



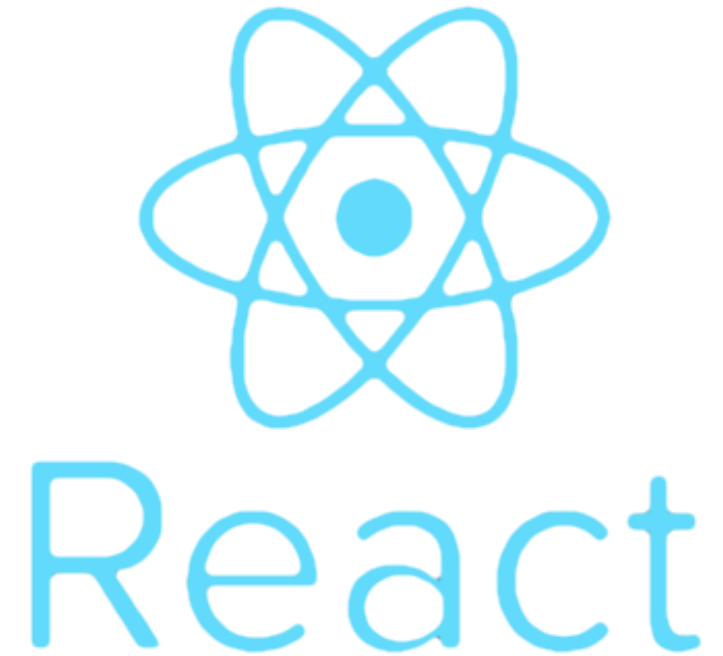
Desenvolvimento Front-End

React



O que é o React?

- Biblioteca JavaScript para construir interfaces de usuário.
- Desenvolvido pelo Facebook, com foco em performance e reuso de componentes.
- Popularidade: usado por grandes empresas como Instagram, Airbnb e Netflix.



Por que aprender React?

- Facilidade de manutenção e escalabilidade.
- Reutilização de componentes.
- Comunidade grande e ativa.



Configuração do Ambiente

Instalação do Node.js

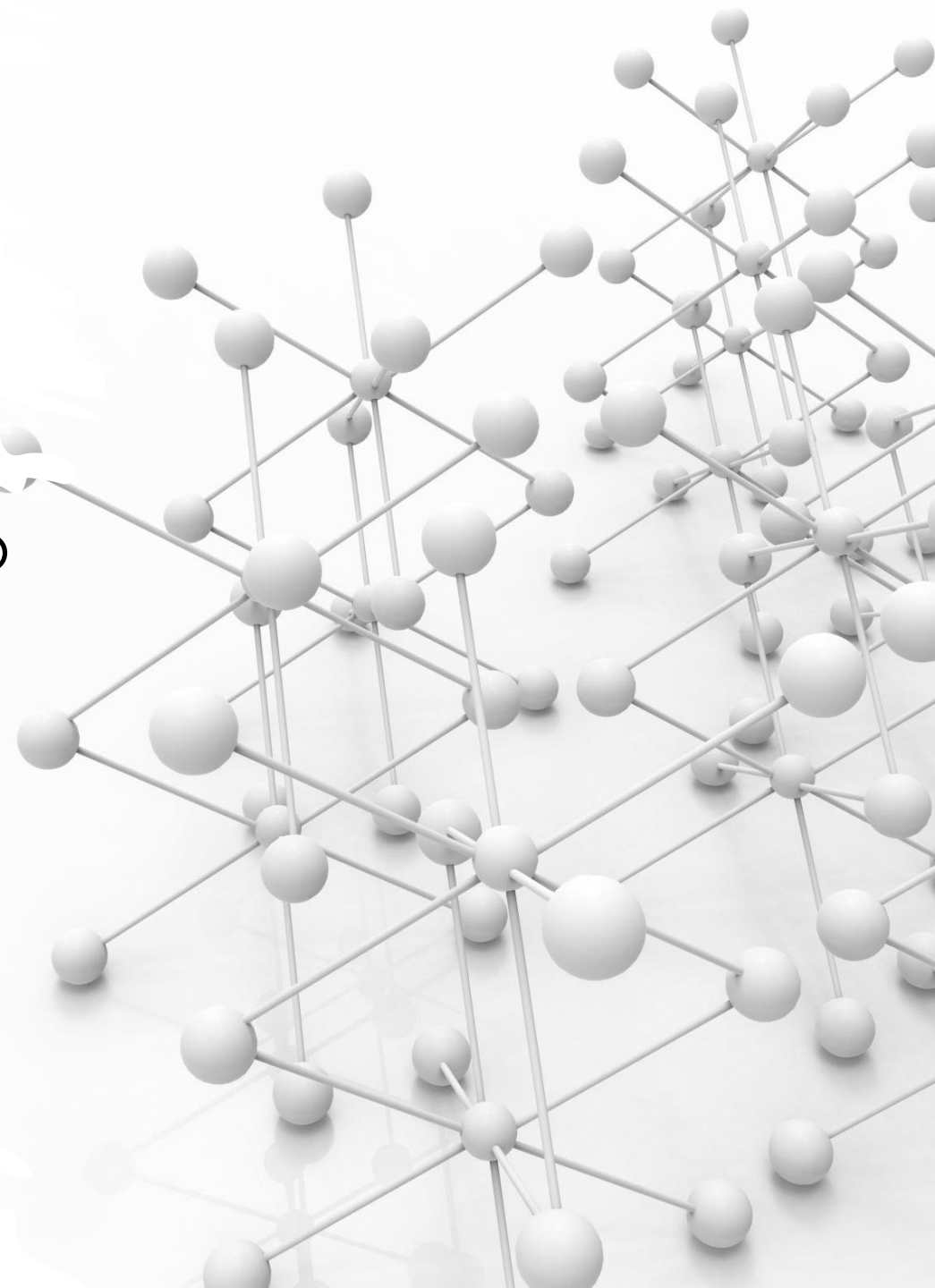
Dependência do Node.js e npm.

- O **React** é uma biblioteca JavaScript focada no desenvolvimento de interfaces de usuário
- O **Node.js** atua como o ambiente de execução necessário para rodar ferramentas de desenvolvimento no lado do servidor
- O **npm** (Node Package Manager) gerencia pacotes e dependências, incluindo o React e suas bibliotecas auxiliares.
- Com o **npm**, é possível:
 - Instalar e atualizar dependências.
 - Gerenciar bibliotecas de forma eficiente.
 - Executar scripts para automatizar tarefas como a inicialização de projetos.
 - Realizar o **build** para produção.

Criar um projeto React

`npx create-react-app nome_aplicação`

```
.Programas_React>npx create-react-app ex01
```



Criando o projeto React ex01

```
Creating a new React app in C:\Users\paulo\OneDrive\01-Documentos\02_SENAI\01_Aulas\01_Graduação\ADS\04_DFRONT\Programas_React\ex01.
```

```
Installing packages. This might take a couple of minutes.
```

```
Installing react, react-dom, and react-scripts with cra-template...
```

```
[██████████] | idealTree:ex01: timing idealTree:#root Completed in 3364ms
```

```
We suggest that you begin by typing:
```

```
cd ex01
npm start
```

```
Happy hacking!
```

```
npm notice
```

```
npm notice New minor version of npm available! 10.5.0 -> 10.8.3
```

```
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.3
```

```
npm notice Run npm install -g npm@10.8.3 to update!
```

```
npm notice
```

```
C:\Users\paulo\OneDrive\01-Documentos\02_SENAI\01_Aulas\01_Graduação\ADS\04_DFRONT\Programas_React>
```

```
Pasta de C:\Users\paulo\OneDrive\01-Documentos\02_SENAI\01_Aulas\01_Graduação\ADS\04_DFRONT\Programas_React
```

```
12/09/2024 14:18 <DIR> .
12/09/2024 14:21 <DIR> ..
12/09/2024 14:20 <DIR> ex01
                0 arquivo(s)                0 bytes
                3 pasta(s) 766.416.334.848 bytes disponíveis
```

Rodando o Projeto

npm start

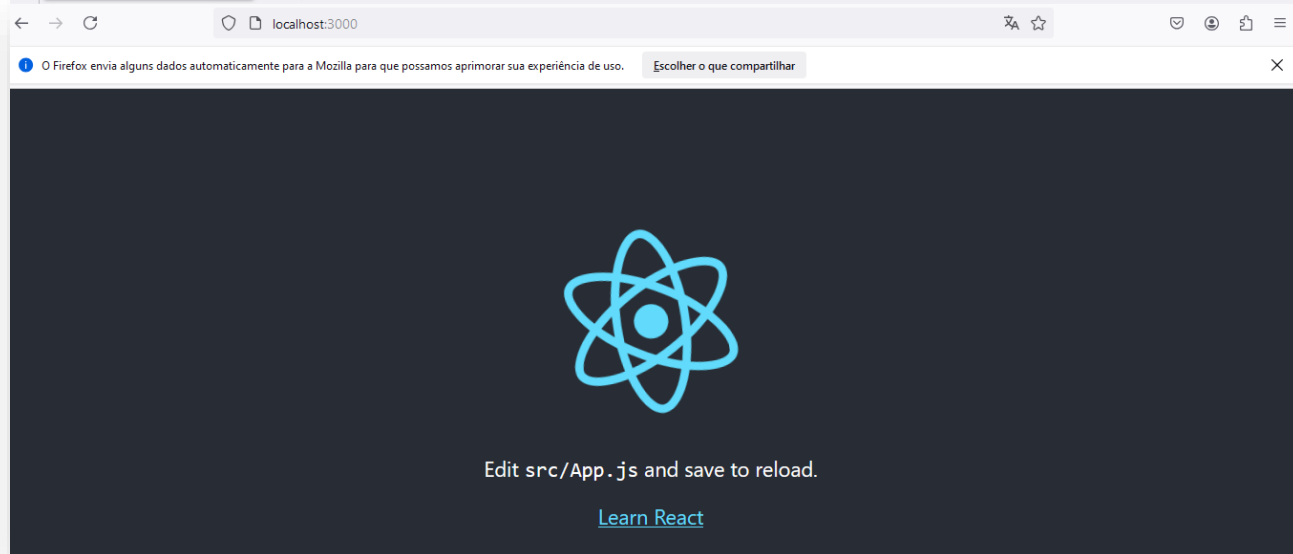
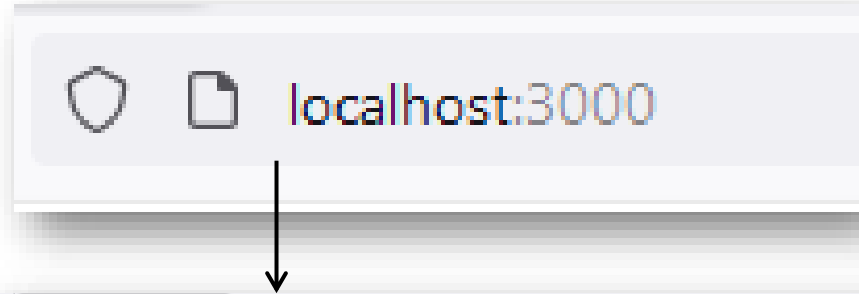
```
\Programas_React\ex01>npm start
```



Edit src/App.js and save to reload.

[Learn React](#)

Rodando o Projeto



```
Compiled successfully!
```

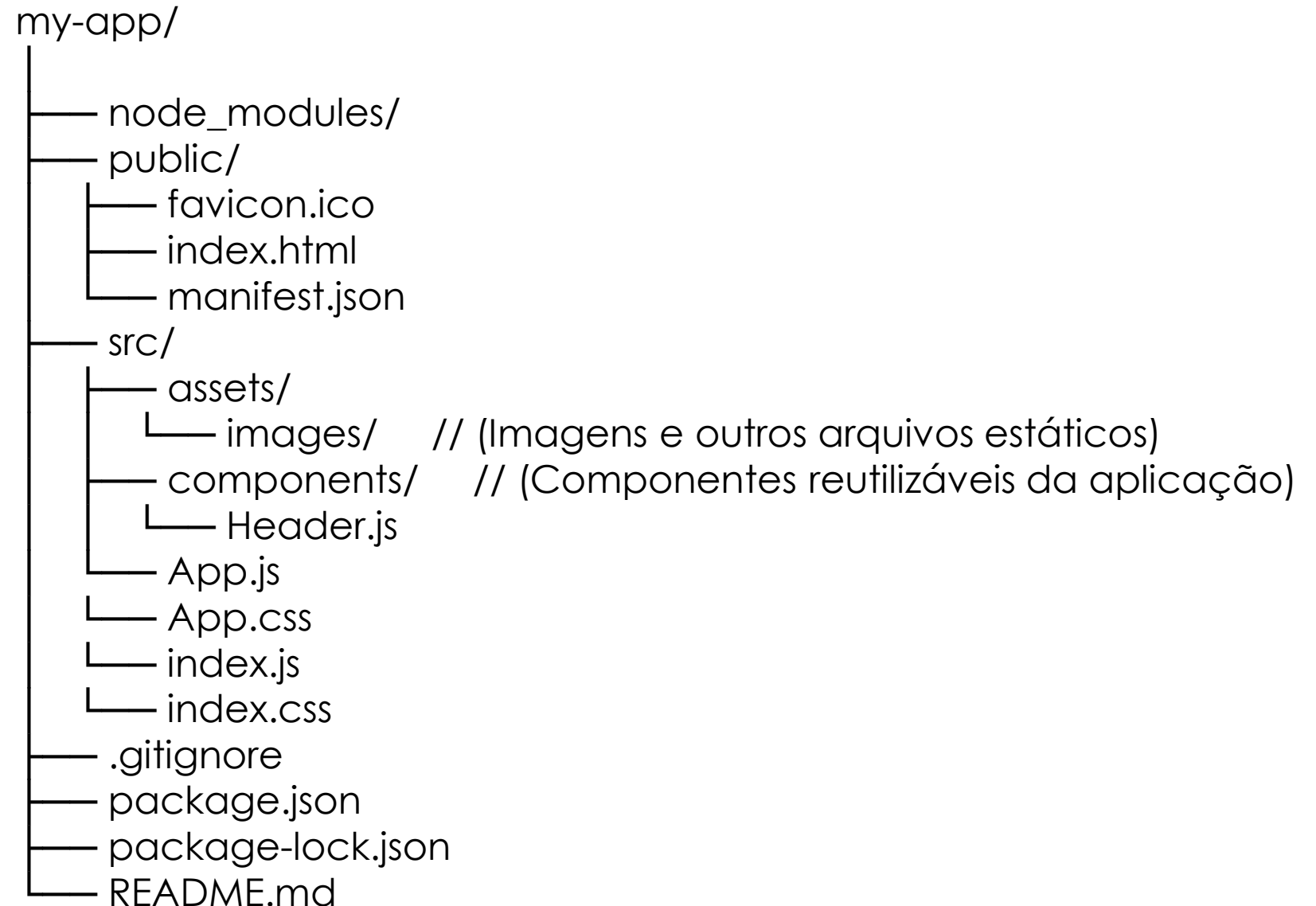
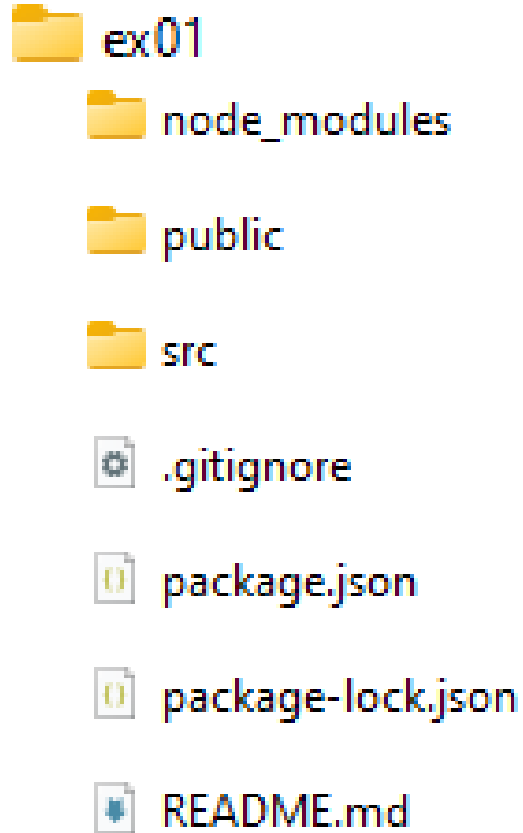
```
You can now view ex01 in the browser.
```

```
http://localhost:3000
```

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

```
webpack compiled successfully
```

Estrutura de Pastas



Estrutura de Pastas

node_modules/

Essa pasta é gerada automaticamente e contém todas as dependências instaladas do projeto. Não é alterada manualmente.

public/

Essa pasta contém arquivos estáticos, que não precisam ser processados pelo Webpack (ferramenta de empacotamento de módulos).

O principal arquivo aqui é o `index.html`, que serve como o único arquivo HTML para a aplicação React. Os componentes React são injetados neste arquivo.

- **index.html:** Este é o ponto de entrada HTML da aplicação. O React vai renderizar toda a interface dentro de uma `div` com o id "root".
- **favicon.ico:** Ícone da aba do navegador.
- **manifest.json:** Usado para configurar como a aplicação se comporta quando instalada como um PWA (Progressive Web App).

Estrutura de Pastas

src/

Essa é a pasta principal onde você irá escrever seu código React. Ela contém os componentes, arquivos CSS e outros arquivos essenciais.

- **App.js:** É o componente principal da aplicação, que normalmente contém a lógica principal e os outros componentes.
- **App.css:** O arquivo de estilos específico para o componente App.js.
- **index.js:** Arquivo que inicializa a aplicação React. Ele importa o App.js e renderiza dentro do HTML na div com id "root".
- **index.css:** Arquivo de estilos globais que pode ser aplicado em toda a aplicação.

Estrutura de Pastas

Subpastas dentro de src/:

- **components/**: É comum organizar os componentes reutilizáveis em uma pasta components/. Cada componente pode ter seu próprio arquivo .js (ou .jsx) e, opcionalmente, um arquivo .css para os estilos específicos desse componente.
- **assets/**: Diretório onde ficam os arquivos estáticos da aplicação, como imagens, fontes e outros recursos que possam ser necessários.

Estrutura de Pastas

package.json

Este arquivo lista as dependências da aplicação e scripts úteis para desenvolvimento, como start, build, e test.

package-lock.json

Este arquivo é gerado automaticamente quando as dependências são instaladas. Ele garante que, em diferentes máquinas, as mesmas versões exatas das dependências sejam usadas.

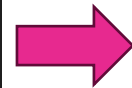
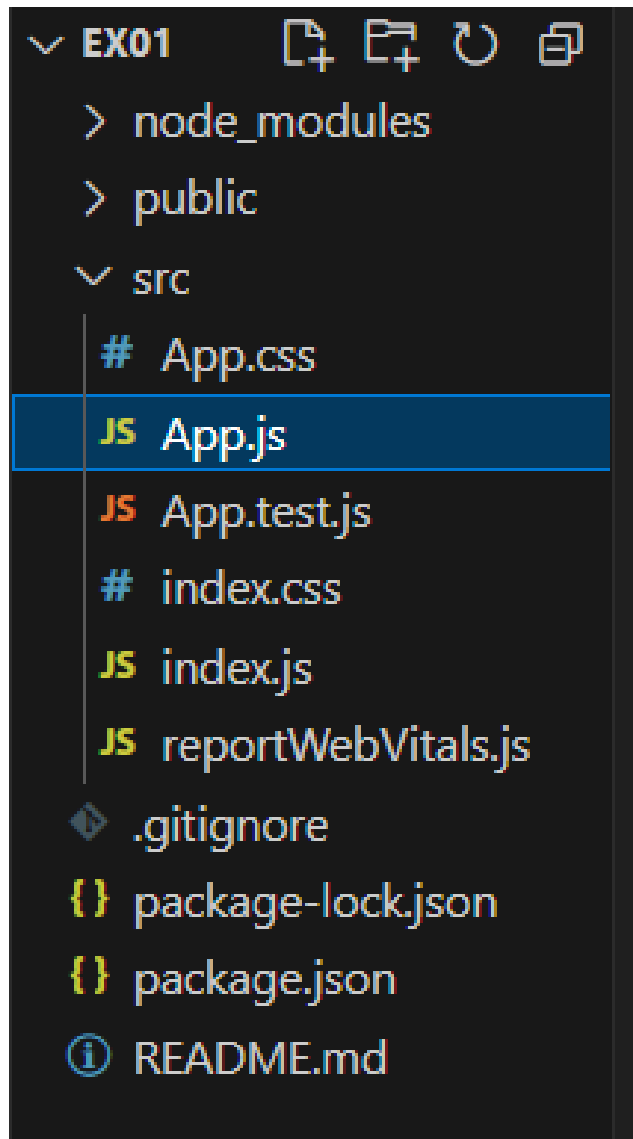
.gitignore

Lista de arquivos e pastas que o Git deve ignorar. Por exemplo, node_modules/ é ignorado, pois pode ser recriado com base no package.json.

README.md

É um arquivo de documentação que explica como configurar e rodar o projeto.

Estrutura de Pastas



Componente principal da aplicação, que normalmente contém a lógica principal e os outros componentes

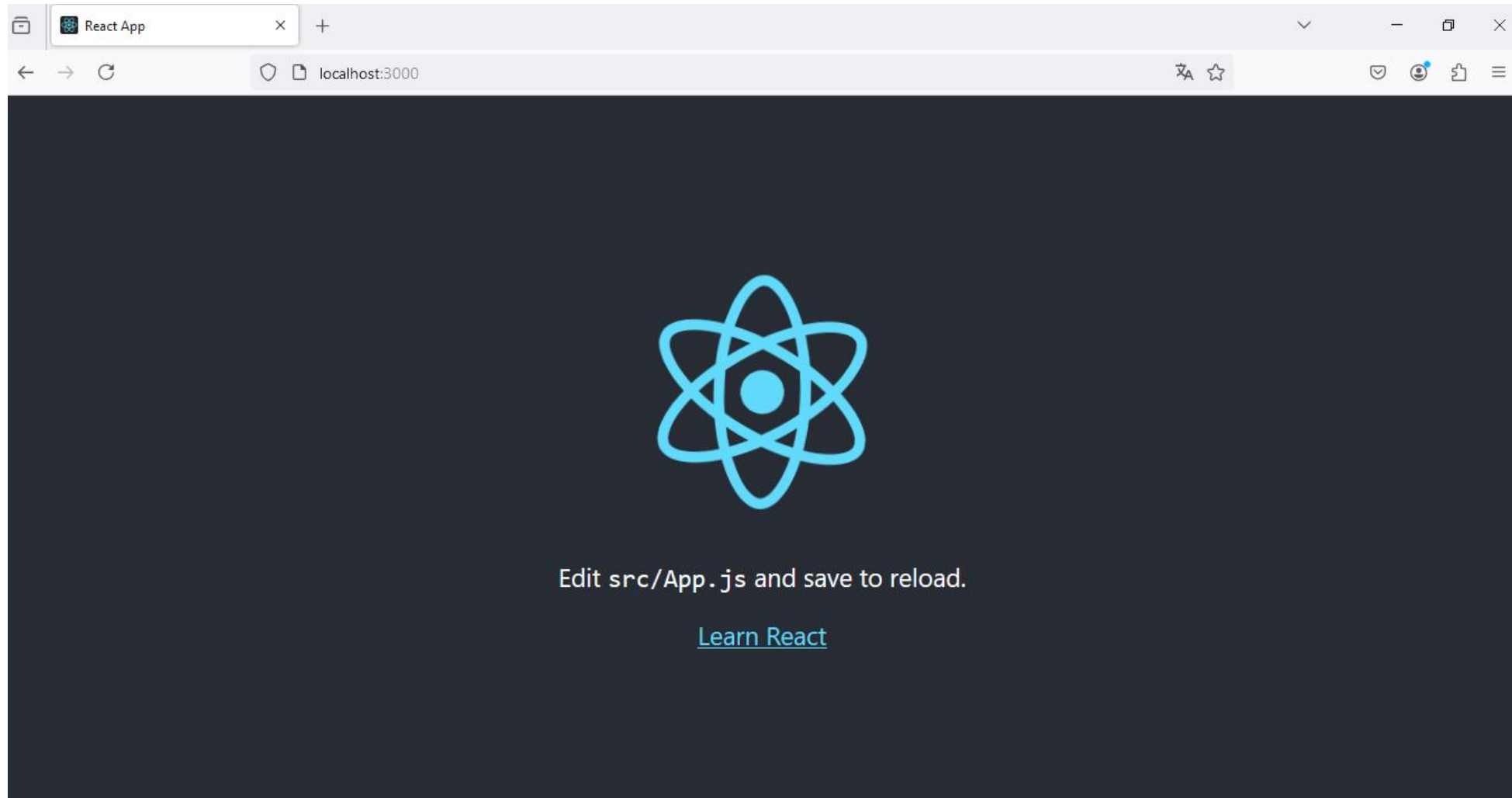
App.js

```
import logo from './logo.svg'; // Importa a imagem logo.svg da pasta local para ser usada no componente.
import './App.css';            // Importa o arquivo de estilos CSS que será aplicado ao componente App.

function App() { // Define o componente funcional "App", que é o principal da aplicação.
  return (      // O retorno da função contém o JSX, que é uma sintaxe similar a HTML para definir a interface do componente.
    <div className="App"> // Cria uma `div` com a classe "App". Esta classe será estilizada usando o CSS importado.
      <header className="App-header"> // Cria uma tag `header` com a classe "App-header"
        <img src={logo} className="App-logo" alt="logo" /> // Adiciona uma imagem que usa o logo importado.
        <p> // Adiciona um parágrafo `<p>` contendo uma instrução para o desenvolvedor.
          Edit <code>src/App.js</code> and save to reload. // A linha de instrução sugere editar o arquivo `src/App.js`
                                                         //e salvar para ver as mudanças na aplicação.
        </p>
        <a
          className="App-link" // Adiciona um link `<a>` com a classe "App-link" que será estilizada no CSS.
          href="https://reactjs.org" // Define a URL para onde o link leva. No caso, o site oficial do React.
          target="_blank" // Abre o link em uma nova aba do navegador.
          rel="noopener noreferrer" // Melhora a segurança do link externo ao prevenir ataques
                                   //relacionados à manipulação do objeto `window.opener`.
        >
          Learn React // Texto exibido para o link.
        </a>
      </header>
    </div>
  );
}

export default App; // Exporta o componente "App" para que ele possa ser usado em outros arquivos do projeto.
```

Página principal




Modificando o projeto ex01

Substituir o conteúdo dentro da tag `<div className="App">` no arquivo `App.js` para criar sua própria interface.

```
import './App.css';

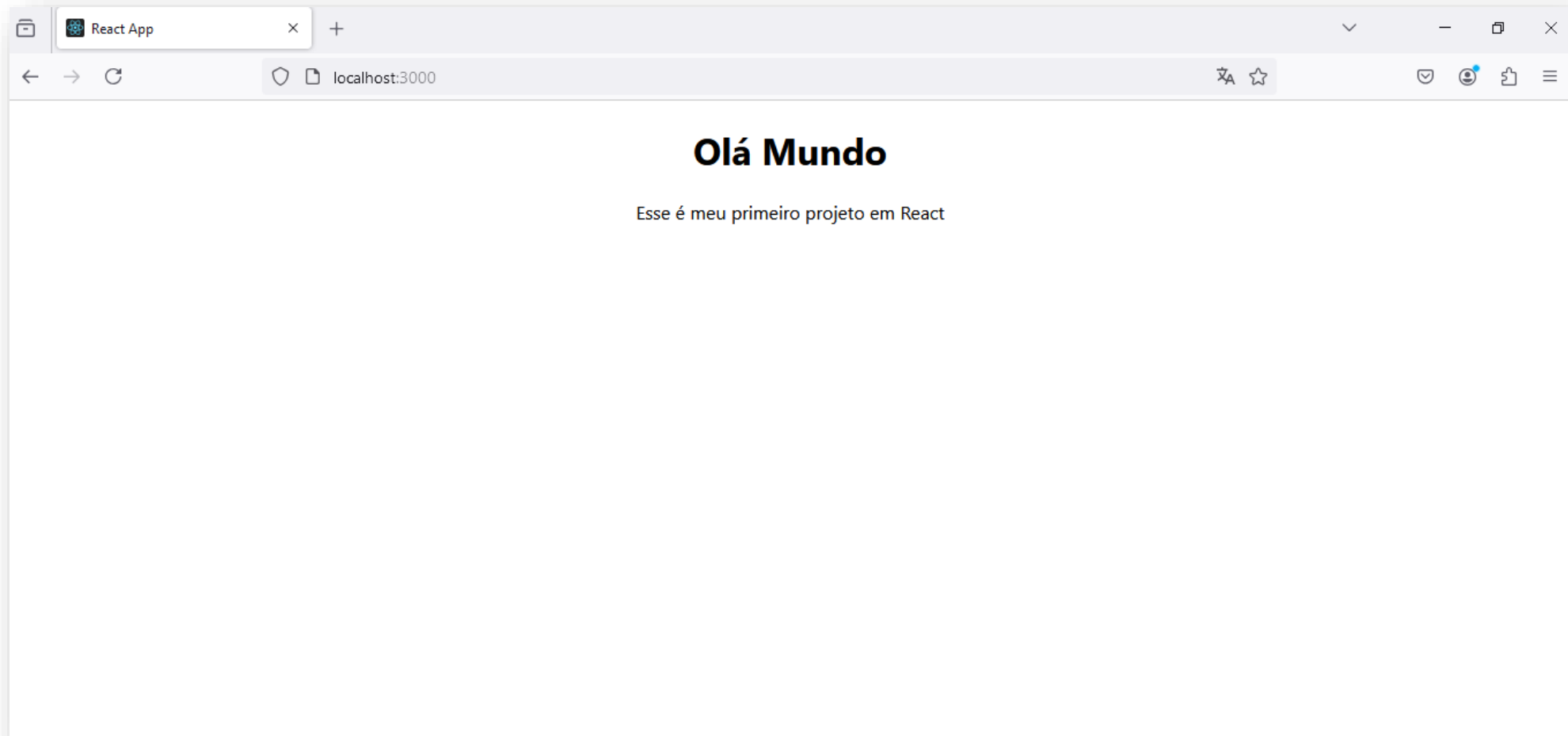
function App() {
  return (
    <div className="App">
      <h1>Olá Mundo</h1>
      <p>Esse é meu primeiro projeto em React</p>
    </div>
  );
}

export default App;
```



Modificar os elementos dentro desta tag

Página modificada



Modificando os estilos dos componentes

Para adicionar ou modificar estilos, você pode editar o arquivo App.css, que está vinculado ao componente App.js.

```
.App h1 {  
  color: blue;  
}
```

```
.App p {  
  font-size: 18px;  
  color: gray;  
}
```

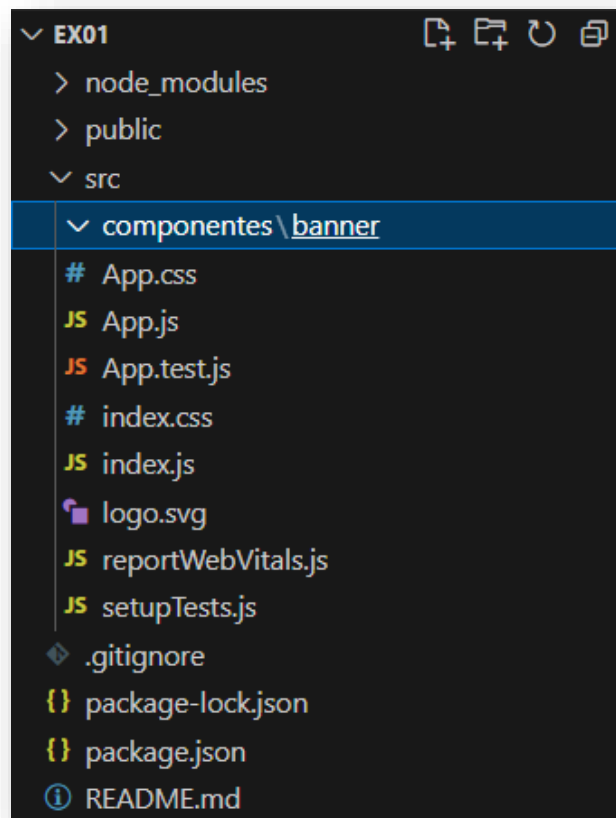
Para testar vamos acrescentar essas duas classes no final do arquivo, para estilizar a tag <h1> e a tag <p>

Página modificada



Componentes

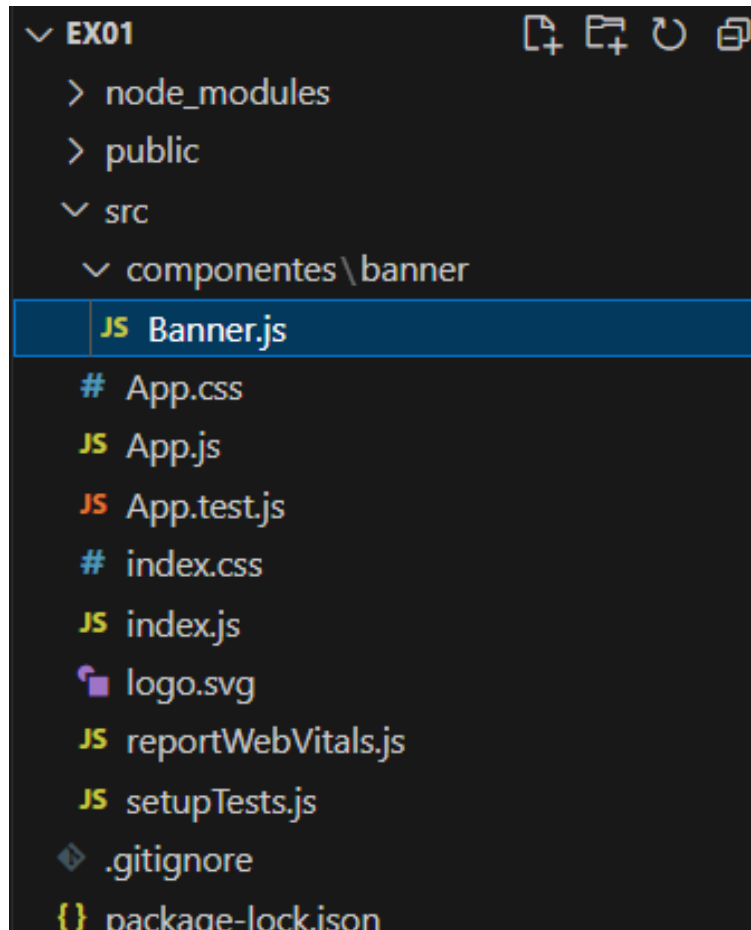
Agora, a forma de trabalharmos é com a forma React. Então em vez de criarmos os elementos e irmos fazendo `document.querySelector`, `document.createElement` para manipularmos o DOM, vamos utilizar o React e uma dessas formas de fazermos isso é criar componentes.



← Criar uma subpasta componentes na pasta src e dentro dela uma pasta chamada banner

Componentes

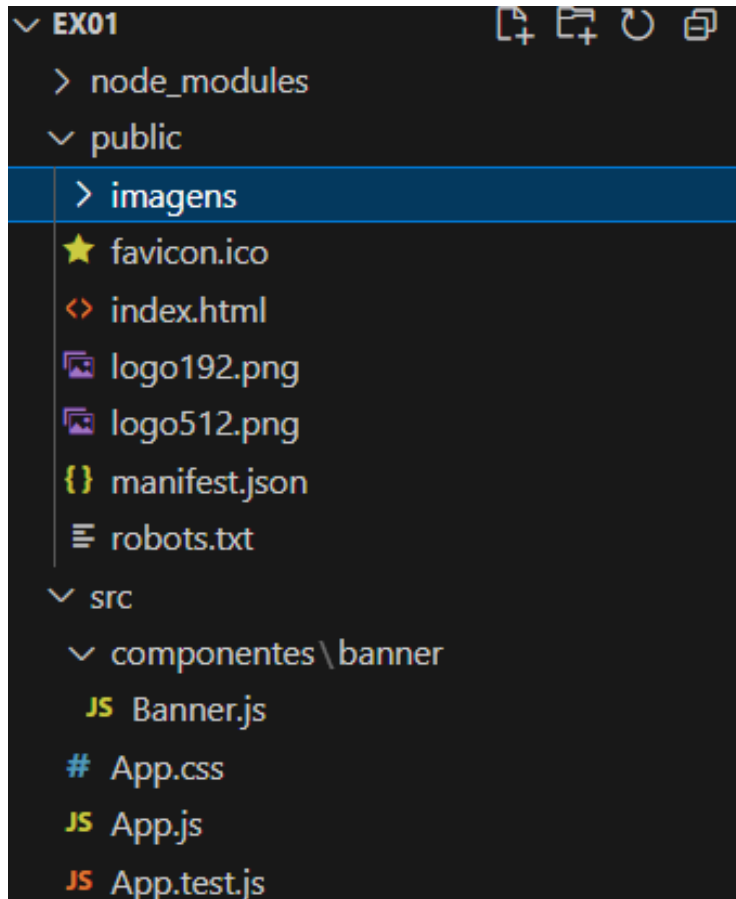
Dentro da pasta banner iremos criar o arquivo Banner.js, onde iremos escrever nosso componente.



Arquivo Banner.js

Componentes

Como esse componente terá uma imagem que será nosso banner, iremos criar uma subpasta na pasta .



A ideia do "public" é que quando criarmos e formos publicar em algum lugar, tudo que chamamos de "imagem estáticas" será disponibilizado colocando dentro da pasta "public".

Componentes

O arquivo banner.js deverá ter essa estrutura.

```
function Banner() {  
  return (  
      
  )  
}  
export default Banner
```

Componentes

Chamada do componente na página principal.

```
import './App.css';  
import Banner from './componentes/banner/Banner';  
  
function App() {  
  return (  
    <div className="App">  
      <Banner />  
    </div>  
  );  
}  
  
export default App;
```

Componentes

Chamada do componente na página principal.



Componentes

Podemos criar uma folha de estilo para o componente. Na pasta do componente criar Banner.css

```
.banner {  
    background-color: black;  
    text-align: center;  
}  
.banner img{  
    max-width: 100%;  
}
```

Componentes

Aplicando css no componente. Importar o Banner.css no Banner.js

```
import './Banner.css' ←
```

```
function Banner() {
```

```
  //JSX
```

```
  return (
```

```
    <header className='banner'> ←
```

```
      
```

```
    </header>
```

```
  )
```

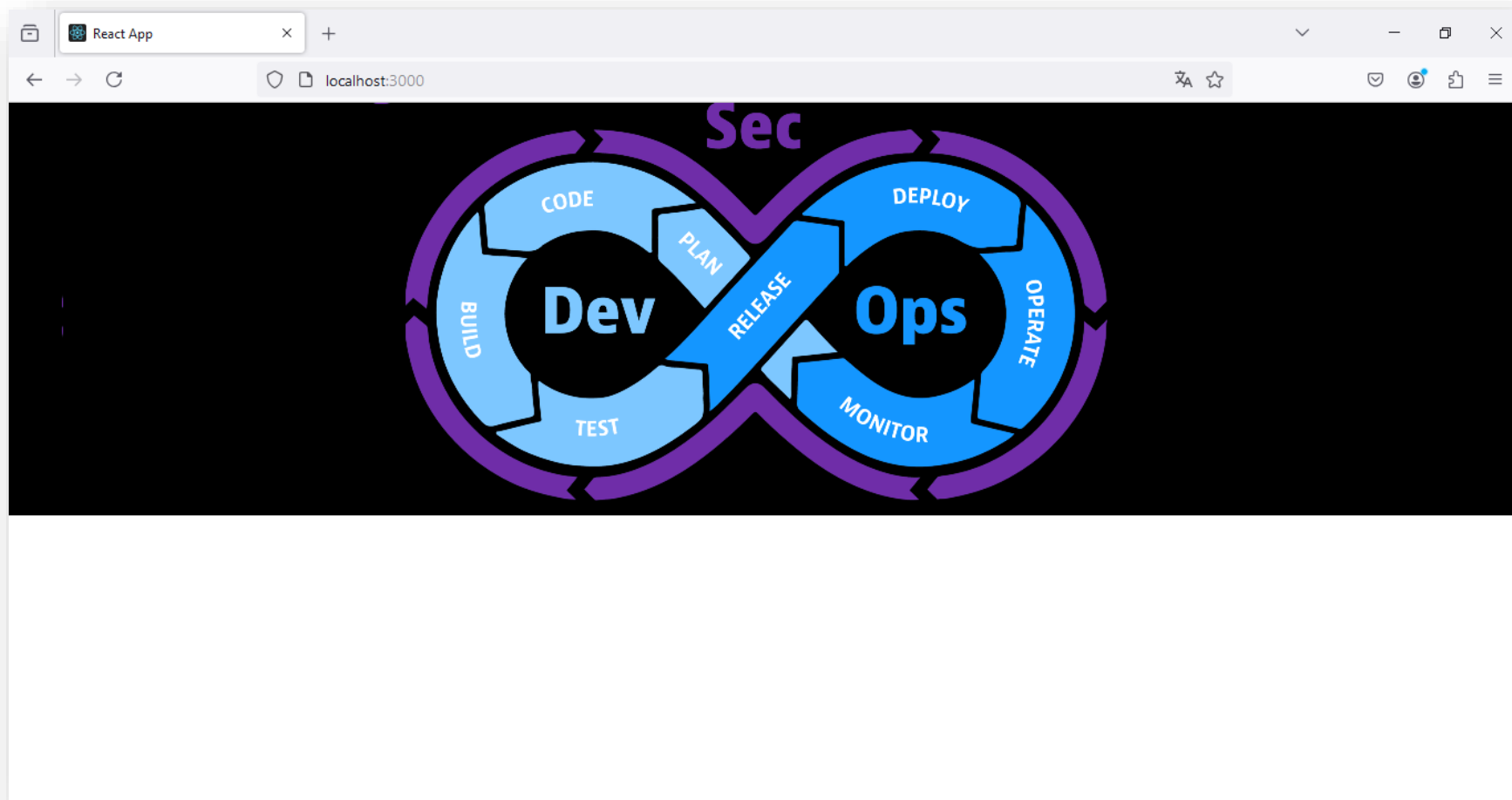
```
}
```

```
export default Banner
```

No JS class é uma palavra reservada, desta forma é necessário utilizar a palavra className

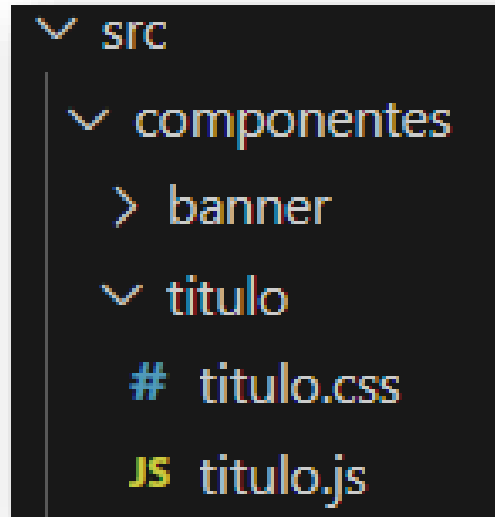
Componentes

Aplicando css no componente. Importar o Banner.css no Banner.js



Componentes

Uma outra forma de criar um componente é utilizando uma ***arrow function***. Vamos criar um componente para inserir um texto na tela. Na pasta Componentes criaremos a pasta titulo e dentro desta pasta os arquivos titulo.js e titulo.css.



Componentes

Criando o componente titulo.js.

```
import './titulo.css';

const CompTitulo = () => {
  return (
    <header className='titulo'>
      <h1>Testando Componentes em React</h1>
    </header>
  )
}

export default CompTitulo;
```

Componentes

Criando folha de estilo titulo.css para o componente titulo.js.

```
.titulo {  
  text-align: center;  
  font-size: 2vw;  
  margin-bottom: 8px;  
}
```

Componentes

Chamada do componente na página principal.

```
import './App.css';
import Banner from './componentes/banner/Banner';
import CompTitulo from './componentes/titulo/titulo'; ←

function App() {
  return (
    <div className="App">
      <Banner />
      <CompTitulo /> ←
    </div>
  );
}

export default App;
```

Componentes

Chamada do componente na página principal.



Componentes

Reutilização de Componentes.

Criando um componente calendário para aquisição de data.

```
import './Compdata.css';

const CompData = () => {
  return (
    <header className='data'>
      <input type='date' />
    </header>
  )
}

export default CompData;
```


Componentes

Reutilização de Componentes.

Estilizando o componente.
Criando o arquivo Compdata.css

```
input {
  color: #fff;
  font-size: 28px;
  width: 40%;
  padding: 20px 5px 5px;
  background-color: #4a4a4a;
  border: none;
  border-radius: 4px;
}

label {
  color: #fff;
  position: absolute;
  left: 0;
  bottom: 0;
  width: 100%;
  height: 100%;
  pointer-events: none;
  border-bottom: 1px solid #ffffff;
}

span {
  position: absolute;
  bottom: 5px;
  left: 5px;
  font-size: 28px;
  transition: all 0.3s ease;
}
```

```
label::after {
  content: "";
  position: absolute;
  left: 0;
  bottom: -1px;
  width: 100%;
  height: 100%;
  border-bottom: 3px solid #e0138c;
  transform: translateX(-100%);
  transition: all 0.3s ease;
}

input:focus + label span,
input:valid + label span {
  transform: translateY(-150%);
  font-size: 14px;
  bottom: 10px;
  color: #e0138c;
}

input:focus + label::after,
input:valid + label::after {
  transform: translateX(0%);
}
```

Componentes

Reutilização de Componentes.

Criando o componente escolha de datas .

```
import CompData from '../compdata/Compdata';  
import './Compescdata.css';
```

```
const CompEscData = () => {  
  return(  
    <header>  
      <div className='col-12'>  
        <h1>Escolha de datas</h1>  
      </div>  
      <div className='row'>  
        <div className='col-6'>  
          <p>Data Atual</p>  
          <CompData />  
        </div>  
        <div className='col-6'>  
          <p>Data Nascimento</p>  
          <CompData />  
        </div>  
      </div>  
    </header>  
  )  
}  
export default CompEscData;
```

Componentes

Reutilização de Componentes.

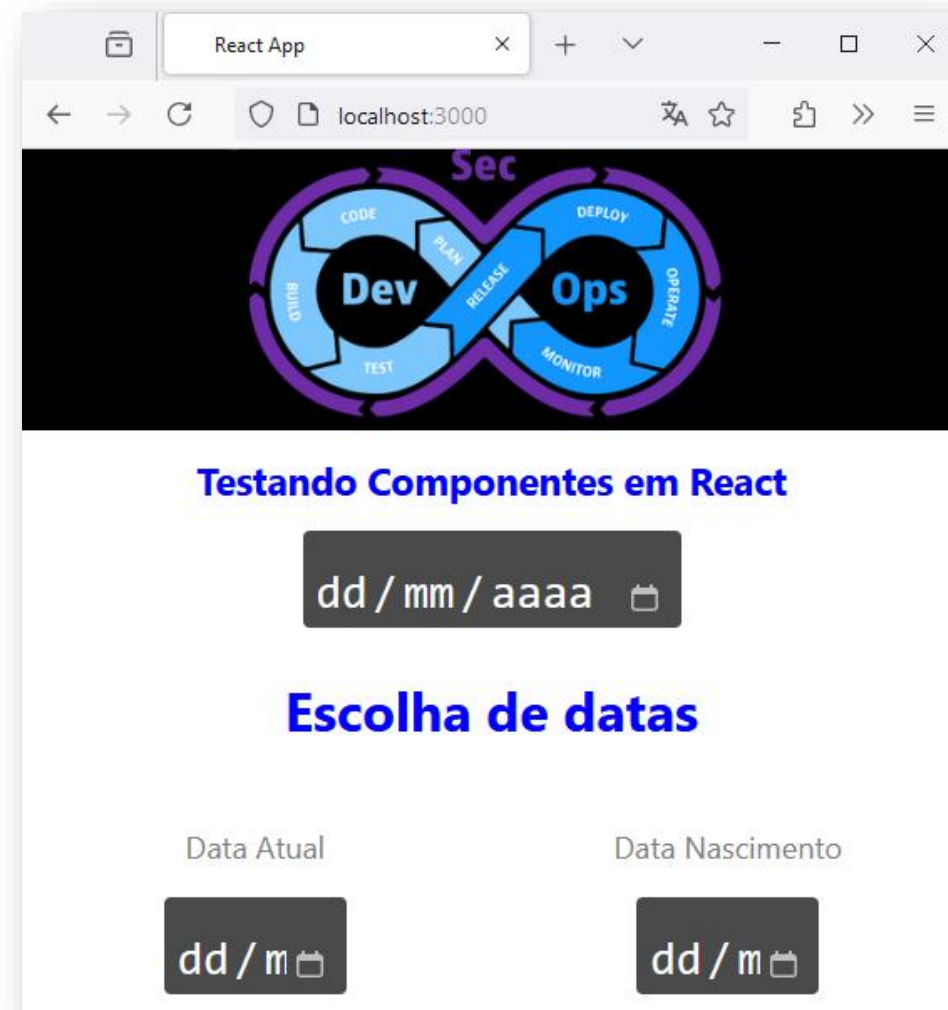
Criando a folha de estilos para o componente escolha de datas . Arquivo Compescdata.css

```
* {  
    box-sizing: border-box;  
}  
  
.row::after {  
    content: "";  
    clear: both;  
    display: table;  
}  
  
[class*="col-"] {  
    float: left;  
    padding: 5px;  
}  
  
.col-1 {width: 8.33%;}  
.col-2 {width: 16.66%;}  
.col-3 {width: 25%;}  
.col-4 {width: 33.33%;}  
.col-5 {width: 41.66%;}  
.col-6 {width: 50%;}  
.col-7 {width: 58.33%;}  
.col-8 {width: 66.66%;}  
.col-9 {width: 75%;}  
.col-10 {width: 83.33%;}  
.col-11 {width: 91.66%;}  
.col-12 {width: 100%;}
```

Componentes

Reutilização de Componentes.

Colocando todos os componentes na página principal.



Exercício

Você foi contratado para desenvolver a página inicial de um pequeno estabelecimento comercial. Esse estabelecimento pode ser uma padaria, um café, uma loja de roupas ou qualquer outro tipo de comércio. O objetivo é criar uma página simples que apresente o nome do estabelecimento, uma breve descrição dos serviços ou produtos oferecidos, e imagens ilustrativas (por exemplo, uma foto da fachada da loja ou de produtos oferecidos).

Objetivo: Desenvolver uma página inicial básica de um estabelecimento comercial utilizando React, onde serão exibidos o nome do estabelecimento, a descrição e imagens ilustrativa do estabelecimento.

Requisitos:

O nome do estabelecimento deve estar em destaque com um tamanho de fonte maior e dentro de um banner.

O layout deve estar centralizado e com cores que combinem com o estilo do comércio. As imagens devem estar alinhadas com os textos e estilizadas adequadamente.