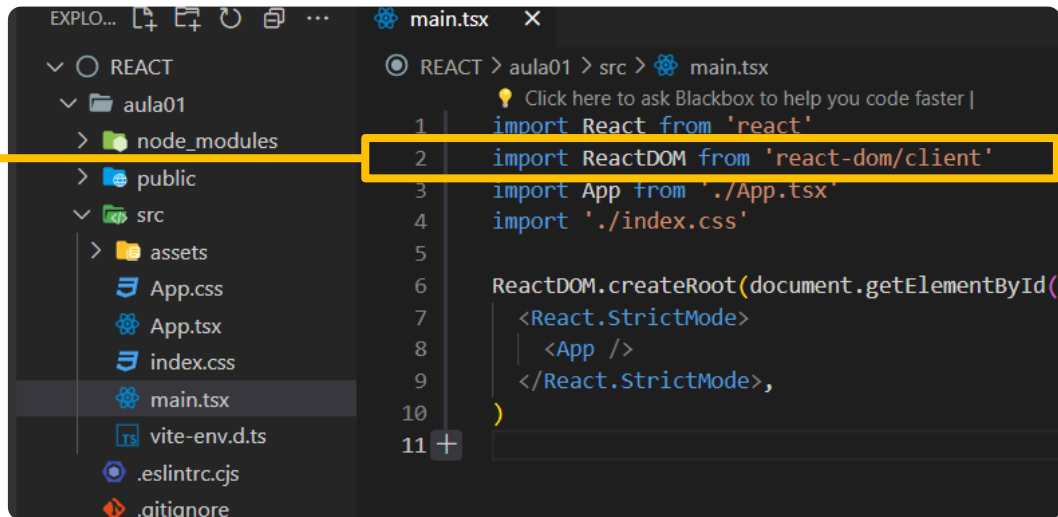
```
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5
6 function App() {
7   const [count, setCount] = useState(0)
8
9   return (
10     <div>
11       <a href="https://vitejs.dev" target="_blank">
12         <img src={viteLogo} className="logo" alt="Vite logo" />
13       </a>
14       <a href="https://react.dev" target="_blank">
15         <img src={reactLogo} className="logo react" alt="React logo" />
16       </a>
17     </div>
18   )
19 }
```

# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ main.tsx

**REACTDOM:** Responsável por transformar o TSX em HTML. Para trabalhar com TSX



```
EXPLO...  main.tsx
└─ ○ REACT
  └─ aula01
    └─ node_modules
    └─ public
    └─ src
      └─ assets
      └─ App.css
      └─ App.tsx
      └─ index.css
      └─ main.tsx
      └─ vite-env.d.ts
      └─ .eslintrc.cjs
      └─ .gitignore

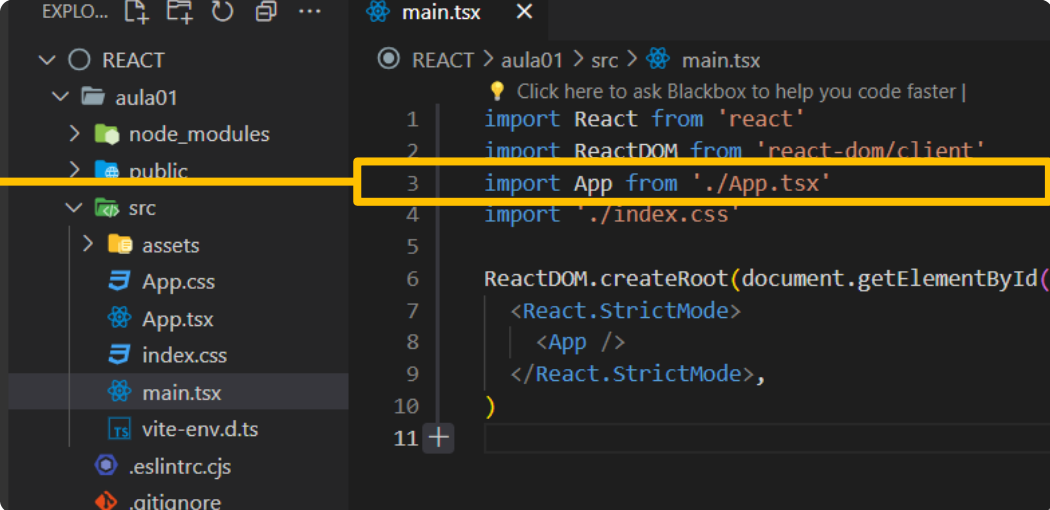
1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import App from './App.tsx'
4  import './index.css'
5
6  ReactDOM.createRoot(document.getElementById(
7    <React.StrictMode>
8      <App />
9    </React.StrictMode>,
10 )
11 +
```

# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ main.tsx

APP: Joga as informações do método ReactDOM.render



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the following structure:

- REACT
  - aula01
    - node\_modules
    - public
    - src
      - assets
        - App.css
        - App.tsx
        - index.css
        - main.tsx
        - vite-env.d.ts
      - .eslintrc.cjs
      - .gitignore

The code editor shows the content of `main.tsx`:

```
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(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>,
10 )
11 +
```

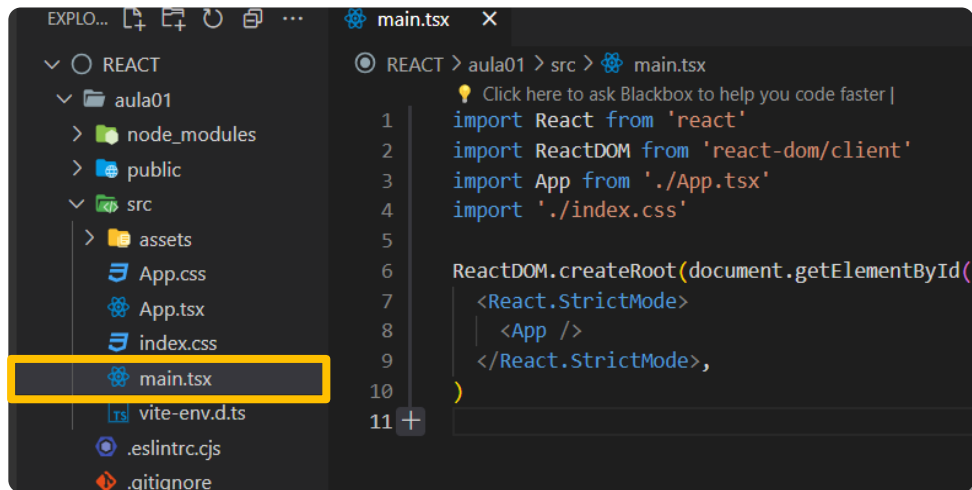
A yellow box highlights the import statement `import App from './App.tsx'` on line 3.

# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ MAIN.TSX

- ✓ **TSX:** A extensão de arquivo “.tsx” é comumente associada a arquivos TypeScript que contêm código React. TypeScript é um superconjunto de JavaScript que adiciona digitação estática à linguagem, e React é uma biblioteca JavaScript para construção de interfaces de usuário. Ao trabalhar com React e TypeScript juntos, os desenvolvedores costumam usar a extensão “.tsx” em seus arquivos para indicar que eles contêm TypeScript e JSX (a extensão de sintaxe do React para JavaScript).

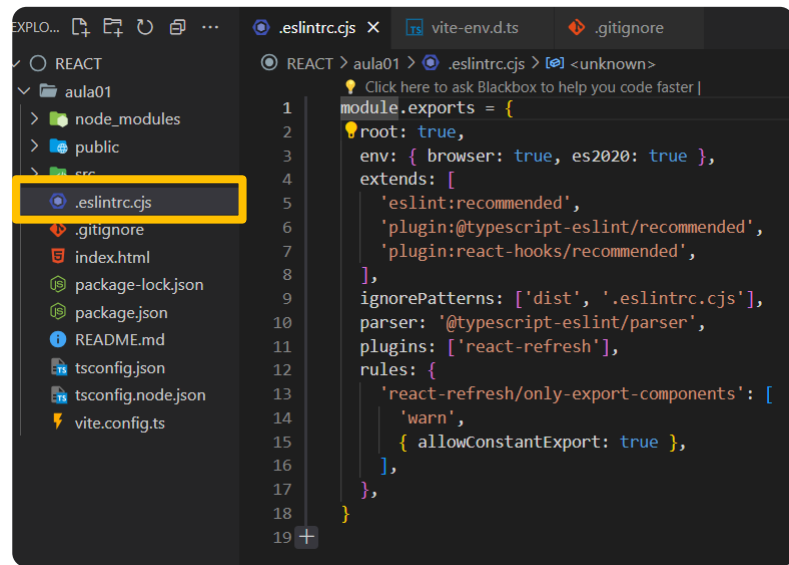


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ .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.

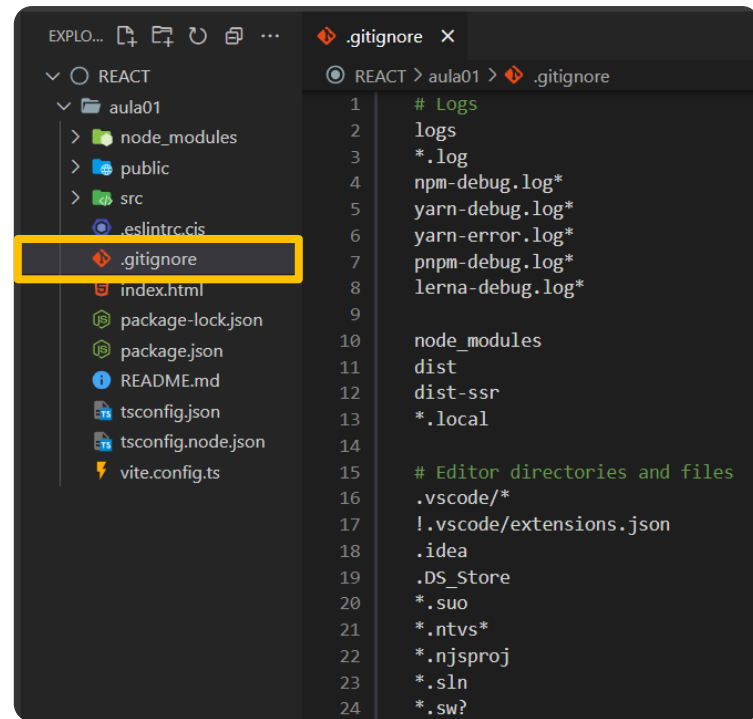


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ .gitignore

- ✓ Para atuar com versionamentos.
- ✓ 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.
- ✓ Os padrões do Git ignore podem ser locais, globais ou compartilhados com a equipe.



```
EXPLO...  .gitignore X
v ○ REACT
  v aula01
    > node_modules
    > public
    > src
    > .eslintrc.cjs
    > .gitignore
    > index.html
    > package-lock.json
    > package.json
    > README.md
    > tsconfig.json
    > tsconfig.node.json
    > vite.config.ts
1  # Logs
2  logs
3  *.log
4  npm-debug.log*
5  yarn-debug.log*
6  yarn-error.log*
7  pnpm-debug.log*
8  lerna-debug.log*
9
10 node_modules
11 dist
12 dist-ssr
13 *.local
14
15 # Editor directories and files
16 .vscode/*
17 !.vscode/extensions.json
18 .idea
19 .DS_Store
20 *.suo
21 *.ntvs*
22 *.njsproj
23 *.sln
24 *.sw?
```

# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Index.html

- ✓ Lembrando que React não é uma linguagem, é uma maneira mais fácil de construir um front com o HTML, CSS e o JS.

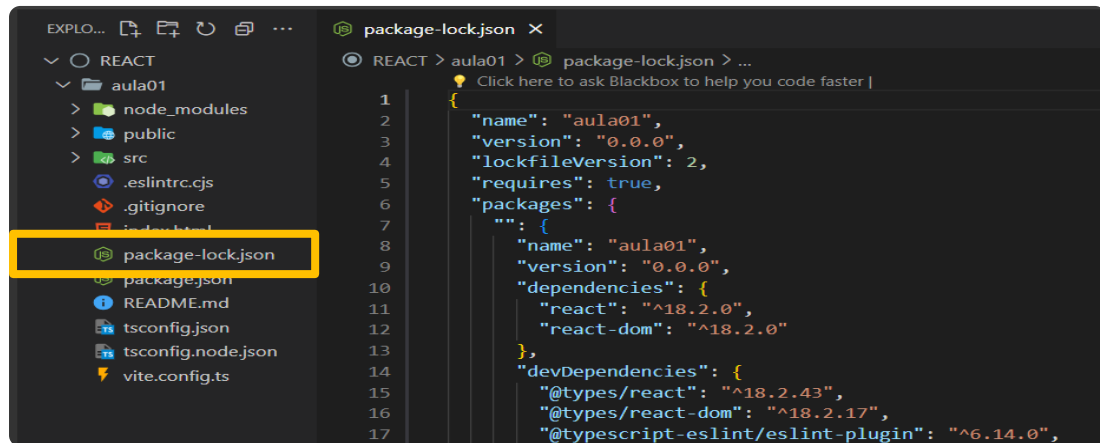


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package-lock.json

- ✓ Além poder ser um problema para você, pode ser um problema para outra pessoa qualquer querendo rodar o seu projeto com **npm install**, podendo acontecer do seu projeto original ficar diferente da nova instalação. Isto implica obviamente em possíveis bugs, mesmo que seja um patch ou minor release.



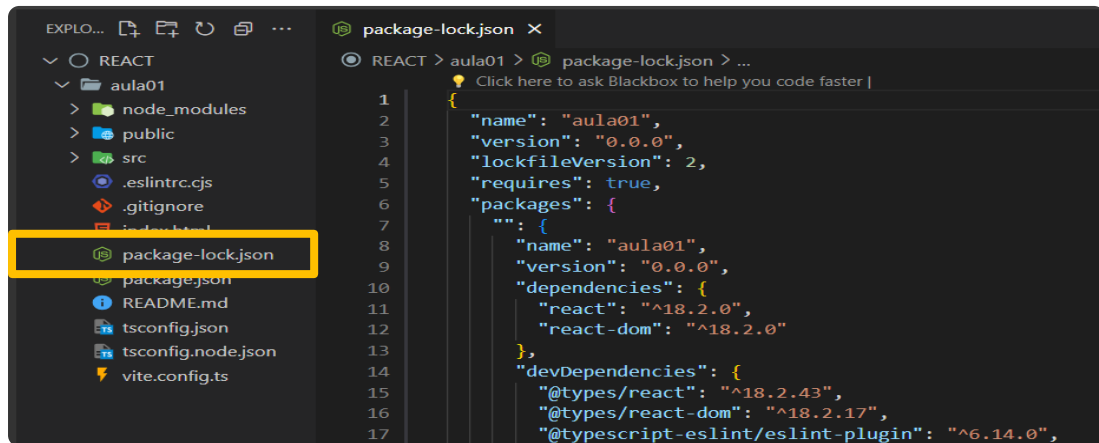


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package-lock.json

- ✓ O arquivo package-lock.json define as versões instaladas de cada pacote de maneira irreversível e o npm usará exatamente estas versões quando você rodar **npm install**.

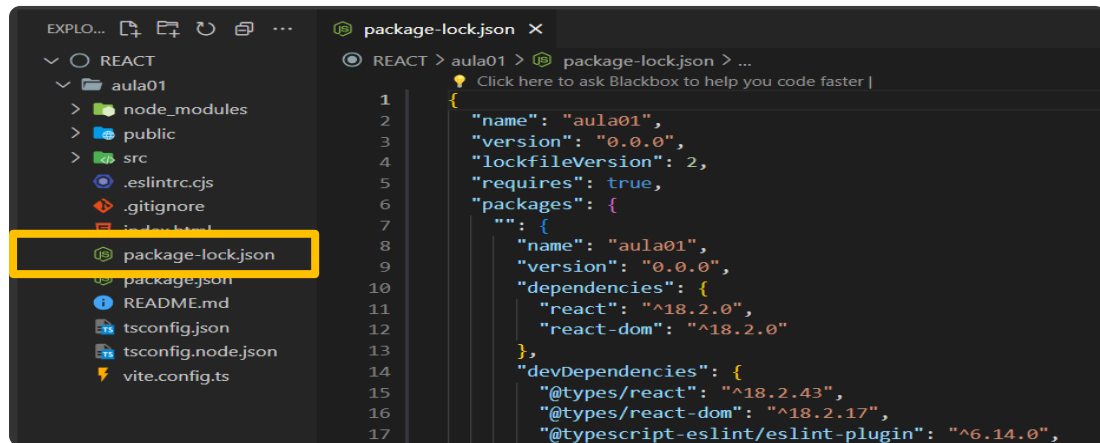


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package-lock.json

- ✓ Este conceito não é novo e outros gerenciadores de pacotes de linguagens de programação (como o Composer do PHP) usam um sistema semelhante por anos.



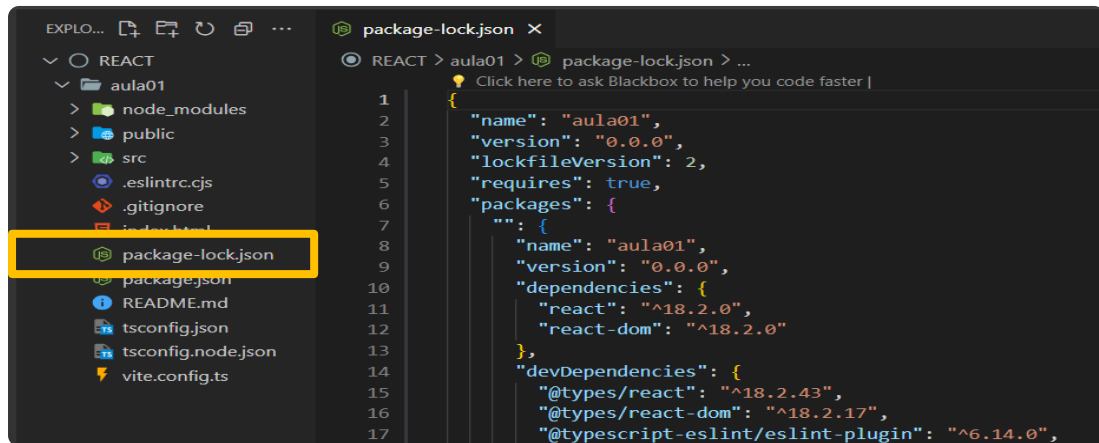
# REACT – INTRODUÇÃO

- ✓ Entendendo a organização e o sistema de arquivos do projeto

- ✓ Package-lock.json

- ✓ Uma lista completa das propriedades, suas definições e sintaxes, pode ser acessada em:

<https://nodejs.reativa.dev/0020-package-lock-json/index>

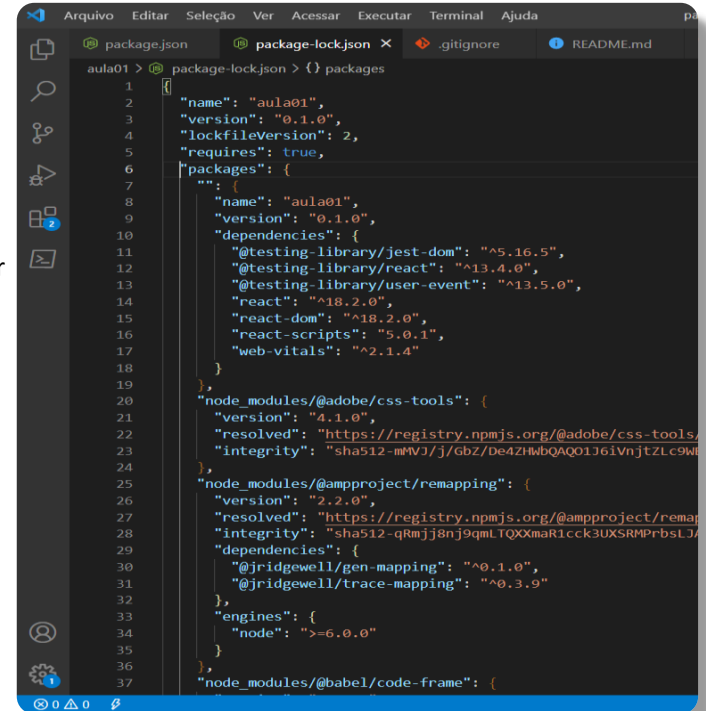


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package-lock.json

- ✓ Na versão 5 o NPM introduziu o arquivo package-lock.json, que é automaticamente gerado quando você instala pacotes Node.
- ✓ O que ele faz? O objetivo deste arquivo é manter registro das **versões** exatas de cada pacote que é instalado para que um produto possa ser 100% reproduzível da mesma maneira mesmo se os pacotes forem atualizados por seus mantenedores.



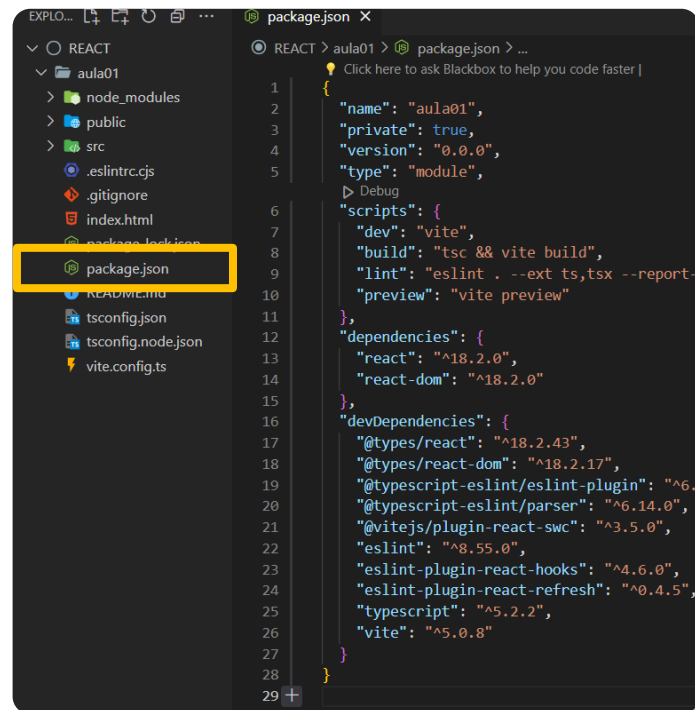
```
1 {
2   "name": "aula01",
3   "version": "0.1.0",
4   "lockfileVersion": 2,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "aula01",
9       "version": "0.1.0",
10      "dependencies": {
11        "@testing-library/jest-dom": "^5.16.5",
12        "@testing-library/react": "^13.4.0",
13        "@testing-library/user-event": "^13.5.0",
14        "react": "^18.2.0",
15        "react-dom": "^18.2.0",
16        "react-scripts": "5.0.1",
17        "web-vitals": "^2.1.4"
18      }
19    },
20    "node_modules/@adobe/css-tools": {
21      "version": "4.1.0",
22      "resolved": "https://registry.npmjs.org/@adobe/css-tools/-/4.1.0",
23      "integrity": "sha512-E98xPpLX87i3e2iL9nB+/B3SfP4YjzAGIYG23dNWsgeN2LbWk5VhRywhJgk2dHr3vabVEUm68Jowj7UOxAg="
24    },
25    "node_modules/@ampproject/remapping": {
26      "version": "2.2.0",
27      "resolved": "https://registry.npmjs.org/@ampproject/remapping/-/2.2.0",
28      "integrity": "sha512-qRkXl9Q5U2mK8Lp09EHx9ahCcDFgf38WJv11Ez8l1kU87B3xNQwyFmYMfBV3N1M/12ZtX1x7OjGVGmGFWA=",
29      "dependencies": {
30        "@jridgewell/gen-mapping": "^0.1.0",
31        "@jridgewell/trace-mapping": "^0.3.9"
32      },
33      "engines": {
34        "node": ">=6.0.0"
35      }
36    },
37    "node_modules/@babel/code-frame": {
```

# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package.json

- ✓ É um descritivo das configurações básicas do projeto e as dependências.
- ✓ O **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.
- ✓ Obs.: Caso abra algum projeto e tenha algum problema na utilização de alguma dependência, no terminal, utilizar a instrução **npm install**.



# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package.json

- ✓ Esse arquivo deve respeitar o formato JSON.
- ✓ Algumas propriedades que encontramos neste arquivo:
  - ✓ **name:** Define o nome do pacote.  
O nome deve ter menos de 214 caracteres, não pode ter espaços, apenas letras minúsculas, hífens (-) ou underlines (\_). Isso ocorre porque quando um pacote é publicado no npm, ele ganha sua própria URL baseada nessa propriedade.

```
2  "name": "aula01",
3  "private": true,
4  "version": "0.0.0",
5  "type": "module",
6  "scripts": {
7    "dev": "vite",
8    "build": "tsc && vite build",
9    "lint": "eslint . --ext ts,tsx --report",
10   "preview": "vite preview"
11 },
12 "dependencies": {
13   "react": "^18.2.0",
14   "react-dom": "^18.2.0"
15 },
16 "devDependencies": {
17   "@types/react": "^18.2.43",
18   "@types/react-dom": "^18.2.17",
19   "@typescript-eslint/eslint-plugin": "^6.
20   "@typescript-eslint/parser": "^6.14.0",
21   "@vitejs/plugin-react-swc": "^3.5.0",
22   "eslint": "^8.55.0",
23   "eslint-plugin-react-hooks": "^4.6.0",
24   "eslint-plugin-react-refresh": "^0.4.5",
25   "typescript": "^5.2.2",
26   "vite": "^5.0.8"
27 }
28
29
```

# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package.json

- ✓ Esse arquivo deve respeitar o formato JSON.
- ✓ Algumas propriedades que encontramos neste arquivo:
  - ✓ **private**: Se definido como true, previne do pacote ser publicado acidentalmente no npm

```
1  {
2    "name": "aula01",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "tsc && vite build",
9      "lint": "eslint . --ext ts,tsx --report",
10     "preview": "vite preview"
11   },
12   "dependencies": {
13     "react": "^18.2.0",
14     "react-dom": "^18.2.0"
15   },
16   "devDependencies": {
17     "@types/react": "^18.2.43",
18     "@types/react-dom": "^18.2.17",
19     "@typescript-eslint/eslint-plugin": "^6.14.0",
20     "@typescript-eslint/parser": "^6.14.0",
21     "@vitejs/plugin-react-swc": "^3.5.0",
22     "eslint": "^8.55.0",
23     "eslint-plugin-react-hooks": "^4.6.0",
24     "eslint-plugin-react-refresh": "^0.4.5",
25     "typescript": "^5.2.2",
26     "vite": "^5.0.8"
27   }
28 }
```

# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package.json

- ✓ Esse arquivo deve respeitar o formato JSON.
- ✓ Algumas propriedades que encontramos neste arquivo:
  - ✓ **version**: Indica a versão atual do pacote.  
Essa propriedade segue o padrão de notação de versionamento semântico chamado **semver**, o que significa que é sempre expresso com 3 números: x.x.x.
  - ✓ O primeiro número é sempre a versão **major**, o segundo a versão **minor** e o terceiro a versão **patch**.

```
1  {
2    "name": "aula01",
3    "version": "0.0.0",
4    "scripts": {
5      "dev": "vite",
6      "build": "tsc && vite build",
7      "lint": "eslint . --ext ts,tsx --report",
8      "preview": "vite preview"
9    },
10   "dependencies": {
11     "react": "^18.2.0",
12     "react-dom": "^18.2.0"
13   },
14   "devDependencies": {
15     "@types/react": "^18.2.43",
16     "@types/react-dom": "^18.2.17",
17     "@typescript-eslint/eslint-plugin": "^6.14.0",
18     "@typescript-eslint/parser": "^6.14.0",
19     "@vitejs/plugin-react-swc": "^3.5.0",
20     "eslint": "^8.55.0",
21     "eslint-plugin-react-hooks": "^4.6.0",
22     "eslint-plugin-react-refresh": "^0.4.5",
23     "typescript": "^5.2.2",
24     "vite": "^5.0.8"
25   }
26 }
```

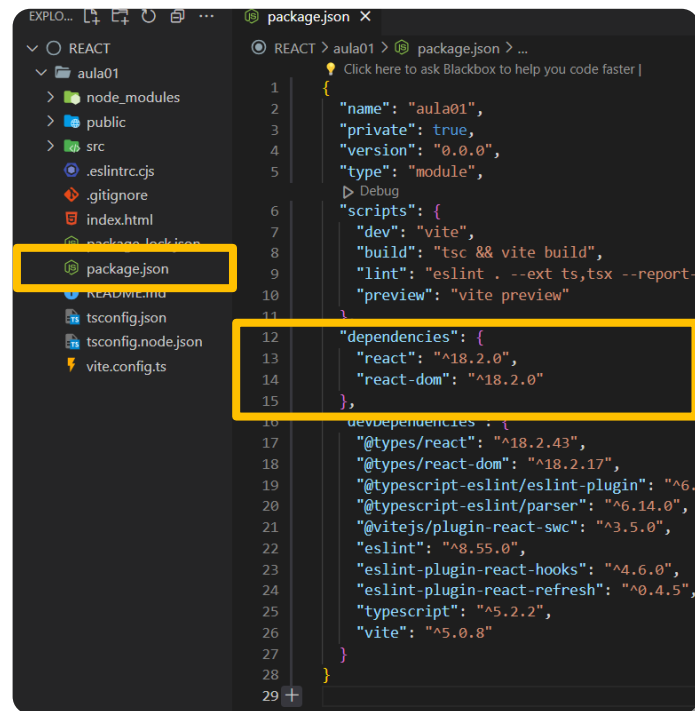


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package.json

- ✓ Esse arquivo deve respeitar o formato JSON.
- ✓ Algumas propriedades que encontramos neste arquivo:
  - ✓ **dependencies:** Define uma lista de pacotes npm instalados como dependências.
  - ✓ Quando você instala um pacote usando npm ou yarn



```
1  {
2    "name": "aula01",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "tsc && vite build",
9      "lint": "eslint . --ext ts,tsx --report-
10     "preview": "vite preview"
11   },
12   "dependencies": {
13     "react": "^18.2.0",
14     "react-dom": "^18.2.0"
15   },
16   "devDependencies": {
17     "@types/react": "^18.2.43",
18     "@types/react-dom": "^18.2.17",
19     "@typescript-eslint/eslint-plugin": "^6.
20     "@typescript-eslint/parser": "^6.14.0",
21     "@vitejs/plugin-react-swc": "^3.5.0",
22     "eslint": "^8.55.0",
23     "eslint-plugin-react-hooks": "^4.6.0",
24     "eslint-plugin-react-refresh": "^0.4.5",
25     "typescript": "^5.2.2",
26     "vite": "^5.0.8"
27   }
28 }
```

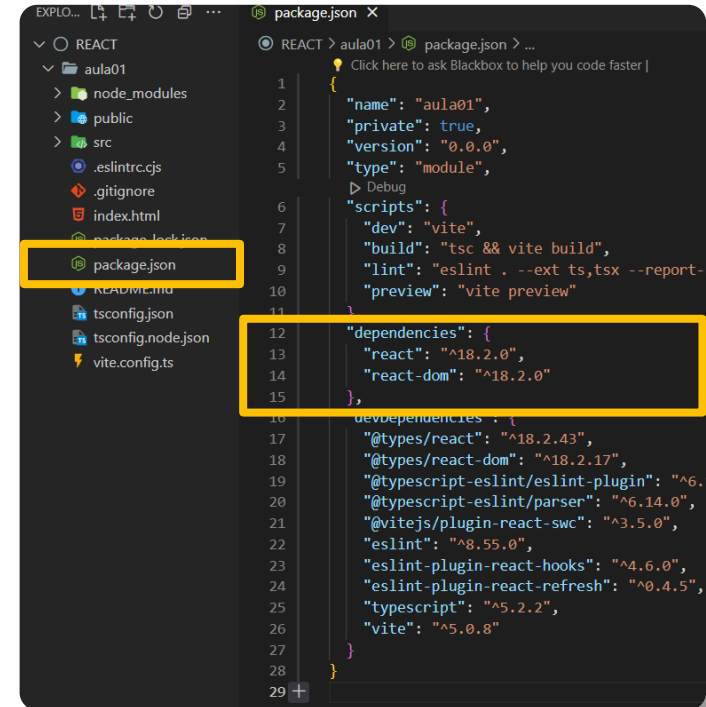
# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ Package.json

- ✓ Neste conteúdo citamos uma pequena lista das propriedades que este arquivo pode ter.
- ✓ Uma lista completa das propriedades, suas definições e sintaxes, pode ser acessada em:

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



```
1  {
2    "name": "aula01",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "tsc && vite build",
9      "lint": "eslint . --ext ts,tsx --report",
10     "preview": "vite preview"
11   },
12   "dependencies": {
13     "react": "^18.2.0",
14     "react-dom": "^18.2.0"
15   },
16   "devDependencies": {
17     "@types/react": "^18.2.43",
18     "@types/react-dom": "^18.2.17",
19     "@typescript-eslint/eslint-plugin": "^6.14.0",
20     "@typescript-eslint/parser": "^6.14.0",
21     "@vitejs/plugin-react-swc": "^3.5.0",
22     "eslint": "^8.55.0",
23     "eslint-plugin-react-hooks": "^4.6.0",
24     "eslint-plugin-react-refresh": "^0.4.5",
25     "typescript": "^5.2.2",
26     "vite": "^5.0.8"
27   }
28 }
```

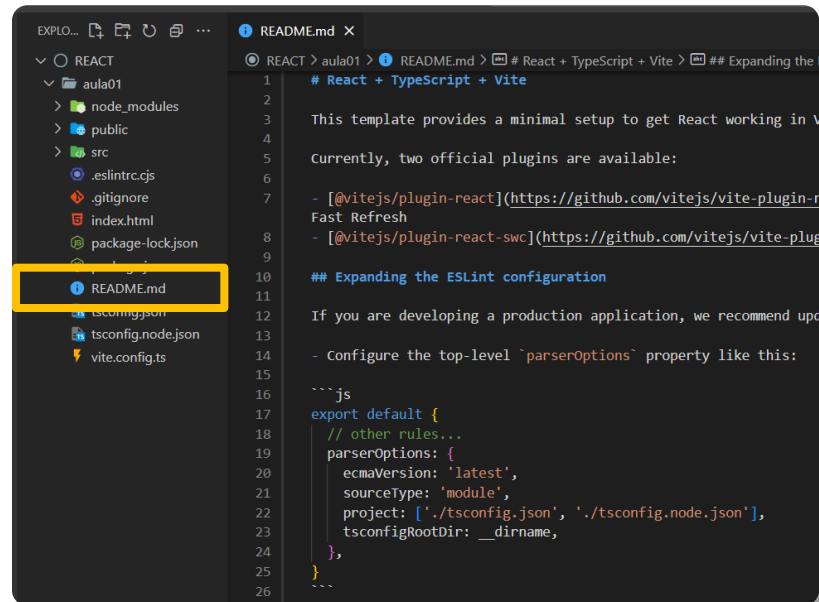
# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ README.md

✓ Mais utilizado para se fazer Commit do projeto, utilizado em projetos no GitHub.

✓ É um arquivo markdown que contém muitas informações úteis sobre o Create React App, tal como um resumo de comandos e links para configurações avançadas.

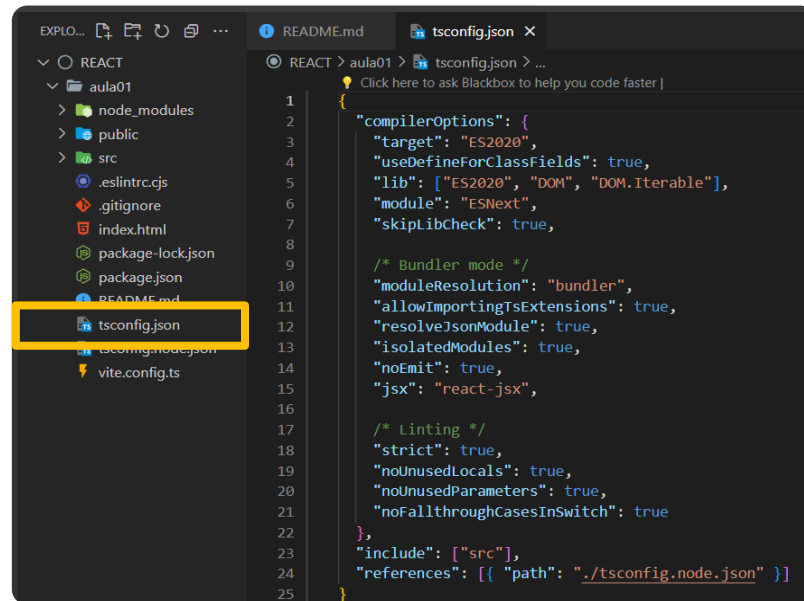


# REACT – INTRODUÇÃO

## ✓ Entendendo a organização e o sistema de arquivos do projeto

### ✓ tsconfig.json e tsconfig.node.json

- ✓ são os arquivos de configuração do TypeScript que definem as opções de compilação para o seu código.
- ✓ O tsconfig.json é usado para o código da aplicação, enquanto o tsconfig.node.json pode ser usado para configurar o TypeScript em ambientes Node.js;



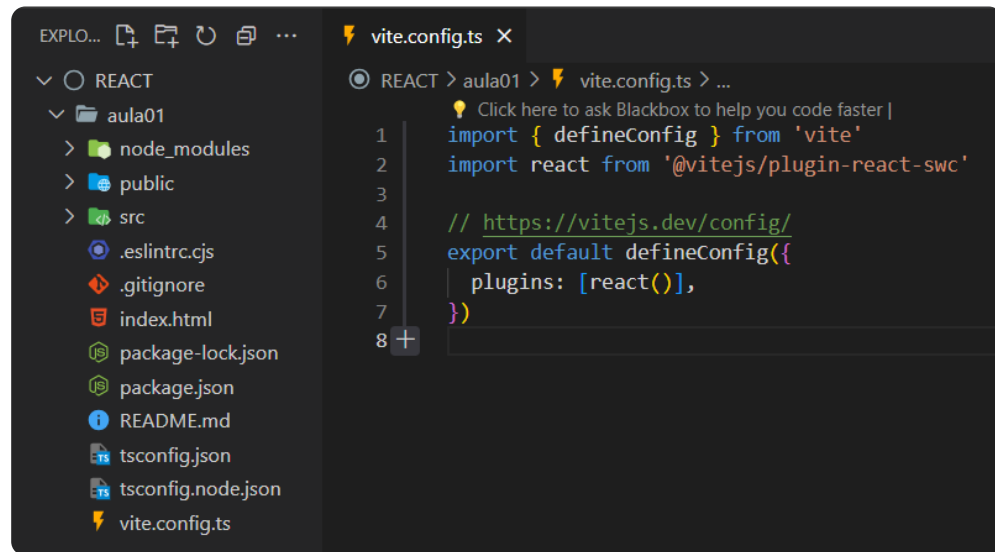
```
EXPLO...  README.md  tsconfig.json x
v O REACT
  v aula01
    > node_modules
    > public
    > src
      .eslintrc.js
      .gitignore
      index.html
      package-lock.json
      package.json
      README.md
      tsconfig.json
      tsconfig.node.json
      vite.config.ts
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
  {
    "compilerOptions": {
      "target": "ES2020",
      "useDefineForClassFields": true,
      "lib": ["ES2020", "DOM", "DOM.Iterable"],
      "module": "ESNext",
      "skipLibCheck": true,

      /* Bundler mode */
      "moduleResolution": "bundler",
      "allowImportingTsExtensions": true,
      "resolveJsonModule": true,
      "isolatedModules": true,
      "noEmit": true,
      "jsx": "react-jsx",

      /* Linting */
      "strict": true,
      "noUnusedLocals": true,
      "noUnusedParameters": true,
      "noFallthroughCasesInSwitch": true
    },
    "include": ["src"],
    "references": [{ "path": "./tsconfig.node.json" } ]
  }
```

# REACT – INTRODUÇÃO

- ✓ **Entendendo a organização e o sistema de arquivos do projeto**
- ✓ **tsconfig.json e tsconfig.node.json**
  - ✓ esse arquivo é usado para configurar o Vite. Ele pode conter configurações relacionadas a plugins, roteamento, aliases de importação, entre outras coisas.



The screenshot shows a code editor with a dark theme. On the left, the file explorer displays the project structure: a folder named 'REACT' containing a sub-folder 'aula01'. Inside 'aula01', there are several files: 'node\_modules', 'public', 'src', '.eslintrc.cjs', '.gitignore', 'index.html', 'package-lock.json', 'package.json', 'README.md', 'tsconfig.json', 'tsconfig.node.json', and 'vite.config.ts'. The 'vite.config.ts' file is selected and open in the editor. The code in the editor is as follows:

```
1 import { defineConfig } from 'vite'
2 import react from '@vitejs/plugin-react-swc'
3
4 // https://vitejs.dev/config/
5 export default defineConfig({
6   plugins: [react()],
7 })
8 +
```

# REFERÊNCIAS BIBLIOGRÁFICAS

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](https://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](https://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](https://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](https://www.hackingwithreact.com/read/1/3/introduction-to-jsx)>. Acesso em: 13 janeiro de 2023.

# REFERÊNCIAS BIBLIOGRÁFICAS

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.

# REFERÊNCIAS BIBLIOGRÁFICAS

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

\_\_\_\_\_. 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-CSS2-20160412/>>. 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.



# REFERÊNCIAS BIBLIOGRÁFICAS

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?
    - Sugestões?
      - Ameaças?