

FrontEnd

FrontEnd - React

Fernanda Fretes



FrontEnd - React

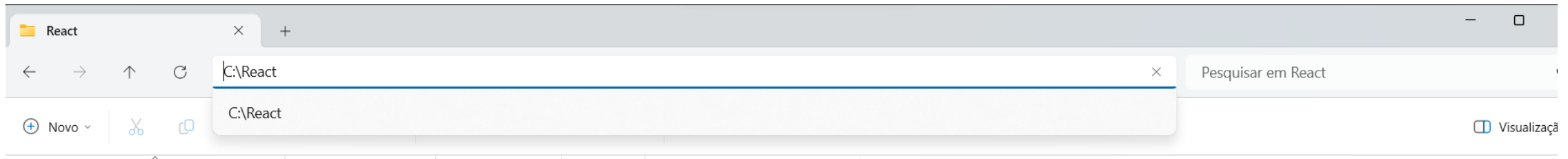
É uma biblioteca usada com a linguagem de programação JavaScript no desenvolvimento Front-end.

O Facebook criou o React em 2011 e disponibilizou a tecnologia em código aberto em 2013.

Desde então, ela é uma das bibliotecas mais populares entre os desenvolvedores. Quando os desenvolvedores do Facebook criaram esta ferramenta, **o objetivo era otimizar a atualização** do feed, mensagens do chat, status e listagem de contatos de forma mais dinâmica e rápida.

Isso acontece porque a conexão entre JavaScript, CSS e HTML, juntamente com os demais componentes, é simplificada.

FrontEnd - React



Vamos criar uma pasta do nosso novo projeto.

Na pasta **C:** criei uma pasta com o nome de React, e vou acessa-la usando o prompt comando.

```
C:\Users\htafr>cd C:\React  
C:\React>
```

FrontEnd - React

Vamos usar o Vite, que é uma das formas mais leves (não carregam tantas bibliotecas específicas na sua instalação) para ter esse primeiro contato com o React.

<https://pt.vitejs.dev/guide/>

E vamos usar um dos comandos que é recomendado pela própria documentação.

The screenshot shows the Portuguese documentation for Vite. At the top, there's a search bar and navigation links like 'Guia', 'Configuração', 'Extensões', 'Recursos', and 'Versões'. Below this is a table listing various frameworks and their corresponding Vite templates. A red box highlights the 'NPM' tab in the 'NOTA DE COMPATIBILIDADE' section, which shows the command `$ npm create vite@latest` in a terminal window.

react	react-ts
preact	preact-ts
lit	lit-ts
svelte	svelte-ts
solid	solid-ts
qwik	qwik-ts

Estruturando o Nosso Primeiro Projeto de Vite

NOTA DE COMPATIBILIDADE

A Vite exige a versão 18+, 20+ da [Node.js](#). No entanto, alguns modelos de projeto exigem uma versão superior da Node.js para funcionarem, devemos atualizar se for o nosso gestor de pacote avisar sobre isto.

NPM Yarn PNPM Bun

```
$ npm create vite@latest
```

Depois seguimos os prontos!

Nós também podemos especificar diretamente o nome do projeto e o modelo que queremos usar através das opções adicionais da linha de comando. Por exemplo, para gerar um projeto Vite + Vue, executamos:

FrontEnd - React

```
C:\Users\htafr>cd C:\React
```

```
C:\React>npm create vite@latest
```

```
C:\Users\htafr>cd C:\React
```

```
C:\React>npm create vite@latest  
Need to install the following packages:  
create-vite@5.2.2  
Ok to proceed? (y)
```

```
C:\React>npm create vite@latest  
Need to install the following packages:  
create-vite@5.2.2  
Ok to proceed? (y) y  
? Project name: » Recomendacoes
```

Selecionando o comando vamos colar no nosso prompt.

Em seguida vamos dar um **“Enter”**.

Ao fazer isso ele vai pedir uma confirmação. Para isso teremos que digitar um **“Y”**.

Ele também pedirá um nome para o projeto, que pode ser por exemplo, recomendações.

FrontEnd - React

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

Através das setas do teclado devemos selecionar o “React”, e dar um “enter”

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

Em seguida selecionar Javascript e dar “enter”

FrontEnd - React

```
C:\Users\htafr>cd C:\React

C:\React>npm create vite@latest
Need to install the following packages:
create-vite@5.2.2
Ok to proceed? (y) y
✓ Project name: ... Recomendacoes
✓ Package name: ... recomendacoes
✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in C:\React\Recomendacoes...

Done. Now run:

  cd Recomendacoes
  npm install
  npm run dev

C:\React>
```

Feito isso, já é apresentado em tela, os passos que devemos fazer.

Vamos ter que digitar essas instruções Uma a Uma.

FrontEnd - React

Done. Now run:

```
cd Recomendacoes  
npm install  
npm run dev
```

```
C:\React>cd Recomendacoes
```

Entrando na pasta do projeto

```
C:\React>cd Recomendacoes
```

```
C:\React\Recomendacoes>
```

FrontEnd - React

```
C:\React\Recomendacoes>npm install  
[██████████] \ idealTree:eslint-plugin-react: sill placeDep ROOT doctrine@3.0.0 OK for: eslint@8.57.0 want: ^3.0.0
```

Fazendo efetivamente todas as instalações necessárias

```
C:\React\Recomendacoes>npm install  
  
added 279 packages, and audited 280 packages in 19s  
  
103 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
  
C:\React\Recomendacoes>
```

FrontEnd - React

```
npm install  
npm run dev
```

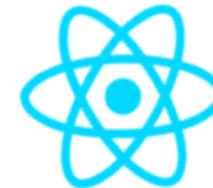
```
VITE v5.1.6 ready in 198 ms  
  
→ Local:   http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

Depois é só pedir para rodar o comando run DEV.

Isso vai fazer com que sua máquina trabalhe como um servidor. Ou seja, só conseguiremos trabalhar com o React caso o terminal esteja aberto.

FrontEnd - React

Seguindo todos esses passos, conseguimos agora através do endereço localhost mostrado anteriormente, acessar uma página React.



Vite + React

count is 0

Edit `src/App.jsx` and save to test HMR

Click on the Vite and React logos to learn more

FrontEnd - React

Uma das grandes características do react, é fazer as manipulações das telas, sem a necessidade de dar o “refresh”, e já nesta página, ele quer demonstrar isso, com o botão Count. Ou seja, caso clicado, ele já adiciona um numero, sem que precise dar um f5.

localhost:5173



Vite + React

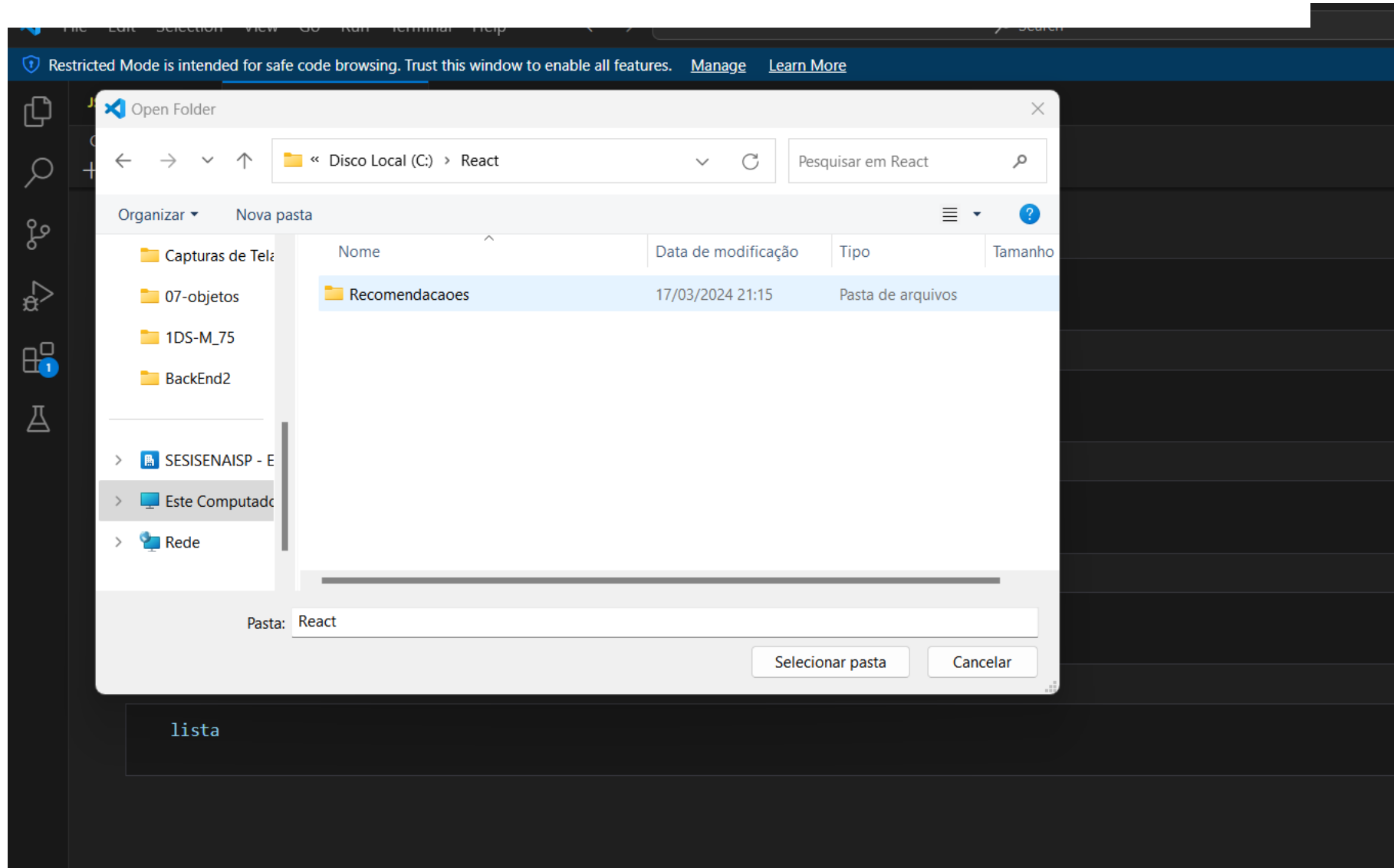
count is 1

Edit `src/App.jsx` and save to test HMR

[Click on the Vite and React logos to learn more](#)

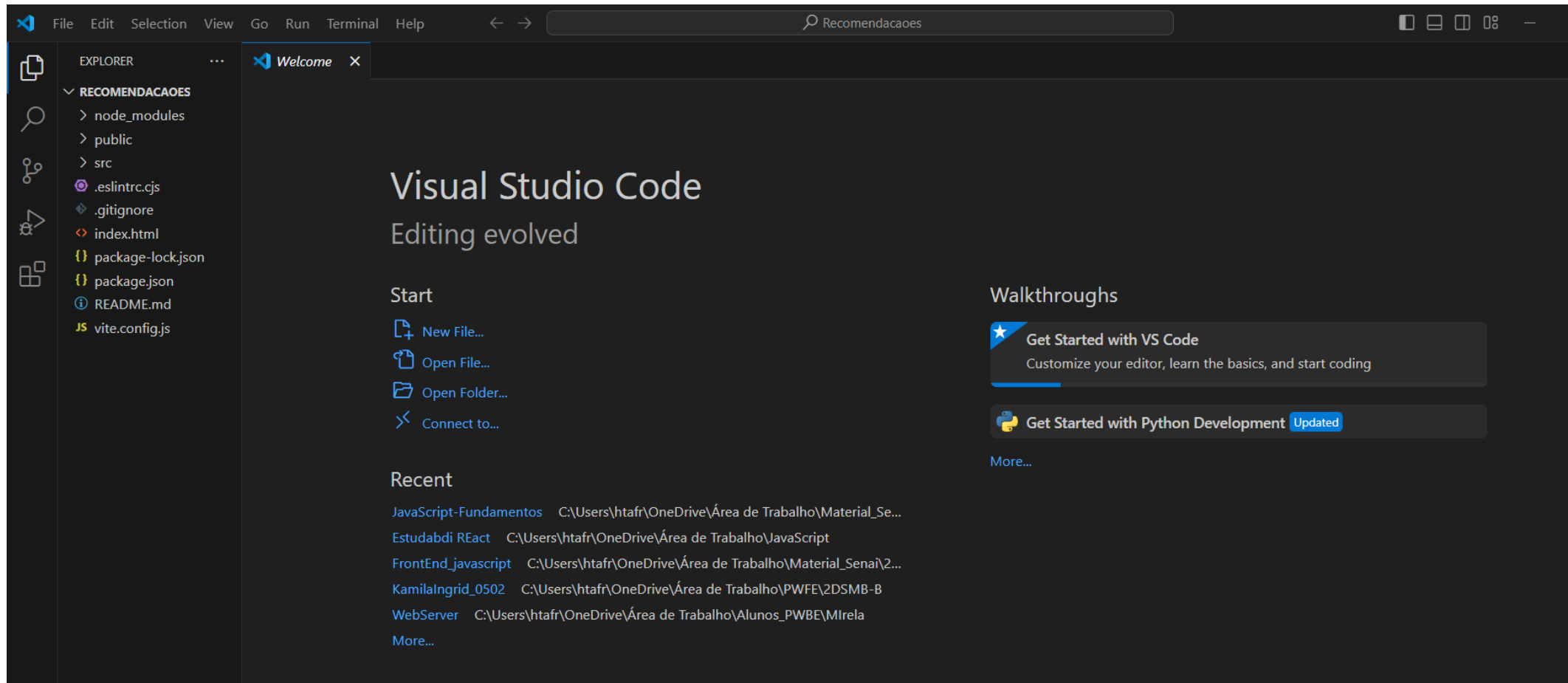
FrontEnd - React

Agora vamos abrir nosso projeto com o VSCode

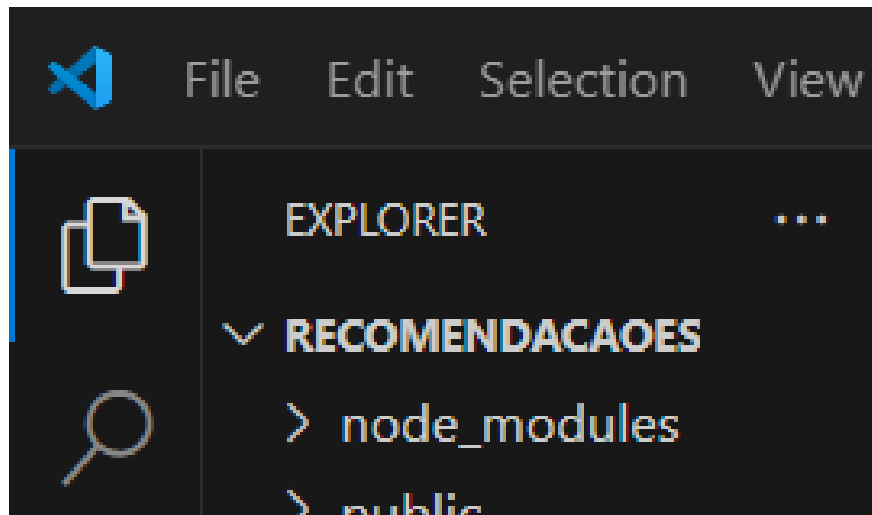


FrontEnd - React

Com aqueles comandos toda a estrutura necessária para fazer uma aplicação usando o React

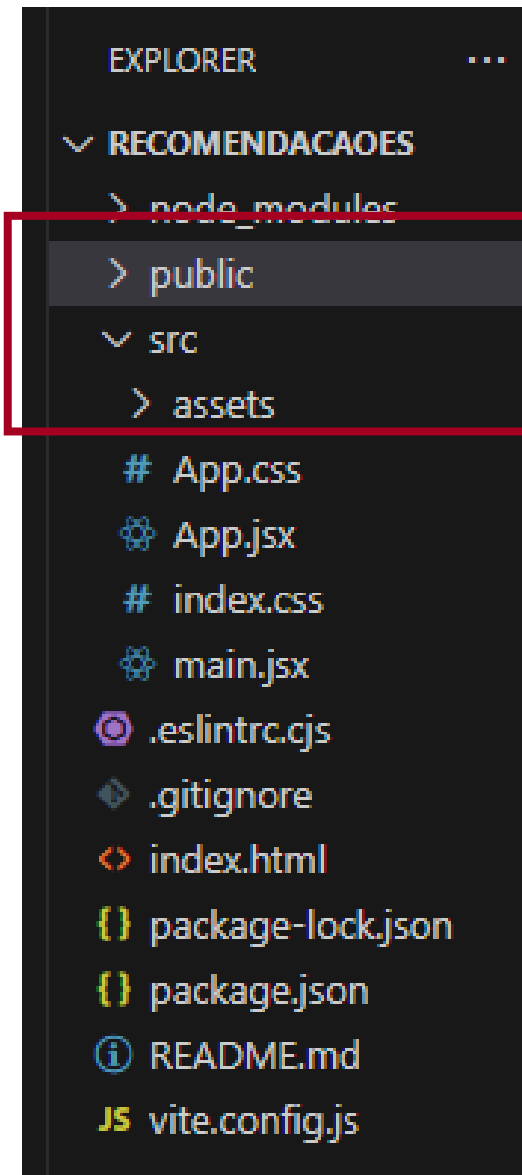


FrontEnd - React



É na pasta `node_modules` que temos grande parte dos pacotes que são usadas localmente ou globalmente para fazer rodar a nossa aplicação.

FrontEnd - React

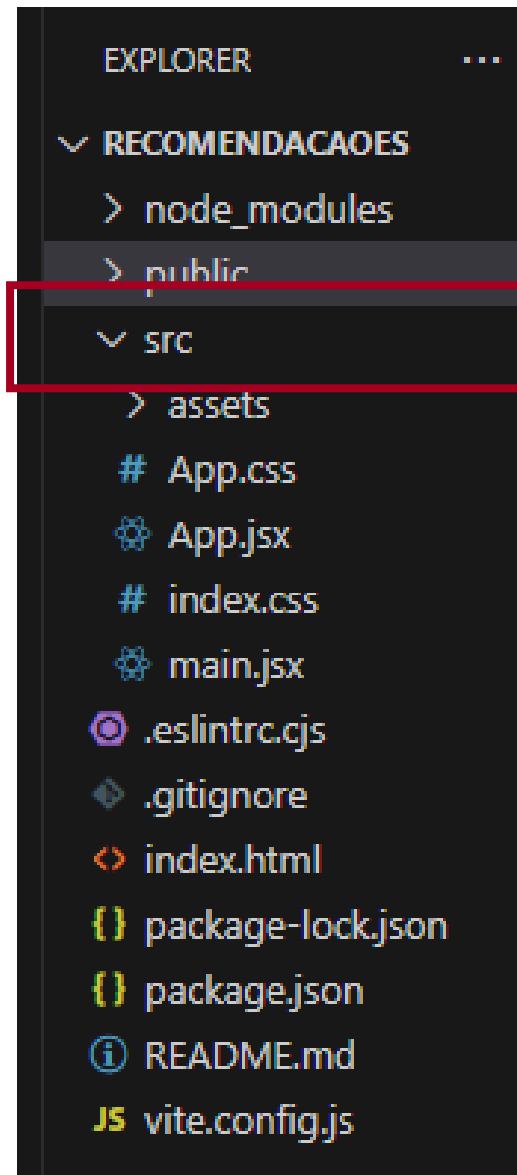


Na pasta public deixo grande parte dos meus artefatos estáticos. Como imagens ou fontes, que serão acessados publicamente.

A pasta SRC armazena as rotas possíveis, arquivos de imagens, ícones

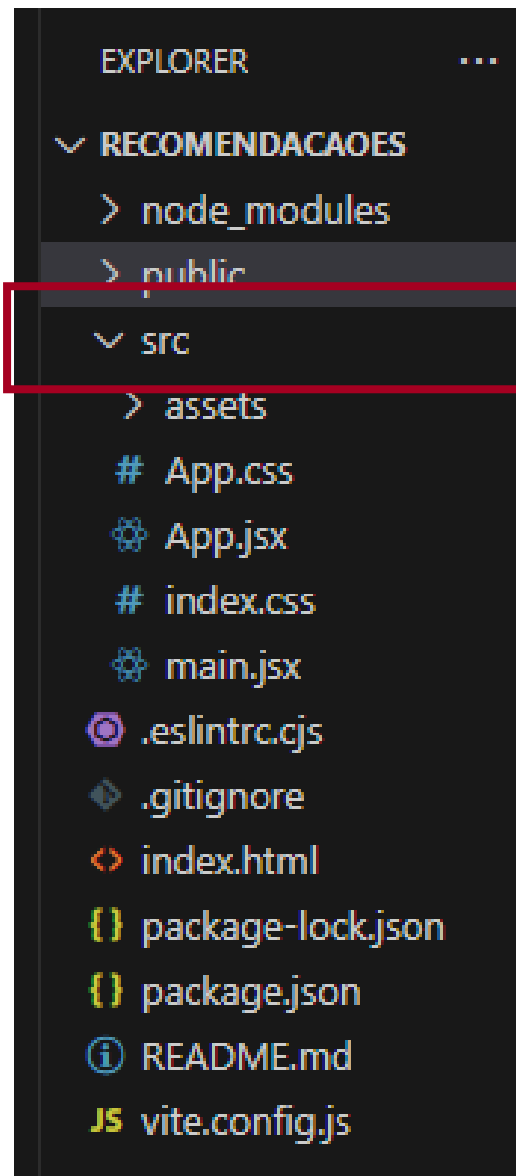
assets/: essa pasta é usada para armazenar arquivos de recursos estáticos, como imagens, fontes, listas, etc., que serão importados no código

FrontEnd - React



- App.css: esse é um arquivo de estilo CSS que contém estilos específicos para o componente App;
- App.jsx: o arquivo App.tsx é o componente principal da sua aplicação React. É onde você define a estrutura e o comportamento geral da sua aplicação;
- index.css: esse é o arquivo de estilos globais da sua aplicação;
- main.tsx: esse é o ponto de entrada da sua aplicação React. Ele renderiza o componente App na página HTML;
- vite-env.d.ts: esse arquivo é usado para declarações de tipos globais que podem ser necessárias no seu projeto.

FrontEnd - React



eslintc.js: esse é o arquivo de configuração do ESLint, que é uma ferramenta para ajudar a manter um código JavaScript/TypeScript limpo e consistente. Ele define as regras e configurações para a análise estática do código;

gitignore: esse arquivo lista os arquivos e pastas que você deseja que o Git ignore ao controlar as mudanças do projeto. Isso geralmente inclui arquivos gerados automaticamente, como node_modules, bem como arquivos de compilação e cache;

index.html: é o arquivo HTML principal da sua aplicação. É aqui que o ponto de entrada do React é incorporado e onde você pode incluir metadados, links para estilos e outros recursos;

package-lock.json: esse arquivo é gerado automaticamente pelo “npm” e registra as versões exatas de todas as dependências do seu projeto. Ele é usado para garantir que as mesmas versões das dependências sejam instaladas em diferentes máquinas;

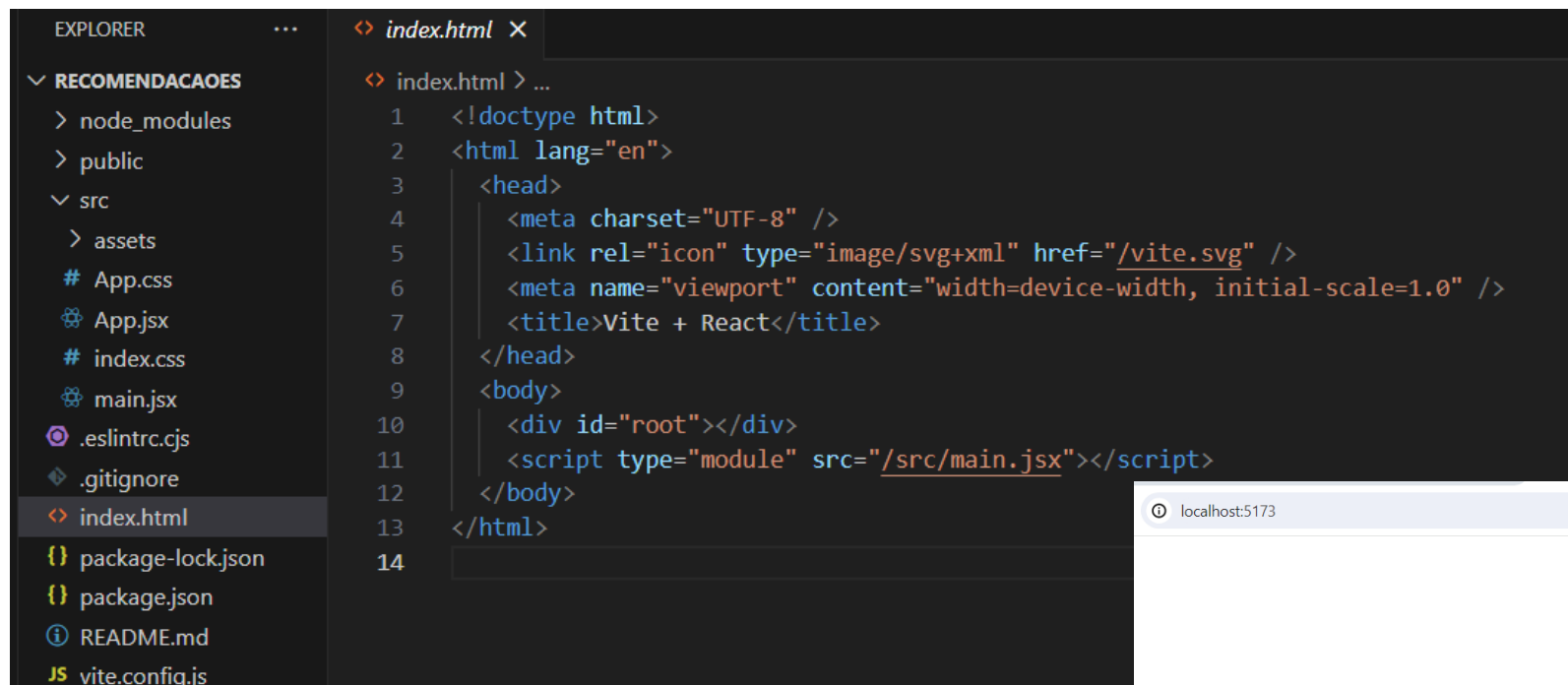
package.json: esse arquivo contém informações sobre o projeto, como nome, versão, dependências e scripts personalizados. Você pode usá-lo para gerenciar dependências e definir scripts para tarefas comuns de desenvolvimento;

README.md: é um arquivo de documentação para o seu projeto. É onde você pode fornecer informações sobre como instalar, configurar e usar a aplicação;

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.

FrontEnd - React

Fernanda Fretes



The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing a project structure under 'RECOMENDACOES'. The files listed are: node_modules, public, src, assets, App.css, App.jsx, index.css, main.jsx, .eslintrc.cjs, .gitignore, index.html (selected), package-lock.json, package.json, README.md, and vite.config.js. The main editor area displays the content of index.html, which is a basic HTML template for Vite + React. The code includes a doctype, html lang, head with meta charset, link for vite.svg, meta viewport, and title 'Vite + React', and a body with a root div and a script tag for main.jsx.

```
<?xml>
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```



Vite + Reaja

contagem é 0

Edite src/App.jsx e salve para testar o HMR

Clique nos logotipos Vite e React para saber mais

FrontEnd - React

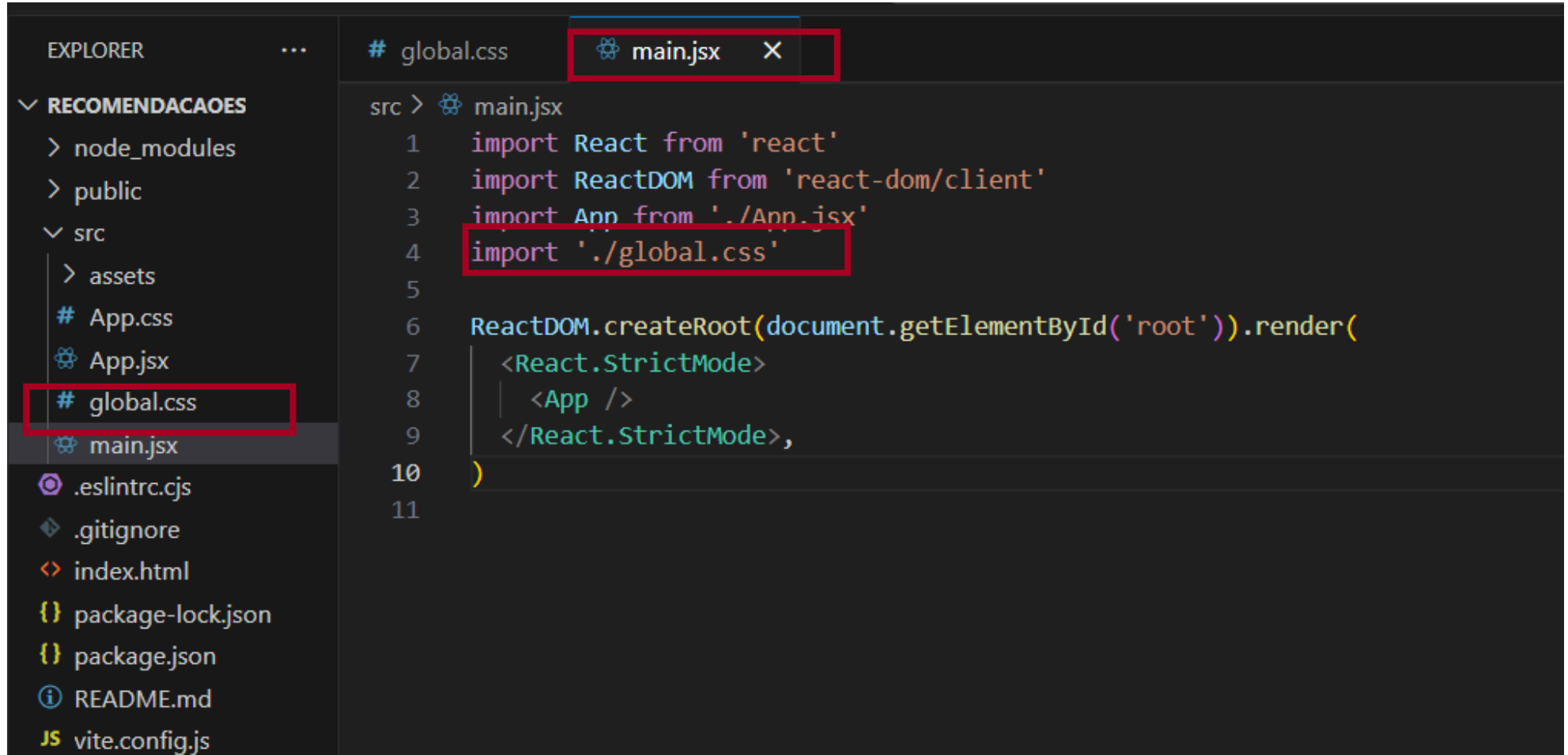
Nós podemos deixar uma página de estilo para cada rota que formos trabalhar, mas caso, algumas características sejam a identidade visual da aplicação, podemos ter um CSS Global.

Então o arquivo index.css, podemos mudar seu nome para “global.css” e apagar aquelas configurações que tem lá, eliminar as características básicas do navegador.

```
> # index.css > ✖ *  
1  *{  
2    margin:0px;  
3    padding: 0px;  
4    box-sizing: border-box;  
5  
6  }
```

FrontEnd - React

Essa folha de estilo é chamada na main, por isso devemos fazer as alterações lá também, para o novo nome do arquivo.



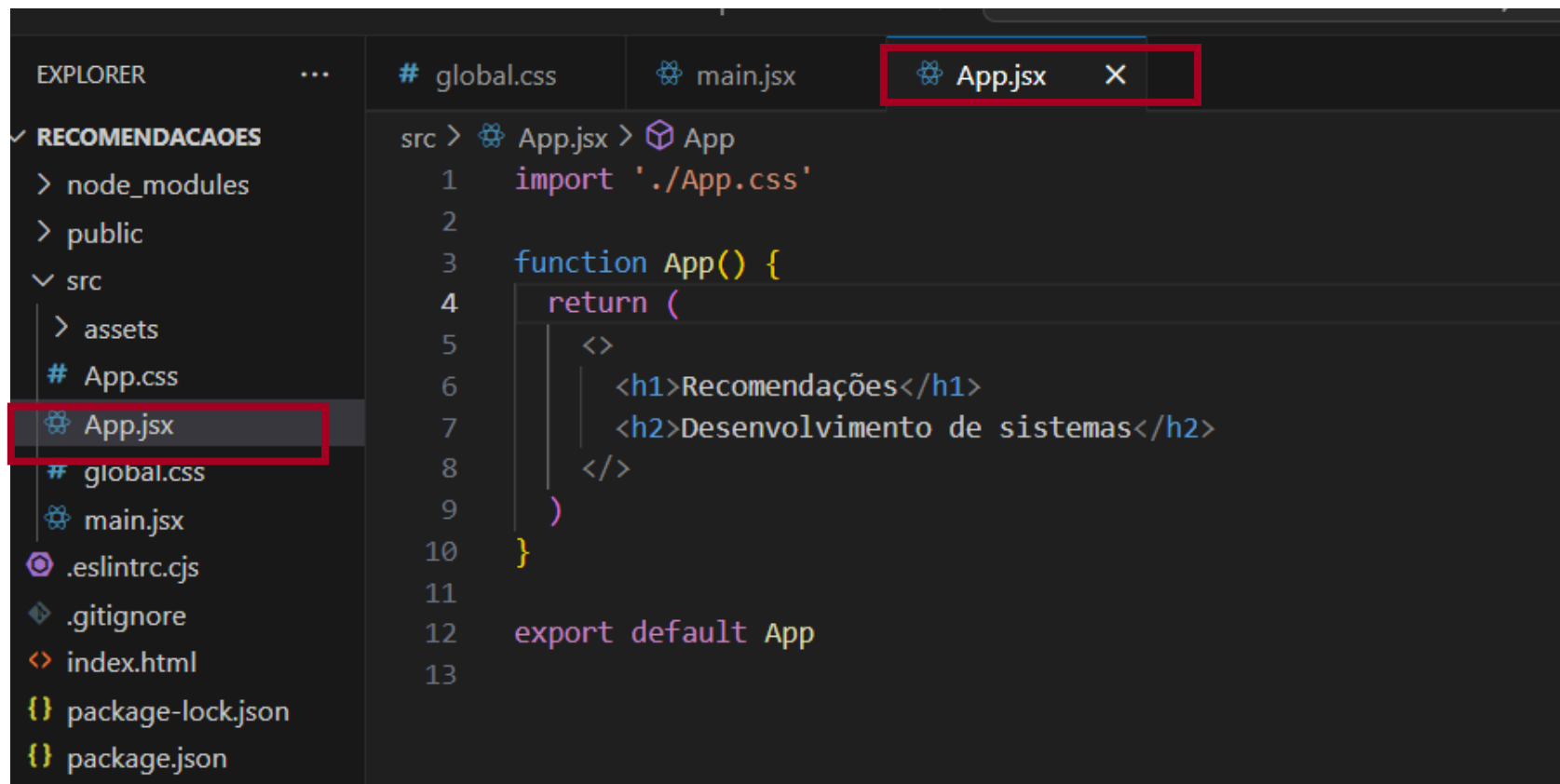
```
EXPLORER  ...  # global.css  main.jsx X

▼ RECOMENDACOES
  > node_modules
  > public
  ▼ src
    > assets
    # App.css
    App.jsx
    # global.css
    main.jsx
    .eslintrc.cjs
    .gitignore
    index.html
    package-lock.json
    package.json
    README.md
    vite.config.js

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

FrontEnd - React

Fernanda Fretes



The screenshot shows the VS Code interface. In the Explorer pane on the left, the file `App.jsx` is selected and highlighted with a red box. The Editor pane on the right displays the content of `App.jsx`, which is also highlighted with a red box in the tab bar. The code in `App.jsx` is as follows:

```
src > App.jsx > App
1  import './App.css'
2
3  function App() {
4    return (
5      <>
6        <h1>Recomendações</h1>
7        <h2>Desenvolvimento de sistemas</h2>
8      </>
9    )
10 }
11
12 export default App
13
```



Usando CSS Module

Motivação

Conforme o projeto vai crescendo, muitos problemas começam a surgir por decisões tomadas no início, quando aquilo não parecia ser um problema ou talvez nem fosse imaginado. Um desses problemas pode ser a falta de padrão nos estilos, e um problema que é originado a partir disso é a confusão gerada com os classNames.

Problema

O problema com os classNames ocorre da seguinte maneira. Imagine um nome de className bastante utilizado por todos, no caso irei explicar utilizando o nome "title". Todos os lugares em que utilizamos o "title" precisamos criar um nome composto para que não haja globalidade entre os estilos, então caso formos utilizar dentro de um cartão, criaremos o "card-title", caso seja um modal, será "modal- title", e cada vez fica mais difícil pensar num bom nome de className para cada componente.

Solução

Desde a versão 2 do CRA (create-react-app) temos a opção de criar estilos exclusivos para cada componente, utilizando os módulos css. Os módulos css são arquivos css em que os classNames e animações são definidos localmente, isso significa que os estilos ali criados, só serão declarados dentro desse escopo, e não globalmente, evitando conflitos entre estilos.

FrontEnd - React

```
src > App.jsx > ...
1 import estilos from './App.module.css'
2
3 function App() {
4   return (
5     <>
6       <h1 className={estilos.titulo}>Recomendações</h1>
7       <h2 className={estilos.subtitulo}>2DSTB</h2>
8     </>
9   )
10 }
11
12 export default App
13
```

```
rc > # App.module.css > .subtitulo
1 .titulo{
2   font-size: 28;
3   color: blue;
4 }
5
6 .subtitulo{
7   font-size: 22;
8   color: red;
9 }
```

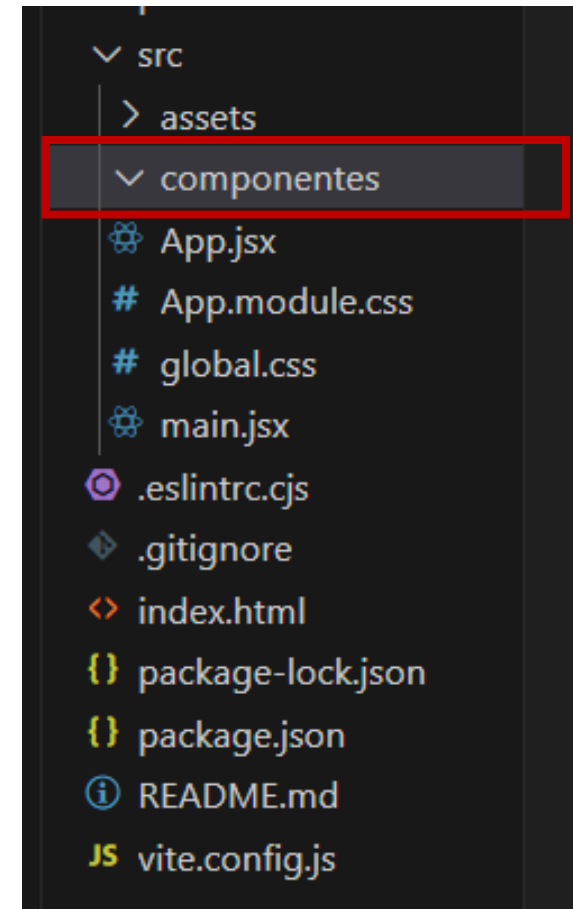
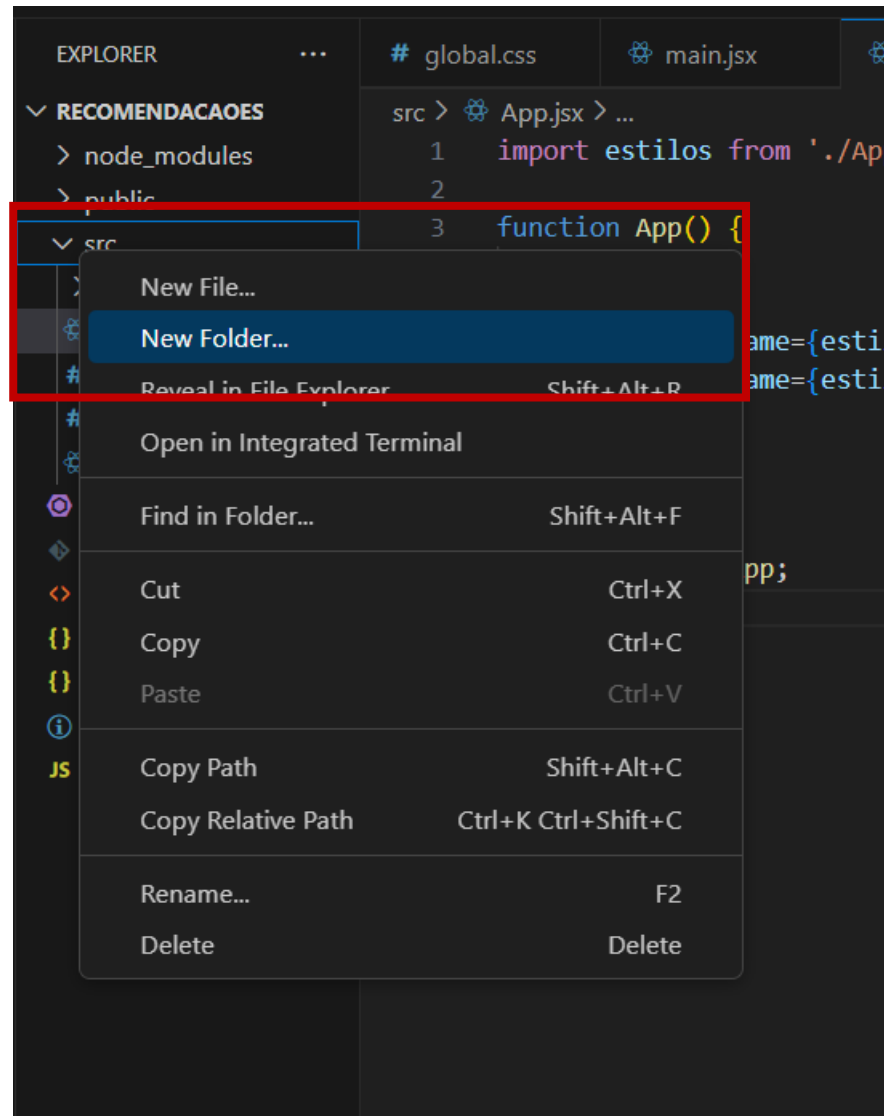
Componentes

Os componentes são os blocos de construção fundamentais de qualquer aplicação React. Eles representam partes isoladas da interface do usuário e podem variar em tamanho e complexidade. Um componente pode ser tão simples quanto um botão ou tão complexo quanto um formulário de registro. A modularidade dos componentes no ReactJS permite que os desenvolvedores os criem, reutilizem e mantenham de maneira eficiente.

FrontEnd - React

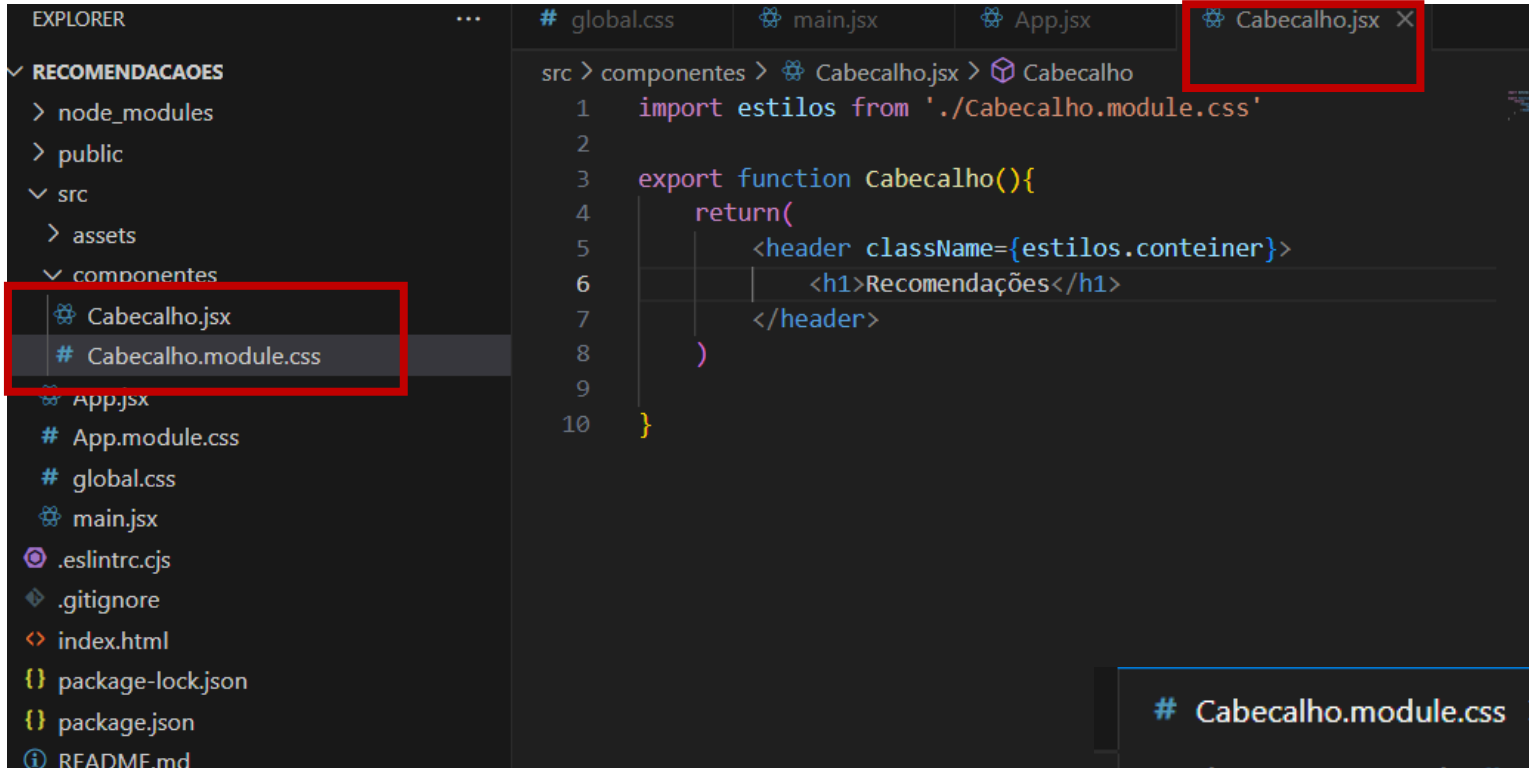
Componentes

Em Src, vamos criar mais uma pasta chamada "componentes".



FrontEnd - React

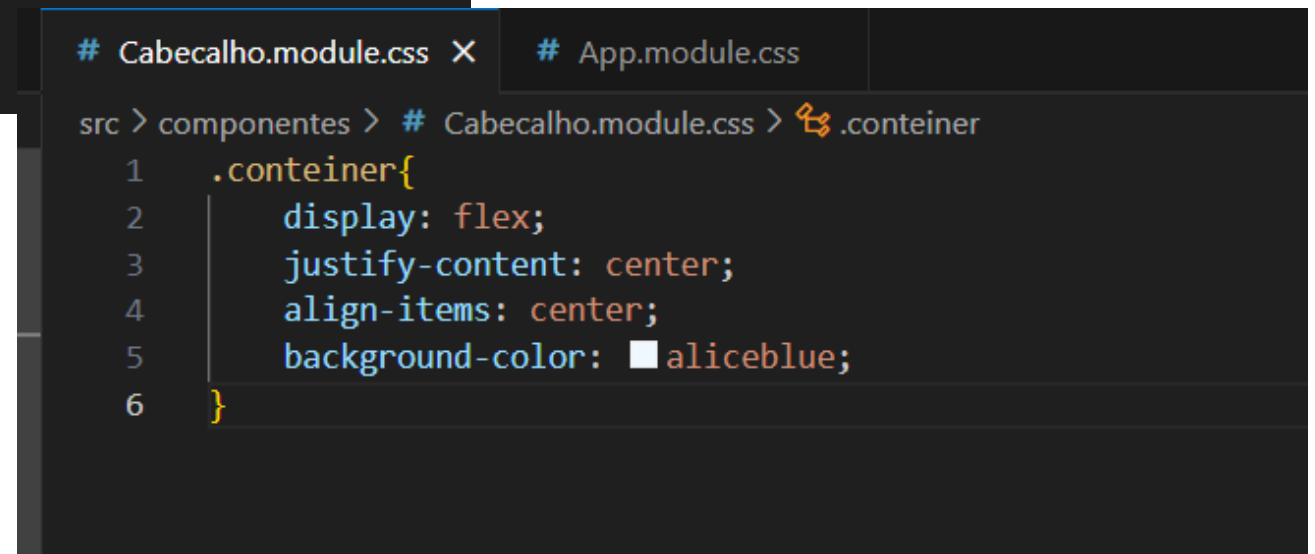
Fernanda Fretes



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a file tree with a red box highlighting the 'componentes' folder and its sub-files 'Cabecalho.jsx' and 'Cabecalho.module.css'. The main editor area shows the 'Cabecalho.jsx' file with the following code:

```
1 import estilos from './Cabecalho.module.css'
2
3 export function Cabecalho(){
4   return(
5     <header className={estilos.container}>
6       <h1>Recomendações</h1>
7     </header>
8   )
9
10 }
```

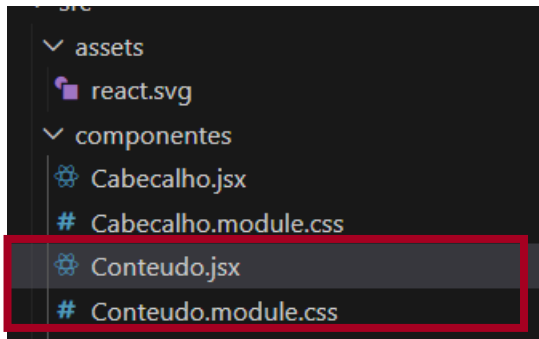
Para cada componente podemos estilizar de uma forma única, necessitando do seu próprio CSS. Então teremos um componente e um css cabeçalho



The screenshot shows the 'Cabecalho.module.css' file in the VS Code editor. The code defines a CSS class for the header container:

```
# Cabecalho.module.css X # App.module.css
src > componentes > # Cabecalho.module.css > .container
1 .container{
2   display: flex;
3   justify-content: center;
4   align-items: center;
5   background-color: #aliceblue;
6 }
```

FrontEnd - React



O mesmo para o conteúdo. Teremos 2 arquivos, um jsx e um css

The image shows two side-by-side code editor windows. The left window is titled 'Conteudo.jsx' and shows the following code:

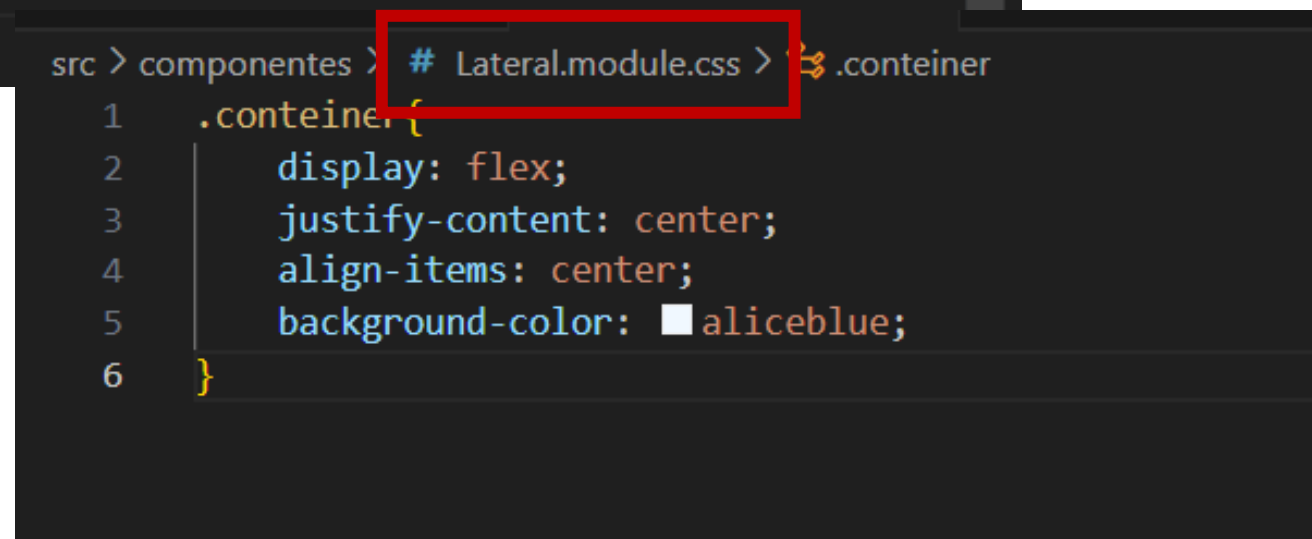
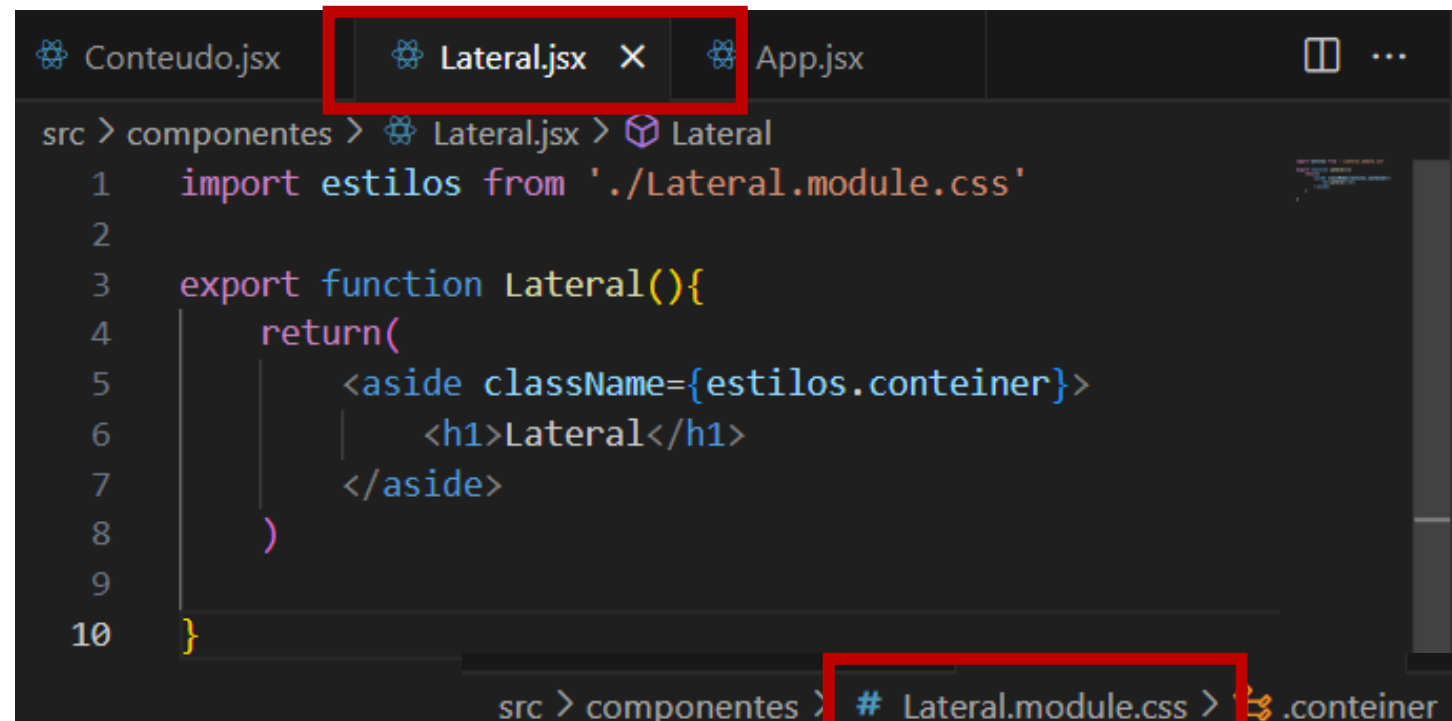
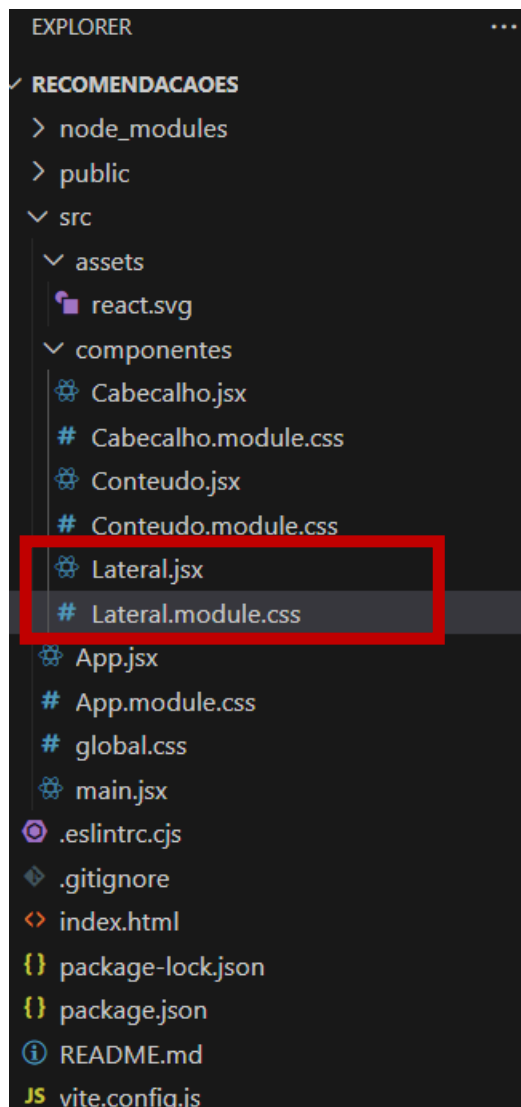
```
src > componentes > Conteudo.jsx > Conteudo
1 import estilos from './Conteudo.module.css'
2
3 export function Conteudo(){
4   return(
5     <main className={estilos.container}>
6       <h2>conteudo</h2>
7     </main>
8   )
9
10 }
```

The right window is titled '# Conteudo.module.css' and shows the following CSS code:

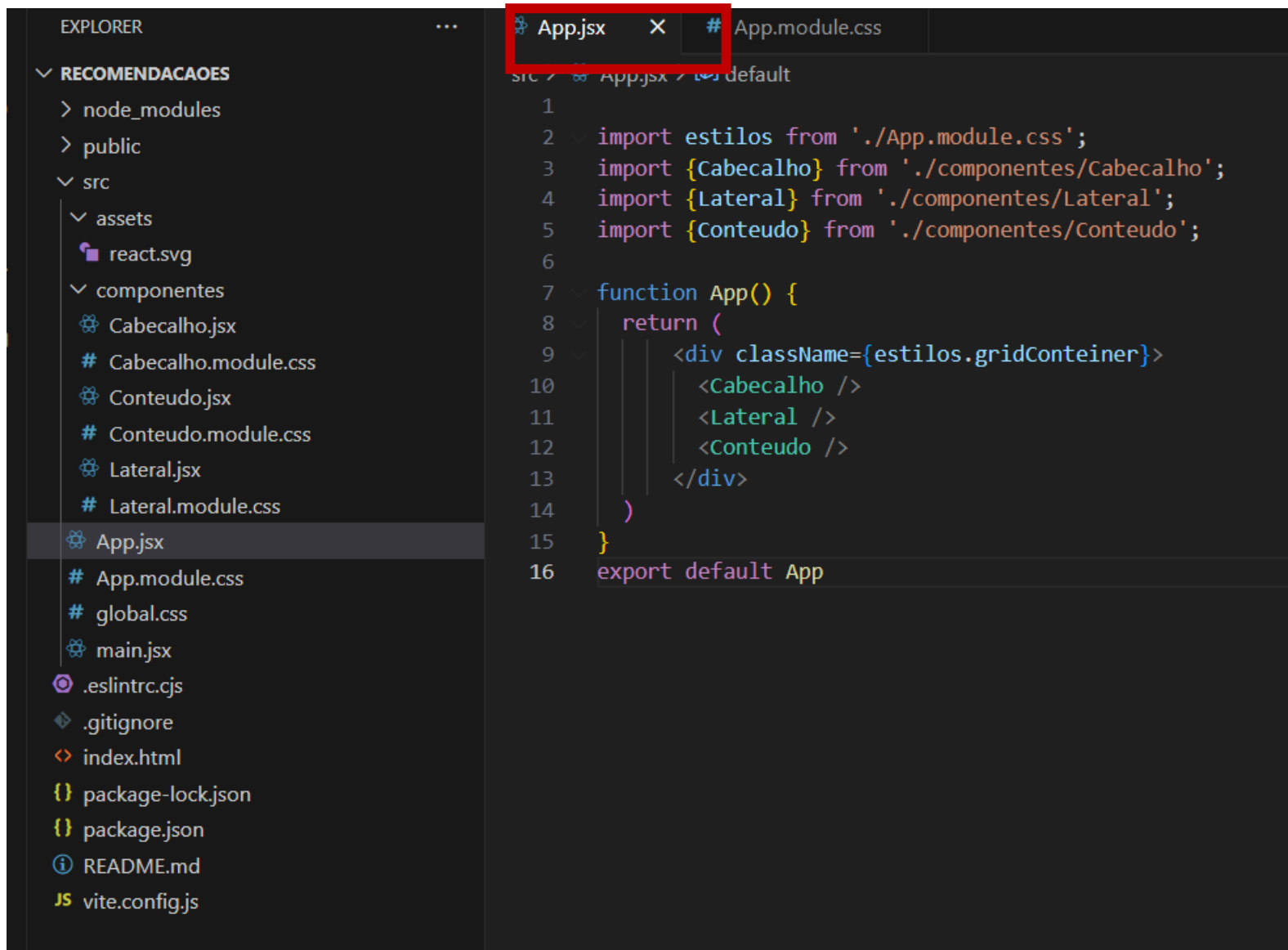
```
src > componentes > # Conteudo.module.css > .container
1 .container{
2   display: flex;
3   justify-content: center;
4   align-items: center;
5   background-color: #aliceblue;
6 }
```

FrontEnd - React

Fernanda Fretes



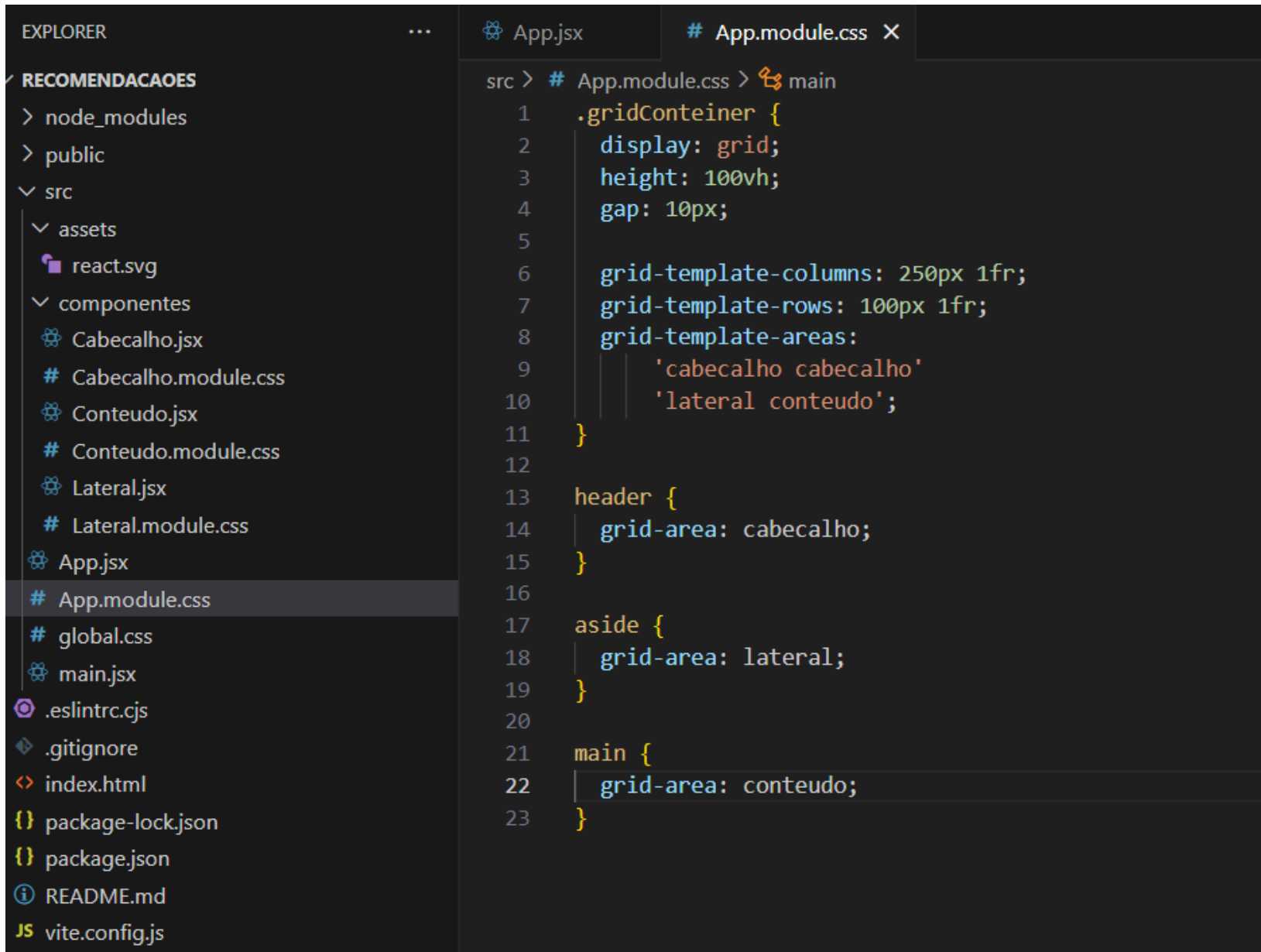
FrontEnd - React



```
1
2 import estilos from './App.module.css';
3 import {Cabecalho} from './componentes/Cabecalho';
4 import {Lateral} from './componentes/Lateral';
5 import {Conteudo} from './componentes/Conteudo';
6
7 function App() {
8   return (
9     <div className={estilos.gridContainer}>
10       <Cabecalho />
11       <Lateral />
12       <Conteudo />
13     </div>
14   )
15 }
16 export default App
```

Já no “App” vamos chamar todos esses componentes que criamos, fazendo o importe deles.

FrontEnd - React



The screenshot shows a VS Code editor with a dark theme. On the left is the Explorer sidebar showing a project structure. The main editor area displays the content of 'App.module.css'.

EXPLORER

- RECOMENDACOES
- > node_modules
- > public
- ▼ src
 - ▼ assets
 - react.svg
 - ▼ componentes
 - Cabecalho.jsx
 - # Cabecalho.module.css
 - Conteudo.jsx
 - # Conteudo.module.css
 - Lateral.jsx
 - # Lateral.module.css
 - App.jsx
 - # App.module.css
 - # global.css
 - main.jsx
- .eslintrc.cjs
- .gitignore
- index.html
- package-lock.json
- package.json
- README.md
- vite.config.js

App.module.css

```
src > # App.module.css > main
1  .gridContainer {
2      display: grid;
3      height: 100vh;
4      gap: 10px;
5
6      grid-template-columns: 250px 1fr;
7      grid-template-rows: 100px 1fr;
8      grid-template-areas:
9          'cabecalho cabecalho'
10         'lateral conteudo';
11 }
12
13 header {
14     grid-area: cabecalho;
15 }
16
17 aside {
18     grid-area: lateral;
19 }
20
21 main {
22     grid-area: conteudo;
23 }
```

Também teremos que dimensionar todos esses elementos na tela.

FrontEnd - React

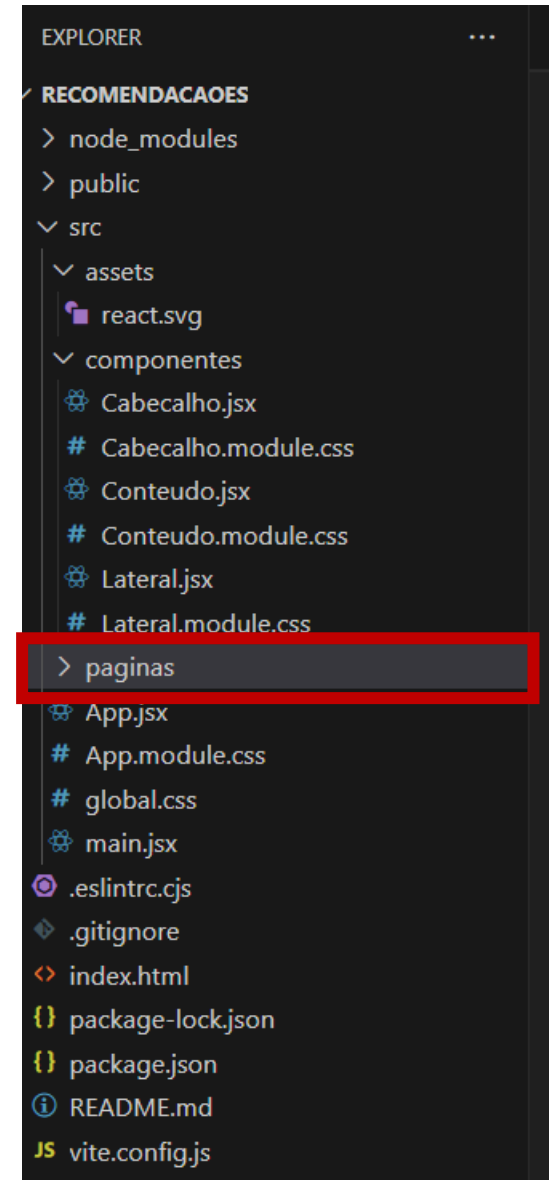
Fernanda Fretes



FrontEnd - React

Mas, para que consigamos interagir de forma a seguir as boas práticas, é interessante que deixemos os componentes na pasta correspondentes, e as páginas em outro.

Então precisaremos mudar um pouco a nossa estrutura. Em SRC vamos criar uma pasta chamada “Páginas”



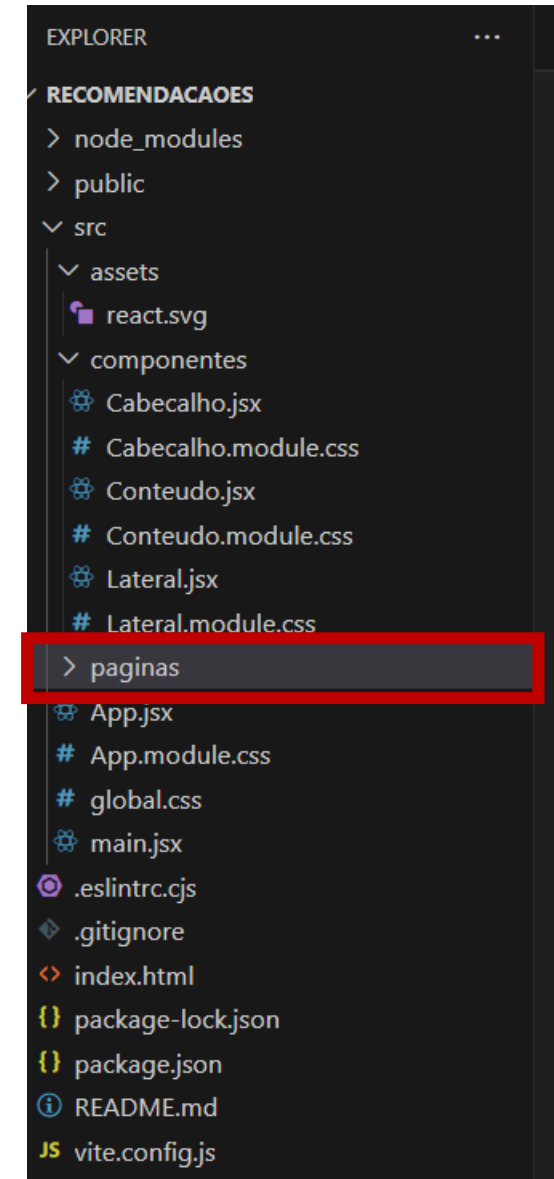
FrontEnd - React

E dentro de pagina, criaremos dois arquivos:

inicial.jsx e inicial.module.css

Em inicial vou copiar tudo que tem em App

E em inicial.module.css, o que tem em app.module.css, alterando o nome das páginas.



FrontEnd - React

Fernanda Fretes

The image shows a VS Code editor with the following components:

- EXPLORER (Left Panel):** Displays the project file structure.
 - RECOMENDACOES
 - node_modules
 - public
 - src
 - assets
 - react.svg
 - componentes
 - Cabecalho.jsx
 - Cabecalho.module.css
 - Conteudo.jsx
 - Conteudo.module.css
 - Lateral.jsx
 - Lateral.module.css
 - paginas
 - Inicial.jsx (selected)
 - Inicial.module.css
 - App.jsx
 - App.module.css
 - global.css
 - main.jsx
 - .eslintrc.cjs
 - .gitignore
 - index.html
 - package-lock.json
 - package.json
 - README.md

- Main Editor (Center):** Shows the file `src > paginas > Inicial.jsx > Inicial`. The code is as follows:

```
1 import estilos from './inicial.module.css';
2 import {Cabecalho} from '../componentes/Cabecalho';
3 import {Lateral} from '../componentes/Lateral';
4 import {Conteudo} from '../componentes/Conteudo';
5
6
7 export function Inicial() {
8   return (
9     <div className={estilos.gridContainer}>
10       <Cabecalho />
11       <Lateral />
12       <Conteudo />
13     </div>
14   )
15 }
```
- Right Editor:** Shows the file `src > paginas > Inicial.module.css > main`. The code is as follows:

```
1 .gridContainer {
2   display: grid;
3   height: 100vh;
4   gap: 10px;
5
6   grid-template-columns: 250px 1fr;
7   grid-template-rows: 100px 1fr;
8   grid-template-areas:
9     'cabecalho cabecalho'
10    'lateral conteudo';
11 }
12
13 header {
14   grid-area: cabecalho;
15 }
16
17 aside {
18   grid-area: lateral;
19 }
20
21 main {
22   grid-area: conteudo;
23 }
```

FrontEnd - React

O Arquivo App.jsx agora vai chamar o inicial :

```
App.jsx  index.html  Inicial.jsx  App.module.css

src > App.jsx > ...
1  import { Inicial } from './paginas/Inicial'
2
3
4  function App() {
5    return (
6      <Inicial />
7    )
8  }
9  export default App
```

```
App.jsx  index.html  Inicial.jsx  App.module.css X

src > # App.module.css
1
```

FrontEnd - React

```
App.jsx  Inicial.jsx  # App.module.css  # global.css X
src > # global.css > body
1  :root{
2    --cinza-claro:#dcdcd;
3    --roxo-escuro: #1f0322;
4    --roxo-claro: #ab10bc;
5  }
6
7  *{
8    margin:0px;
9    padding: 0px;
10   box-sizing: border-box;
11 }
12
13 body{
14   background-color: var(--roxo-escuro);
15 }
```

Uma coisa bem legal que temos como ferramenta na estilização, é a declaração de variáveis (cores) que podem ser chamadas a qualquer momento, colocando-as na “global”.

Para fazer isso, coloco em “:root”
E com “--” declaro um nome para receber a cor, por exemplo:

FrontEnd - React

Fernanda Fretes

Se acessarmos
nossa aplicação,
vemos que está
funcionando
normalmente

