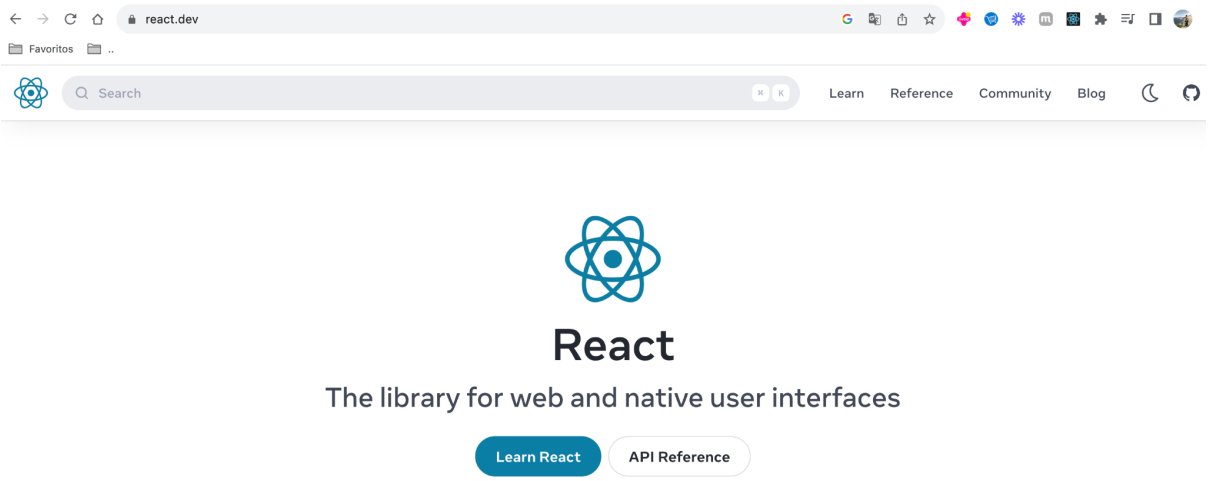
INTRODUÇÃO AO REACT

O React é uma biblioteca JavaScript de código aberto desenvolvida pelo Facebook para criar interfaces de usuário (UI) interativas e eficientes. Ele é utilizado para construir componentes reutilizáveis para interfaces de usuário (UI) e facilita o desenvolvimento de aplicativos de página única (SPA) onde as atualizações são feitas de forma eficiente, sem a necessidade de recarregar a página inteira.



O React utiliza uma abordagem baseada em componentes, o que significa que você pode dividir a interface do usuário em componentes independentes e reutilizáveis, tornando o desenvolvimento mais modular e fácil de manter. Cada componente do React representa uma parte específica da interface do usuário e pode ter seu próprio estado interno.

Uma das principais características do React é o uso do conceito de Virtual DOM (Documento de Objeto Modelo), que é uma representação em memória da estrutura da página. O Virtual DOM permite que o React faça atualizações eficientes na interface do usuário, minimizando a quantidade de manipulação direta do DOM, o que geralmente é uma operação custosa em termos de desempenho.

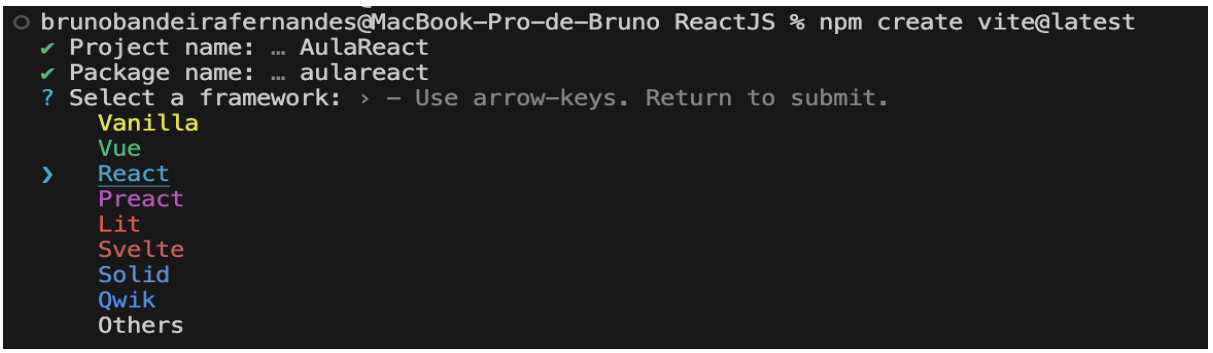
Além disso, o React é frequentemente utilizado em conjunto com outras bibliotecas e ferramentas, como o React Router para controle de navegação, o Redux para gerenciamento de estado e o webpack para empacotamento e construção de aplicativos.

CRIANDO PROJETO EM REACT

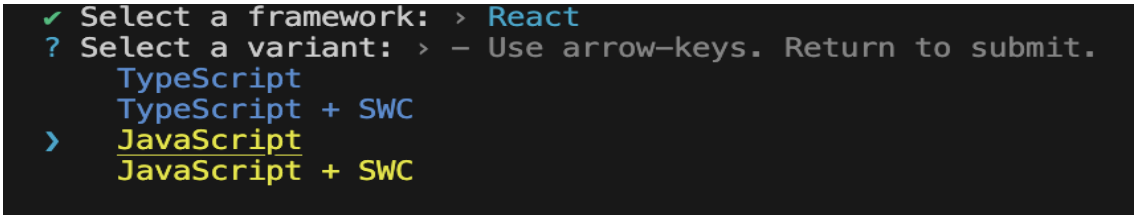
Para criar um projeto utilizando a estrutura Vite, utilize o seguinte comando:

npm create vite@latest

Será solicitado o nome do Projeto. Informei AulaReact.

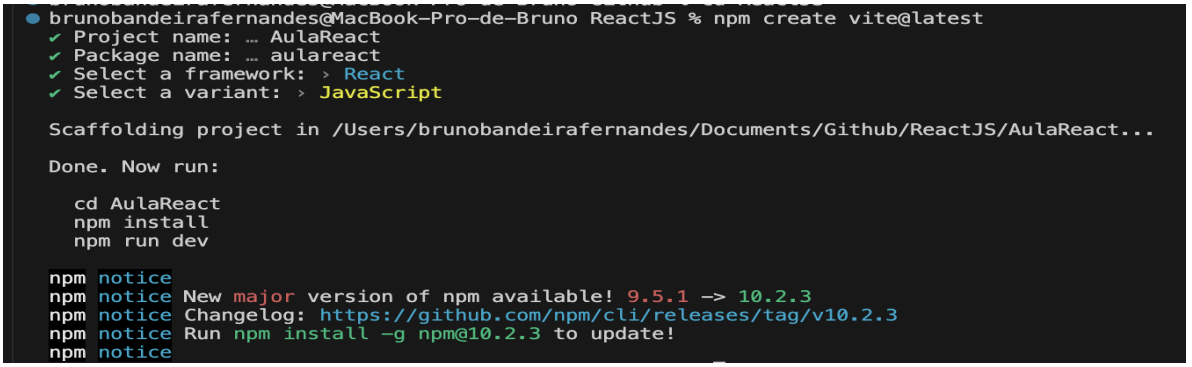


Em seguida, deverá selecionar a variante (JS ou TS). Selecionei JS.



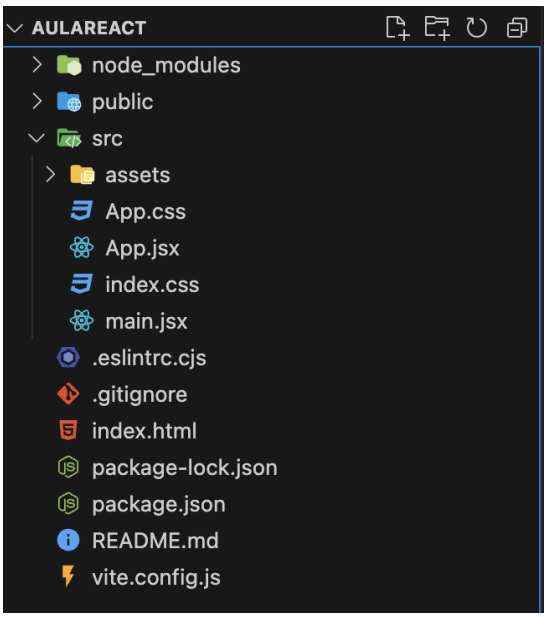
Pronto, o projeto será criado. Porém, o Vite não realiza a instalação do pacote node_modules (por isso a instalação é tão leve). Neste caso, entre na pasta e faça a instalação utilizando comando

npm install



Para executar o projeto, utilize o comando:
npm run dev

ESTRUTURA DO PROJETO



src/main.jsx ou src/main.tsx
Este é o ponto de entrada principal para o seu aplicativo. Dependendo se você está usando JavaScript ou TypeScript, o arquivo pode ser main.jsx ou main.tsx. Aqui é onde você pode importar seu componente raiz React e renderizá-lo no DOM.

src/App.jsx ou src/App.tsx:
Este é o componente raiz do seu aplicativo. Aqui é onde você pode definir a estrutura geral do seu aplicativo e incorporar outros componentes.

index.html:
Este é o arquivo HTML principal do seu aplicativo. Vite injetará automaticamente os scripts necessários no arquivo HTML durante o desenvolvimento.

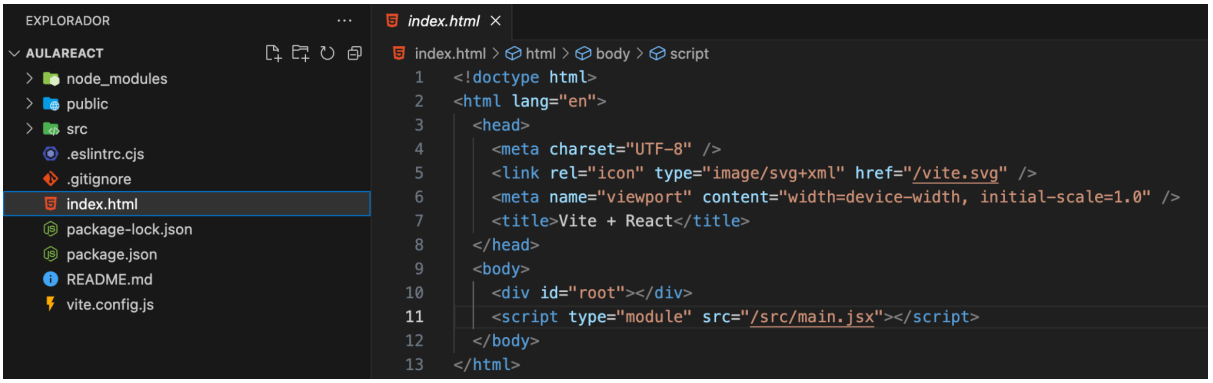
vite.config.js:
Este arquivo contém a configuração do Vite para o seu projeto. Aqui você pode ajustar opções de construção, configurar plugins e definir outras configurações específicas do Vite.

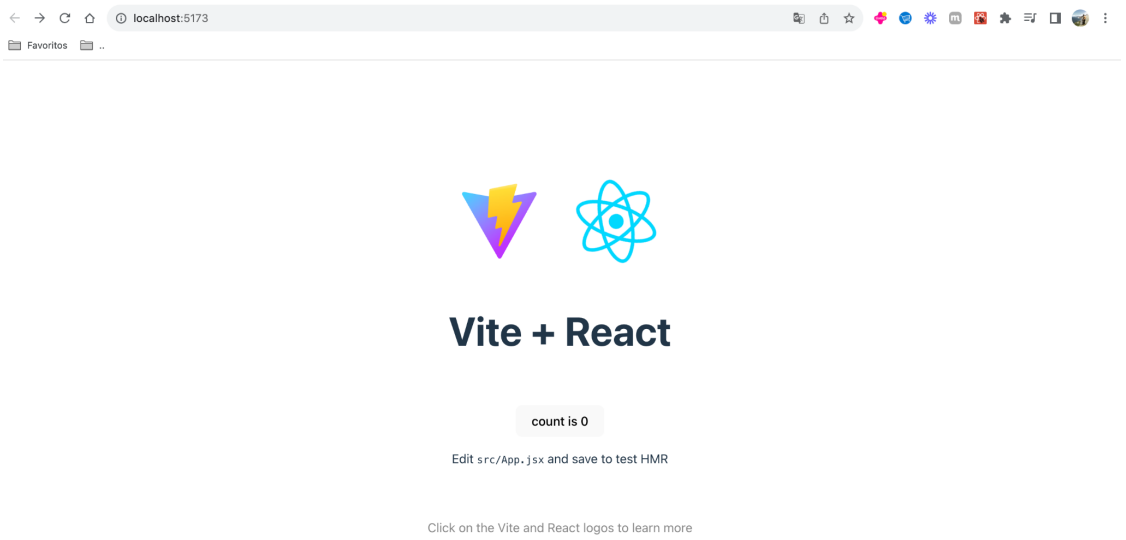
package.json:
O arquivo package.json contém as dependências do seu projeto, scripts de construção, versão do Node.js e outras configurações relacionadas ao projeto. Vite também usa este arquivo para definir scripts de construção personalizados.

node_modules/:
Este é o diretório onde as dependências do seu projeto são instaladas pelo npm.

public/:
Similar ao projeto Create React App, o diretório public contém arquivos estáticos como o HTML principal, ícones e outros recursos que você deseja acessar diretamente.

O padrão é que seja inicializado o arquivo index.html. E observe que neste arquivo não existe o conteúdo que está aparecendo na página.





Isso acontece porque o React utiliza o conceito de SPA. O termo "SPA" refere-se a "Single Page Application" (Aplicação de Página Única) e React é frequentemente utilizado para desenvolver esse tipo de aplicativo.

Single Page Application (SPA):
Uma SPA é uma aplicação web que carrega uma única página HTML e atualiza dinamicamente o conteúdo desta página à medida que o usuário interage com a aplicação. Em vez de carregar páginas HTML separadas para cada ação, o aplicativo carrega e manipula dados no fundo, modificando apenas a parte da página que precisa ser alterada.

Como o React Contribui para SPAs:

- **Componentização:** O React promove a construção de interfaces de usuário por meio de componentes reutilizáveis. Cada componente React pode ser pensado como uma parte isolada e interativa da interface.
- **Virtual DOM:** O React utiliza o Virtual DOM para otimizar as atualizações na interface do usuário. Ele compara o estado atual do DOM com uma representação virtual em memória e atualiza apenas as partes que foram alteradas, minimizando a manipulação direta do DOM.
- **State Management:** O React oferece um sistema de gerenciamento de estado eficiente, permitindo que os componentes reajam a mudanças no estado e atualizem dinamicamente a interface do usuário.
- **React Router:** O React Router é uma biblioteca comumente usada para adicionar roteamento à aplicação React. Ele permite a navegação entre "páginas" ou "rotas" sem recarregar a página.

Vantagens das SPAs com React:

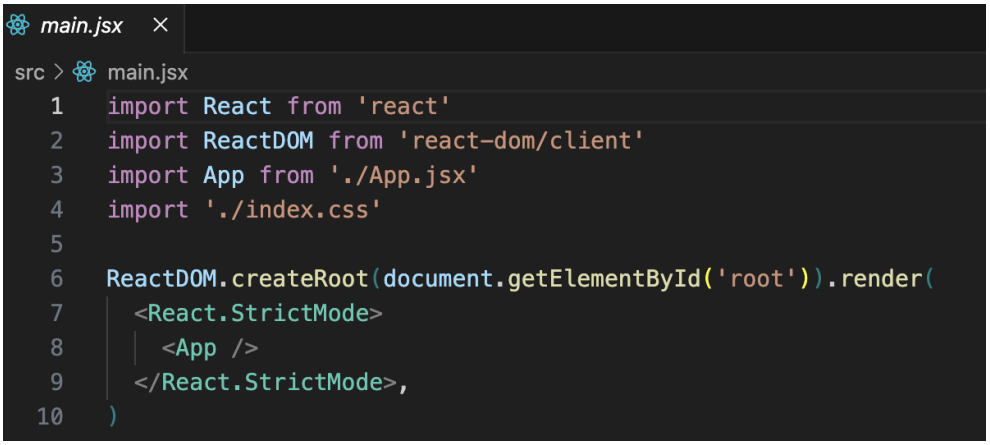
- **Experiência do Usuário Aprimorada:** A navegação sem recarregar páginas resulta em uma experiência de usuário mais fluida e rápida.
- **Manutenção Simplificada:** A abordagem de componentização do React facilita a manutenção e a organização do código.
- **Desenvolvimento Eficiente:** O Hot Module Replacement (HMR) no ambiente de desenvolvimento do React permite atualizar componentes em tempo real, acelerando o ciclo de desenvolvimento.
- **Integração com APIs:** Facilita a integração com APIs assíncronas para carregar dados dinamicamente sem recarregar a página.

Em resumo, entenda como funciona o carregamento:
Primeiramente será carregado o arquivo index.html e realizada a leitura do body. No body, contém um id "root" e a chamada de um arquivo na pasta src/main.jsx.

```
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
```

O arquivo src/main.jsx utiliza o método createRoot de ReactDOM (linha 6) para criar um "root" (raiz) React. O método createRoot é uma abordagem mais moderna para a renderização de componentes em React. O código

document.getElementById('root') obtém o elemento HTML com o ID "root" do documento (na página index.html) e o méotod render inicia o processo de renderização.



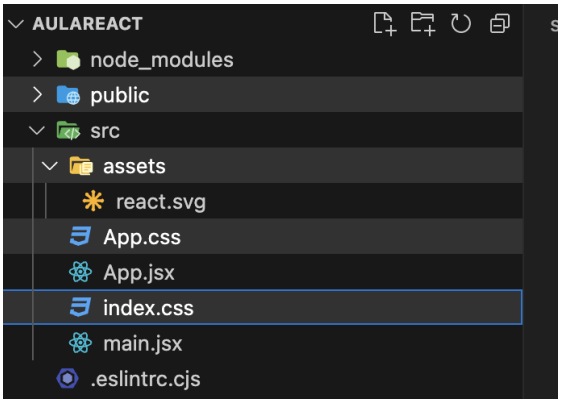
Esse trecho de código configura o ambiente React, importa o componente principal (App), aplica estilos globais e inicia o processo de renderização no elemento HTML com o ID "root". O uso de React.StrictMode ajuda a identificar e corrigir práticas não recomendadas durante o desenvolvimento.

O arquivo App.jsx apresenta o conteúdo apresentado na página e o arquivo index.css apresenta a estilização da página.

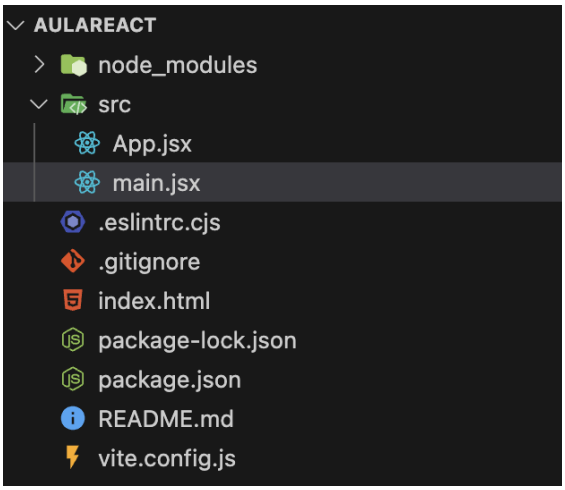
REESTRUTURANDO O PROJETO

Vamos reestruturar o projeto, apagando a página padrão e ajustando conforme o projeto a ser implementado.

Inicialmente, apague as pastas public e assets; e os arquivos App.css e index.css.



Ficará assim:



Remova a linha 5 do arquivo index.html, que contém a importação do icon: <link rel="icon" type="image/svg+xml" href="/vite.svg" />.

Remova a importação do css no arquivo main.jsx (import './index.css') e exclua todo o conteúdo do arquivo App.jsx.

Crie uma pasta chamada pages e arraste o arquivo App.jsx para dentro da pasta criada. Por fim, renomeie o arquivo App.jsx para Home.jsx.

Observe que os arquivos possuem extensão jsx. Os arquivos JSX no React são arquivos que contêm código JavaScript estendido com a sintaxe JSX (JavaScript

XML). O JSX é uma extensão de sintaxe que lembra XML/HTML e é usada para descrever a estrutura da interface do usuário em aplicações React.

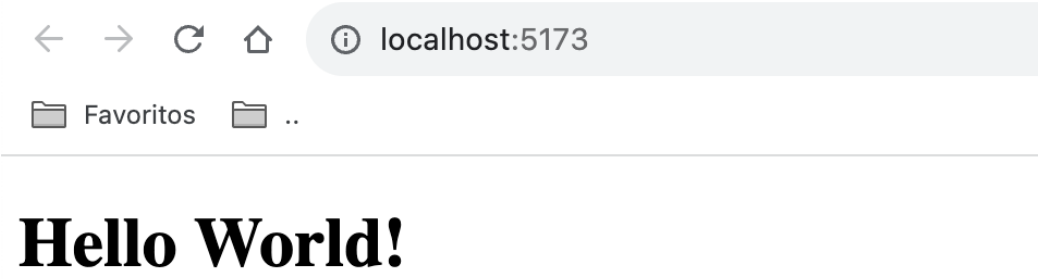
Arquivo Home.jsx

```
export function Home(){


   return(
       <h1>Hello World!</h1>
   )
}
```

Arquivo main.jsx.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import { Home } from './pages/Home.jsx'

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

Por fim, o resultado da alteração:



UTILIZANDO O CONCEITO DE FRAGMENT

No React, um "fragment" é uma maneira de agrupar múltiplos elementos filhos sem a necessidade de criar um elemento DOM adicional. Ele permite que você retorne vários elementos sem adicionar um nó extra à árvore DOM.

Antes dos fragments, se você quisesse retornar vários elementos adjacentes em um componente React, você precisaria envolvê-los em um único elemento pai. Por exemplo:

```
export function Home(){


   return(
       <div>
           <h1>Hello World!</h1>
           <span>Bruno Bandeira Fernandes</span>
       </div>
   )
}
```

Nesse exemplo, o é usado como um wrapper em torno dos elementos <h1> e dois <p>. Embora isso funcione, às vezes você não quer adicionar um elemento extra ao DOM, especialmente se não tiver um propósito semântico.

É aí que os fragments entram. Com a introdução dos fragments no React, você pode reescrever o exemplo acima usando um fragment, que é uma sintaxe mais concisa:

```
export function Home(){


   return(
       <>
           <h1>Hello World!</h1>
           <span>Bruno Bandeira Fernandes</span>
       </>
   )
}
```

O <> e </> são a sintaxe de fragmentos. Eles são equivalentes a <React.Fragment>.

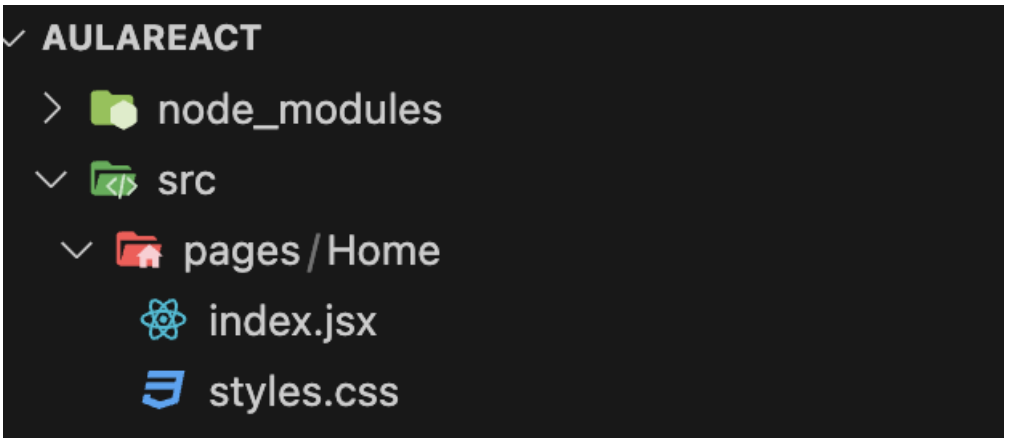
Vantagens dos fragments:

- **Não Adiciona Nós Extras ao DOM:** Como mencionado, ao usar fragments, você evita adicionar um nó DOM extra ao seu componente, resultando em uma árvore DOM mais limpa.
- **Sintaxe Mais Concisa:** A sintaxe do fragment é mais concisa e legível, especialmente quando você tem vários elementos adjacentes.
- **Melhora a Semântica:** Em alguns casos, a adição de um elemento extra só para envolver outros elementos pode prejudicar a semântica do seu código.

ADICIONANDO CSS E ESTRUTURANDO O PROJETO

Para cada página na aplicação, iremos criar uma pasta dentro da página 'pages'.

Crie uma pasta chamada Home, dentro da pasta pages. Arraste o arquivo Home.jsx para dentro da pasta Home e renomeie o arquivo para index.jsx. Por padrão, o React vai chamar inicialmente o arquivo chamado index. Agora, crie um arquivo chamado styes.css. Este arquivo irá conter o estilo de somente a página Home.



Lembre-se de corrigir as importações.

Para conferir, segue o arquivo index.jsx da pasta Home:

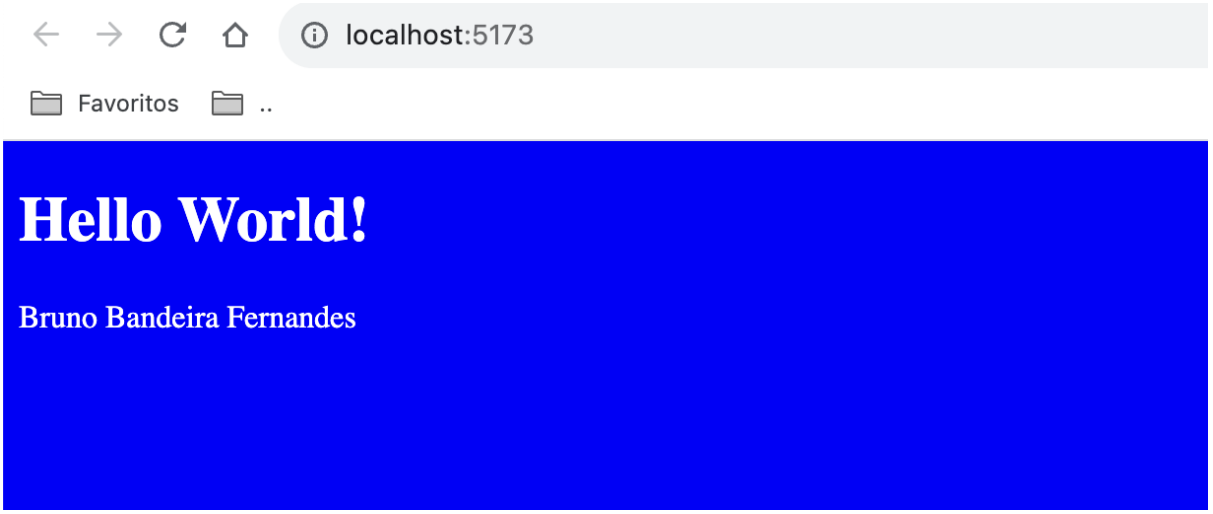
```jsx
import "./styles.css"

export function Home(){

   return(
       <>
           <h1>Hello World!</h1>
           <span>Bruno Bandeira Fernandes</span>
       </>
   )
}
```

Um estio (qualquer) para teste, no arquivo stules.css:

```css
body{
   background: blue;
}

h1, span{
```

```
    color: white;
}
```

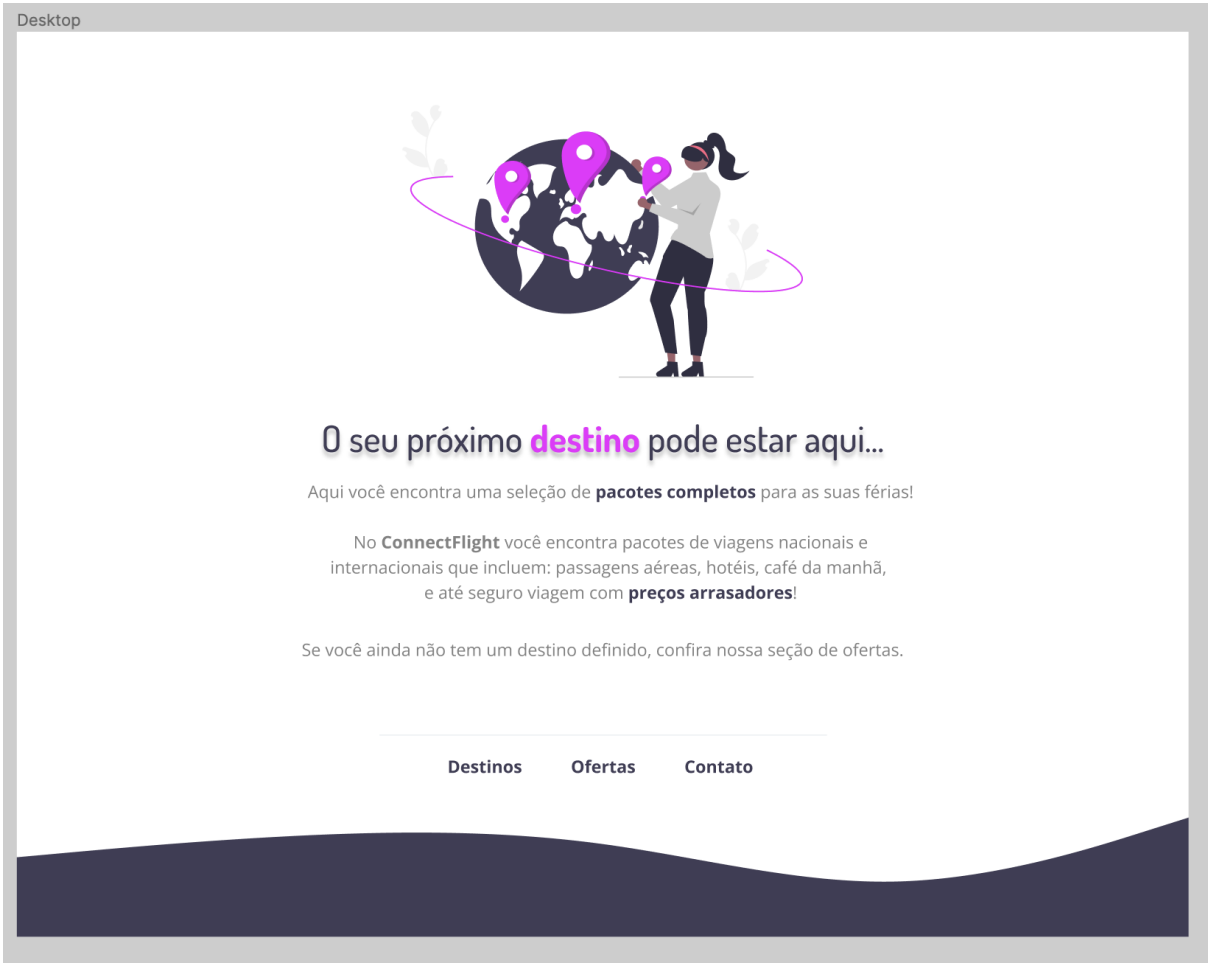
O resultado:



Link do Github:
https://github.com/brunobandeiraf/Aulas_React/commit/cb3bb33721210ffe86c710557df8f1308abe1048

ATIVIDADE DE SALA

Desenvolva o projeto em React.



Link do Figma:
https://www.figma.com/file/U9czFYiF7DVGINZWMeMUGx/Aula-01---React?type=design&node-id=0-1&mode=design&t=r0cbCvs7vzSI1yFk-0

Habilidades Técnicas:

- **HTML e JSX:**
  - Conhecimento sólido em HTML para estruturar a página.
  - Familiaridade com JSX, a sintaxe de extensão de JavaScript usada no React.
- **CSS:**
  - Habilidade para criar estilos atraentes.
  - Conhecimento em seletores CSS e propriedades de estilo.
  - Compreensão de Flexbox e/ou Grid para layouts flexíveis.
- **React Basics:**
  - Entendimento dos conceitos fundamentais do React, mesmo que a aplicação seja simples.

Habilidades Comportamentais:

- Resolução de Problemas:
    - Capacidade de identificar e resolver problemas relacionados à renderização e estilo da página.
- Aptidão para Detalhes:
    - Atenção aos detalhes é crucial para garantir que a página seja visualmente atraente e funcional em diferentes dispositivos e navegadores.
- Organização:
    - Manter um código bem organizado e estruturado, seguindo as melhores práticas do React.

Entregar o link do projeto no github.