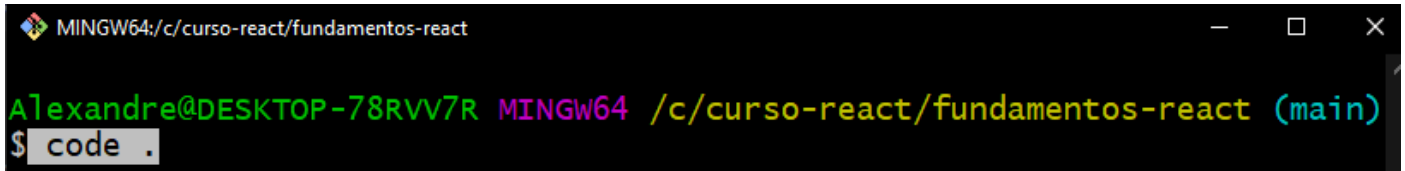


React.JS Fundamentos e Intermediário 13/12/2021

Na última aula havíamos criado o projeto na pasta:

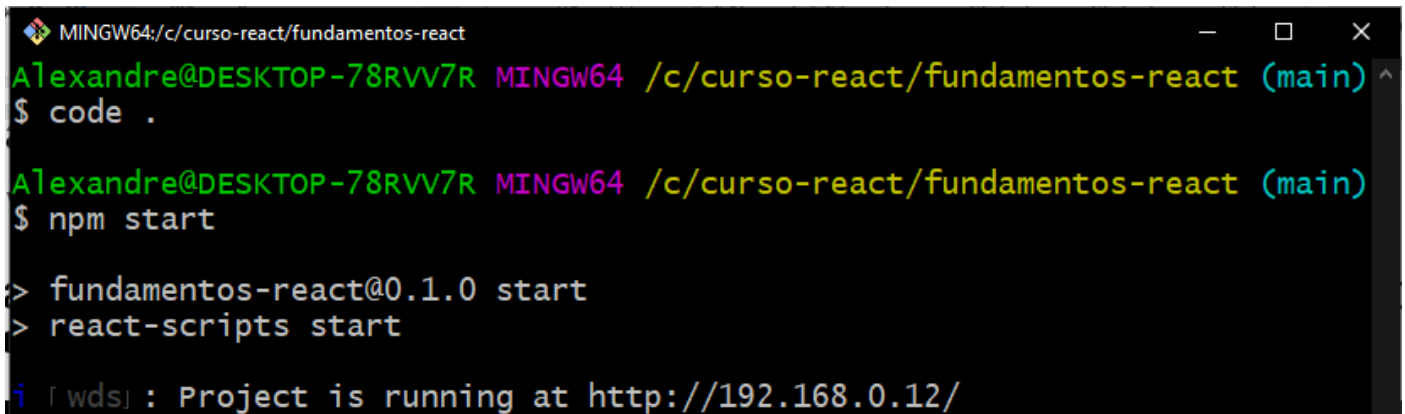
c:\curso-react\fundamentos-react

Vamos dar um Git Bash Here nessa pasta e iniciar o VSCode com o comando **code .**



```
MINGW64:/c:/curso-react/fundamentos-react
Alexandre@DESKTOP-78RVV7R MINGW64 /c:/curso-react/fundamentos-react (main)
$ code .
```

Em seguida iniciar nosso servidor com o comando **npm start**



```
MINGW64:/c:/curso-react/fundamentos-react
Alexandre@DESKTOP-78RVV7R MINGW64 /c:/curso-react/fundamentos-react (main)
$ code .

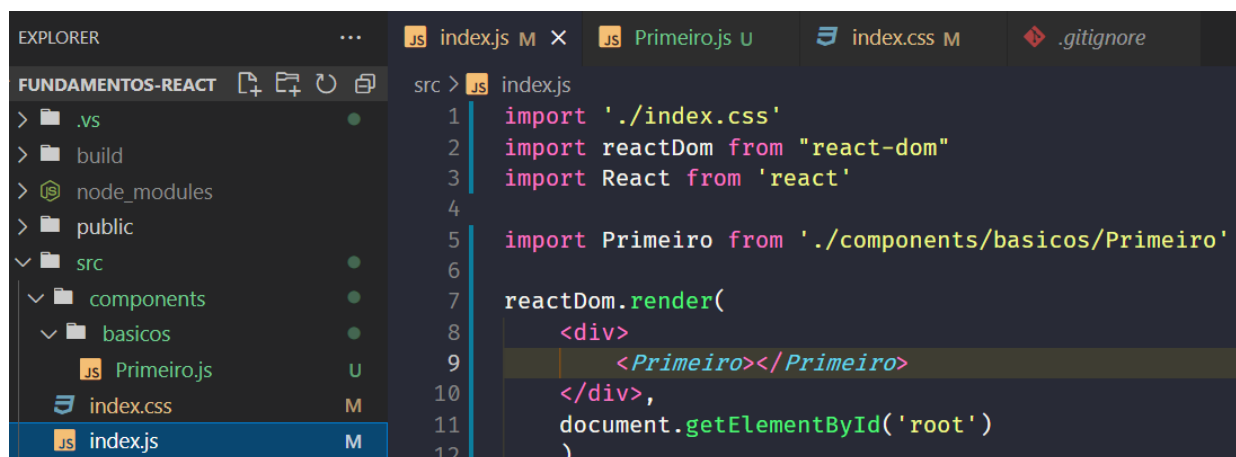
Alexandre@DESKTOP-78RVV7R MINGW64 /c:/curso-react/fundamentos-react (main)
$ npm start

> fundamentos-react@0.1.0 start
> react-scripts start

i [wds] : Project is running at http://192.168.0.12/
```

Tínhamos também os seguintes arquivos e estruturas:

Arquivo index.js



```
EXPLORER
FUNDAMENTOS-REACT
  > .vs
  > build
  > node_modules
  > public
  > src
    > components
      > basicos
        JS Primeiro.js
        index.css
        JS index.js
src > JS index.js
1 import './index.css'
2 import ReactDOM from 'react-dom'
3 import React from 'react'
4
5 import Primeiro from './components/basicos/Primeiro'
6
7 ReactDOM.render(
8   <div>
9     <Primeiro></Primeiro>
10   </div>,
11   document.getElementById('root')
12 )
```

Arquivo index.css

The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure: `FUNDAMENTOS-REACT` (expanded) contains `.vs`, `build`, `node_modules`, `public`, and `src`. The `src` folder is expanded, showing `components` (expanded) which contains `basicos` (expanded) with `Primeiro.js`, and `index.css` (selected). The main editor shows the content of `index.css`:

```
src > index.css > ...
1  body {
2      background-color: #222;
3      color: #fff;
4  }
5
```

E o arquivo componente Primeiro.js

The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure: `FUNDAMENTOS-REACT` (expanded) contains `.vs`, `build`, `node_modules`, `public`, and `src`. The `src` folder is expanded, showing `components` (expanded) which contains `basicos` (expanded) with `Primeiro.js` (selected). The main editor shows the content of `Primeiro.js`:

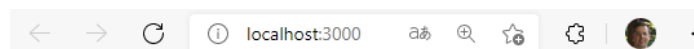
```
src > components > basicos > JS Primeiro.js > Primeiro > msg
1  import React from "react"
2
3  export default function Primeiro() {
4      const msg = 'Seja bem-vindo(a)!'
5      return (
6          <div>
7              <h2>'Primeiro Componente!</h2>
8              <p>{ msg }</p>
9          </div>
10     )
11 }
12
```

Até o momento, nossos componentes são considerados componentes funcionais pois são baseados em funções.

Agora vamos criar nosso segundo componente com parâmetros. Chamaremos eles de `ComParametro.jsx`

Mudaremos a extensão dos arquivos JS para JSX apenas para facilitar nosso trabalho na IDE. Por padrão o React aceita as duas extensões.

Quando mudarmos o nome do arquivo para `Primeiro.jsx`, a aplicação dará erro.

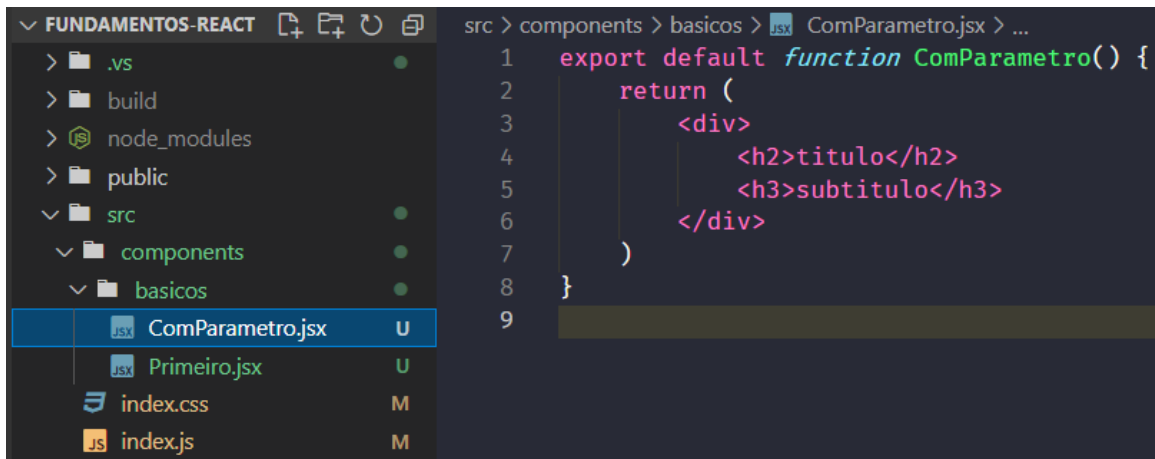


Failed to compile

```
./src/components/basicos/Primeiro.js
Error: ENOENT: no such file or directory,
open 'C:\curso-react\fundamentos-
react\src\components\basicos\Primeiro.js'
```

No terminal Git dê um <CTRL> + <C> e a aplicação irá parar. Reinicie com npm start.

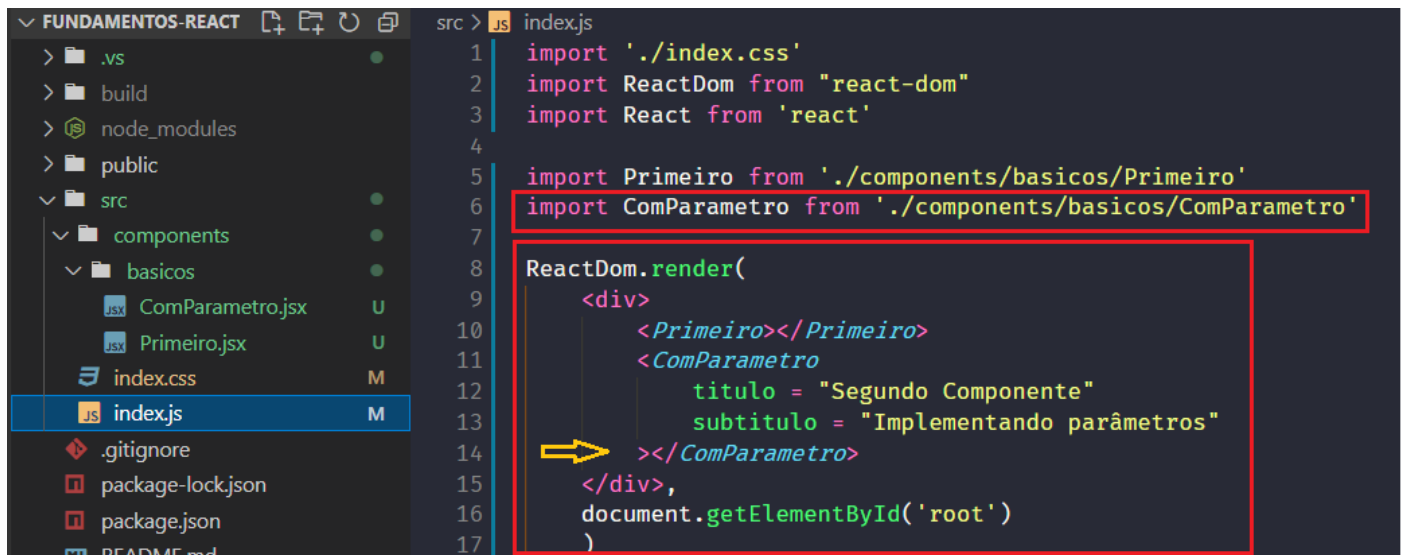
Vamos criar o arquivo ComParametro.jsx com o seguinte código passando parâmetros para o título e o subtítulo mostrarem na nossa aplicação.



The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure: **FUNDAMENTOS-REACT** with folders `.vs`, `build`, `node_modules`, `public`, and `src`. Inside `src`, there are folders `components` and `basicos`. The `ComParametro.jsx` file is selected in the `basicos` folder. On the right, the code editor shows the content of `ComParametro.jsx`:

```
1 export default function ComParametro() {
2   return (
3     <div>
4       <h2>titulo</h2>
5       <h3>subtitulo</h3>
6     </div>
7   )
8 }
9
```

Precisamos fazer o **import** do arquivo **ComParametro.jsx** lá no arquivo **index.js** e adicionar o código entre `<ComParametro>``</ComParametro>`



The screenshot shows the VS Code interface with the `index.js` file open. The file explorer on the left shows the `index.js` file selected in the `src` folder. The code editor shows the content of `index.js`:

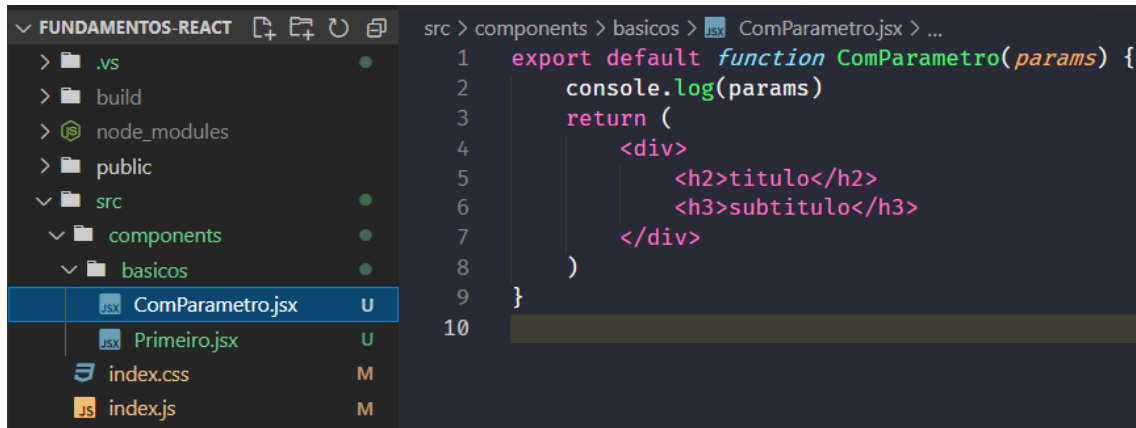
```
1 import './index.css'
2 import ReactDOM from 'react-dom'
3 import React from 'react'
4
5 import Primeiro from './components/basicos/Primeiro'
6 import ComParametro from './components/basicos/ComParametro'
7
8 ReactDOM.render(
9   <div>
10     <Primeiro></Primeiro>
11     <ComParametro
12       titulo = "Segundo Componente"
13       subtitulo = "Implementando parâmetros"
14     ></ComParametro>
15   </div>,
16   document.getElementById('root')
17 )
```

Red boxes highlight the import statements on lines 5 and 6, and the `<ComParametro>` JSX element on lines 11-14. A yellow arrow points to the closing tag `></ComParametro>` on line 14.

Nossa aplicação permanece inalterada, pois passamos nenhum parâmetro.



Agora passaremos como parâmetro a variável *params* e darei um `console.log(params)`

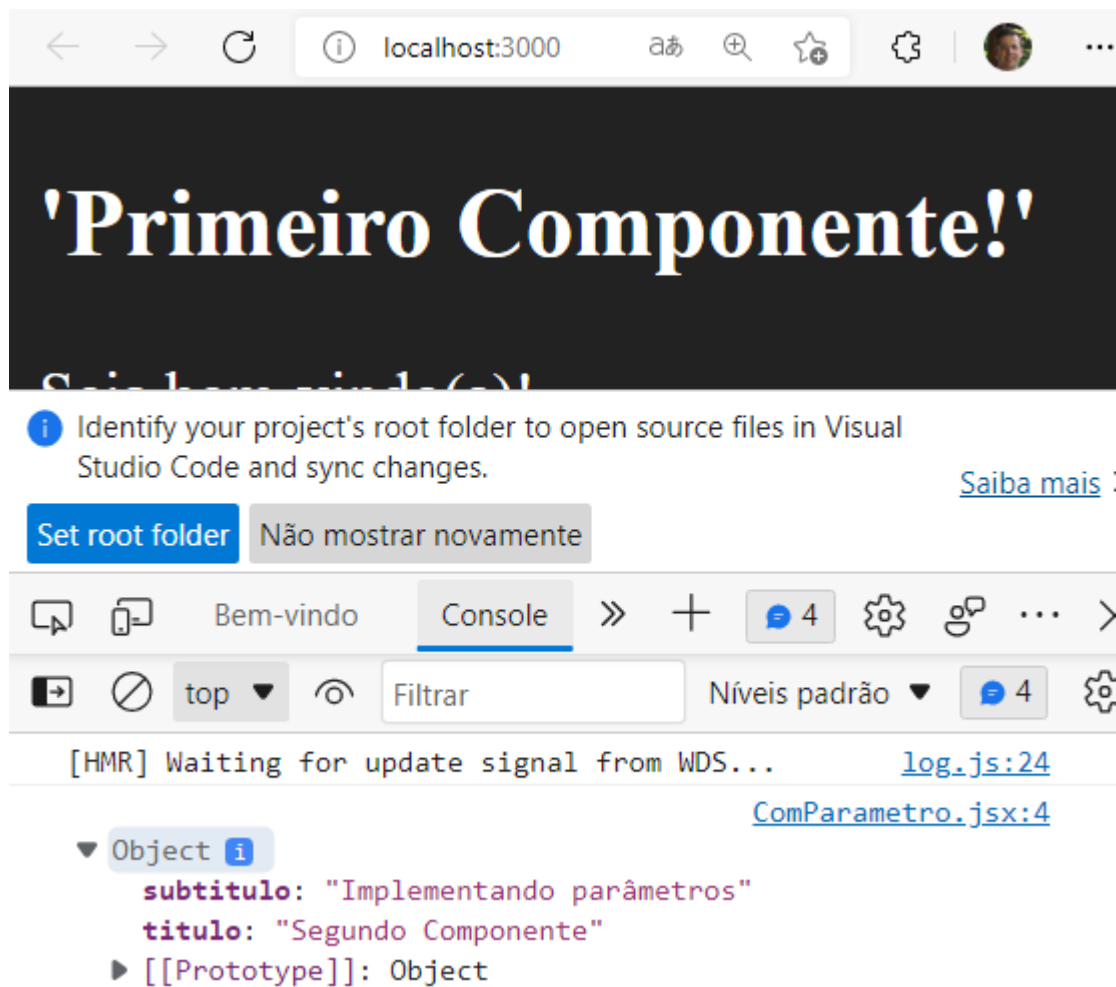


```

1  export default function ComParametro(params) {
2    console.log(params)
3    return (
4      <div>
5        <h2>titulo</h2>
6        <h3>subtitulo</h3>
7      </div>
8    )
9  }
10

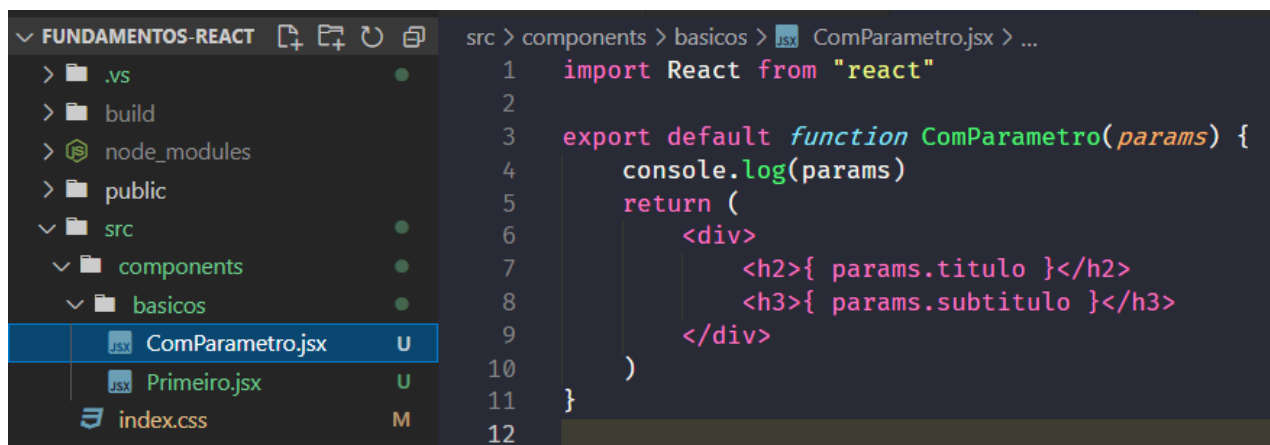
```

Daremos um refresh na nossa página e abriremos o console com <F12>.

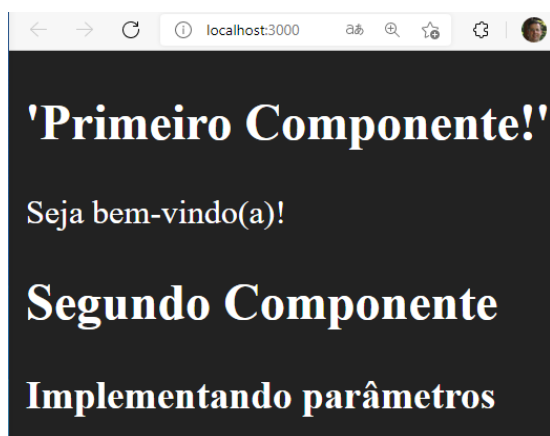


Note acima que é um objeto que tem um título e um subtítulo.

Passaremos esses parâmetros para serem apresentados na nossa página.



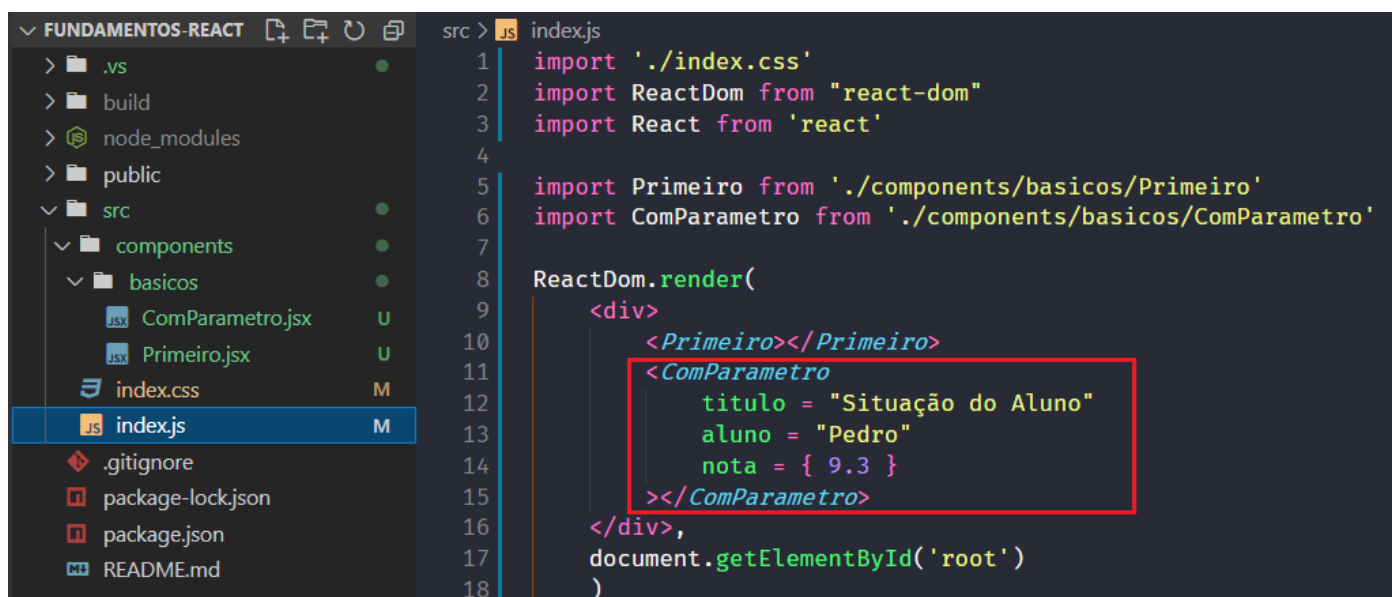
```
src > components > basicos > ComParametro.jsx > ...
1  import React from "react"
2
3  export default function ComParametro(params) {
4    console.log(params)
5    return (
6      <div>
7        <h2>{ params.titulo }</h2>
8        <h3>{ params.subtitulo }</h3>
9      </div>
10   )
11 }
12
```



Via de regra geral, não trabalhamos com a palavra *params* e sim *props* que significa *propriedades*

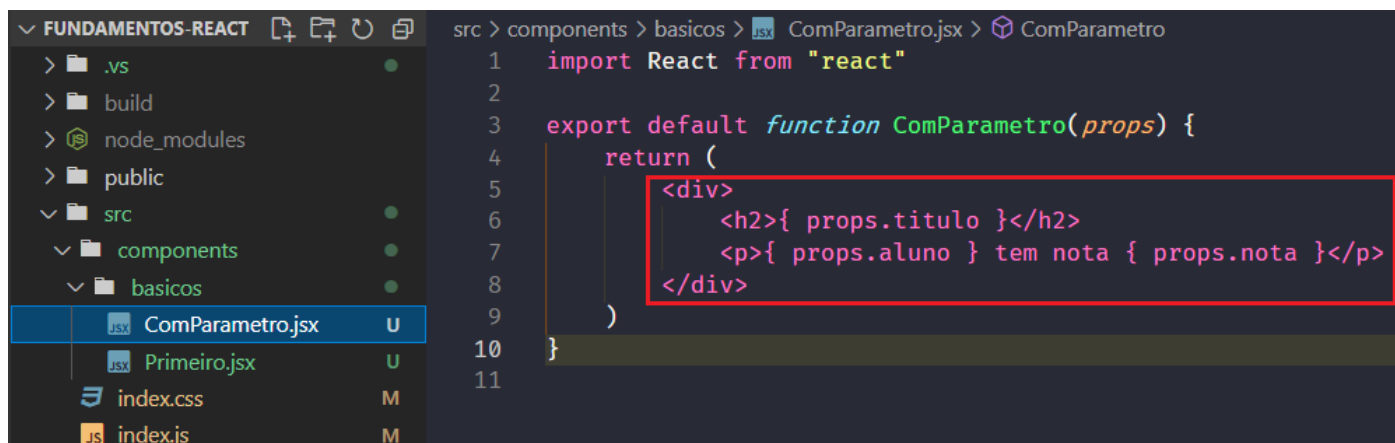
As propriedades desse componente são titulo e subtitulo , que foram passadas para a função ComParametro e conseguimos usar essa função a partir da interpolação.

Mudaremos nosso index.js para trabalhar com Aluno.



```
src > index.js
1  import './index.css'
2  import ReactDOM from "react-dom"
3  import React from 'react'
4
5  import Primeiro from './components/basicos/Primeiro'
6  import ComParametro from './components/basicos/ComParametro'
7
8  ReactDOM.render(
9    <div>
10      <Primeiro></Primeiro>
11      <ComParametro
12        titulo = "Situação do Aluno"
13        aluno = "Pedro"
14        nota = { 9.3 }
15      >></ComParametro>
16    </div>,
17    document.getElementById('root')
18  )
```

E o arquivo ComParametro.jsx

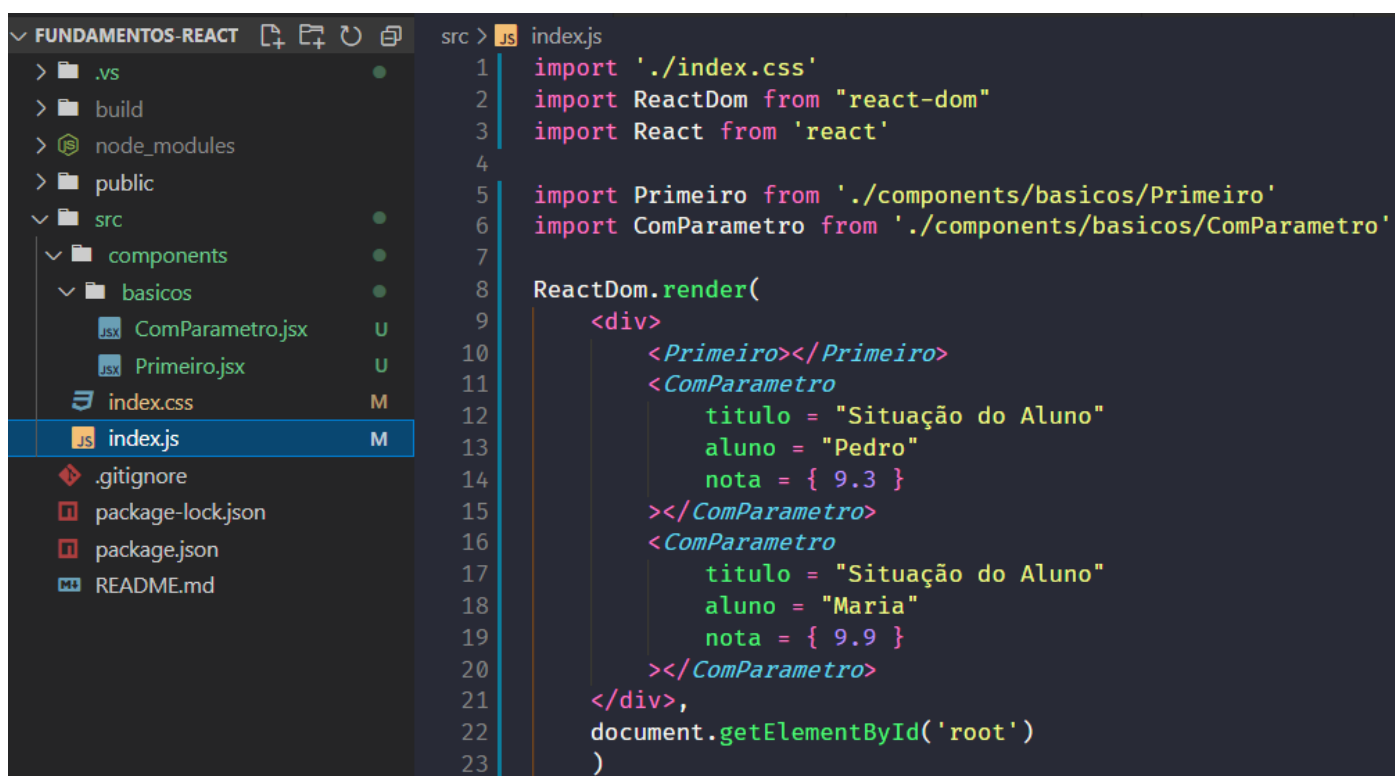


```

src > components > basicos > ComParametro.jsx > ComParametro
1  import React from "react"
2
3  export default function ComParametro(props) {
4    return (
5      <div>
6        <h2>{ props.titulo }</h2>
7        <p>{ props.aluno } tem nota { props.nota }</p>
8      </div>
9    )
10 }
11

```

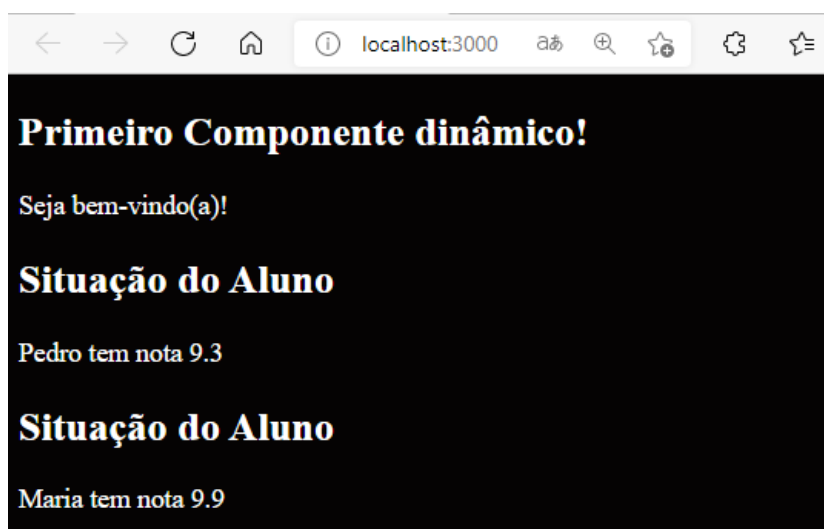
Agora que temos esse componente podemos passar outros alunos.



```

src > index.js
1  import './index.css'
2  import ReactDOM from "react-dom"
3  import React from 'react'
4
5  import Primeiro from './components/basicos/Primeiro'
6  import ComParametro from './components/basicos/ComParametro'
7
8  ReactDOM.render(
9    <div>
10     <Primeiro></Primeiro>
11     <ComParametro
12       titulo = "Situação do Aluno"
13       aluno = "Pedro"
14       nota = { 9.3 }
15     ></ComParametro>
16     <ComParametro
17       titulo = "Situação do Aluno"
18       aluno = "Maria"
19       nota = { 9.9 }
20     ></ComParametro>
21   </div>,
22   document.getElementById('root')
23 )

```



Essa é a grande ideia atrás do componente. Você define ele e usa quantas vezes for necessário para ter o máximo de reuso possível.

Podemos ainda incrementar esse componente colocando o status de aprovado ou em recuperação.

```

1 import React from "react"
2
3 export default function ComParametroNovo(props) {
4   const status = props.nota >= 7 ? 'Aprovado' : 'Em recuperação'
5   return (
6     <div>
7       <h2>{ props.titulo }</h2>
8       <p>{ props.aluno }, tem nota { props.nota }, e esta { status }.</p>
9     </div>
10  )
11 }
12

```



Receber propriedades no seu componente dá a possibilidade de personalizar seu HTML.

Propriedades são somente leitura

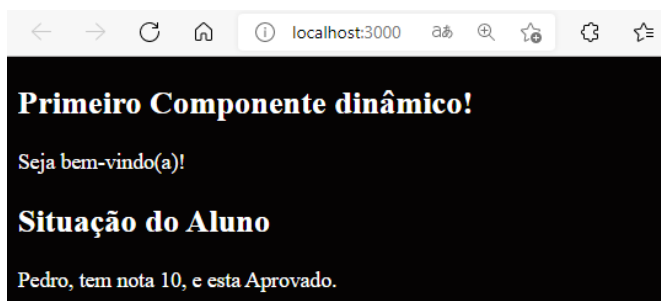
Um erro é que você pode achar que pode alterar o valor das propriedades. As propriedades são somente leitura e não conseguimos alterá-las.

Eventualmente você pode fazer um processamento e atribuir a um outro valor para ser mostrado. Exemplo arredondar nota para cima.

```

1 import React from "react"
2
3 export default function ComParametroNovo(props) {
4   const status = props.nota >= 7 ? 'Aprovado' : 'Em recuperação'
5   const notaInt = Math.ceil(props.nota)
6   return (
7     <div>
8       <h2>{ props.titulo }</h2>
9       <p>{ props.aluno }, tem nota { notaInt }, e esta { status }.</p>
10     </div>
11  )
12 }

```



Se for necessário alterar uma propriedade durante o processamento da aplicação, podemos fazer isso também através de outras estruturas que são estados que podem evoluir com o tempo.

React fragment

O objetivo é simular erro com componentes adjacentes. Criaremos o arquivo `Fragmento.jsx`

```

src > components > basicos > Fragmento.jsx > ...
1  import React from "react"
2
3  export default function Fragmento(props) {
4    return (
5      <div>
6        <h2>Fragmento</h2>
7        <p>Cuidado com esse erro!</p>
8      </div>
9    )
10  }
11

```

E no arquivo `index.js` faremos a importação.

```

src > JS index.js
1  import './index.css'
2  import ReactDOM from "react-dom"
3  import React from "react"
4
5  import Primeiro from './components/basicos/Primeiro'
6  import ComParametro from './components/basicos/ComParametro'
7  import Fragmento from './components/basicos/Fragmento'
8
9  ReactDOM.render(
10    <div>
11      <Primeiro></Primeiro>
12      <ComParametro
13        titulo = "Situação do Aluno"
14        aluno = "Pedro"
15        nota = { 9.3 }
16      ></ComParametro>
17      <Fragmento></Fragmento>
18    </div>,
19    document.getElementById('root')
20  )
21

```

Retirando a `<div>`

```

src > components > basicos > Fragmento.jsx > ...
1  import React from "react"
2
3  export default function Fragmento(props) {
4    return (
5      <h2>Fragmento</h2>
6      <p>Cuidado com esse erro!</p>
7    )
8  }
9
10
11

```




Failed to compile

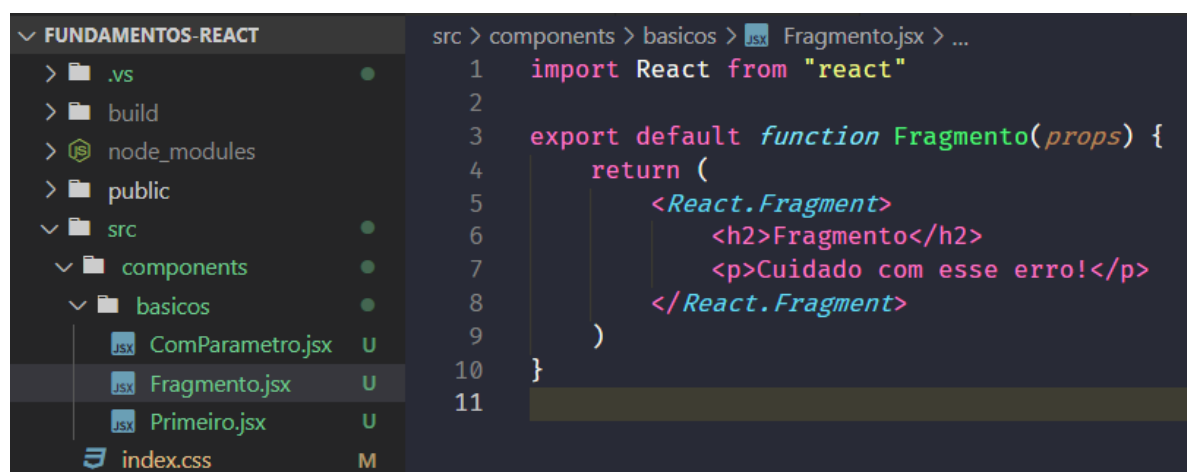
src\components\basicos\Fragmento.jsx
Syntax error: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>? (7:13)

```

5 |
6 |         <h2>Fragmento</h2>
> 7 |         <p>Cuidado com esse erro!</p>
  |           ^
8 |
9 |     )
10 | }

```

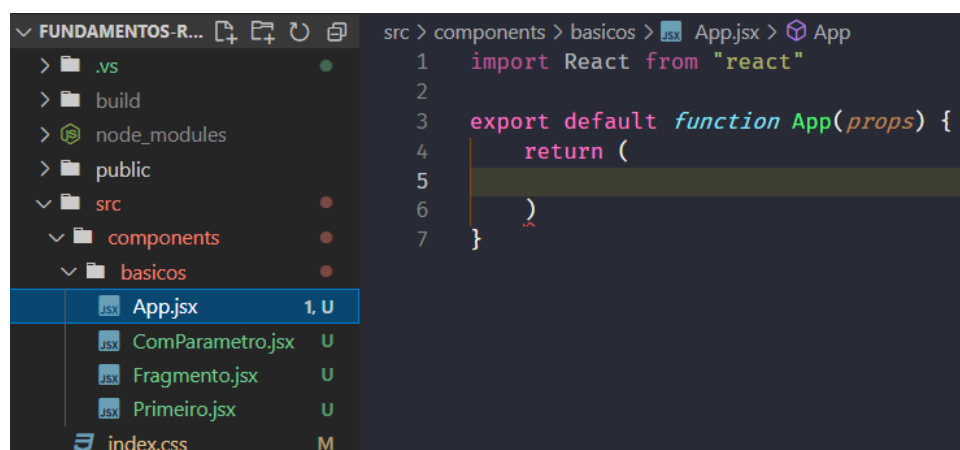
Mas você pode dizer que não quer envolver em uma <div> e sim retornar esses elementos diretamente. Podemos usar a tag <React.Fragment> ou ainda a tag <></> e não usar atributos.



Criando uma forma mais enxuta no React

Usamos uma Arrow function.

Criaremos o arquivo App.jsx



Pegaremos esse trecho de código.

```

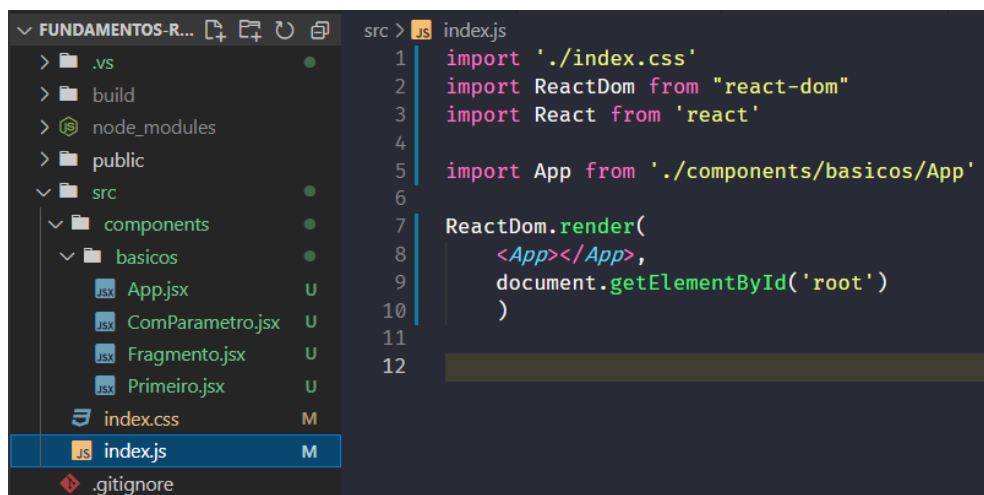
src > index.js
1  import './index.css'
2  import ReactDOM from 'react-dom'
3  import React from 'react'
4
5  import Primeiro from './components/basicos/Primeiro'
6  import ComParametro from './components/basicos/ComParametro'
7  import Fragmento from './components/basicos/Fragmento'
8
9  ReactDOM.render(
10     <div>
11       <Primeiro></Primeiro>
12       <ComParametro
13         titulo = "Situação do Aluno"
14         aluno = "Pedro"
15         nota = { 9.3 }
16       ></ComParametro>
17       <Fragmento></Fragmento>
18     </div>,
19     document.getElementById('root')
20   )
21
22
  
```

E colocaremos esse trecho de código no arquivo App.jsx e os imports

```

src > components > basicos > App.jsx
1  import React from 'react'
2
3  import Primeiro from './Primeiro'
4  import ComParametro from './ComParametro'
5  import Fragmento from './Fragmento'
6
7  export default function App(props) {
8    return (
9      <div>
10        <h1>Fundamentos React</h1>
11        <Fragmento></Fragmento>
12        <ComParametro
13          titulo="Situação do Aluno"
14          aluno="Pedro"
15          nota={9.3}
16        ></ComParametro>
17        <Primeiro></Primeiro>
18      </div>
19    )
20  }
  
```

E o arquivo final index.js ficará

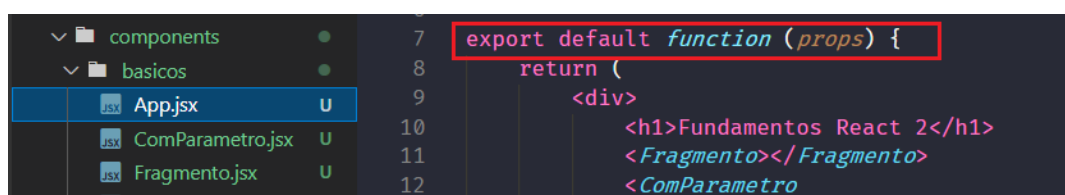


```

1 import './index.css'
2 import ReactDOM from 'react-dom'
3 import React from 'react'
4
5 import App from './components/basicos/App'
6
7 ReactDOM.render(
8   <App></App>,
9   document.getElementById('root')
10 )
11
12

```

Podemos deixar como um a função anônima

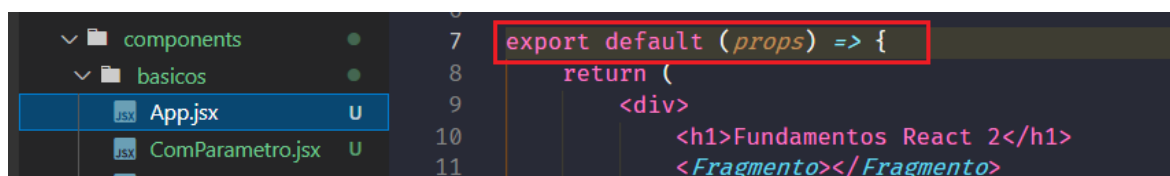


```

7 export default function (props) {
8   return (
9     <div>
10       <h1>Fundamentos React 2</h1>
11       <Fragmento></Fragmento>
12       <ComParametro

```

E tirar o nome *function* e transformar em uma *arrow function*.



```

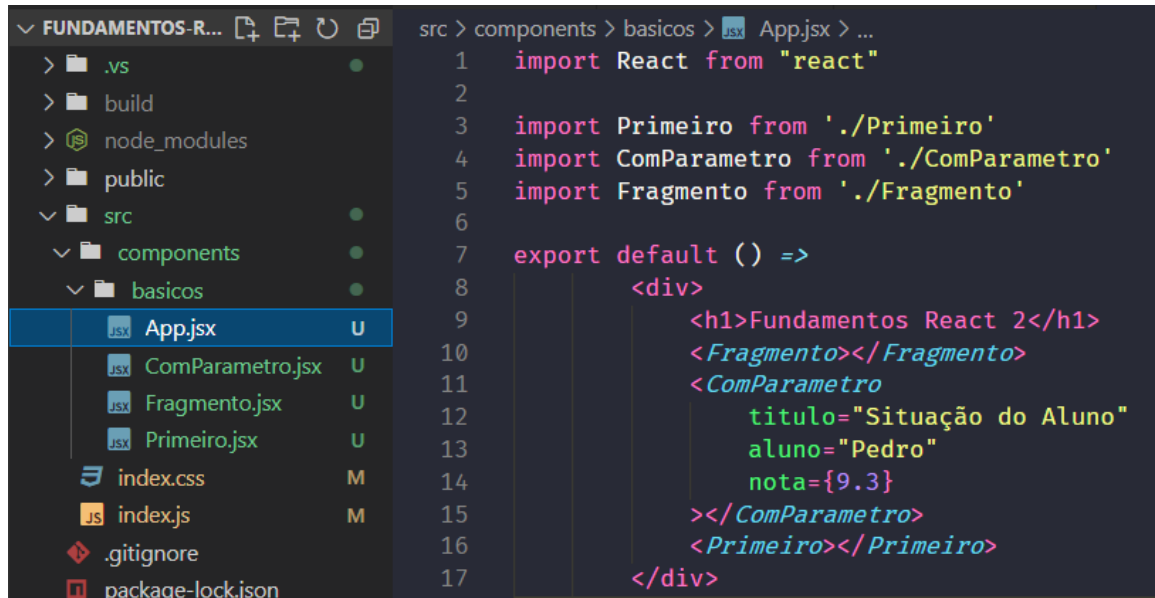
7 export default (props) => {
8   return (
9     <div>
10       <h1>Fundamentos React 2</h1>
11       <Fragmento></Fragmento>

```

E por fim como estou passando apenas um parâmetro posso colocar somente os parênteses e eliminar o corpo da função.

Eliminando o corpo da função retiramos também o return, pois entendemos que irá retornar de forma implícita toda a função.

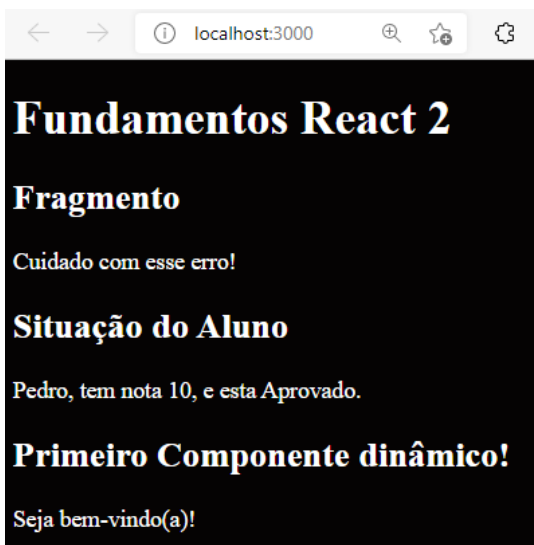
Temos a forma mais enxuta da função.



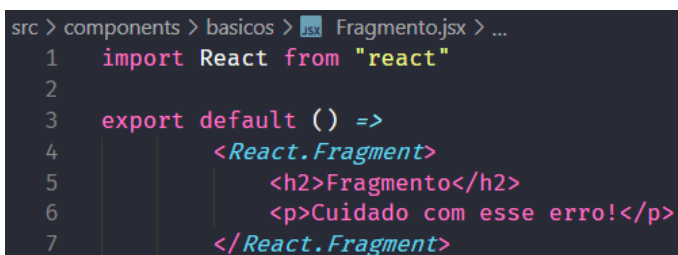
```

src > components > basicos > App.jsx > ...
1  import React from "react"
2
3  import Primeiro from './Primeiro'
4  import ComParametro from './ComParametro'
5  import Fragmento from './Fragmento'
6
7  export default () =>
8      <div>
9          <h1>Fundamentos React 2</h1>
10         <Fragmento></Fragmento>
11         <ComParametro
12             titulo="Situação do Aluno"
13             aluno="Pedro"
14             nota={9.3}
15         ></ComParametro>
16         <Primeiro></Primeiro>
17     </div>

```



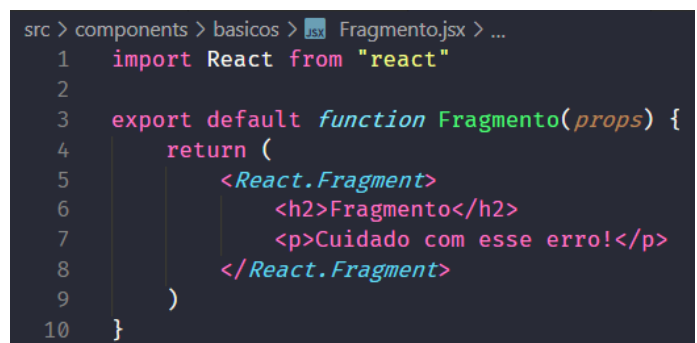
Pode ser dessa forma mais enxuta da esquerda ou da direita a normal.



```

src > components > basicos > Fragmento.jsx > ...
1  import React from "react"
2
3  export default () =>
4      <React.Fragment>
5          <h2>Fragmento</h2>
6          <p>Cuidado com esse erro!</p>
7      </React.Fragment>

```



```

src > components > basicos > Fragmento.jsx > ...
1  import React from "react"
2
3  export default function Fragmento(props) {
4      return (
5          <React.Fragment>
6              <h2>Fragmento</h2>
7              <p>Cuidado com esse erro!</p>
8          </React.Fragment>
9      )
10 }

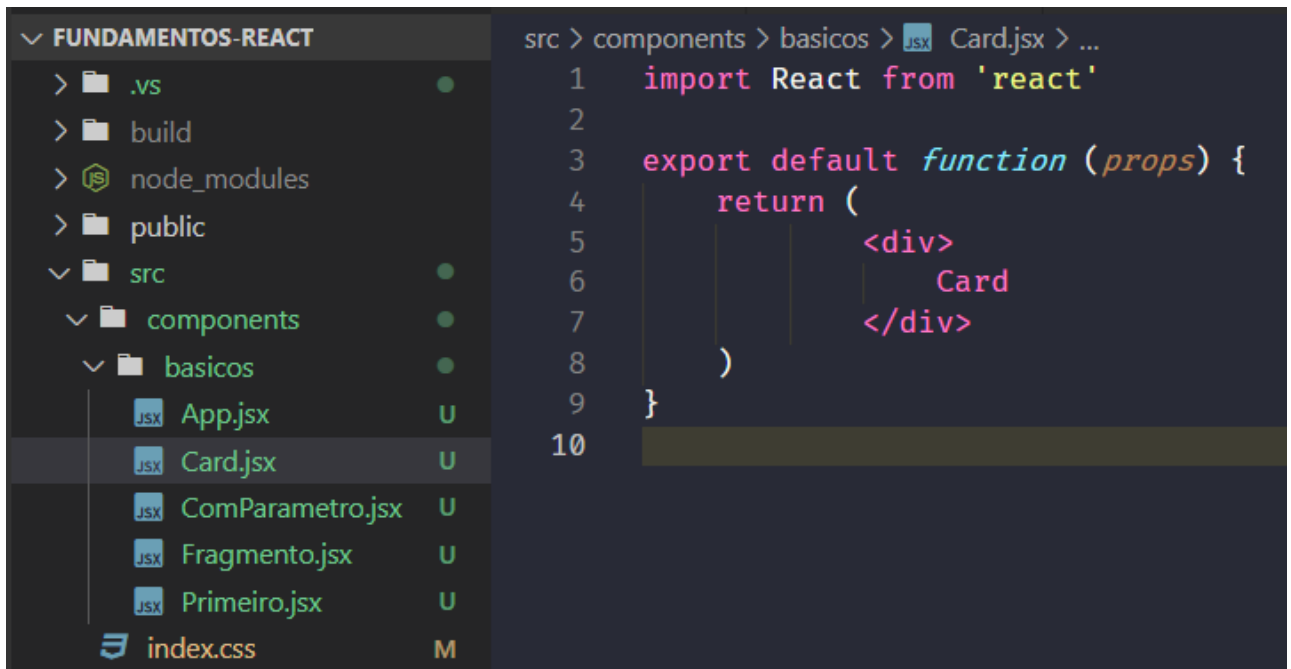
```

Para alinhar a indentação no Windows é o <Alt> + <Shift> + <F>

E no Linux é <Ctrl> + <Shift> + <i>

Vamos agora trabalhar com componentes estilizados com CSS.

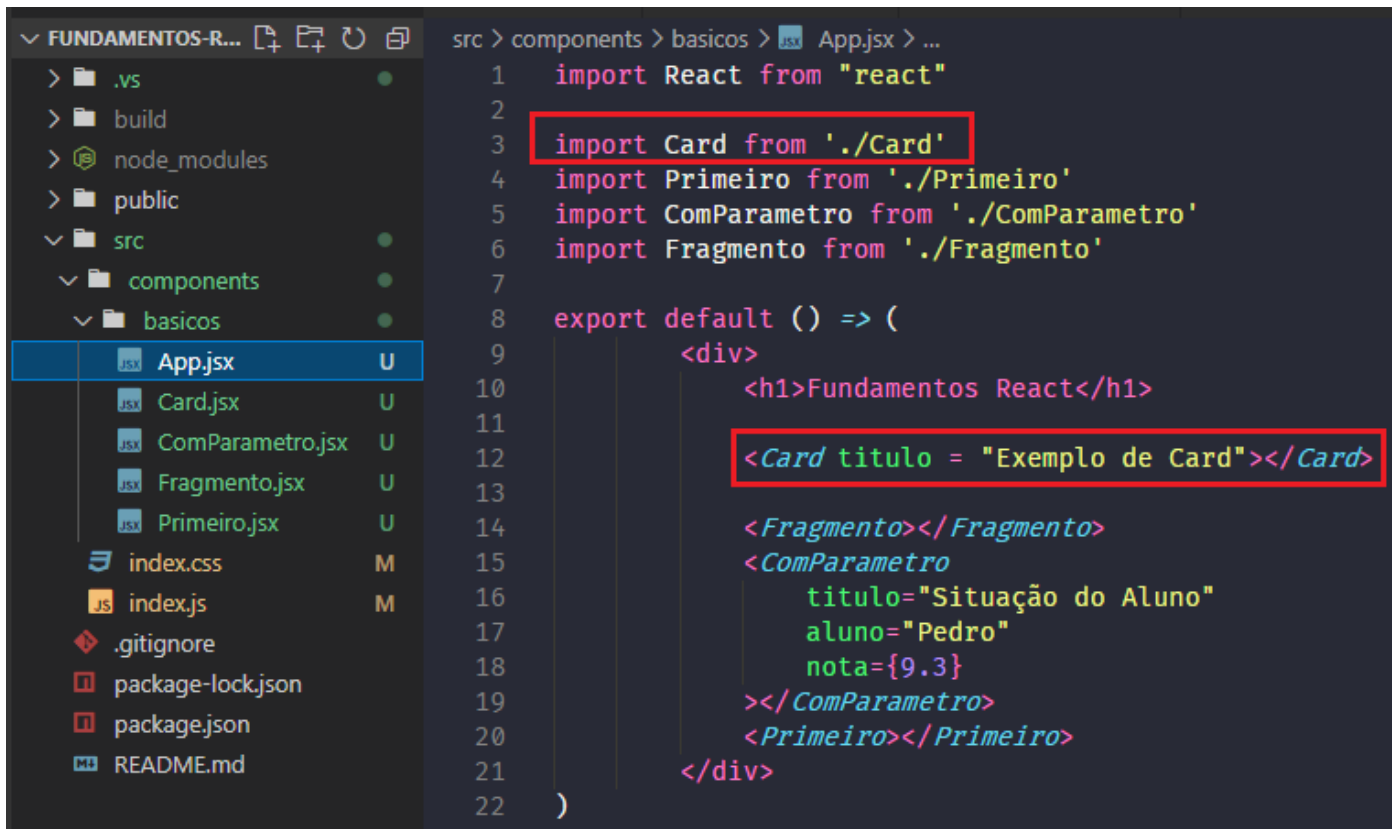
Criaremos um arquivo Card.jsx e outro Card.css na pasta basicos.



The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure: `FUNDAMENTOS-REACT` (expanded) contains `.vs`, `build`, `node_modules`, `public`, and `src`. The `src` folder is expanded, showing `components` (expanded) and `basicos` (expanded). Inside `basicos`, the files `App.jsx`, `Card.jsx`, `ComParametro.jsx`, `Fragmento.jsx`, `Primeiro.jsx`, and `index.css` are listed. `Card.jsx` is selected. On the right, the code editor shows the content of `Card.jsx` at the path `src > components > basicos > Card.jsx`. The code is as follows:

```
1 import React from 'react'
2
3 export default function (props) {
4   return (
5     <div>
6       Card
7     </div>
8   )
9 }
10
```

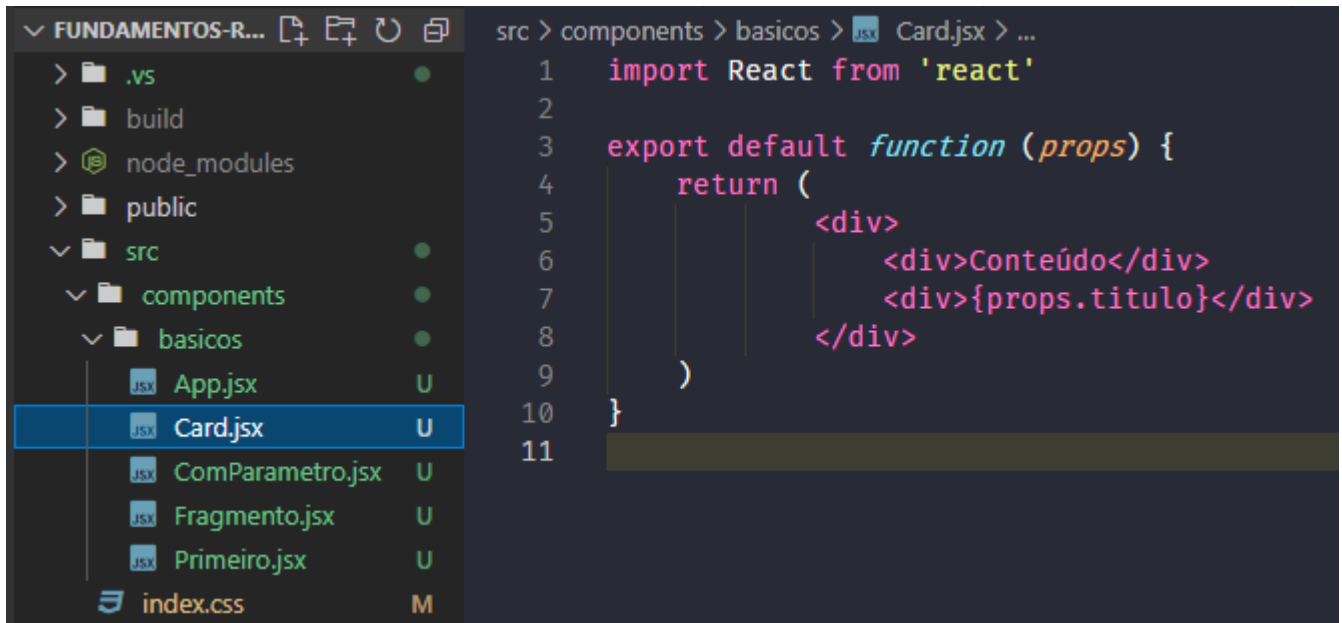
E vamos importar no arquivo App.jsx



The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure, with `App.jsx` selected in the `basicos` folder. On the right, the code editor shows the content of `App.jsx` at the path `src > components > basicos > App.jsx`. The code is as follows:

```
1 import React from "react"
2
3 import Card from './Card'
4 import Primeiro from './Primeiro'
5 import ComParametro from './ComParametro'
6 import Fragmento from './Fragmento'
7
8 export default () => (
9   <div>
10     <h1>Fundamentos React</h1>
11
12     <Card titulo = "Exemplo de Card"></Card>
13
14     <Fragmento></Fragmento>
15     <ComParametro
16       titulo="Situação do Aluno"
17       aluno="Pedro"
18       nota={9.3}
19     ></ComParametro>
20     <Primeiro></Primeiro>
21   </div>
22 )
```

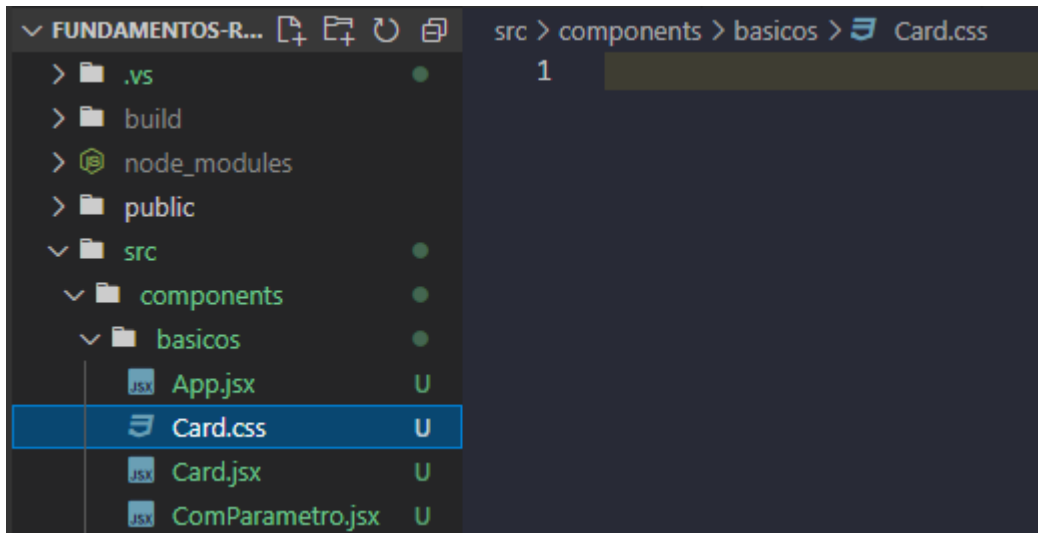
Na pasta Card.jsx faremos as mudanças.



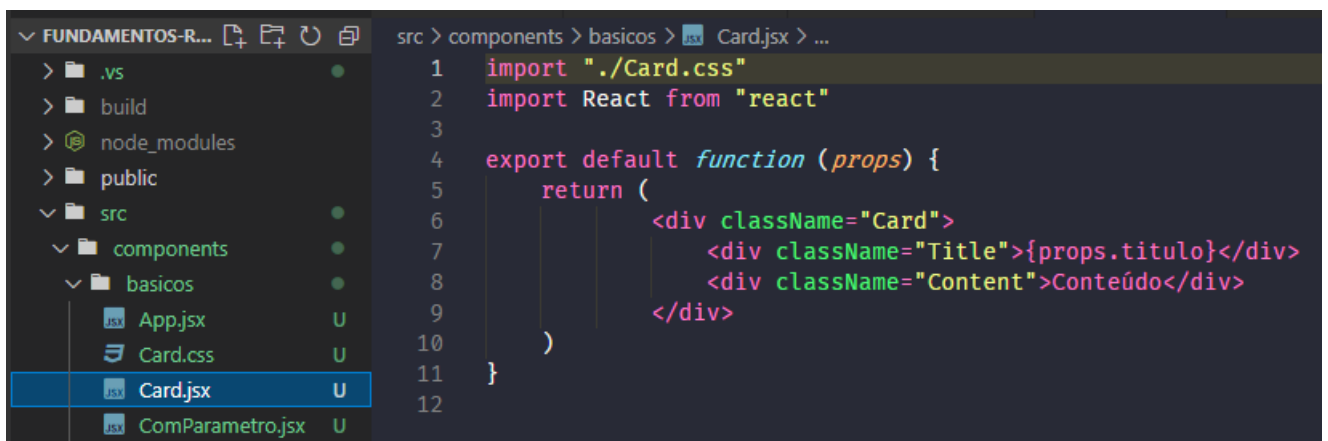
```
src > components > basicos > Card.jsx > ...
1  import React from 'react'
2
3  export default function (props) {
4    return (
5      <div>
6        <div>Conteúdo</div>
7        <div>{props.titulo}</div>
8      </div>
9    )
10 }
11
```



Vamos criar o arquivo Card.css



Criado o arquivo Card.css, vamos referencia-lo no arquivo Card.jsx, fazendo o import Card.css e criando as className para pegar os estilos do arquivo Card.css



Podemos selecionar uma fonte no Google fonts e escolhe a font Oswald.

Selecione os tamanhos das fontes necessárias e copie o link.

```
<link rel="preconnect" href="https://fonts.googleapis.com">
```

```
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
```

```
<link
```

```
href="https://fonts.googleapis.com/css2?family=Oswald:wght@200;300;400;500;600;700&display=swap" rel="stylesheet">
```

Oswald

Select styles Glyphs About License

Medium 500

Almost before we knew it, we h — Remove this style

Semi-bold 600

Almost before we knew it, we — Remove this style

Bold 700

Almost before we knew it, we — Remove this style

Download family

Selected family

You can now share your selected families with others. Dismiss

Light 300 —

Regular 400 —

Medium 500 —

Semi-bold 600 —

Bold 700 —

☒ <link> ☐ @import

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Oswald:wght@200;300;400;500;600;700&display=swap" rel="stylesheet">
```

Vamos incluir esse estilo no nosso arquivo index.html

```
public > index.html > html > head > link
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8" />
6   <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
7   <meta name="viewport" content="width=device-width, initial-scale=1" />
8   <meta name="theme-color" content="#000000" />
9   <meta name="description" content="Web site created using create-react-app" />
10  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
11
12  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
13
14  <link rel="preconnect" href="https://fonts.googleapis.com">
15  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
16  <link href="https://fonts.googleapis.com/css2?family=Oswald:wght@200;300;400;500;600;700&display=swap" rel="stylesheet">
17
18  <title>React App</title>
19
20 </head>
21
```

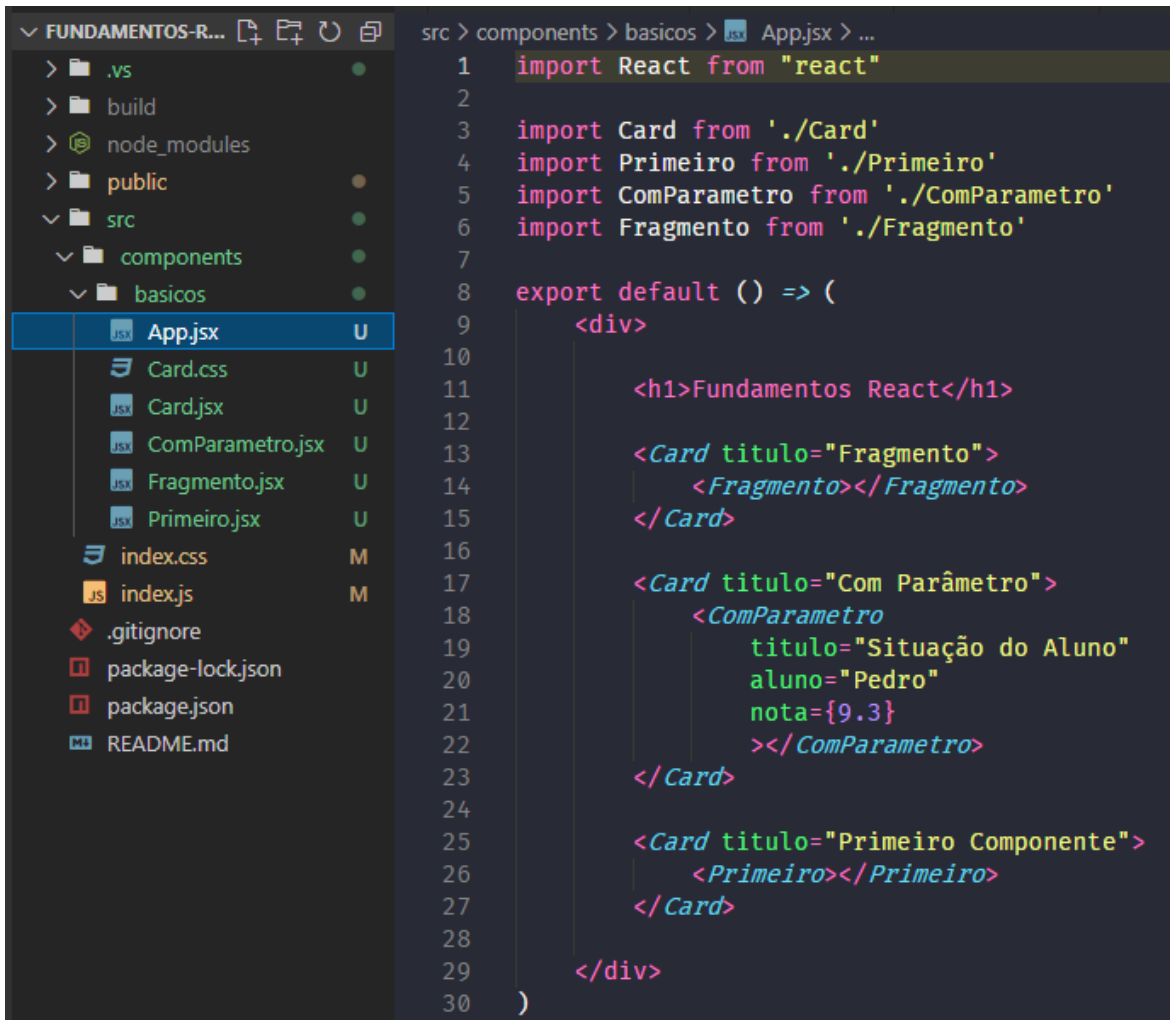
E colocar no index.css. Vamos colocar outros estilos para ganhar tempo.

```
src > index.css > body
1 body {
2   font-family: Oswald;
3   background-color: #222;
4   color: #fff;
5   text-align: center;
6   font-size: 1.4rem;
7 }
8
9 h1, h2, h3 {
10  font-weight: 200;
11 }
12 h2, h3 {
13  margin: 0;
14 }
```


Nossa página já está estilizada.

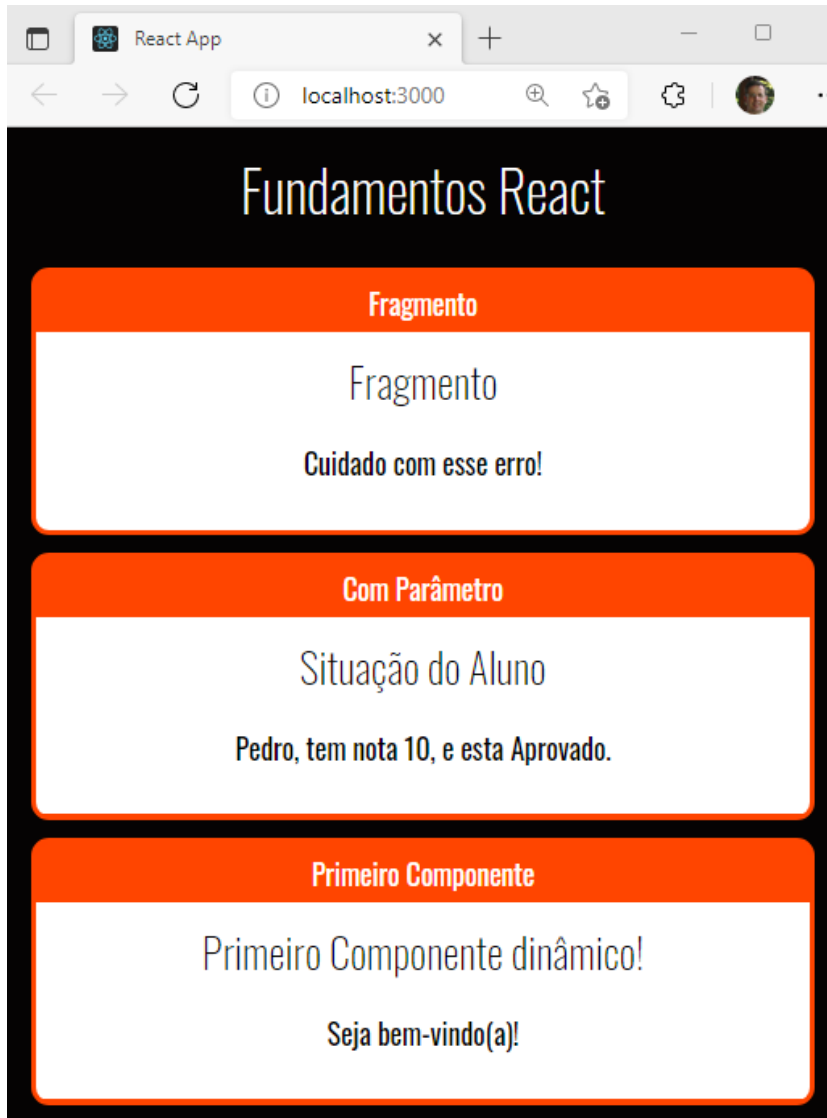


Agora é colocar os exemplos para dentro do conteúdo que é fixo



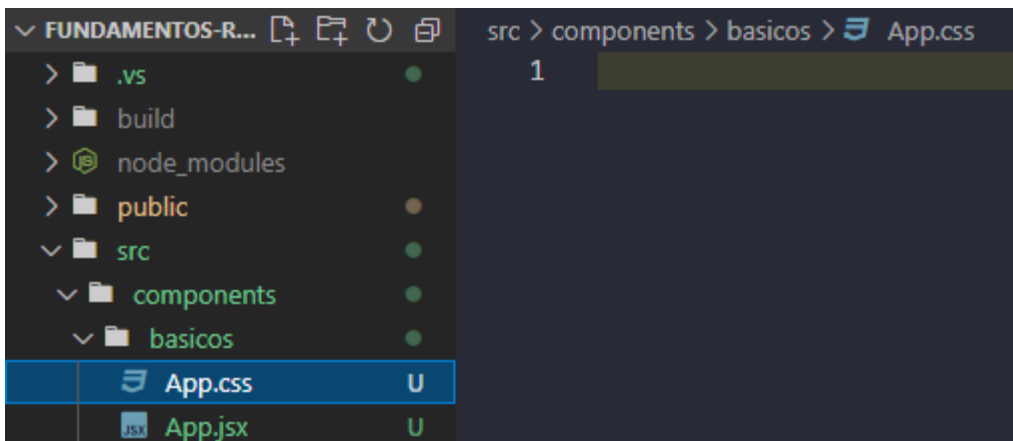
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a folder named 'FUNDAMENTOS-R...' containing subfolders like '.vs', 'build', 'node_modules', 'public', and 'src'. The 'src' folder is expanded, showing 'components' and 'basicos'. The 'App.jsx' file is selected in the 'basicos' folder. The code editor shows the content of 'App.jsx' with the following code:

```
src > components > basicos > App.jsx > ...
1  import React from "react"
2
3  import Card from './Card'
4  import Primeiro from './Primeiro'
5  import ComParametro from './ComParametro'
6  import Fragmento from './Fragmento'
7
8  export default () => (
9    <div>
10
11      <h1>Fundamentos React</h1>
12
13      <Card titulo="Fragmento">
14        <Fragmento></Fragmento>
15      </Card>
16
17      <Card titulo="Com Parâmetro">
18        <ComParametro
19          titulo="Situação do Aluno"
20          aluno="Pedro"
21          nota={9.3}
22        ></ComParametro>
23      </Card>
24
25      <Card titulo="Primeiro Componente">
26        <Primeiro></Primeiro>
27      </Card>
28
29    </div>
30  )
```

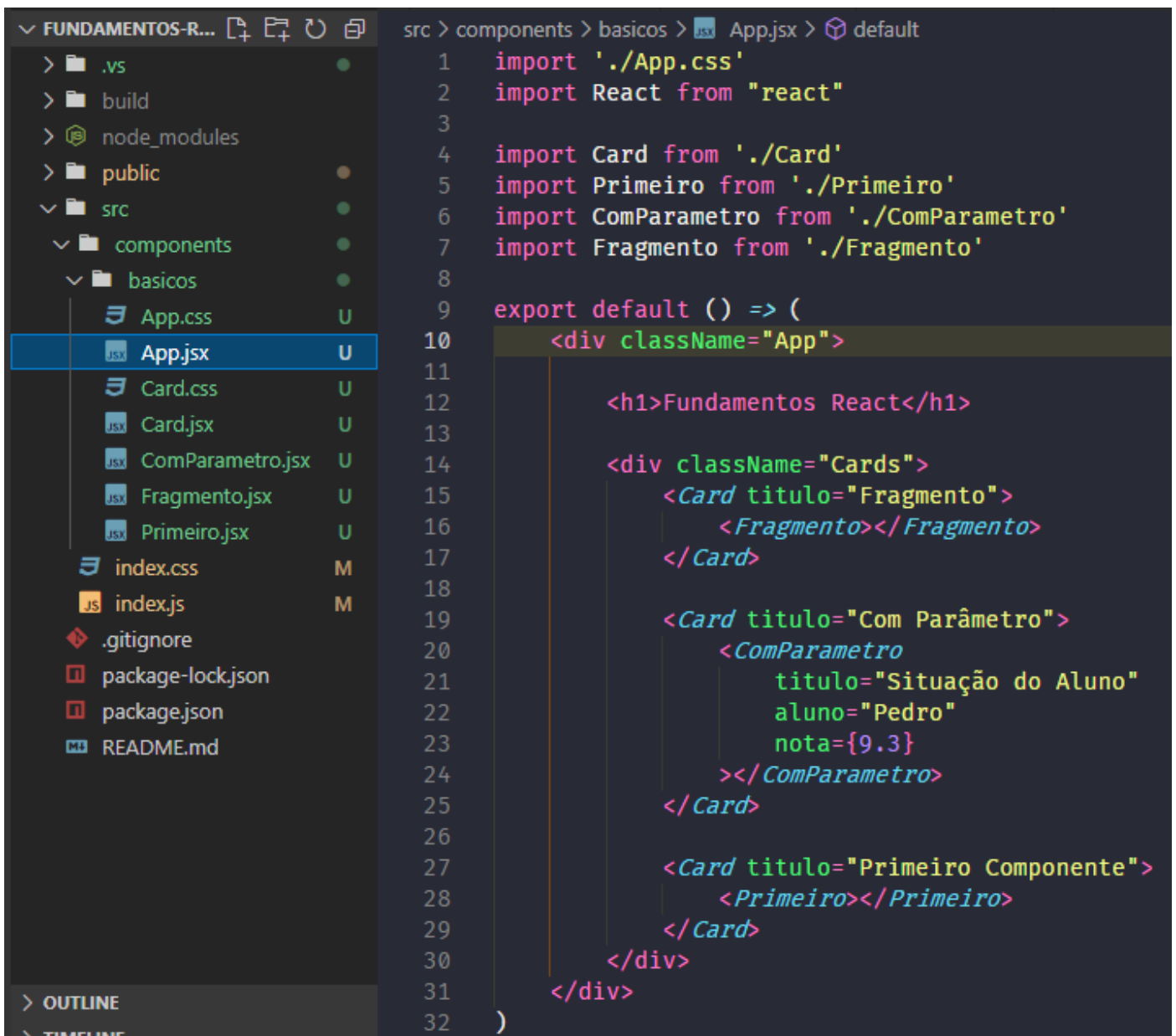


Podemos melhorar a responsividade do nosso projeto com flex-box.

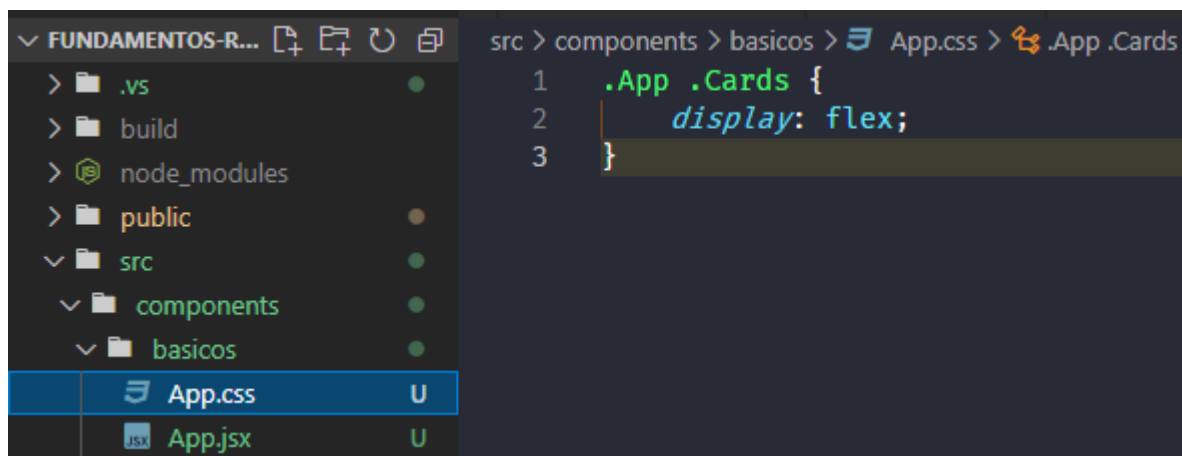
Vamos criar o arquivo App.css para estilizar



No nosso arquivo App.jsx, colocaremos todos os cards em uma `<div>` e fazer o impor do App.css.



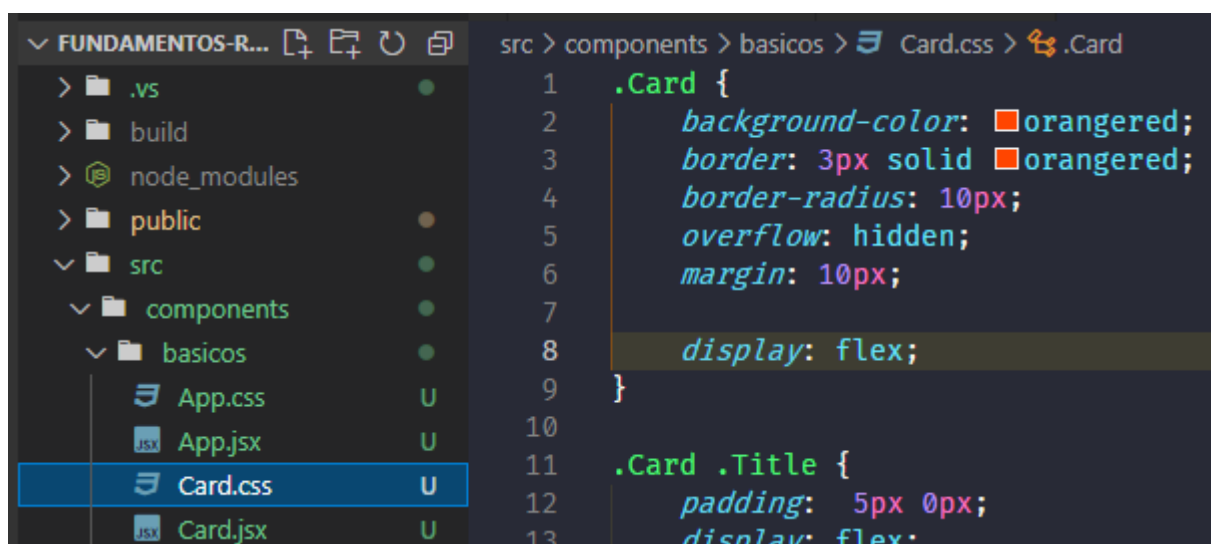
Apenas colocando display flex no App.css já notamos uma grande diferença.



```
src > components > basicos > App.css > .App .Cards
1  .App .Cards {
2      display: flex;
3  }
```



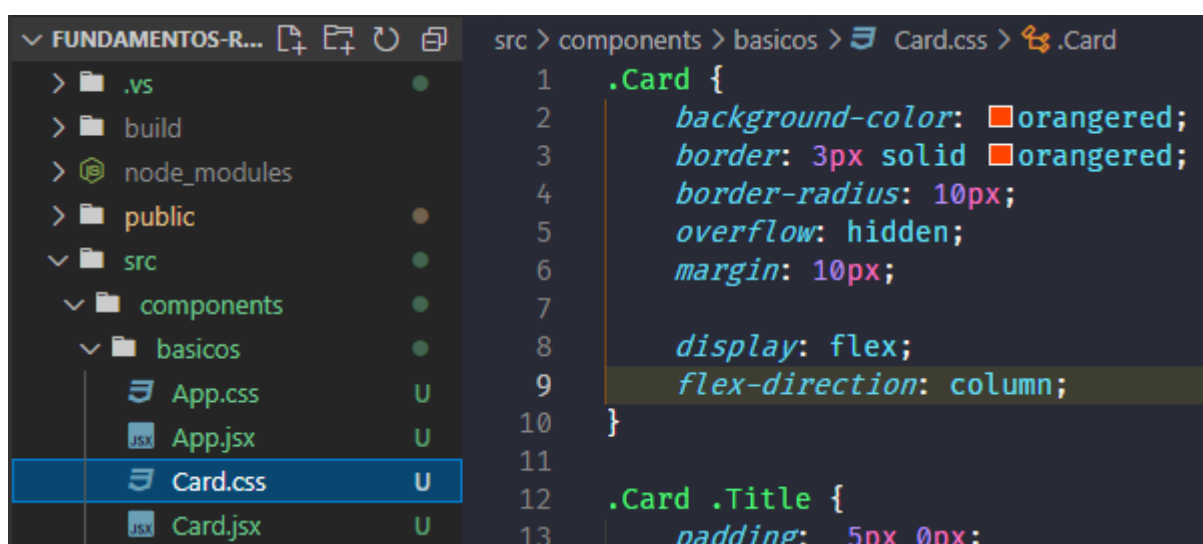
Aplicando display flex no Card.css



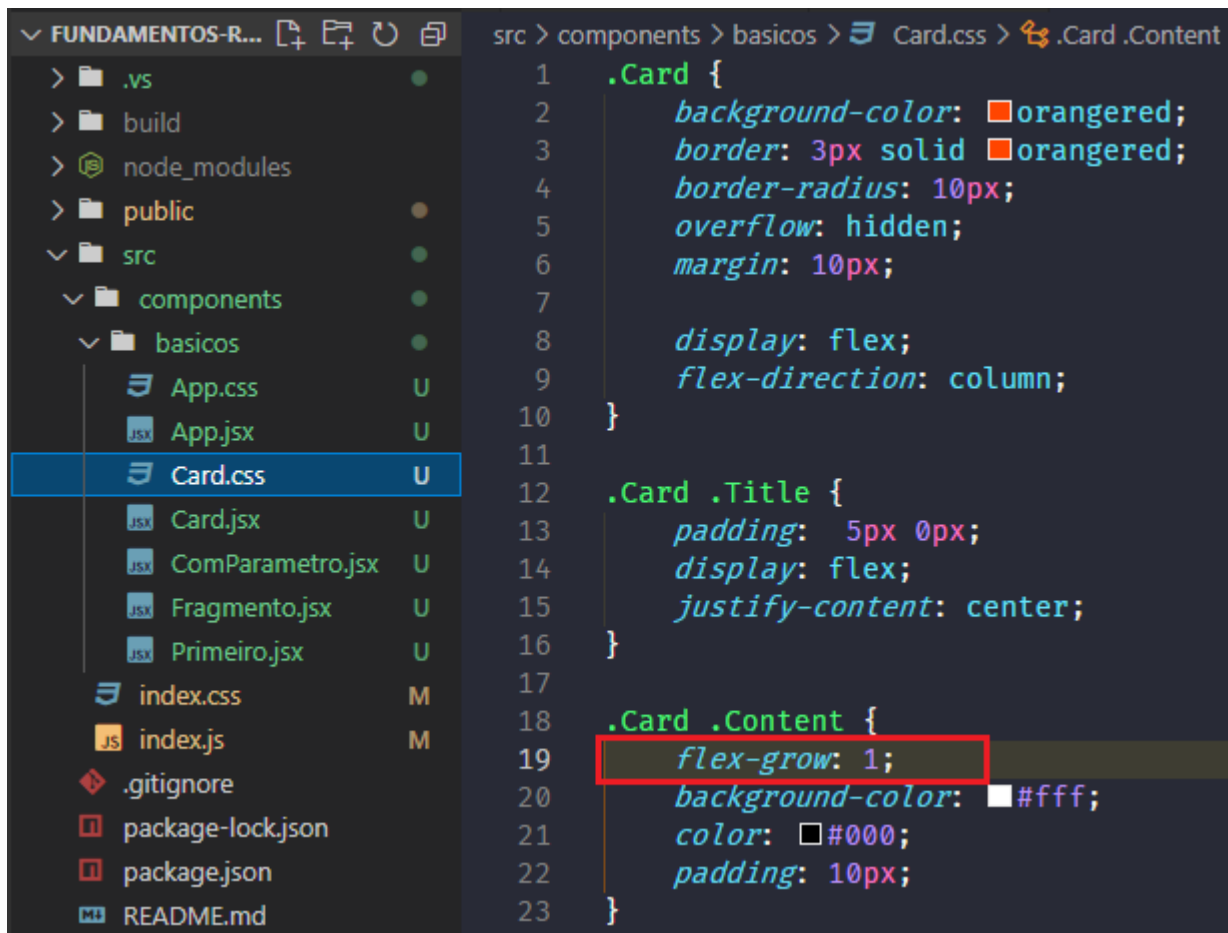
```
src > components > basicos > Card.css > .Card
1  .Card {
2      background-color: orange;
3      border: 3px solid orange;
4      border-radius: 10px;
5      overflow: hidden;
6      margin: 10px;
7
8      display: flex;
9  }
10
11 .Card .Title {
12     padding: 5px 0px;
13     display: flex;
```



Aplicando flex-direction: column;



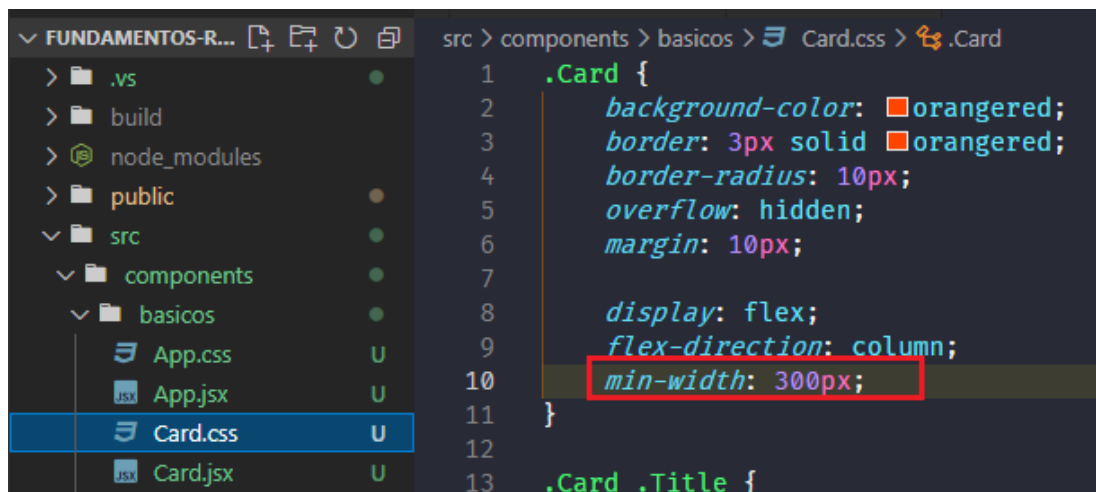
Agora o conteúdo crescerá e o título ficará fixo.



```
src > components > basicos > Card.css > .Card .Content
1  .Card {
2      background-color: orangered;
3      border: 3px solid orangered;
4      border-radius: 10px;
5      overflow: hidden;
6      margin: 10px;
7
8      display: flex;
9      flex-direction: column;
10 }
11
12 .Card .Title {
13     padding: 5px 0px;
14     display: flex;
15     justify-content: center;
16 }
17
18 .Card .Content {
19     flex-grow: 1;
20     background-color: #fff;
21     color: #000;
22     padding: 10px;
23 }
```



É interessante colocar um tamanho mínimo para os cards, no nosso caso 300px.

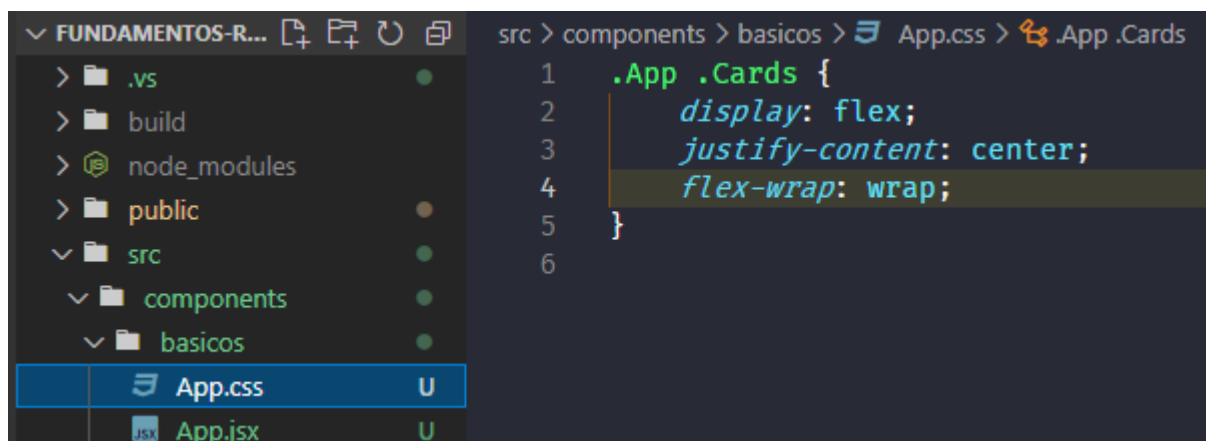


```
1 .Card {
2   background-color: #f4a460;
3   border: 3px solid #f4a460;
4   border-radius: 10px;
5   overflow: hidden;
6   margin: 10px;
7
8   display: flex;
9   flex-direction: column;
10  min-width: 300px;
11 }
12
13 .Card .Title {
```

Porém os cards permanecem na mesma linha.



A propriedade flex-wrap: wrap; faz os cards pularem para a próxima linha e de forma centralizada com justify-content: center;

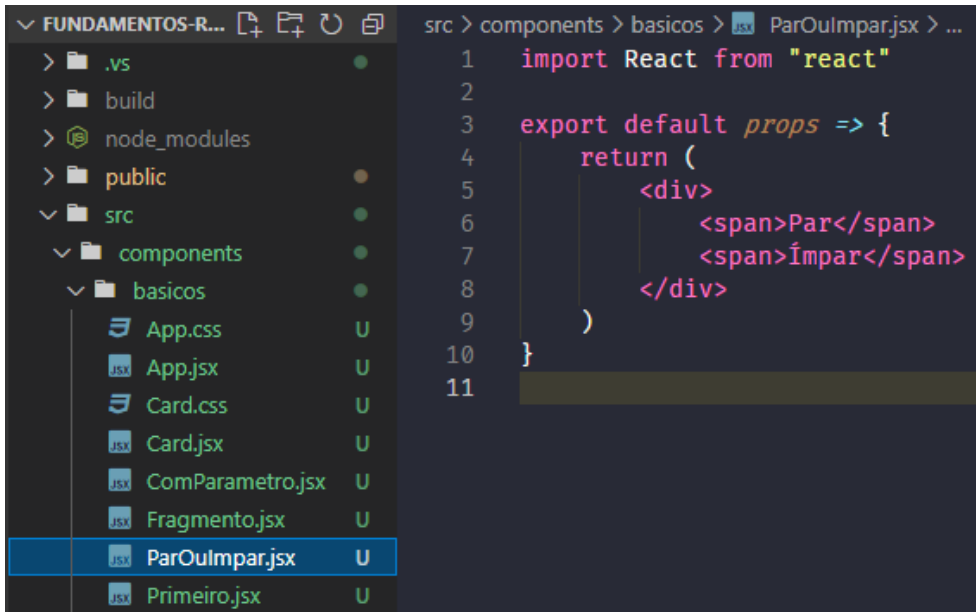


```
1 .App .Cards {
2   display: flex;
3   justify-content: center;
4   flex-wrap: wrap;
5 }
6
```




Renderização condicional

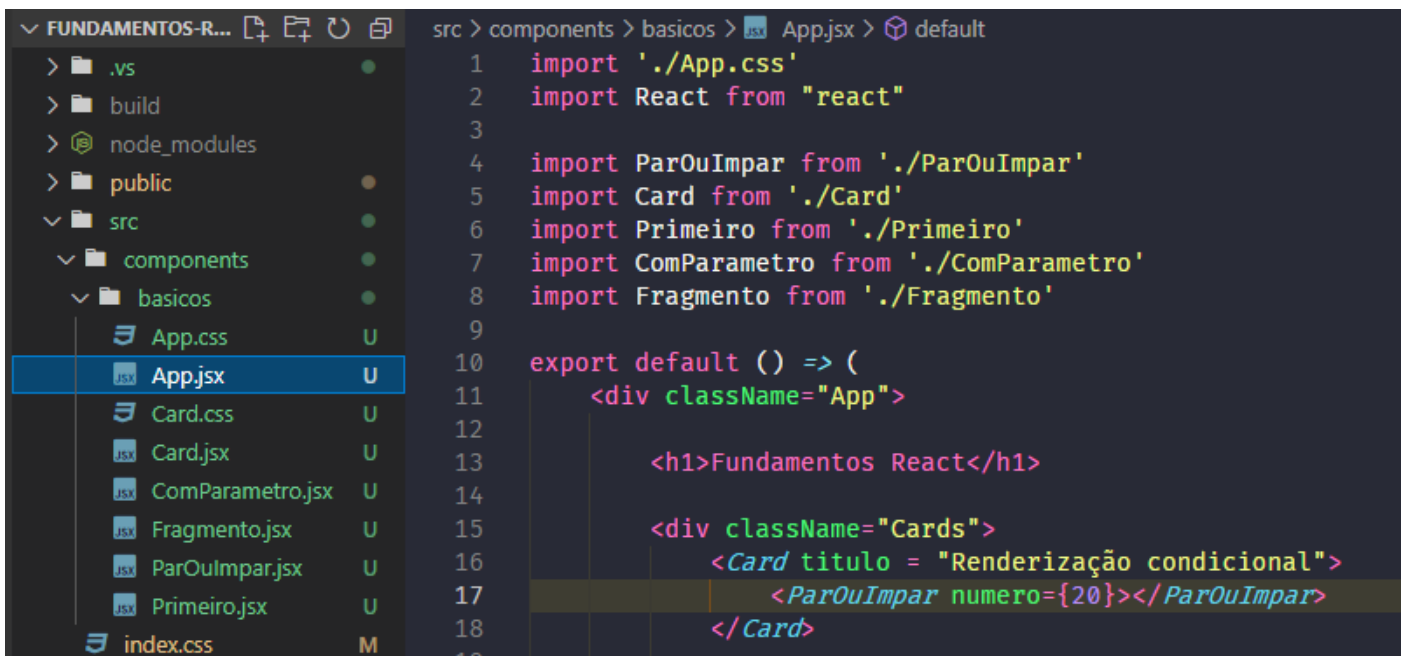
Vamos criar uma aplicação para renderizar somente se o número for par. Utilizaremos o arquivo ParOuImpar.jsx



```
src > components > basics > ParOuImpar.jsx > ...
1  import React from "react"
2
3  export default props => {
4    return (
5      <div>
6        <span>Par</span>
7        <span>Ímpar</span>
8      </div>
9    )
10 }
11
```

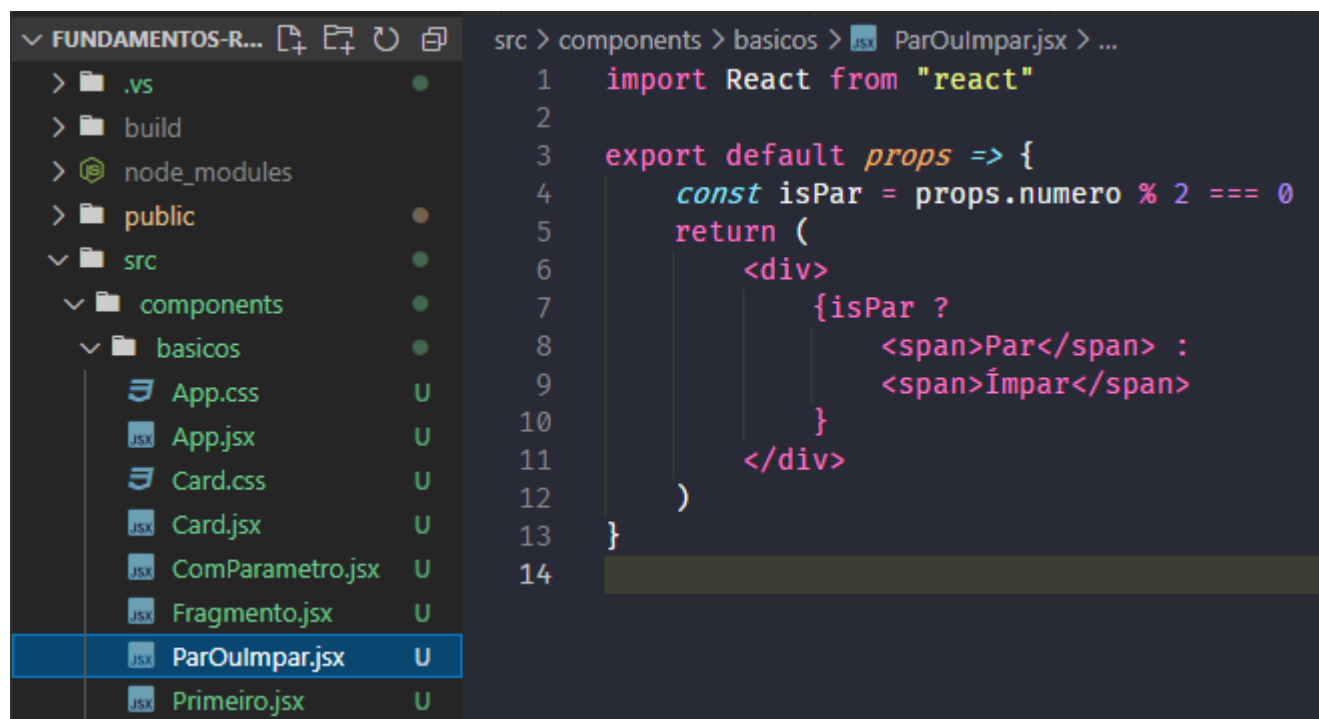
No App.jsx vamos referenciar essa aplicação.

Começaremos com o número 20.



```
src > components > basics > App.jsx > default
1  import './App.css'
2  import React from "react"
3
4  import ParOuImpar from './ParOuImpar'
5  import Card from './Card'
6  import Primeiro from './Primeiro'
7  import ComParametro from './ComParametro'
8  import Fragmento from './Fragmento'
9
10 export default () => (
11   <div className="App">
12
13     <h1>Fundamentos React</h1>
14
15     <div className="Cards">
16       <Card titulo = "Renderização condicional">
17         <ParOuImpar numero={20}></ParOuImpar>
18       </Card>
19     </div>
20   </div>
21 )
```

A constante `isPar` armazena o resultado da divisão como sendo `true` ou `false` e a condicional ternária apresenta Par ou Ímpar.



```
src > components > basicos > ParOuImpar.jsx > ...
1  import React from "react"
2
3  export default props => {
4      const isPar = props.numero % 2 === 0
5      return (
6          <div>
7              {isPar ?
8                  <span>Par</span> :
9                  <span>Ímpar</span>
10             }
11          </div>
12      )
13  }
14
```

