

React JS

★ AULA 02 ★

★ PROFESSOR
MAROMO ★

React JS

Componentes

Criando os primeiros componentes



Agenda

- **Conceitos sobre componentes que serão trabalhados**
 - **1.Criar Componentes**
 - **2.Exportar Componentes**
 - **3.Importar Componentes**
 - **4.Estilizar e Organizar Componentes**



Componentes

Componentes representam fundamentalmente elementos visuais que são apresentados ao usuário na interface da tela. Considerando, por exemplo, uma página Web, cada elemento de interação, seja um botão, um item do menu ou até mesmo um campo para edição de dados, são definidos como componentes. Além disso, é importante salientar que a página Web como um todo, com todas as suas partes interconectadas, também é caracterizada como um componente - este que abriga seus próprios componentes filhos.



Componentes no React

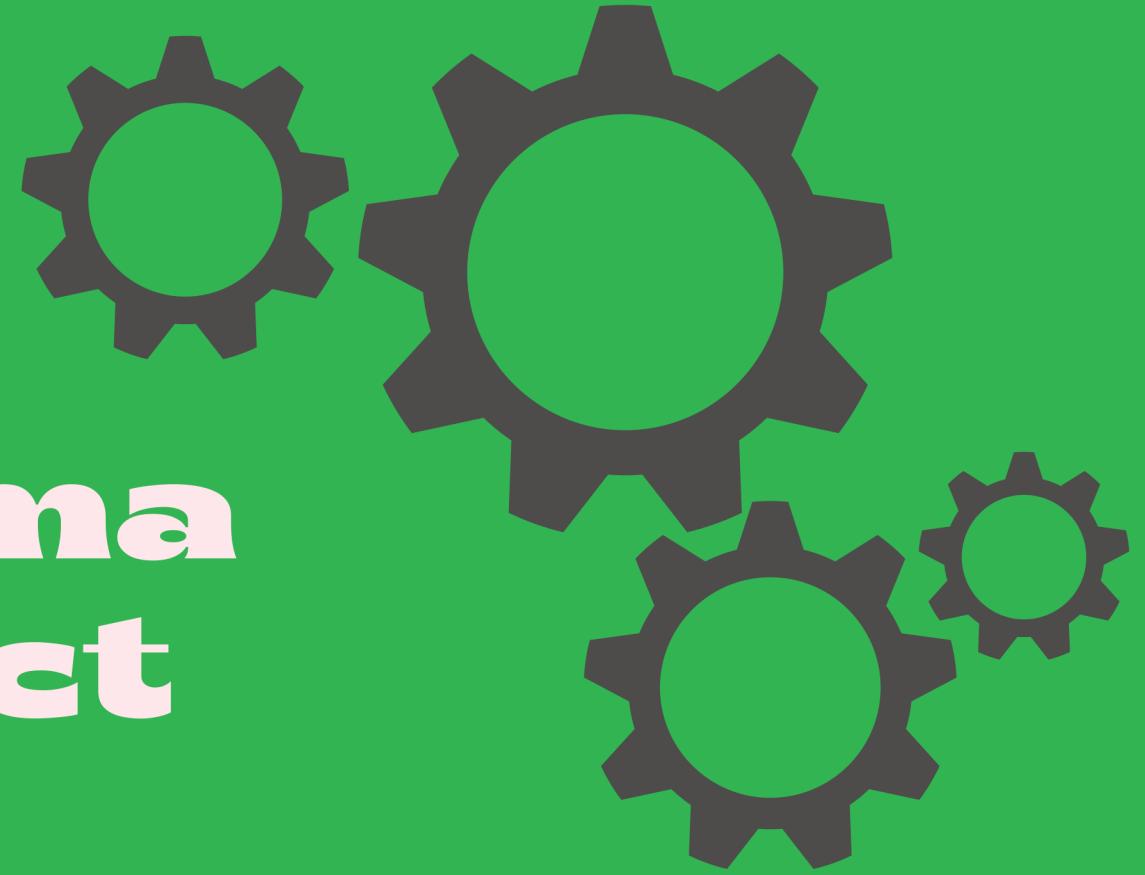
No universo do React, cada elemento visual é considerado um componente. A adoção de componentes se mostra extremamente útil para a manutenção da aplicação, visto que permite a divisão de um código complexo em partes mais digeríveis e menores, facilitando a leitura e a gestão. Outro benefício notável é a possibilidade de reutilizar nosso código, maximizando a eficiência do desenvolvimento.



Pais e Filhos

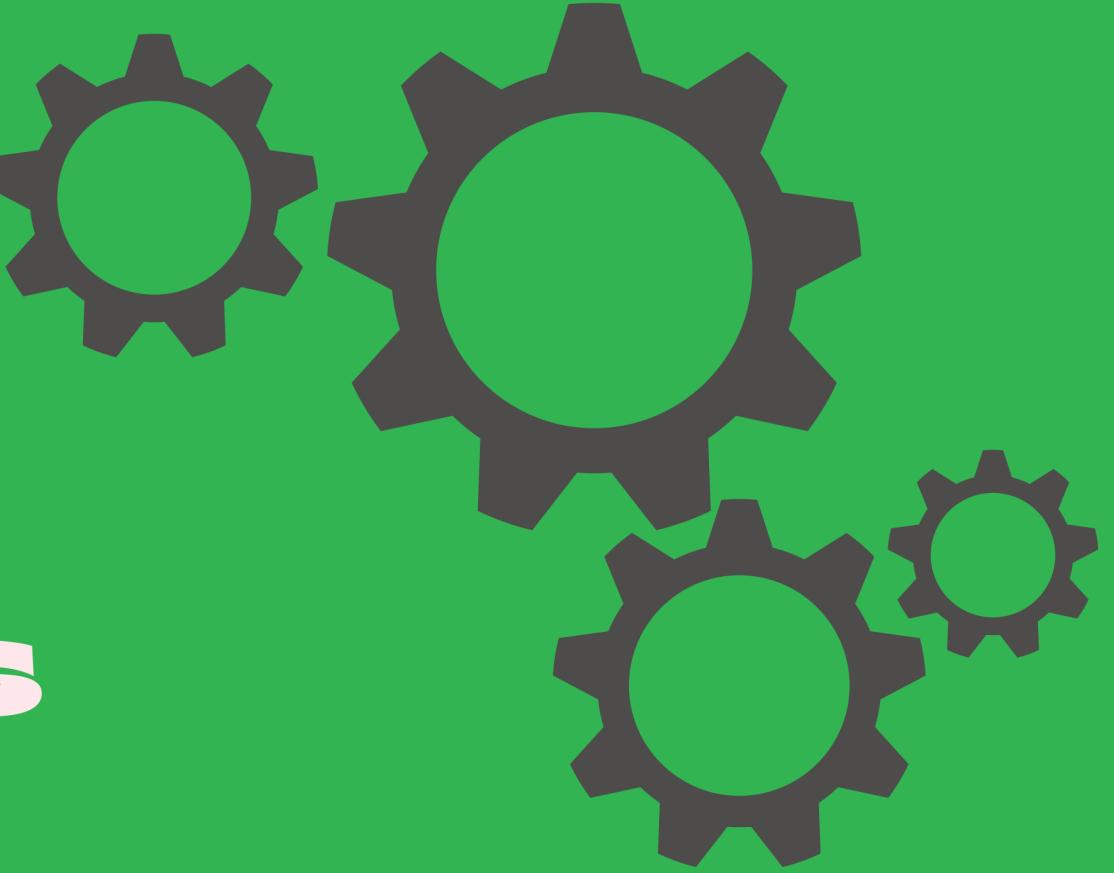
A tarefa de desenvolver uma aplicação em React implica essencialmente na concepção dos primeiros componentes 'filhos'.

Posteriormente, esses componentes são 'importados' e integrados ao componente 'pai' principal - o componente fundamental que representa a totalidade da página. Vale salientar que, neste exemplo, estamos tratando de uma aplicação que consiste em uma única página.



Mecânica de uma aplicação React

1. Primeiro você vai codificar o seu componente (por exemplo, um botão na tela);
2. Com o componente codificado, o segundo passo é exportar esse componente - o que significa deixá-lo exposto para que possa ser usado na sua página;
3. Por último você vai 'importar' o componente previamente exposto e dessa forma apresentá-lo na tela.



Em termos mais técnicos

- 1.Criar o componente filho;
- 2.Exportar o componente filho;
- 3.Importar o componente filho dentro do componente pai
(neste exemplo, na própria página).

passo I



Criar
componente
filho



Componente

Header

Componente

Card

Componente

Footer

passo 2



Exportar o
componente
filho



passo 3



Importar o
componente
filho dentro do
pai



Arquitetura do exemplo



Arquitetura do Exemplo

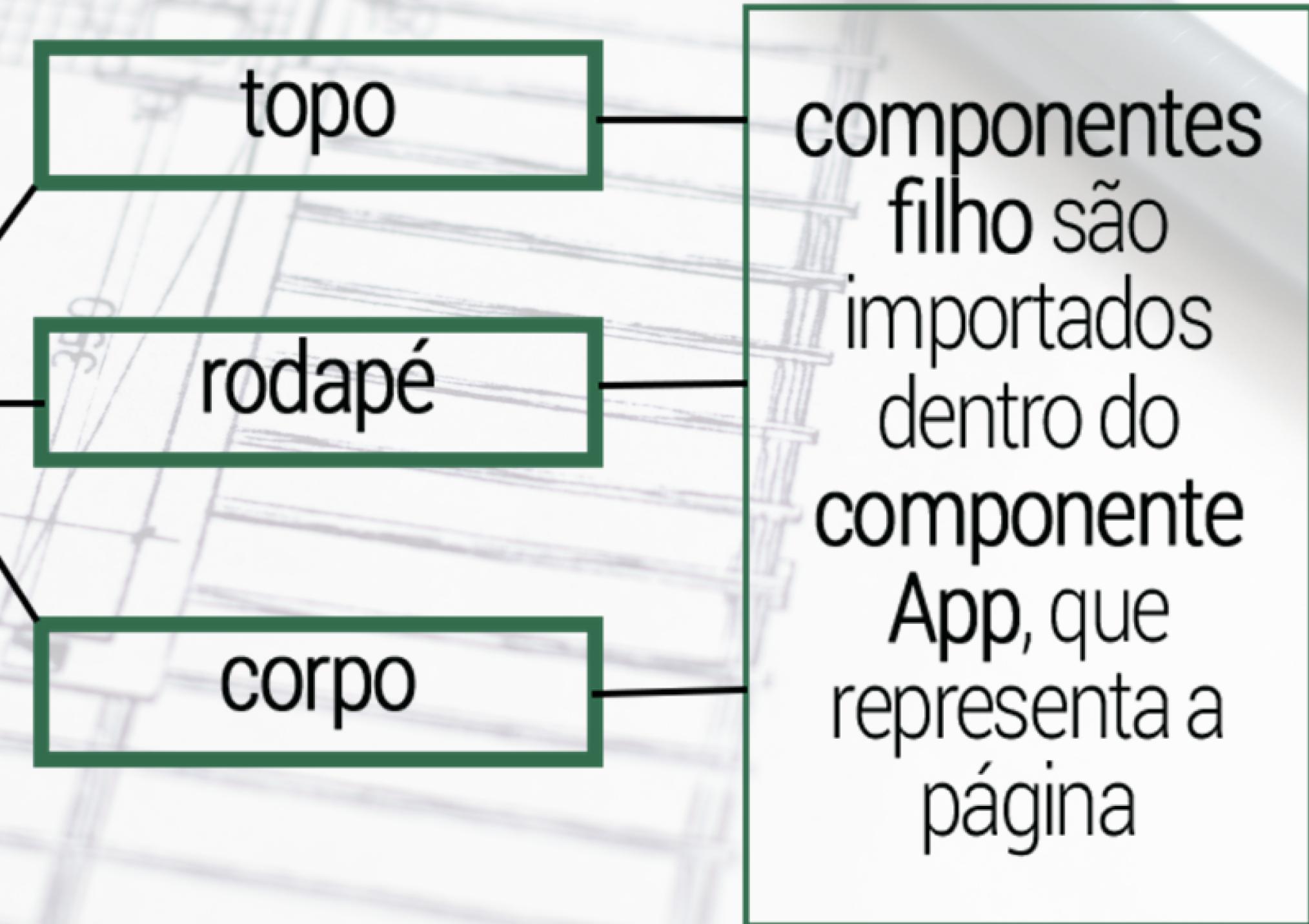


É um exemplo simples, contendo apenas uma página Web com alguns componentes estáticos.

Não há entrada de dados do usuário, apenas a exibição de componentes visuais, tais como topo, rodapé, card.

A arquitetura deste exemplo consiste na mecânica de importação de componentes filho dentro de um componente pai principal, chamado App, como mostra a Figura.

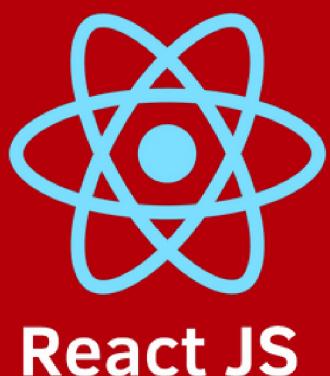
Componentes filhos



Laboratório



Desenvolvimento de uma aplicação SPA usando o React JS. Nome da aplicação **sample02**.

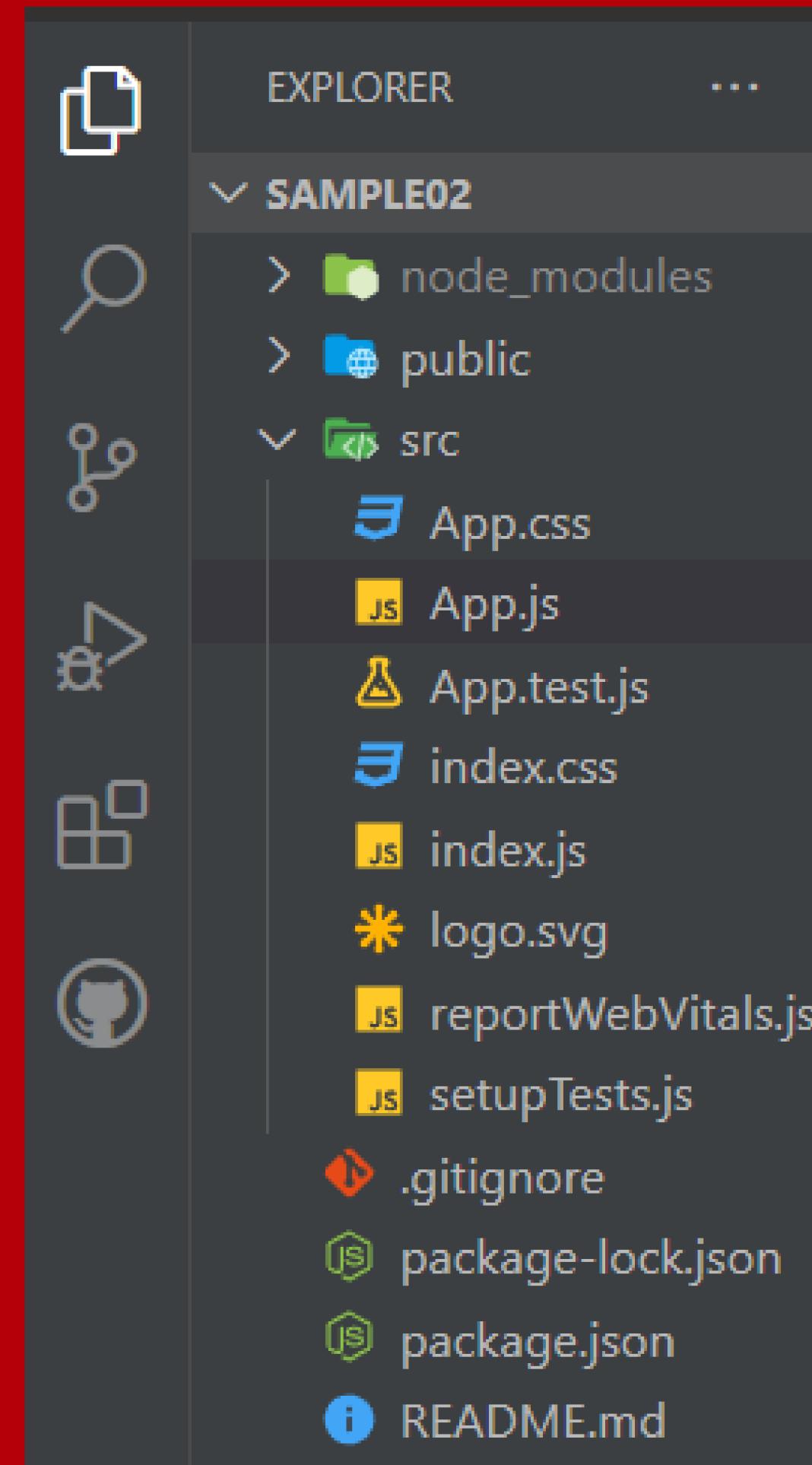




Passo 1: Configuração do ambiente de desenvolvimento

1. Execute o comando `npx create-react-app sample02` para criar um novo projeto React chamado "sample02". Aguarde até que a criação do projeto seja concluída.
2. O processo anterior, de criação do projeto, pode levar alguns instantes enquanto create-react-app monta todo o **boilerplate** mínimo que uma aplicação React para a web precisa ter.
3. Entre no diretório [cd sample02] e digite [code .] para abrir o VS Code.

Passo 2: Estrutura do projeto React





Antes de começarmos a criar nossos componentes filho como o Header, o Footer, o Card e até mesmo a página HomePage, precisamos lembrar que eles farão parte de um componente maior, como vemos na Figura.





Passo 3: Preparando o componente pai

Esse componente é chamado **App** e já foi criado

junto com o projeto na aula anterior.

O que precisamos fazer é remover o conteúdo que foi automaticamente criado dentro dele e inserir o nosso código.

- É o código estrutural do componente **App**. Por padrão esse é o primeiro componente a ser exibido na aplicação e será nosso componente pai.

- Agora, com nosso componente pai pronto, podemos prosseguir e criar o primeiro componente filho, Header.

```
src > JS App.js > [e] default
1 | import React from "react";
2 | import './App.css';
3 |
4 | function App(){
5 |   return (
6 |     <div className="App">
7 |       </div>
8 |     );
9 |   }
10| }
11| export default App;
```



Passo 4: Preparando os filhos

Componente Header

Com o nosso componente App criado, agora podemos começar a construir os componentes filho que vão compor nossa página. Nessa aula vamos criar o componente Header, como mostra a Figura.

The diagram illustrates a React component structure for a Header. At the top, a green arrow points to the title "Componente Header". Below it is a red header bar containing the text "Trabalhando com componentes". The main content area is a white box with the heading "Esse é o topo da nossa aplicação". This section is divided into three columns, each containing a "Componentes:" heading and a descriptive paragraph. At the bottom of the white box is a red footer bar with the text "Desenvolvido com React".

Componente Header

Trabalhando com componentes

Esse é o topo da nossa aplicação

Componentes:
Facilita em manter partes menores funcionando corretamente
Você pode reutilizá-los, ou seja, menos código para se escrever

Componentes:
Facilita em manter partes menores funcionando corretamente
Você pode reutilizá-los, ou seja, menos código para se escrever

Componentes:
Facilita em manter partes menores funcionando corretamente
Você pode reutilizá-los, ou seja, menos código para se escrever

Desenvolvido com React



Passo 4: Preparando os filhos

Componente Header

Para organizar melhor os arquivos do nosso código vamos criar uma pasta chamada '**components**' dentro do diretório **src**.

Dentro do diretório componentes vamos criar uma pasta chamada **Header** e nela um arquivo chamado **index.js**. Esse arquivo terá o código do nosso componente Header, como mostra a Figura.

```
> node_modules
> public
< src
  < components\Header
    index.js
  App.css
  App.js
  App.test.js
  index.css
```



Passo 4: Preparando os filhos

Componente Header

Pasta Header, arquivo: index.js

```
src > components > Header > js index.js > [d] default

1 import React from "react";
2
3 function Header(){
4     return(
5         <header>Trabalhando com componentes</header>
6     );
7 }
8
9 export default Header;
```

Nota (1/2)



Ao darmos vida a um novo componente - ou seja, um elemento que será projetado na tela - é fundamental importar o componente React da dependência 'react'. Contudo, note que essa operação não é mais necessária, pois o processo se tornou automático.

Adotar essa prática permite ao React interpretar corretamente o código, reconhecendo-o como um componente destinado a ser renderizado na tela. Isso faz parte da mecânica padrão no processo de construção de componentes utilizando React.



Nota (2/2)

Em seguida, observamos uma função JavaScript que porta o mesmo nome do componente em questão - este é um comportamento padrão na codificação de componentes em React.

Todo componente é obrigado a retornar algo que será renderizado na tela. Este retorno é executado por meio da palavra-chave 'return'. Dentro deste 'return', encontramos uma tag JSX <header> (veja a nota) com o texto "Trabalhando com componentes".

Com o componente finalmente criado, é necessário exportá-lo para permitir que o componente App possa utilizá-lo. Esta exportação é realizada através do uso da palavra-chave 'export'.



Passo 5: Preparando os filhos

Componente Footer

Dentro do diretório components vamos criar uma pasta chamada **Footer** e nela um arquivo chamado **index.js**. Esse arquivo terá o código do nosso componente Header, como mostra a Figura.

```
> node_modules
> public
< src
  < components
    index.js
  < Header
    index.js
```



Passo 5: Preparando os filhos

Componente Footer

Pasta Footer, arquivo: index.js

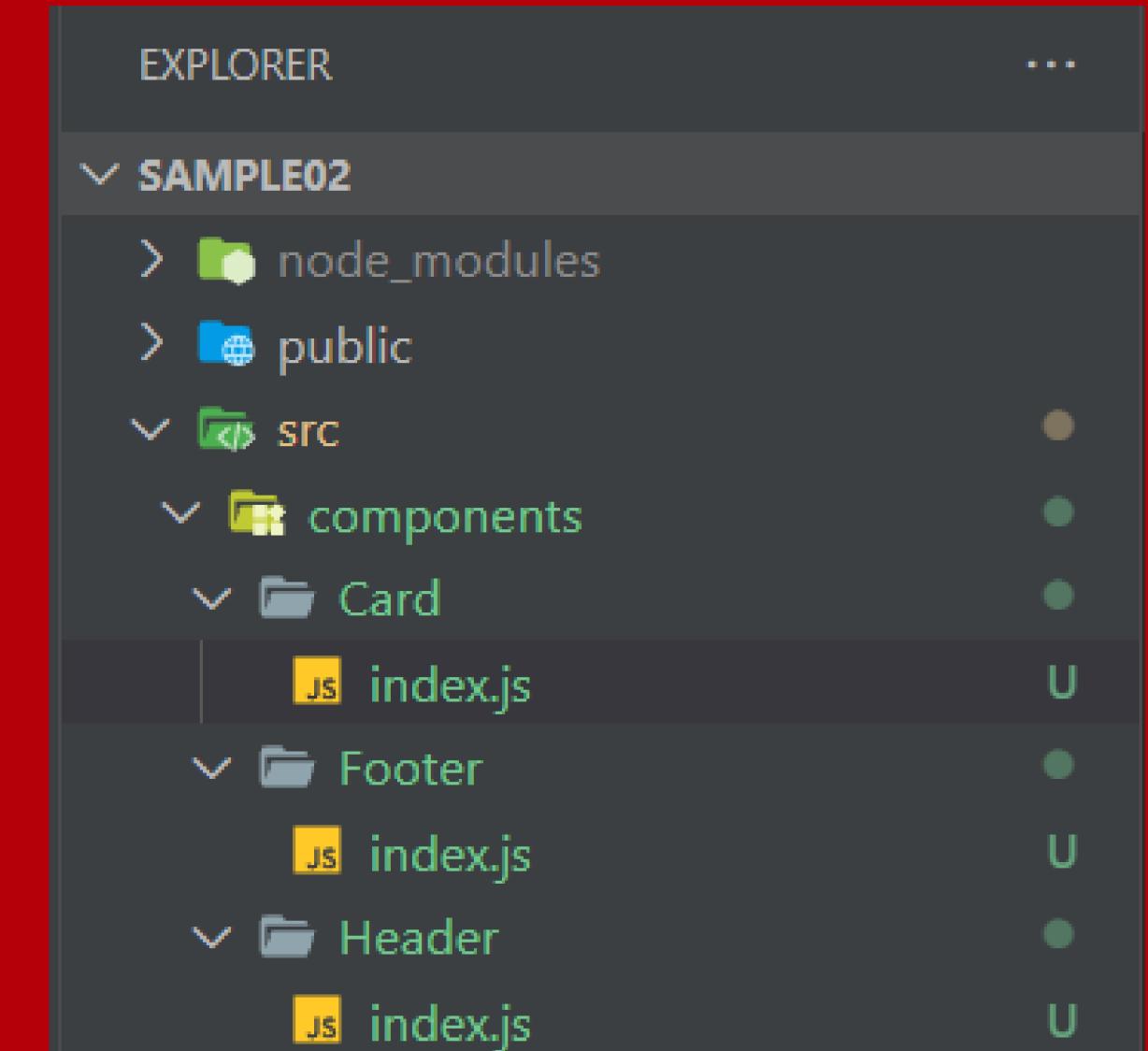
```
src > components > Footer > js index.js > ...
1 import React from "react";
2
3 function Footer(){
4     return (
5         <footer>Desenvolvimento com React</footer>
6     );
7 }
8
9 export default Footer;
```



Passo 6: Preparando os filhos

Componente Card

Dentro do diretório components vamos criar uma pasta chamada **Card** e nela um arquivo chamado **index.js**. Esse arquivo terá o código do nosso componente **Card**, como mostra a Figura.





Passo 6: Preparando os filhos

Componente Card

Pasta Card, arquivo: index.js

```
src > components > Card > JS index.js > ...
1 import React from "react";
2
3 function Card(){
4     return(
5         <div>
6             <p>Componentes:</p>
7             <p>Facilita manter partes menores funcionando</p>
8             <p>Você pode reutilizá-los, ou seja, reaproveitamento</p>
9         </div>
10    );
11 }
12 export default Card;
```



Passo 6: Preparando os filhos

Componente Card

Pasta Card, arquivo: index.js

```
src > components > Card > JS index.js > ...
1 import React from "react";
2
3 function Card(){
4     return(
5         <div>
6             <p>Componentes:</p>
7             <p>Facilita manter partes menores funcionando</p>
8             <p>Você pode reutilizá-los, ou seja, reaproveitamento</p>
9         </div>
10    );
11 }
12 export default Card;
```



Passo 7: Importando componentes criados

Com nossos dois componentes criados (**Header, Card e Footer**) chegou a hora de exibi-los no navegador.

A princípio **não vamos nos preocupar com a estilização e sim em aprender a parte lógica**. No final deste curso vamos estilizar toda nossa aplicação.

Nesta aula vamos importar os componentes **Header, Card e Footer** para nosso componente principal **App**.

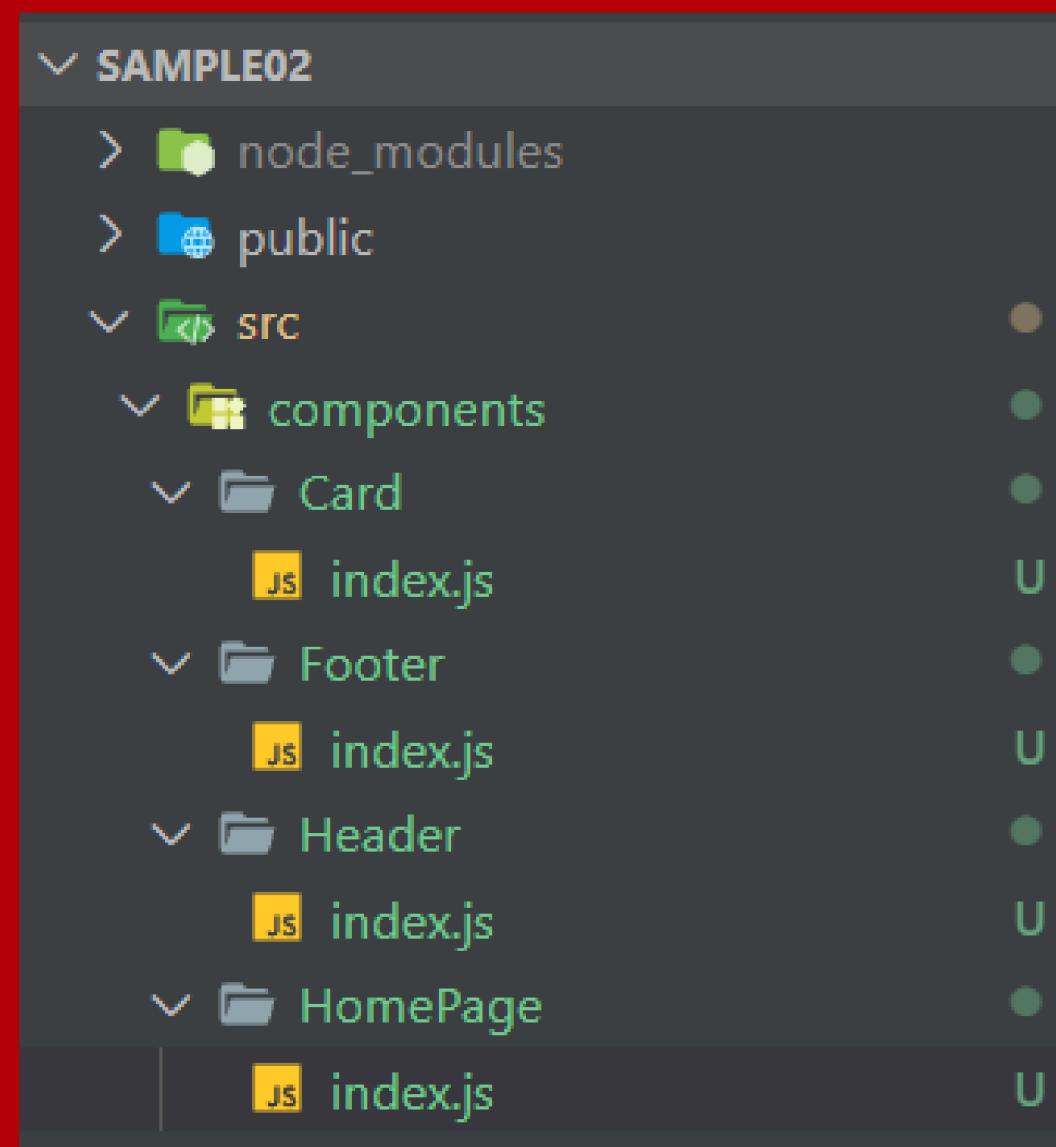


Passo 8: Criando nossa HomePage

Componente:

HomePage - index.js

Crie uma pasta com o nome **HomePage** e dentro um arquivo chamado **index.js** e insira o seguinte código no arquivo que acabamos de criar, como vemos no Código.



```
src > components > HomePage > JS index.js > [e] default  
1 import React from "react";  
2 import Card from "../Card";  
3  
4 function HomePage(){  
5     return(  
6         <div>  
7             <Card />  
8             <Card />  
9             <Card />  
10        </div>  
11    );  
12 }  
13  
14 export default HomePage;
```



Passo 9: Importando componente filho HomePage no arquivo principal App.js:

Com nosso último componente criado só precisamos importá-lo no componente principal e dessa forma teremos o seguinte resultado apresentado na Figura.

```
src > JS App.js > ...
1 import React from "react";
2 import Header from './components/Header';
3 import Footer from './components/Footer';
4 import './App.css';
5 import HomePage from './components/HomePage';
6 function App(){
7   return (
8     <div className="App">
9       <Header />
10      <HomePage />
11      <Footer />
12    </div>
13  );
14}
15 export default App;
```



Resultado da Execução

React App

localhost:3000

110%

Introdução Amazon Booking.com (1) #77 - App IMCCalc... Elai.io - Create AI vide... (6) Flutter ChatGPT us...

Trabalhando com componentes

Componentes:

Facilita manter partes menores funcionando

Você pode reutilizá-los, ou seja, reaproveitamento

Componentes:

Facilita manter partes menores funcionando

Você pode reutilizá-los, ou seja, reaproveitamento

Componentes:

Facilita manter partes menores funcionando

Você pode reutilizá-los, ou seja, reaproveitamento

Desenvolvimento com React



A black and white photograph of a person from the side, wearing a dark, zip-up jacket with a high collar. They are standing in front of a wall made of large, light-colored rectangular tiles. The person's hair is dark and messy. A green speech bubble shape is positioned in the upper right corner of the image, containing white text.

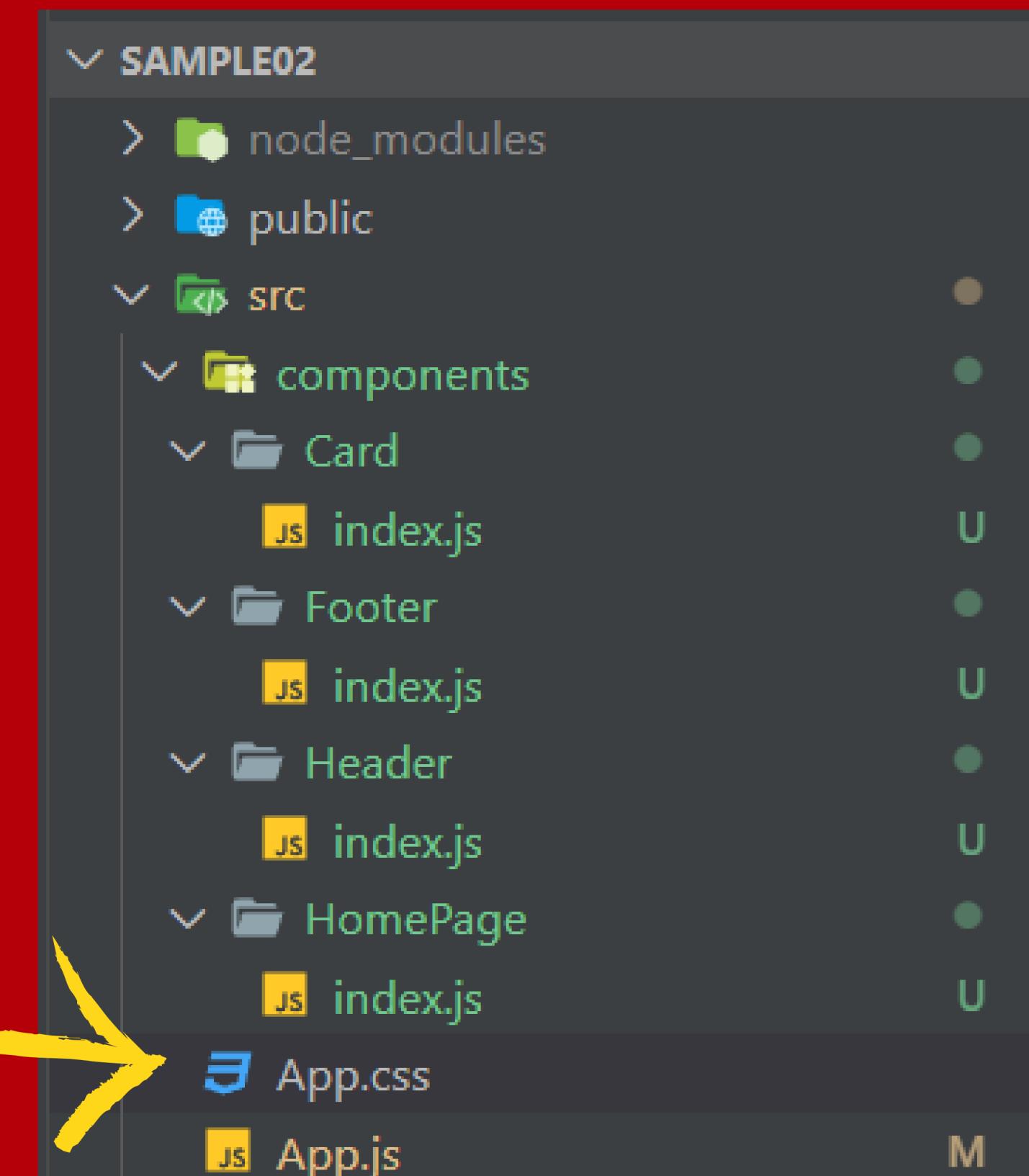
Estilizando os
componentes

STYLE. STYLE.
STYLE. STYLE.
STYLE. STYLE.



Passo 10: ESTILIZANDO CSS

Agora que temos a estrutura da nossa aplicação pronta, vamos melhorar a aparência dela. O primeiro passo é estilizar o componente pai **App**. Para isso vamos editar o arquivo **App.css**, que encontramos na estrutura apresentada na Figura.





Passo 10: ESTILIZANDO CSS

Arquivo App.css



```
1 ~ .App {  
2   display: flex;  
3   flex-direction: column;  
4   justify-content: space-between;  
5   height: 100vh;  
6 }
```



Passo 10: ESTILIZANDO CSS

App.css

O que temos aqui é o uso de display flex para organizar os elementos da página.

Dizemos que flex-direction é column, pois queremos um elemento abaixo do outro (Header acima de HomePage e HomePage acima de Footer).

O mais importante, para o entendimento do nosso código React é o seguinte: na primeira linha estamos identificando para o CSS que todo elemento que tiver a classe App será estilizado com as propriedades definidas dentro das chaves:

```
.App{
```

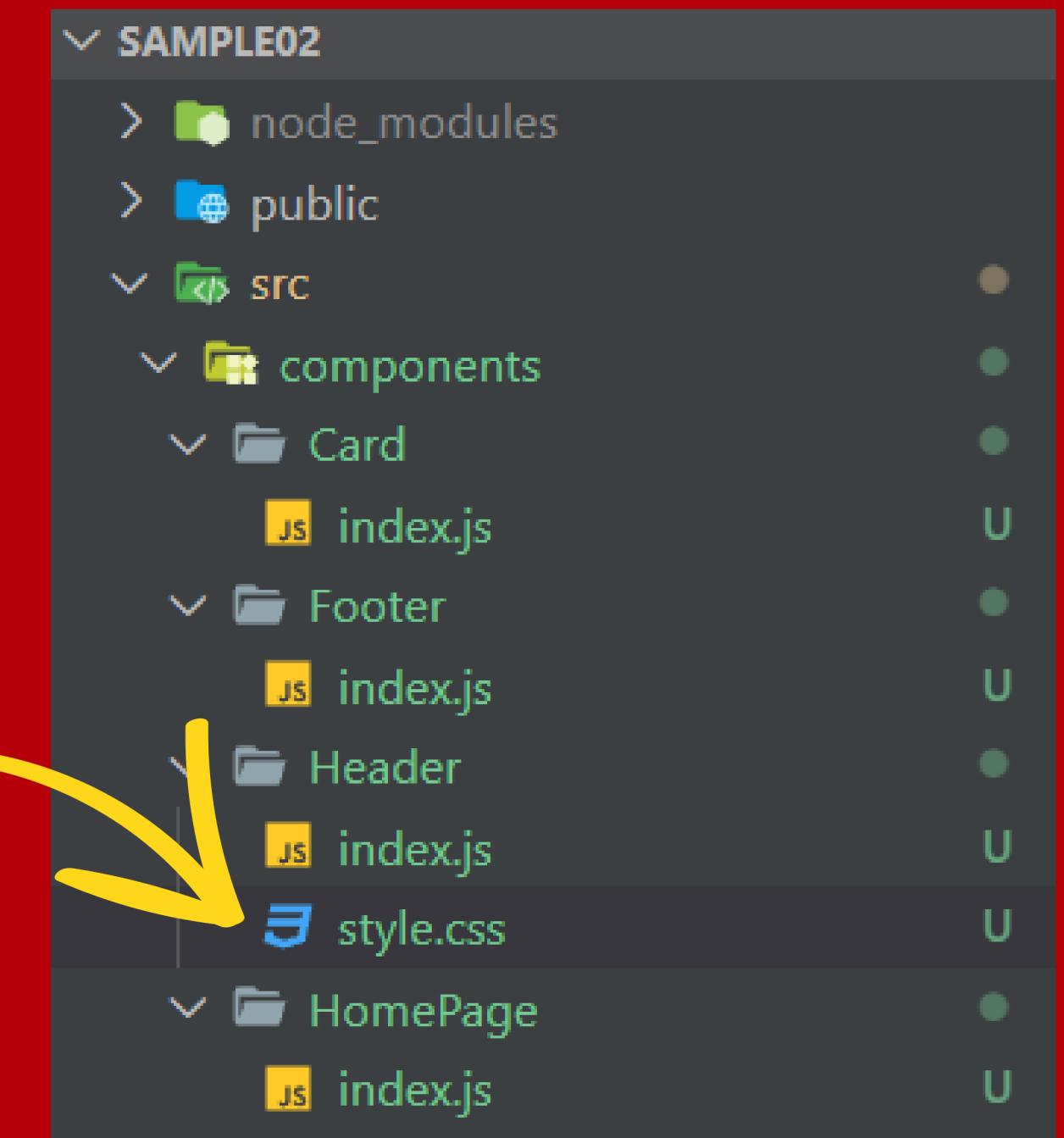
```
}
```



Passo 11: ESTILIZANDO CSS Header

Para estilizar o componente **Header** vamos precisar criar um novo arquivo **.css**, pois ao contrário de App, este componente não possui um arquivo CSS previamente associado.

Crie um arquivo chamado **style.css** na mesma pasta do componente Header, como vemos na Figura.





Passo 11: ESTILIZANDO CSS

stile.css (Header)

```
1 header{  
2     margin: 5px;  
3     background-color: #296027;  
4     border-radius: 10px;  
5     font-weight: 500;  
6     font-size: 1.4rem;  
7     color: #fff;  
8     height: 60px;  
9     display: flex;  
10    justify-content: center;  
11    align-items: center;  
12 }
```



Passo 11: ESTILIZANDO CSS

Header - alterando o index.js

```
src > components > Header > js index.js > [d] default
1 import React from "react";
2 import "./style.css"; 
3
4 function Header(){
5     return(
6         <header>Trabalhando com componentes</header>
7     );
8 }
9
10 export default Header;
```

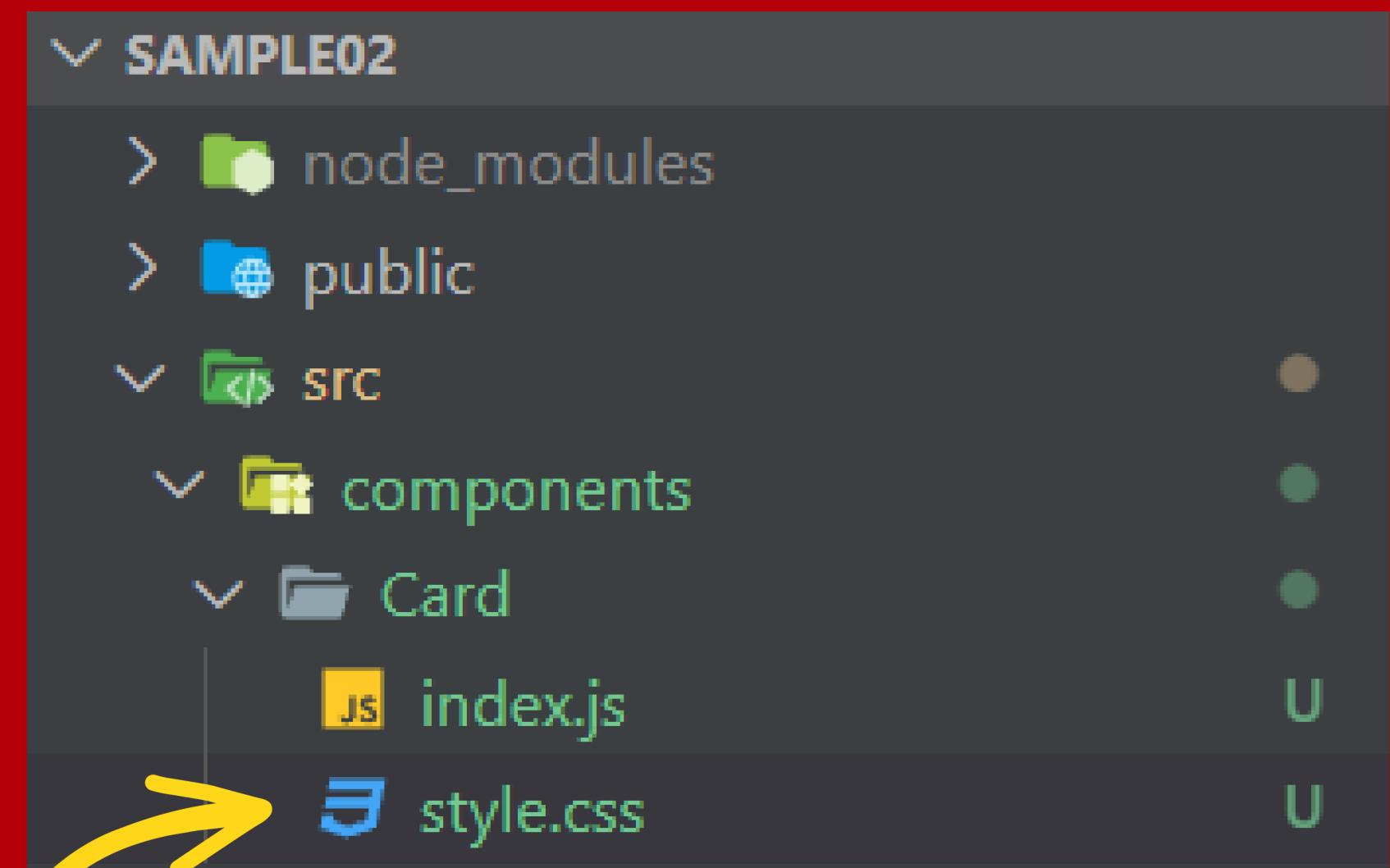


Passo 12: ESTILIZANDO CSS

Card - alterando o index.js

Para estilizar o componente Card vamos precisar criar um novo arquivo .css, pois ao contrário de App, este componente não possui um arquivo CSS previamente associado.

Crie um arquivo chamado style.css na mesma pasta do componente Header, como vemos na Figura.





Passo 12: ESTILIZANDO CSS

stile.css (Card)

```
1 .card-container{  
2     border: 1px solid #296027;  
3     border-radius: 4px;  
4     width: 200px;  
5     height: 200px;  
6     background-color: #f1f1f1;  
7     margin: 10px;  
8     padding: 10px;  
9 }
```



Passo 12: ESTILIZANDO CSS

Card - alterando o index.js

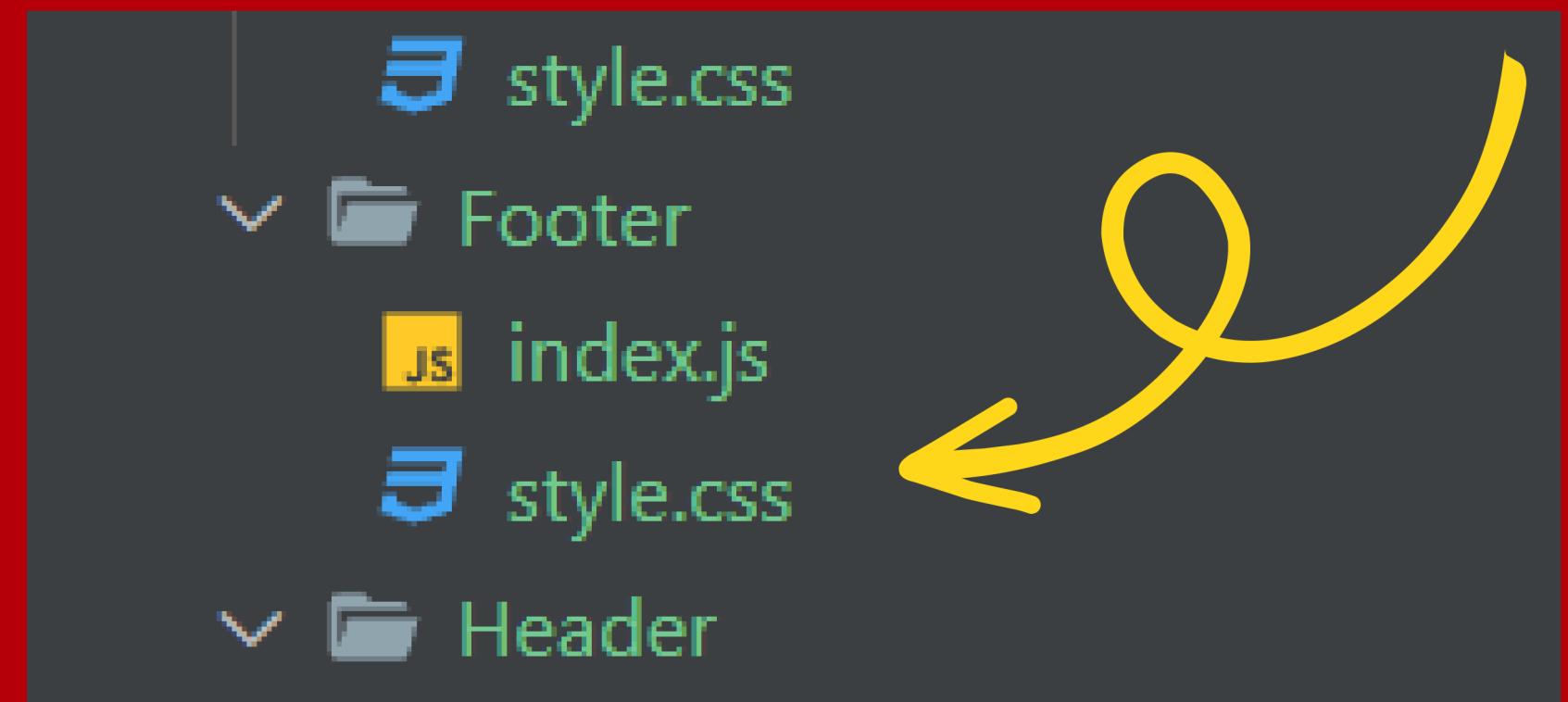
```
1 import React from "react";
2 import "./style.css"; ←
3
4 function Card(){
5   return(
6     <div className="card-container"> ←
7       <p>Componentes:</p>
8       <p>Facilita manter partes menores funcionando</p>
9       <p>Você pode reutilizá-los, ou seja, reaproveitamento</p>
10    </div>
11  );
12}
13 export default Card;
```



Passo 13: ESTILIZANDO CSS Footer

Para estilizar o componente Footer vamos precisar criar um novo arquivo .css, pois ao contrário de App, este componente não possui um arquivo CSS previamente associado.

Crie um arquivo chamado style.css na mesma pasta do componente Header, como vemos na Figura.





Passo 13: ESTILIZANDO CSS

stile.css (Footer)

```
1 footer{  
2     margin: 5px;  
3     background-color: #296027;  
4     font-weight: 500;  
5     font-size: 1rem;  
6     border-radius: 10px;  
7     color: white;  
8     height: 60px;  
9     display: flex;  
10    justify-content: center;  
11    align-items: center;  
12 }
```



Passo 13: ESTILIZANDO CSS

Footer - alterando o index.js

```
1 import React from "react";
2 import './style.css'; ←
3
4 function Footer(){
5     return (
6         <footer>Desenvolvimento com React</footer>
7     );
8 }
9
10 export default Footer;
```

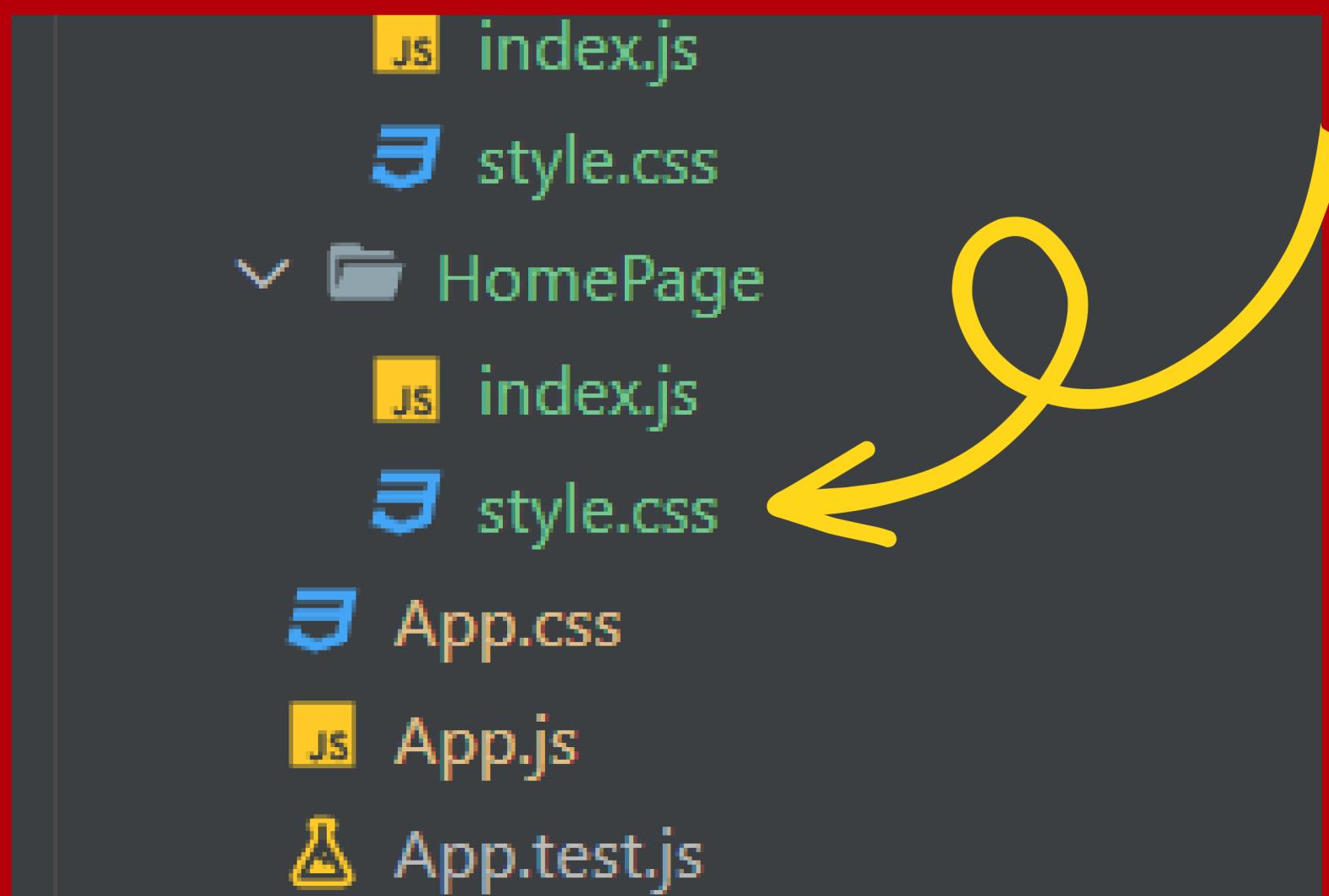


Passo 14: ESTILIZANDO CSS

Finalmente a Homepage

Para estilizar o componente **HomePage** vamos precisar criar um novo arquivo .css, pois ao contrário de **App**, este componente não possui um arquivo CSS previamente associado.

Crie um arquivo chamado **style.css** na mesma pasta do componente **Header**, como vemos na Figura.





Passo 14: ESTILIZANDO CSS

stile.css (HomePage)

src > components > HomePage >  style.css >  .home-page

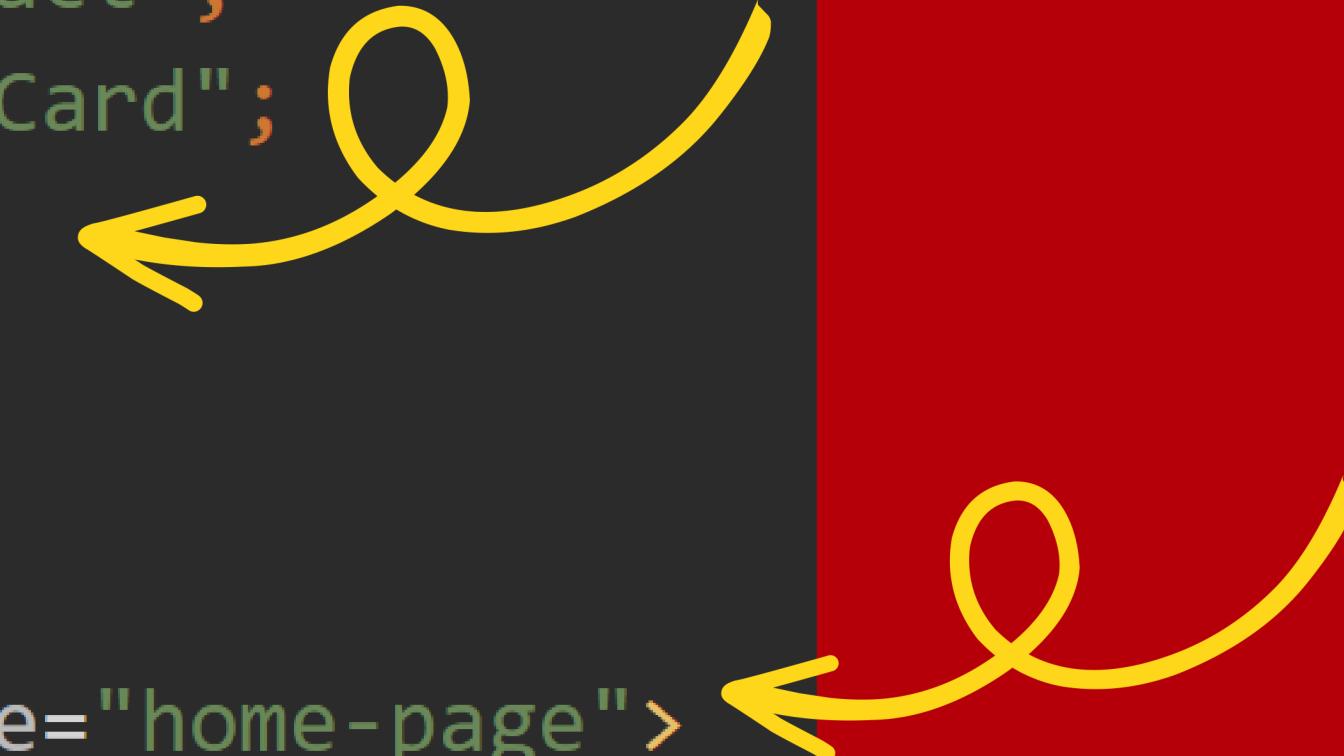
```
1 .home-page{  
2     display: flex;  
3     justify-content: center;  
4     align-items: center;  
5     flex-wrap: wrap;  
6     margin: 0 auto;  
7 }
```



Passo 14: ESTILIZANDO CSS

HomePage - alterando o index.js

```
1 import React from "react";
2 import Card from "../Card";
3 import './style.css'; ←
4
5 function HomePage(){
6   return(
7     <div className="home-page"> ←
8       <Card />
9       <Card />
10      <Card />
11    </div>
12  );
13}
```



React App

172.17.16.1:3000

Introdução Amazon Booking.com (1) #77 - App IMCCalc... Elai.io - Create AI vide...

Trabalhando com componentes

Componentes:
Facilita manter partes menores funcionando
Você pode reutilizá-los, ou seja, reaproveitamento

Componentes:
Facilita manter partes menores funcionando
Você pode reutilizá-los, ou seja, reaproveitamento

Componentes:
Facilita manter partes menores funcionando
Você pode reutilizá-los, ou seja, reaproveitamento

Desenvolvimento com React





Conclusão

No React, criamos componentes modulares e reutilizáveis para melhor manutenção do código. Exportamos e importamos esses componentes para utilização em diferentes partes do sistema, otimizando a eficiência do código. A estilização e organização apropriada dos componentes asseguram uma experiência de usuário eficaz e um código limpo.

FIM



Não espere por
oportunidades, crie-as.