

Imersão React da Alura

Boas vindas ao eBook de exercícios e apoio de estudo da **#ImersãoReact** da [Alura](#). Criado e revisado por:

- [Joviane Jardim](#)
- [Marco Bruno](#)

Tire suas dúvidas ao vivo às **18:33** na twitch.tv/marcobrunodev nos dias 27, 28, 29, 30, 31 de Julho e pelo Discord da **#ImersãoReact**, entre nele no link a seguir:

- [Discord](#)

Capítulos

1. [Como funciona esse eBook?](#)
2. [Antes de começar](#)
3. [React, component, properties e deploy](#)

Como funciona esse eBook?

Esse eBook é um material de apoio das 5 aulas gratuitas da **#ImersãoReact** feita pela **Alura**, com as professoras:

- [Ju Negreiros](#)
- [Mario Souto](#)
- [Marco Bruno](#)

Nele tem todos os exercícios feitos durante as aulas, além dos desafios no final de cada aula. Cada capítulo é um dia de aula, com um ou mais exercícios.

No início de cada exercício tem um *spoiler* (revelação do que acontecerá) em seguida duas seções:

- **Tarefas;**
- **Passo a passo.**

Depois de todos exercícios para finalizar a aula, tem mais duas seções:

- **Desafios;**
- **Estude mais.**

A seguir explico com funciona cada uma das seções que acabei de citar.

Tarefas

Com o conteúdo apresentado na aula do dia, você já consegue fazer o exercício com a descrição apresentada na seção **Tarefas**, mas caso você tenha qualquer dificuldade poderá assistir a aula novamente ou seguir para próxima seção, ela te ajudará.

Passo a passo

Aqui tem uma explicação similar a feita em aula bem completa e com toda a informação detalhada que você precisa para executar o exercício. Lembre-se que se tiver qualquer dúvida durante a **#ImersãoReact** não fique com ela, compartilhe com a comunidade no Discord:

- [Discord](#)

Desafios

Essa é a parte mais importante da aula, é onde você validará se absorveu o conhecimento e poderá acompanhar sua evolução.

Estude mais

É uma seção feliz com conteúdos da comunidade sobre os assuntos apresentados na aula, para você estudar ainda mais sobre os temas abordados. Essa curadoria fizemos com carinho e felicidade.

Antes de começar

O dia zero não existe, mas aqui mostrarei o que você precisa fazer antes de começar a assistir as aulas.

Exercício 00 - Preparando ambiente

Nesse exercício instalaremos tudo que precisamos para começar nossos estudos da biblioteca **React** e todo o seu ecosistema. Usaremos um editor de texto conhecido como *VSCode* (*Visual Studio Code* - Estúdio visual de código), o *Firefox Developer* ou *Chrome* por serem navegadores modernos e muito utilizados pelas pessoas desenvolvedoras, o sistema de versionamento chamado *Git*, além do *Node* e o *npm*. Uma parte feliz é que todas essas ferramentas funcionam nos 3 SOs (Sistemas Operacionais) mais usados no mercado: *Windows*, *Linux* e *MacOS*.

Tarefas

Entre nos sites a seguir e faça a instalação do *VSCode*, *Firefox Developer* ou *Chrome*, *Git*, *Node* e *npm* (*Node package manager* - Gerenciador de pacotes do Node). Sobre o *Node* instale a versão recomendada, a famosa versão *LTS* (*long-term support* - suporte de longo prazo):

1. <https://code.visualstudio.com>
2. <https://www.mozilla.org/en-US/firefox/developer> ou <https://www.google.com/chrome>
3. <https://git-scm.com/book/pt-br/v2/Come%CA7ando-Instalando-o-Git>
4. <https://nodejs.org/pt-br>

Passo a passo

1. Instale o VSCode

Entre no site <https://code.visualstudio.com> faça o download (baixar um arquivo) para o SO que você está usando e em seguida instale-o, se tiver qualquer problema com a instalação não deixe de pedir ajuda no Discord da **#ImersãoReact**:

- [Discord](#)

2. Instale o Firefox Developer ou Chrome

Acesse o site <https://www.mozilla.org/en-US/firefox/developer> ou <https://www.google.com/chrome> e realize o download para o SO que você está usando, em seguida faça a instalação, no mesmo site terá todas as instruções para você realizar a instalação.

3. Instale Git

Acesso o eBook do Git no site <https://git-scm.com/book/pt-br/v2/Come%CA7ando-Instalando-o-Git> que lá terá todo processo de download e instalação para o SO que você estiver usando.

4. Instale Node e npm

Entre no site <https://nodejs.org/pt-br> faça o download e instale a versão recomendada que é famosa versão *LTS*.

Pronto! Agora que tudo está instalado você já pode partir para a primeira aula da **#ImersãoReact**. Se teve qualquer problema com o processo de instalação, por favor não deixe de pedir ajuda no Discord da **#ImersãoReact**. Ah! Se conseguiu passe lá para ajudar as pessoas que tiveram algum problema, é assim que construímos uma comunidade feliz e com muita troca de conhecimento!

React, component, properties e deploy

O dia zero não existe, mas aqui mostrarei o que você precisa fazer antes de começar a assistir as aulas.

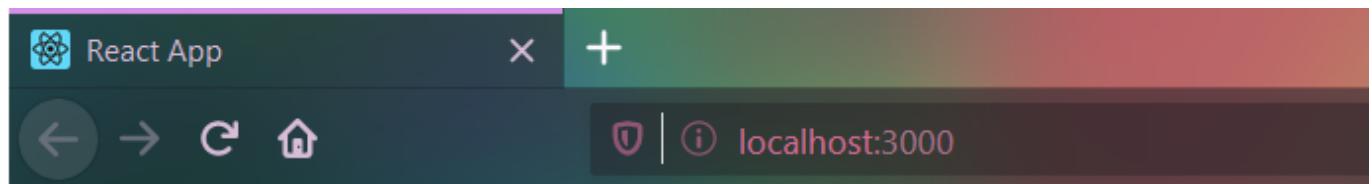
Exercício 01 - Inicie o projeto

Para criar a estrutura inicial do projeto usaremos o `create-react-app` que é o *starter-kit* (kit de ferramentas para iniciar projetos) mais usado no mercado de trabalho. Ele não foi o primeiro a aparecer, mas de longe é o mais usado, além de que foi criado pelo *Facebook* que continua mantendo o projeto atualizado, dessa forma temos tranquilidade e confiança para usar o `create-react-app` em nossos projetos.

Tarefas

Faça as 4 tarefas a seguir:

1. Crie o projeto com o nome de **aluraflix** usando o `create-react-app` sem ter que instalá-lo de forma global na sua máquina;
2. Remova os arquivos que não usaremos inicialmente: `App.test.js`, `setupTests.js`, `logo.svg`, `App.css` e `serviceWorker.js`;
3. Remova tudo sobre o `serviceWorker.js` que está no arquivo `index.js`;
4. Altere o arquivo `App.js` para que tenha apenas uma função retornando uma `div` que contenha a `className` com o valor `App`, além do conteúdo da `div` com o texto `Vamos começar`;
5. Com o projeto iniciado você terá algo similar em seu navegador:



Hello world

Passo a passo

1 - Abra o terminal (tela preta da NASA) do seu SO no qual você tem acesso ao **Node** e **npm** que instalou anteriormente e rode o comando a seguir, para iniciar o projeto **aluraflix** com o famoso **create-react-app**:

```
npx create-react-app aluraflix
```

Você receberá umas das duas saídas no terminal após o **create-react-app** finalizar a inicialização do projeto **AluraFlix**.

Saída similar a seguir caso não tenha o **yarn** instalado:

```
Success! Created aluraflix at C:\Users\marco\github\aluraflix-apostila\projeto\aluraflix
Inside that directory, you can run several commands:
```

```
npm start
  Starts the development server.
```

```
npm run build
  Bundles the app into static files for production.
```

```
npm test
  Starts the test runner.
```

```
npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!
```

We suggest that you begin by typing:

```
cd aluraflix
npm start
```

Happy hacking!

Se o `yarn` estiver instalado, a saída será similar a esta:

```
Success! Created aluraflix at C:\Users\marco\github\aluraflix-apostila\projeto\aluraflix
Inside that directory, you can run several commands:
```

```
yarn start
  Starts the development server.
```

```
yarn build
  Bundles the app into static files for production.
```

```
yarn test
  Starts the test runner.
```

```
yarn eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!
```

We suggest that you begin by typing:

```
cd aluraflix
yarn start
```

Happy hacking!

`npx` é uma ferramenta disponível no `npm` que você pode usar toda vez que precisar executar um pacote de forma temporária e global. Simples e feliz de usar o `npx`

O `create-react-app` tem muitas outras funcionalidades, como por exemplo criar um template utilizando *TypeScript* (linguagem de programação criado pela Microsoft). Recomendo que, em algum momento, você invista algumas horas lendo a documentação no link a seguir: <https://create-react-app.dev/docs/getting-started>

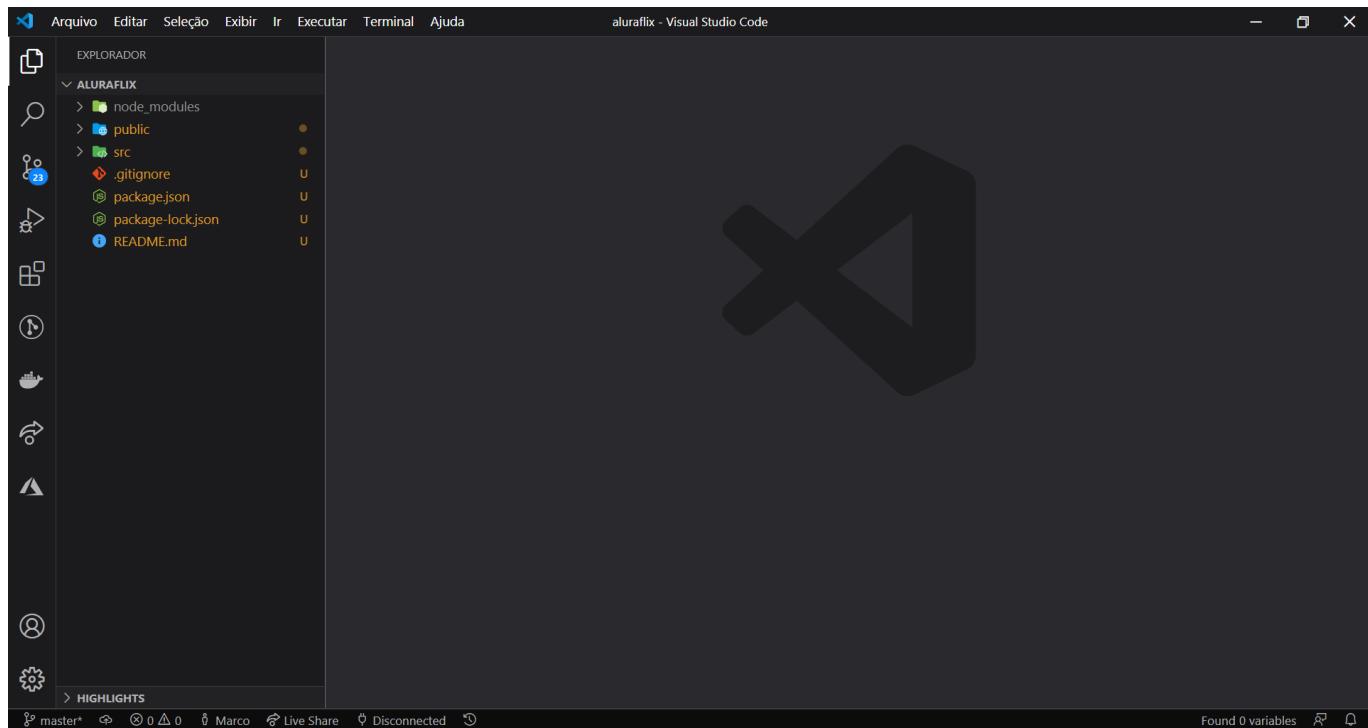
2 - Vamos rodar o projeto que acabamos de criar. O `create-react-app` criou a pasta do projeto chamada **aluraflix**, entre nela ainda pelo terminal com o comando a seguir:

```
cd aluraflix
```

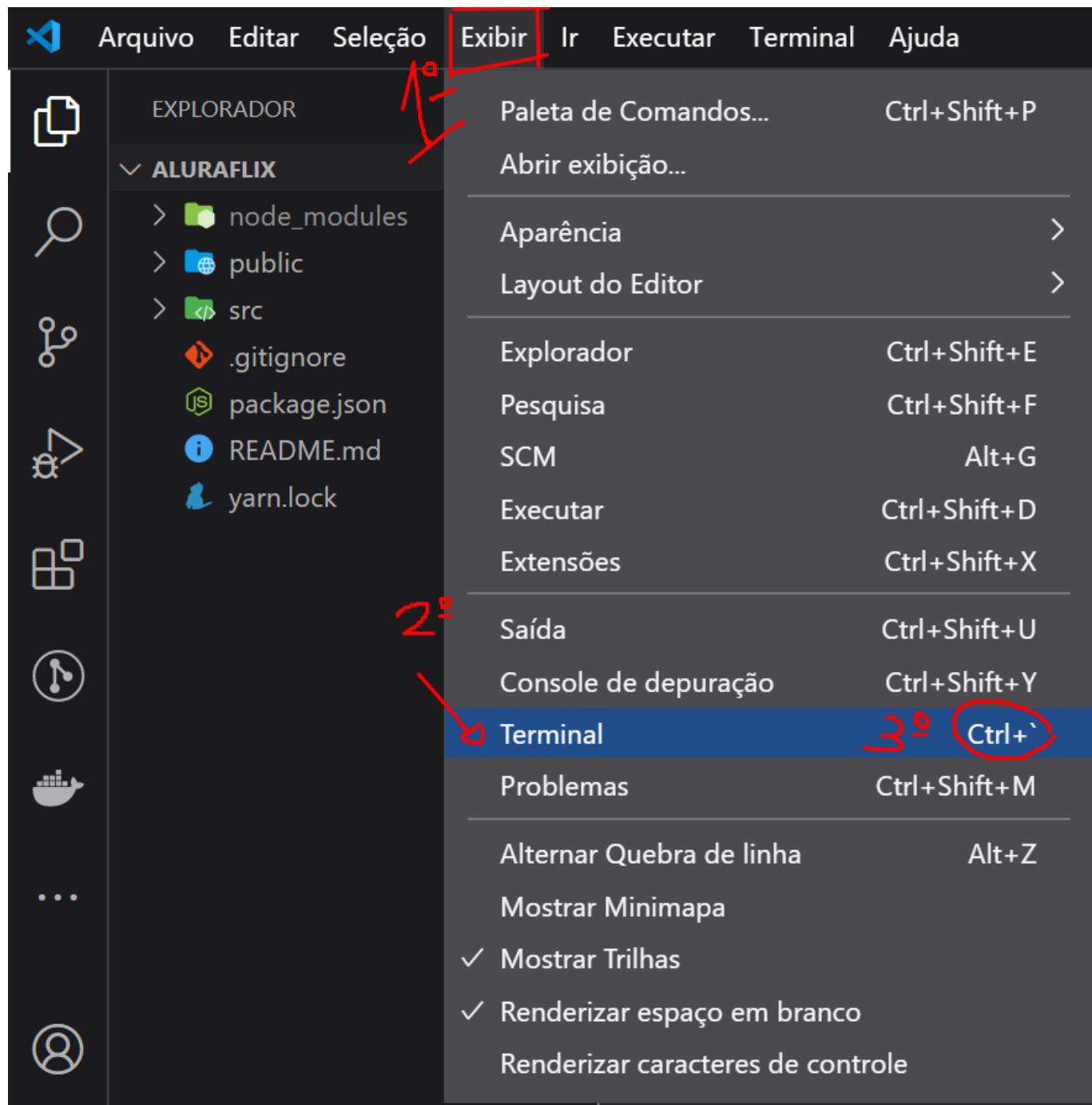
Agora, abra o projeto no seu *VSCode* com o comando a seguir, isso no seu terminal:

```
code .
```

Automaticamente o *VSCode* será aberto e com a pasta do projeto **aluraflix** aberta, e você verá algo similar:



Dentro do seu VSCode clique no menu **Exibir** que está localizado no topo e em seguida na opção **Terminal**.
Outra opção é usar o atalho pelo teclado: **ctrl + `** (control + crase).



Com o terminal aberto como na imagem a seguir:

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with icons for files, folders, and symbols. The 'ALURAFLIX' folder is expanded, showing 'node_modules', 'public', 'src', '.gitignore', 'package.json', 'README.md', and 'yarn.lock'. Below the sidebar is a terminal window titled '1: powershell' containing the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

O carregamento de perfis pessoais e do sistema levou 1270ms.
→ aluraflix git:(master)
```

Vamos executar o comando `npm start` que subirá o nosso projeto e magicamente abrirá o navegador que você tiver definido como padrão em seu sistema operacional. A seguir tem quatro imagens mostrando todo processo:

This screenshot shows the terminal window again, with the command `npm start` being typed at the prompt. The output from the previous screenshot is still visible at the top.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

O carregamento de perfis pessoais e do sistema levou 769ms.
→ aluraflix git:(master) npm start
```

This screenshot shows the terminal window after the command has been executed. The message 'Starting the development server...' is displayed in blue text, indicating the process is underway.

```
Starting the development server...
```

PROBLEMAS SAÍDA TERMINAL CONSOLE DO DEPURADOR 1: node

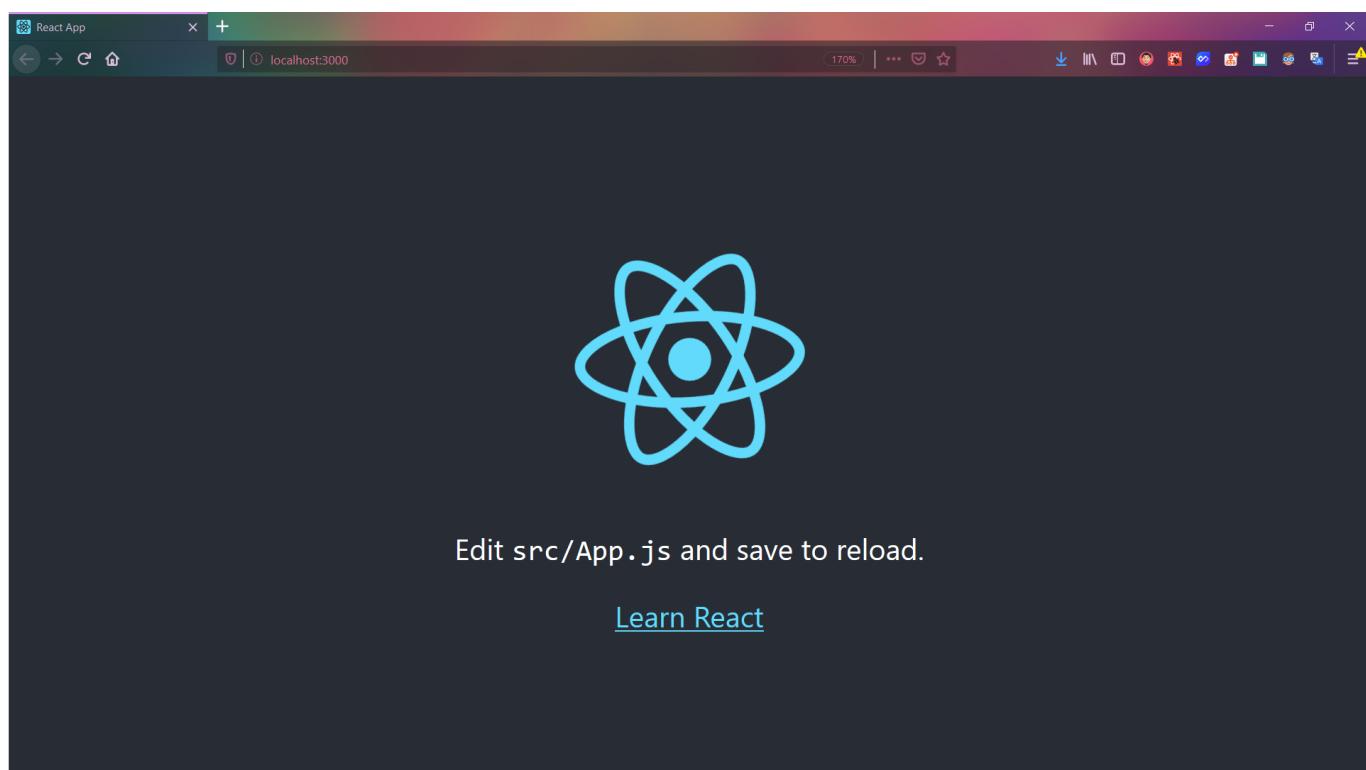
Compiled successfully!

You can now view **aluraflix** in the browser.

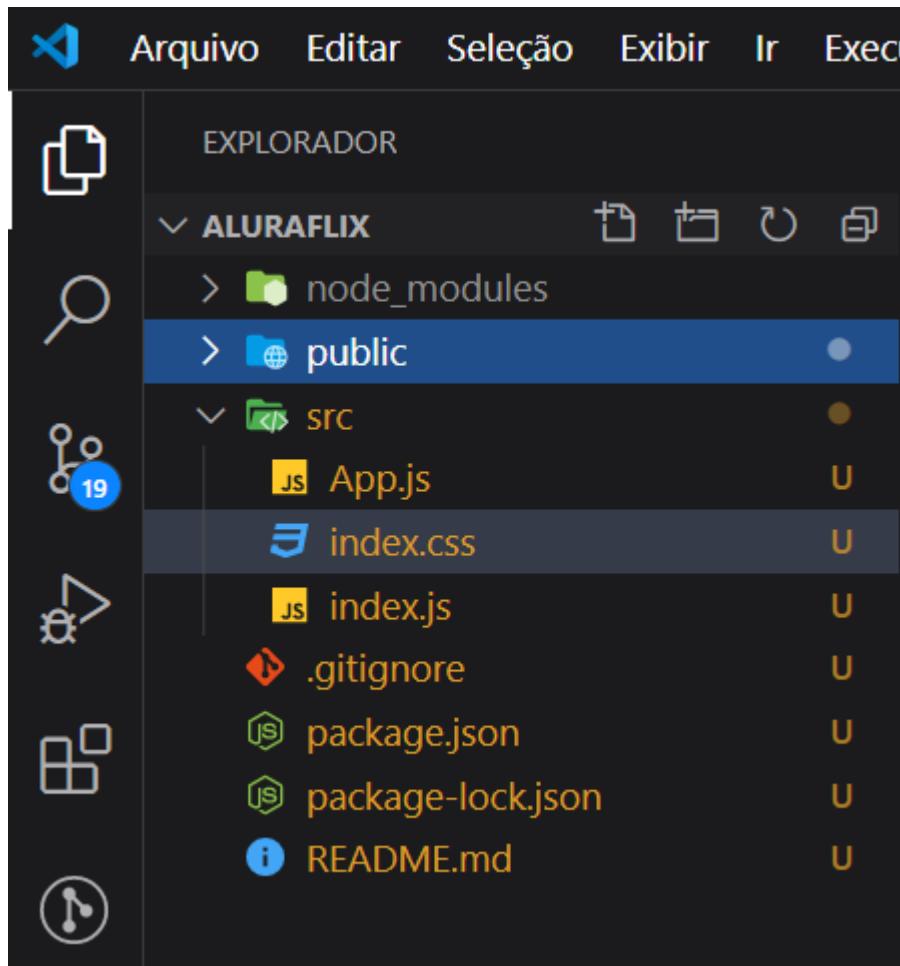
Local: http://localhost:**3000**
On Your Network: http://192.168.0.4:**3000**

Note that the development build is not optimized.
To create a production build, use [yarn build](#).

□



3 - Deletaremos algumas coisas que não precisamos para esse projeto. Dentro da pasta `src` remova os arquivos: `App.test.js`, `setupTests.js`, `logo.svg`, `App.css` e `serviceWorker.js`, seu *Explorer* (Explorador ou árvore de arquivos) do *VSCode*, ficará assim:



Se o seu terminal do VSCode estiver aberto você verá o seguinte erro:

```
PROBLEMAS SAÍDA TERMINAL CONSOLE DO DEPURADOR 1: node + □ □ ^ ×
Failed to compile.

./src/serviceWorker.js
Error: ENOENT: no such file or directory, open 'C:\Users\marco\github\alura\aluraflix\src\serviceWorker.js'
```

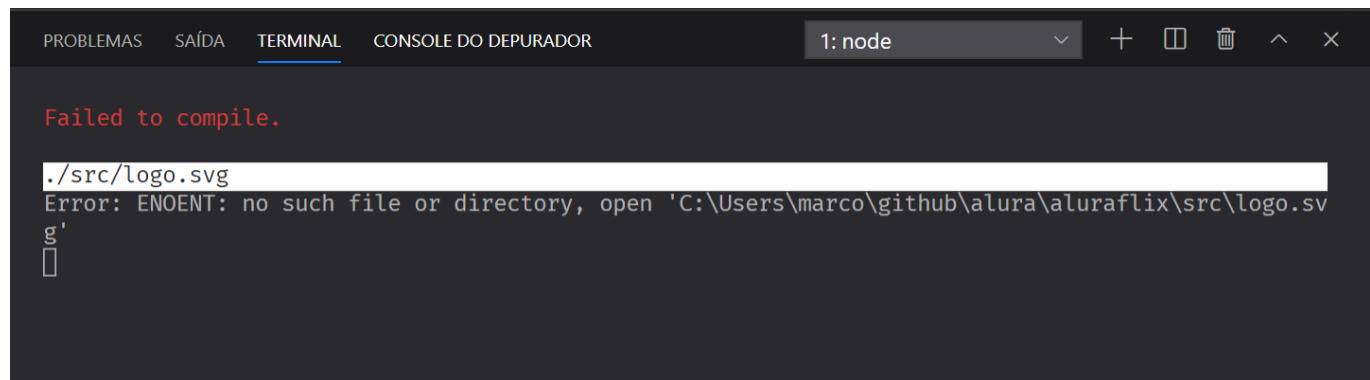
The screenshot shows the VSCode interface with the 'TERMINAL' tab active. The terminal window displays an error message: 'Failed to compile.' followed by the command 'node' and the path './src/serviceWorker.js'. Below this, an 'Error: ENOENT: no such file or directory, open 'C:\Users\marco\github\alura\aluraflix\src\serviceWorker.js'' is shown, indicating that the file was not found. The terminal window has a dark background with light-colored text.

Para o erro sumir, abra o arquivo `index.js` e remova tudo que estiver relacionado ao `serviceWorker.js`, o código ficará assim:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Infelizmente no seu terminal aparecerá um novo erro:



The screenshot shows the VSCode interface with the 'TERMINAL' tab selected. The terminal window displays the following output:

```
PROBLEMAS SAÍDA TERMINAL CONSOLE DO DEPURADOR 1: node + □ □ ^ ×
Failed to compile.

./src/logo.svg
Error: ENOENT: no such file or directory, open 'C:\Users\marco\github\alura\aluraflix\src\logo.svg'
```

Para resolver o erro do logo e outros futuros, dentro do arquivo `App.js` precisamos remover o `import` dos arquivos `logo.svg` e `App.css`, além de deixar o conteúdo da `<div className="App">` mais simples, apenas com o texto `Hello world`. Ah! aproveite e remova a `className="App"` da tag `div`:

```
import React from 'react';

function App() {
  return (
    <div>
      Hello world
    </div>
  );
}

export default App;
```

No seu terminal do VSCode você terá a seguinte saída:

Compiled successfully!

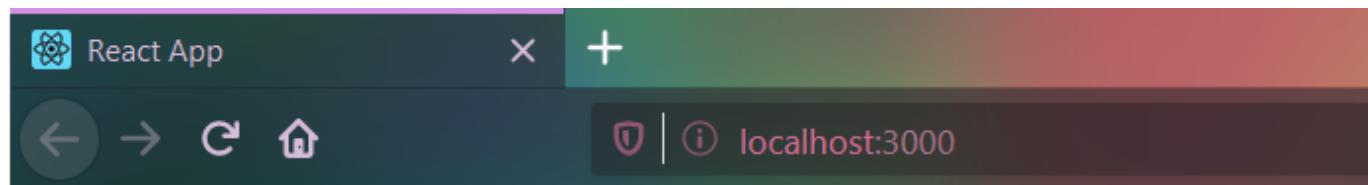
You can now view aluraflix in the browser.

Local: <http://localhost:3000>

On Your Network: <http://192.168.0.4:3000>

Note that the development build is not optimized.
To create a production build, use `npm run build`.

Toda vez que você salva um arquivo do seu projeto, o navegador se atualiza automaticamente (chamamos esse processo de `hot reload`). Volte no seu navegador e você verá isso:

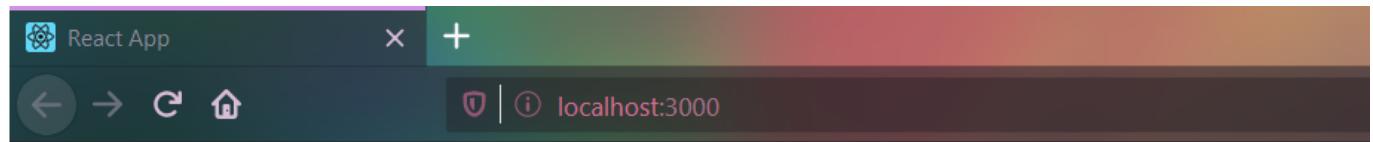


Hello world

Exercício 02 - Reset CSS e variáveis do CSS

Todo projeto *FrontEnd* tem que ter um *Reset CSS* que nos ajuda a garantir que o *layout* do nosso site funcionará igual em todos os navegadores. Além disso veremos uma forma feliz de deixar a paleta de cores do nosso site em variáveis no CSS.

O resultado final nesse exercício será assim:



Hello world

Tarefas

Faça os items a seguir tudo dentro do arquivo `index.css`.

1. Crie um seletor que pega todos os elementos da tela e adicione nele o `box-sizing: border-box;` e a `font-family: 'Roboto', sans-serif;`
2. Crie oito variáveis no seu CSS com as cores a seguir: `#2A7AE4, #000000, #9E9E9E, #F5F5F5, #e5e5e5, #FFFFFF, #6BD1FF e #00C86F;`
3. Zere a `margin` e `padding` do `html` e `body`;
4. Faie para todas tags `a` herdar a cor do elemento pai;

Dentro do arquivo `index.html`:

5. Adicione o link para a *font* Roboto do *Google fonts*.

Passo a passo

1 - Abra o arquivo `index.css` que está dentro da pasta `src` e apague todo código. Crie o seletor que pega todos os elementos da página e adicione duas propriedades conforme está a seguir:

```
* {  
  box-sizing: border-box;  
  font-family: 'Roboto', sans-serif;  
}
```

2 - A seguir crie o pseudo-seletor `:root` e dentro dele crie 8 variaveis que guardarão as cores que utilizaremos no site:

```
:root {  
  --primary: #2A7AE4;  
  --black: #000000;  
  --blackLighter: #9E9E9E;  
  --grayLight: #F5F5F5;  
  --grayMedium: #e5e5e5;  
  --white: #FFFFFF;  
  --frontEnd: #6BD1FF;  
  --backEnd: #00C86F;  
}
```

Para saber mais sobre variaveis no CSS ou *custom properties* (propriedades customizadas) acesse a documentação na MDN: https://developer.mozilla.org/en-US/docs/Web/CSS/-*

3 - Crie um seletor para a tag `html` e `body` e aplique `margin` e `padding` com o valor `0`:

```
html,  
body {  
  margin: 0;  
  padding: 0;  
}
```

4 - Adicione um seletor da tag `a` e aplique a propriedade `color` com o valor `inherit`:

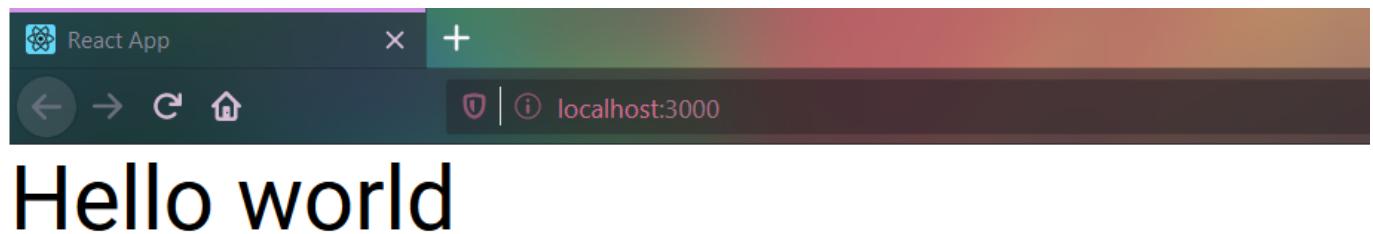
```
a {  
  color: inherit;  
}
```

5 - Para a `font-family: Roboto, sans-serif` funcionar, abra o arquivo `index.html` que está dentro da pasta `public` e adicione dentro do tag `<head>` a tag `<link>` que importará a *font* Roboto pelo *Google fonts*:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- muitos códigos aqui -->
    <link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap" rel="stylesheet">
    <!-- muitos códigos aqui -->
  </head>
  <body>
    <!-- alguns códigos aqui -->
    <div id="root"></root>
    <!-- alguns códigos aqui -->
  </body>
</html>
```

Importante: Não delete nada do arquivo `index.html`, apenas adicione a tag `<link>` apontando para *font* Roboto do *Google fonts* dentro do `<head>`.

Pronto! Agora se você voltar no navegador verá que a alteração visual é pequena mas muito importante para mantermos um layout mais próximo em navegadores diferentes.

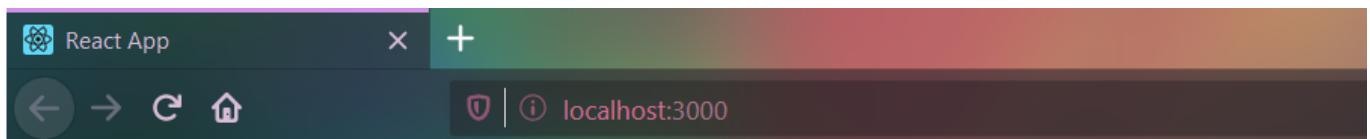


Exercício 3 - Primeiro component

React é uma *lib* (biblioteca) de componentização. Chegou a hora de vermos uma das coisas mais legais que ela tem. Bora criar nosso primeiro component que será o <Menu>.

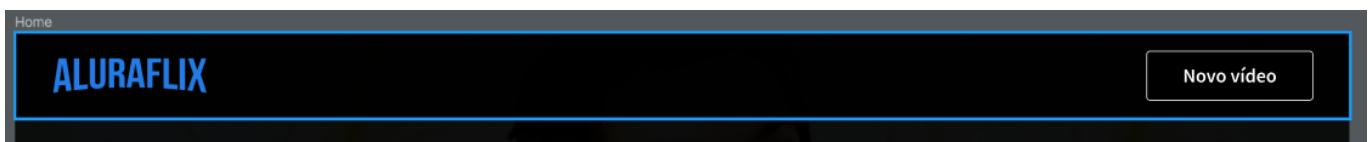
Tarefas

1. Dentro da pasta `src` crie um pasta chamada `components` (componentes) e dentro de outras pasta com o nome de `Menu` e dentro desta pasta um arquivo `index.js`;
2. Dentro do arquivo `index.js` crie um *component* que terá o conteúdo da imagem a seguir:



LogoNovovídeo

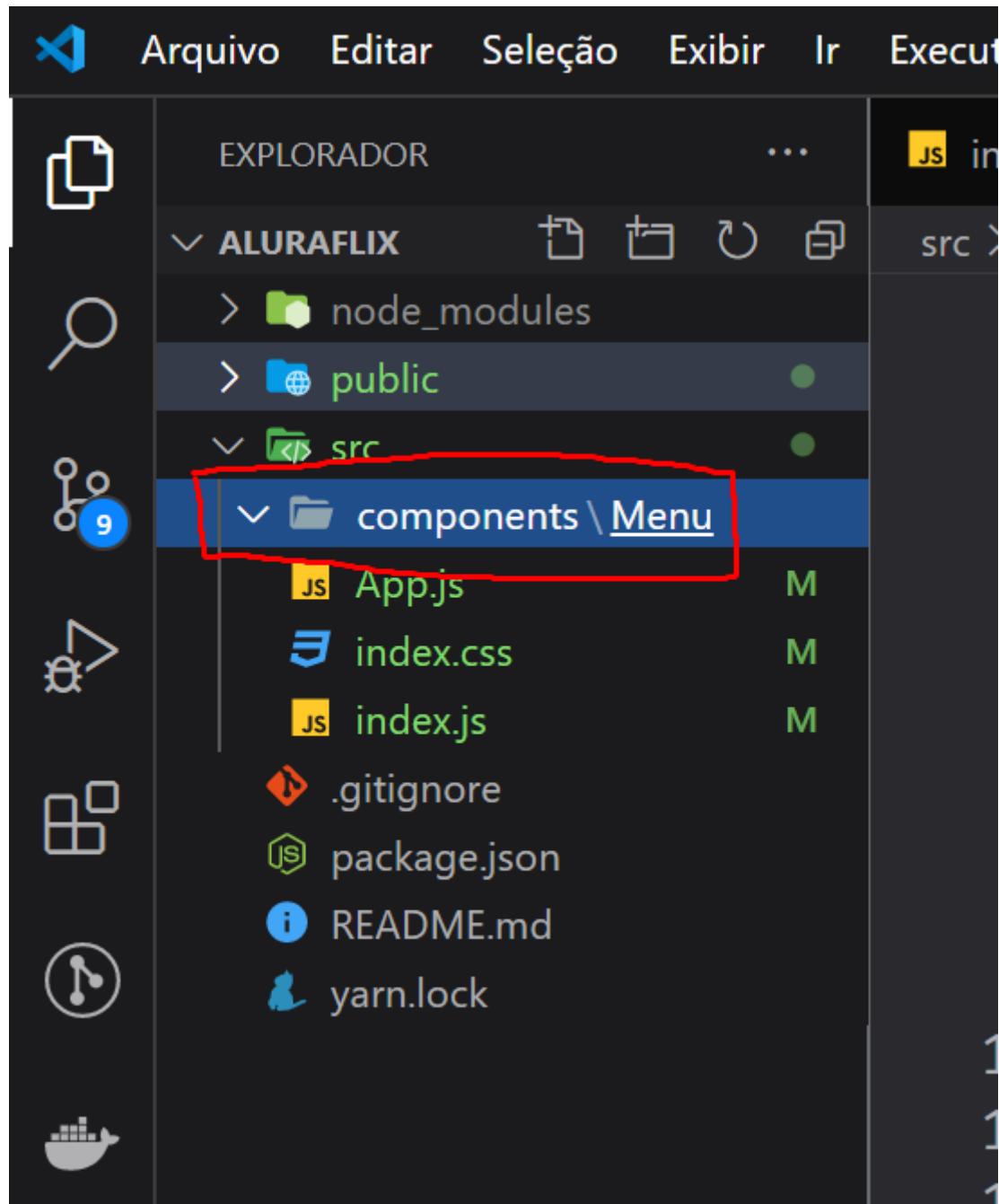
3. Dentro da pasta `Menu` crie um arquivo chamado `Menu.css` que terá todo o CSS para deixar o *component* com esse comportamento visual:



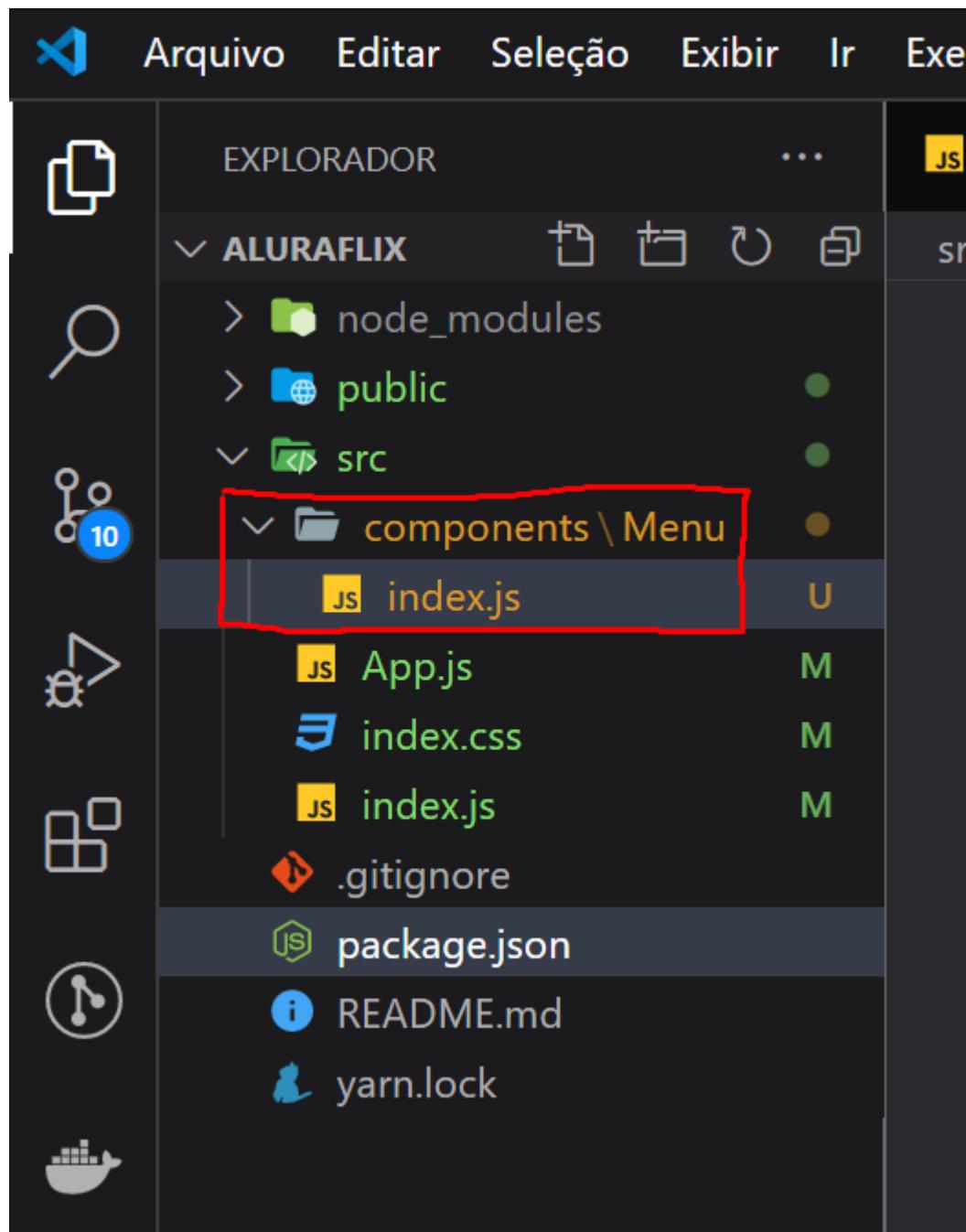
4. Abra o arquivo `App.js` e utilize o *component* <Menu>.

Passo a passo

1 - Criaremos a estrutura de pastas que cuidará dos nossos *components*. Dentro da pasta `src` crie uma pasta chamada `components` e dentro dela a pasta `Menu` que é o nosso primeiro *component*:



Dentro da pasta **Menu** adicione um arquivo chamado **index.js**:



2 - Dentro do arquivo `index.js` criaremos o código necessário para um *component* com *React*, só o CSS do *component* `Menu` ficará em outro arquivo.

Na primeira linha importaremos o *React*, na linha seguinte criaremos uma função com o nome de `Menu` e na última linha faremos o *export* (exportar) da função `Menu` para ser usada em outro arquivo:

```
import React from 'react'

function Menu() {

}

export default Menu
```

Isso ainda não é uma *component* válida porque dentro da função precisamos retornar um *HTML (HyperText Markup Language - Linguagem de Marcação de Hipertexto)*, na verdade um *JSX (JavaScript XML)*. Adicione o `return` (retorno) dentro da função `Menu` que retornará uma tag `<nav>` com dois *links*, representados pela tag `<a>`:

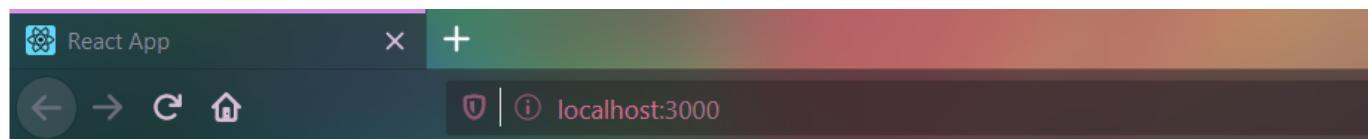
```
import React from 'react'

function Menu() {
  return (
    <nav className="Menu">
      <a href="/">
        Logo
      </a>
      <a className="ButtonLink" href="/">Novo vídeo</a>
    </nav>
  )
}

export default Menu
```

Todo *component* (componente) em *React* damos o nome com a primeira letra maiúscula

Se for até o navegador não verá nada de diferente porque não usamos o *component* `<Menu>`:



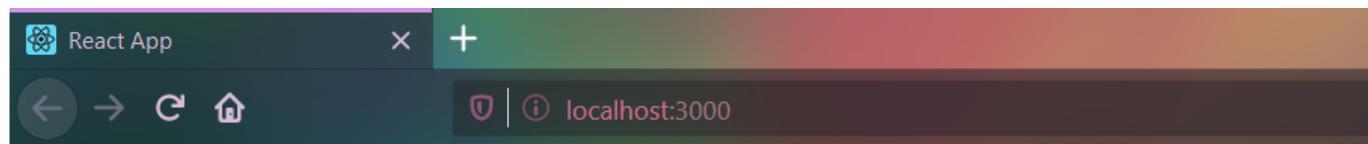
Abra o arquivo `App.js` e na linha 2 faça o `import` do componente `Menu`, depois use o `<Menu>` dentro do `return` da função `App` e apague o texto `Hello World`:

```
import React from 'react';
import Menu from './components/Menu';

function App() {
  return (
    <div>
      <Menu />
    </div>
  );
}

export default App;
```

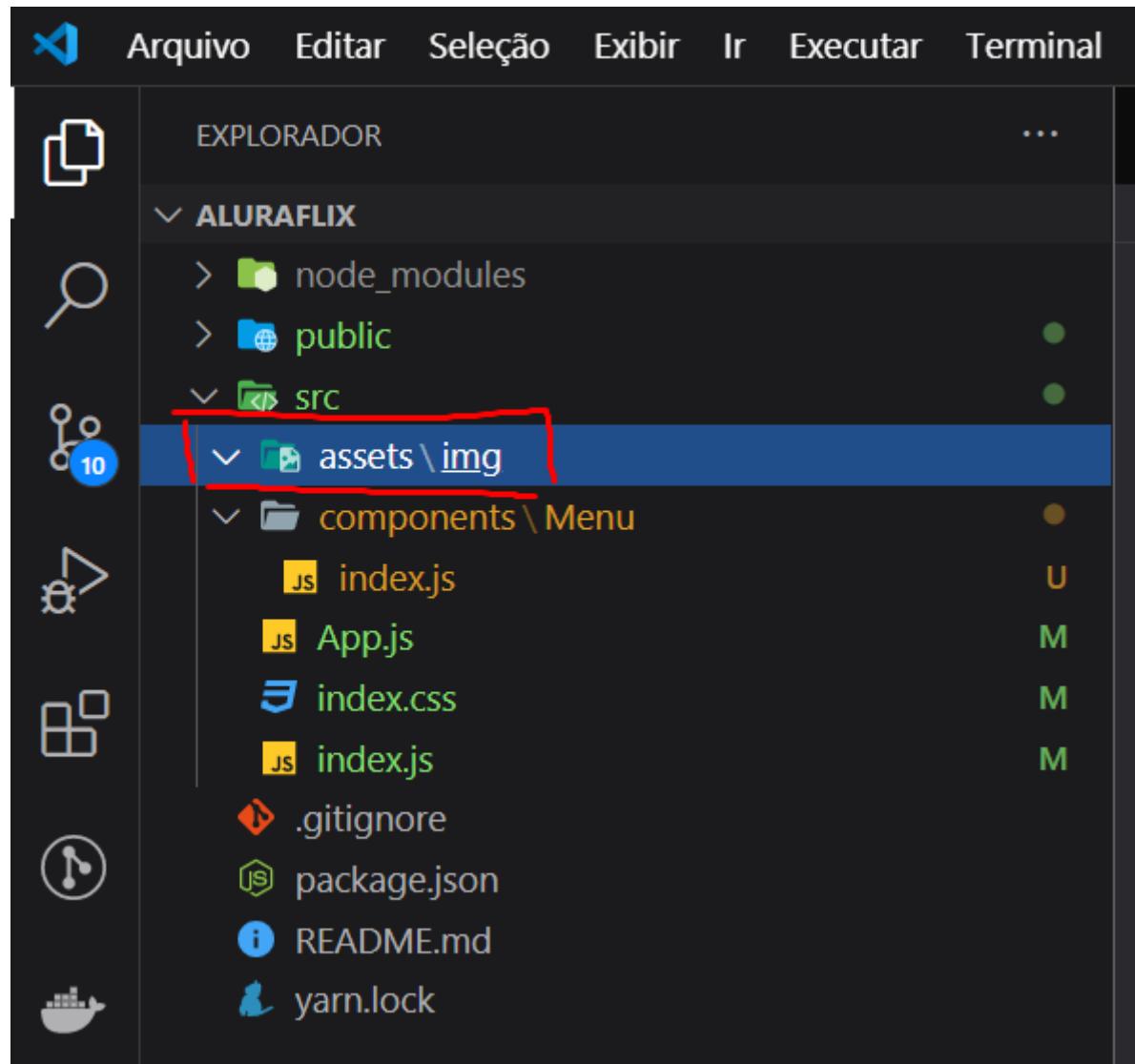
Vá até o navegador e você terá o seguinte resultado:



LogoNovo vídeo

Trocaremos o texto `Logo` da nossa primeira tag `<a>` por uma imagem do logo da **AluraFlix** que está no layout do *Figma* (programa para criar design de interface).

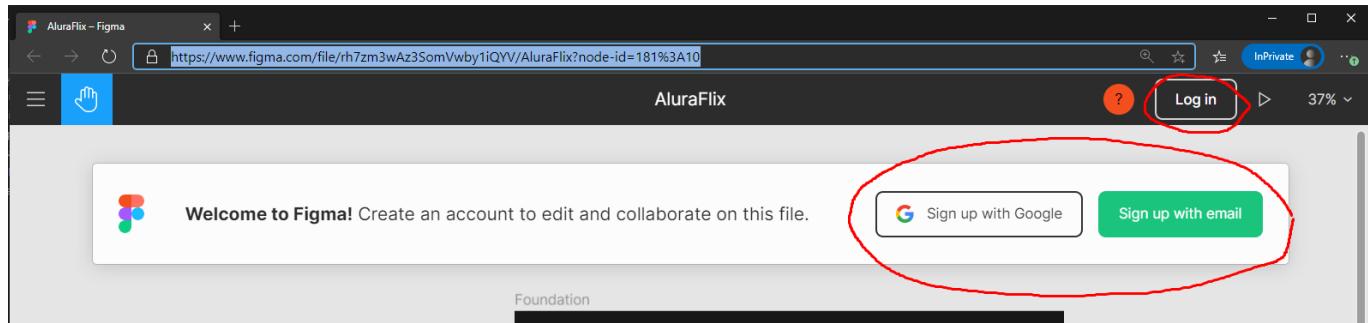
Dentro da pasta `src` crie uma pasta chamada `assets`. Dentro dessa nova pasta crie uma pasta chamada `img`, onde colocaremos todas as imagens do projeto:



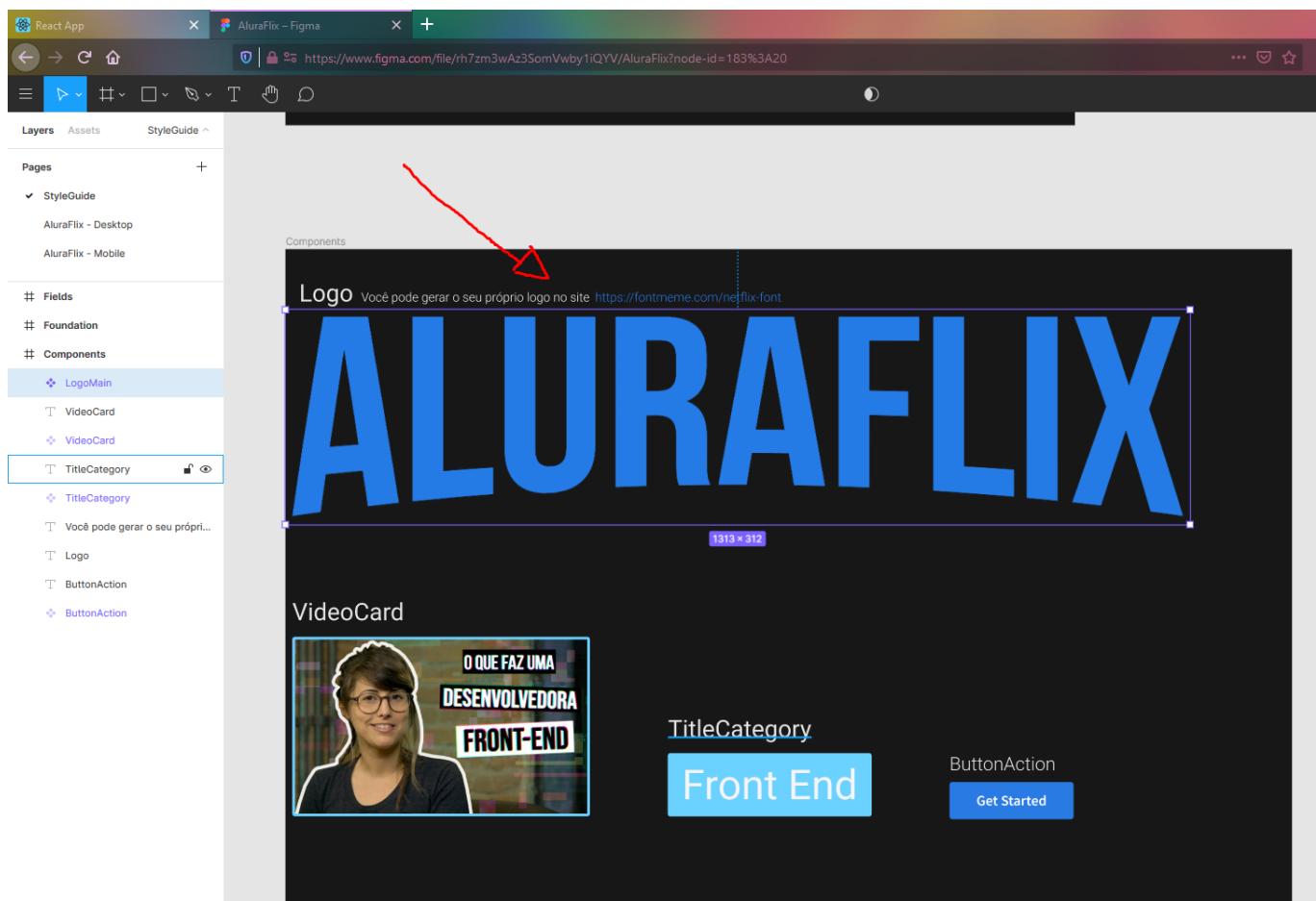
Abra o layout do projeto AluraFlix no link a seguir:

<https://www.figma.com/file/rh7zm3wAz3SomVwby1iQYV/AluraFlix?node-id=181%3A10>

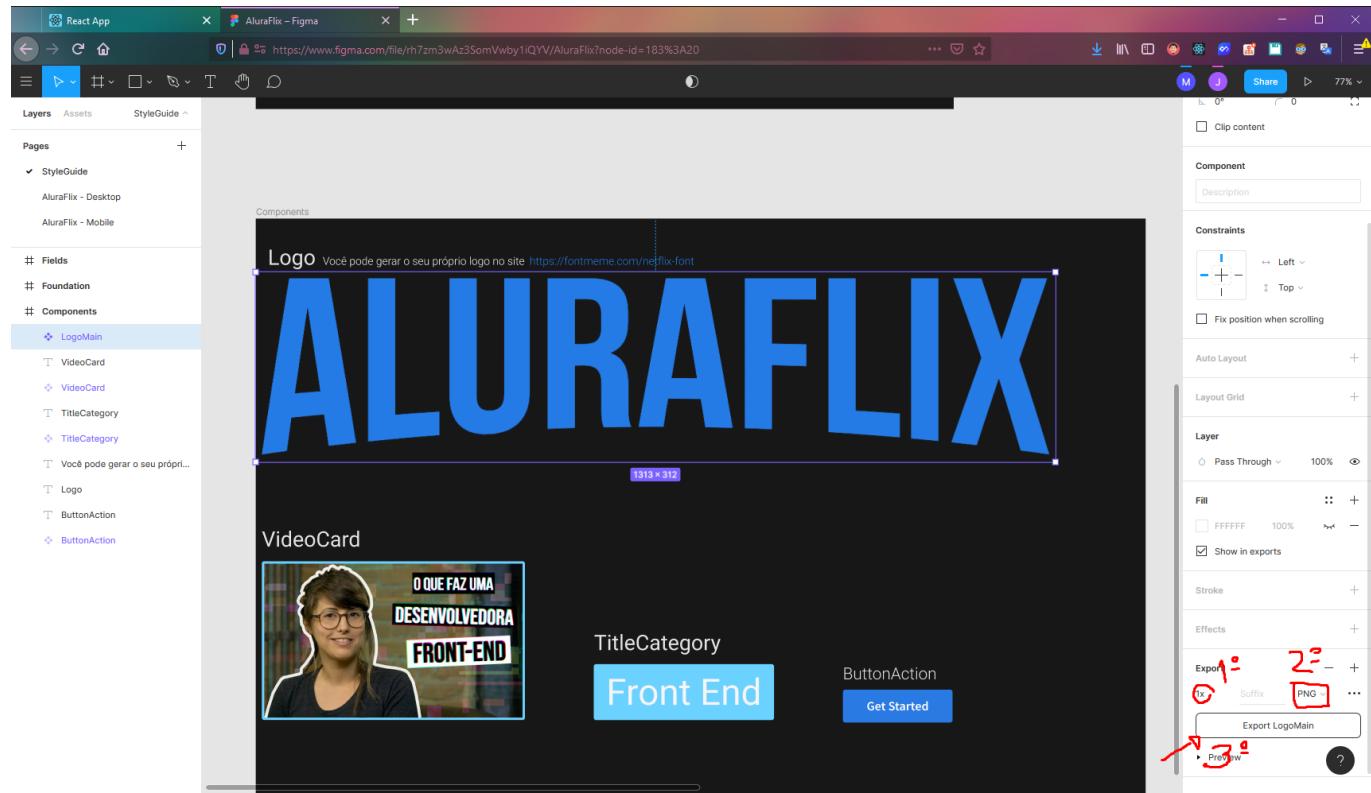
Só clicando no link você já consegue ver o layout do projeto, mas para fazer o download do logo é necessário realizar um cadastro, ou login caso já tenha uma conta no Figma:



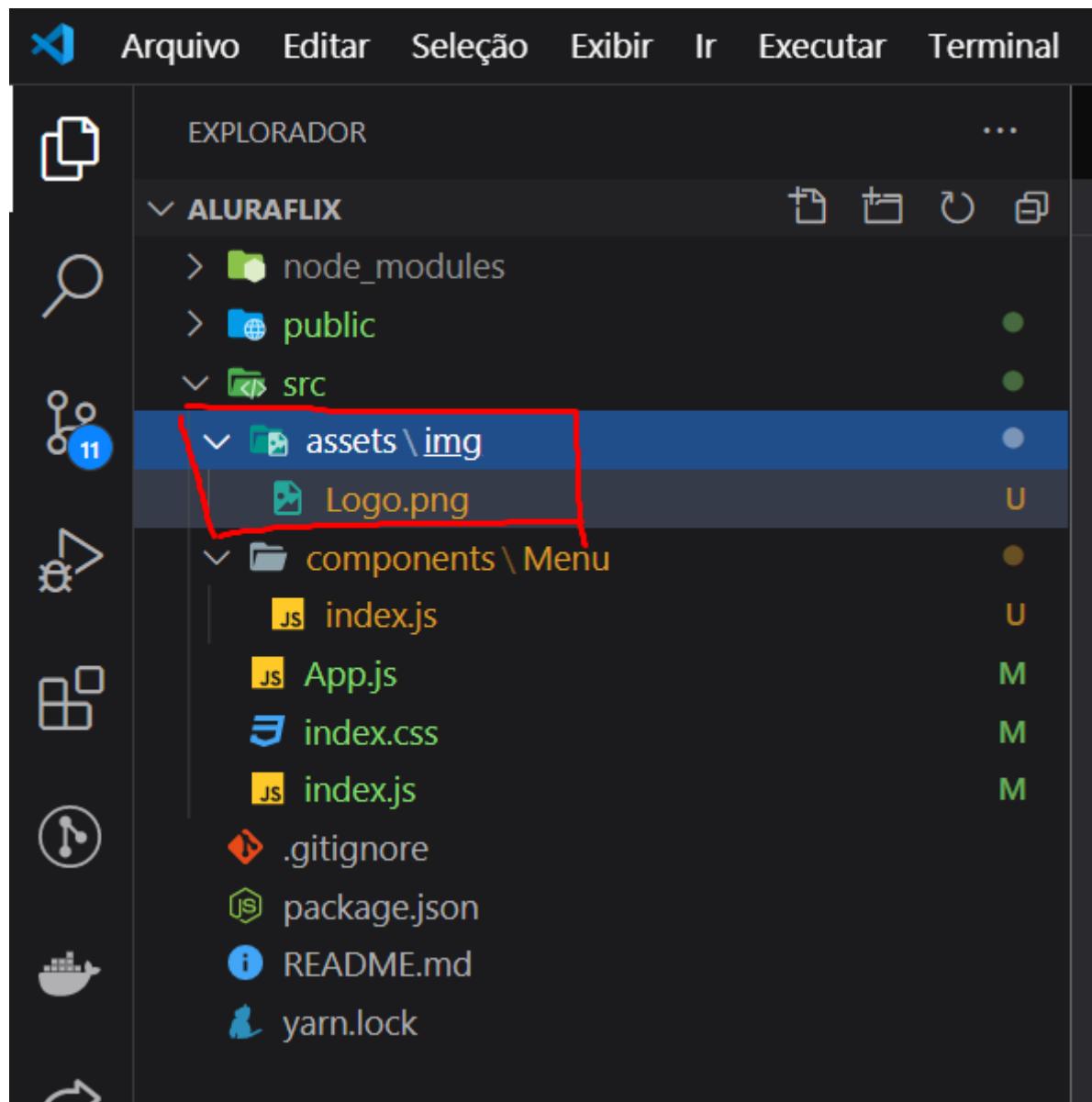
Agora que você fez seu cadastro ou login, clique sobre o logo **AluraFlix**:



Depois, na parte inferior direita do *Figma* faça o *download* clicando no botão *Export LogoMain*:



Renomeie o arquivo do logo para **Logo.png** e mova-o para a pasta **img**:



Abra o arquivo do componente `<Menu>` que está no caminho, `src/component/Menu/index.js`, e na linha 2 faça o `import` do arquivo `Logo.png`:

```
import React from 'react'
import Logo from '../../../../../assets/img/Logo.png'

function Menu() {
  return (
    <nav className="Menu">
      <a href="/">
        Logo
      </a>
      <a className="ButtonLink" href="/">Novo vídeo</a>
    </nav>
  )
}

export default Menu
```

Agora troque o texto **Logo** pela tag ****:

```
import React from 'react'
import Logo from '../../assets/img/Logo.png'

function Menu() {
  return (
    <nav className="Menu">
      <a href="/">
        <img className="Logo" src={Logo} alt="Logo da AluraFlix" />
      </a>
      <a className="ButtonLink" href="/">Novo vídeo</a>
    </nav>
  )
}

export default Menu
```

Abra o seu navegador e você verá o seguinte resultado visual:



É, precisamos urgentemente aplicar um CSS no nosso menu, como o foco da apostila é *React* não vou explicar no detalhe sobre o CSS, mas você pode aparecer na live às **18:33** na twitch.tv/marcobrunodev que farei uma explicação feliz para você, aparece lá também se tiver dúvidas.

Dentro da pasta **src/components/Menu** crie um arquivo chamado **Menu.css** e coloque o seguinte CSS dentro dele:

```
.Logo {
  max-width: 168px;
```

```
}

@media (max-width: 800px) {
  .Logo {
    max-width: 105px;
  }
}

.Menu {
  width: 100%;
  height: 94px;
  z-index: 100;

  display: flex;
  justify-content: space-between;
  align-items: center;

  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  padding-left: 5%;
  padding-right: 5%;

  background: var(--black);
  border-bottom: 2px solid var(--primary);
}

body {
  --bodyPaddingTop: 94px;
  padding-top: var(--bodyPaddingTop);
}

@media (max-width: 800px) {
  .Menu {
    height: 40px;
    justify-content: center;
  }
  body {
    --bodyPaddingTop: 40px;
    padding-top: var(--bodyPaddingTop);
  }
}

.ButtonLink {
  color: var(--white);
  border: 1px solid var(--white);
  box-sizing: border-box;
  cursor: pointer;
  padding: 16px 24px;
  font-style: normal;
  font-weight: bold;
  font-size: 16px;
  outline: none;
  border-radius: 5px;
  text-decoration: none;
  display: inline-block;
```

```
    transition: opacity .3s;
}
(ButtonLink:hover,
.ButtonLink:focus {
  opacity: .5;
}

@media (max-width: 800px) {
  a.ButtonLink {
    position: fixed;
    left: 0;
    right: 0;
    bottom: 0;
    background: var(--primary);
    border-radius: 0;
    border: 0;
    text-align: center;
  }
}
```

Agora faça o `import` desse arquivo dentro do arquivo `src/components/Menu/index.js` na linha 3:

```
import React from 'react'
import Logo from '../../assets/img/Logo.png'
import './Menu.css'

function Menu() {
  return (
    <nav className="Menu">
      <a href="/">
        <img className="Logo" src={Logo} alt="Logo da AluraFlix" />
      </a>
      <a className="ButtonLink" href="/">Novo vídeo</a>
    </nav>
  )
}

export default Menu
```

No navegador você verá o seguinte resultado:

