FIAP

GRADUAÇÃO

45697056

TDS

Responsive Web Development

Prof. Alexandre Carlos : profalexandre.jesus@fiap.com.br
Prof. Luís Carlos : lsilva@fiap.com.br
Prof. Wellington Cidade : profwellington.tenorio@fiap.com.br

45697056

INTRODUÇÃO

VITE + REACT

REACT

45697056

O React é uma biblioteca JavaScript de código aberto utilizada para construir interfaces de usuário interativas e eficientes.

Ele permite que os desenvolvedores dividam a interface em componentes reutilizáveis e gerenciem automaticamente as atualizações do DOM, resultando em um desenvolvimento mais organizado e um desempenho otimizado.

O React utiliza uma abordagem baseada em componentes e utiliza o conceito de "Virtual DOM" para minimizar as atualizações diretas no DOM, proporcionando uma experiência de usuário fluida em aplicações web e móveis.

VITE + REACT

VITE

45697056

Vite é uma build tool de desenvolvimento web ultrarrápido e minimalista.

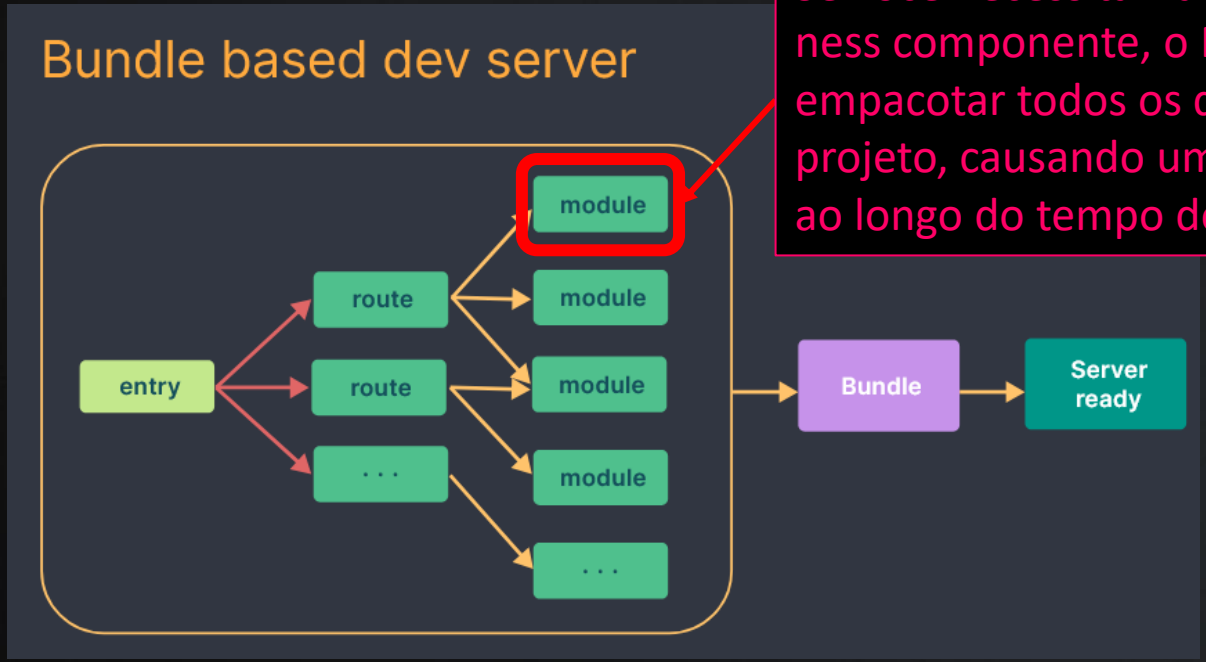
Ele adota uma abordagem de desenvolvimento baseada em módulos ES e oferece tempos de compilação extremamente rápidos, eliminando a necessidade total de bundling(empacotamento total dos componentes) durante o request do usuário.

Isso resulta em um processo de carregamento mais ágil e eficiente, melhorando a experiência do usuário.

VITE + REACT

REACT SEM VITE

45697056

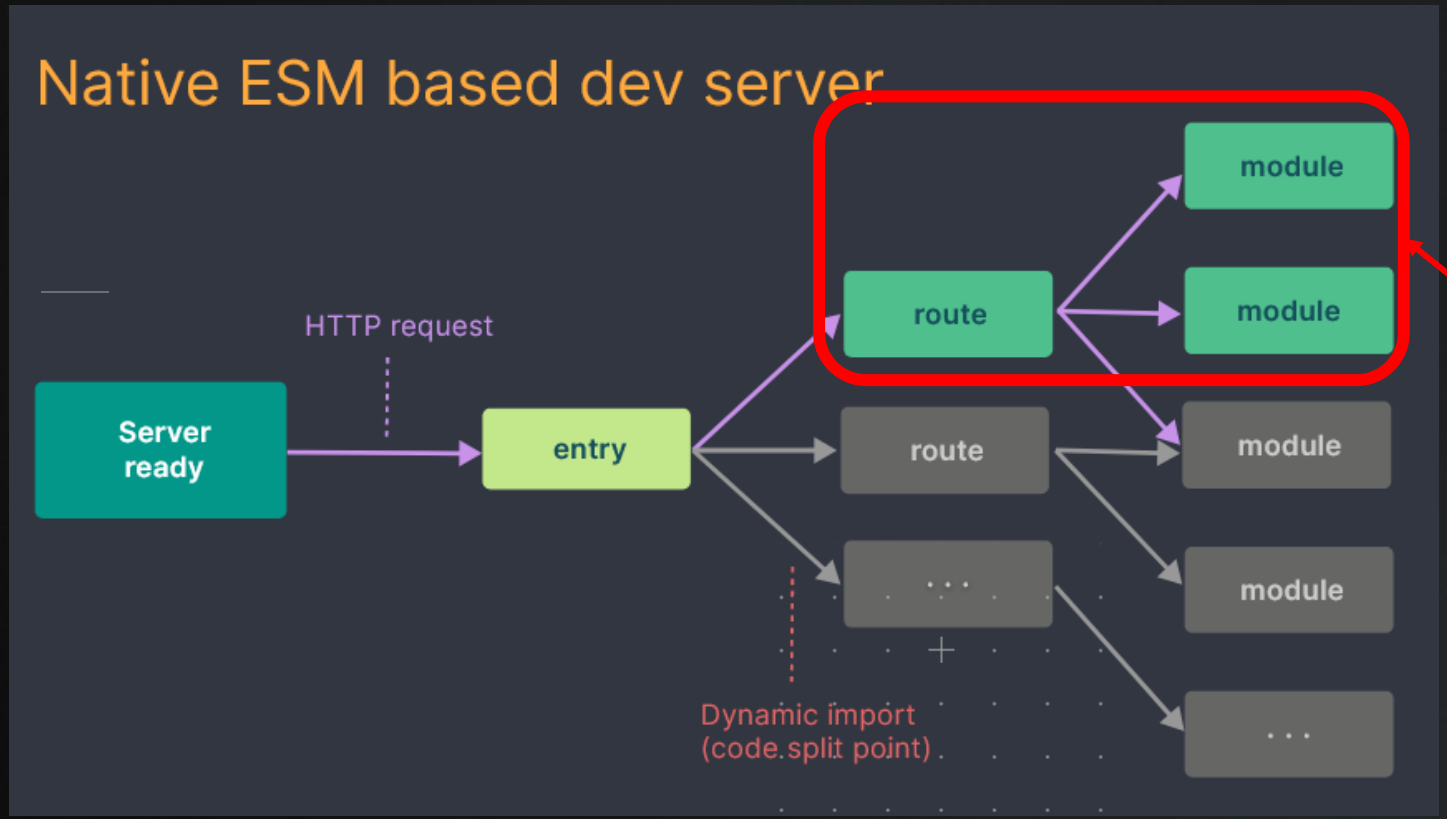


Se você necessitar fazer uma alteração somente ness componente, o React, vai Transpilar e empacotar todos os demais componentes do projeto, causando um impacto no processamento ao longo do tempo de vida do projeto.

VITE + REACT

REACT COM VITE

45697056



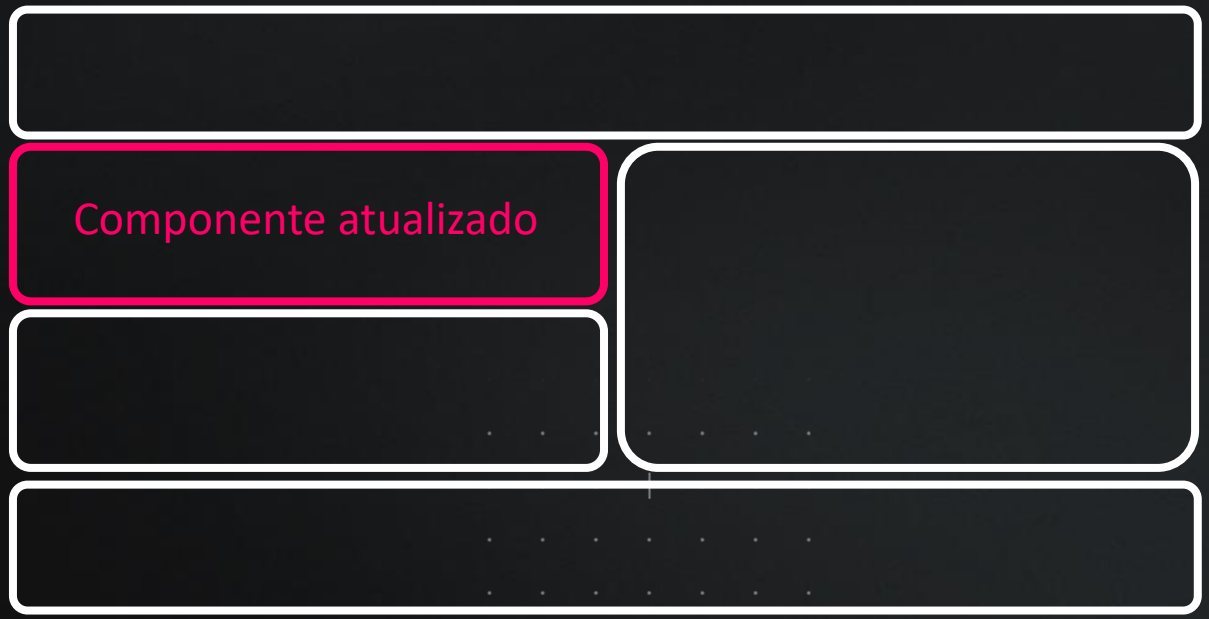
Utilizando o Vite, ele faz o fluxo inverso até a solicitação, excluindo assim os componentes que não participaram do processo, tornando mais rápido o carregamento dos componentes. Isso graças ao ESBuild.

VITE + REACT

REACT

45697056

Exemplo de renderezação de componente isolado em uma SPA(Single Page Application).

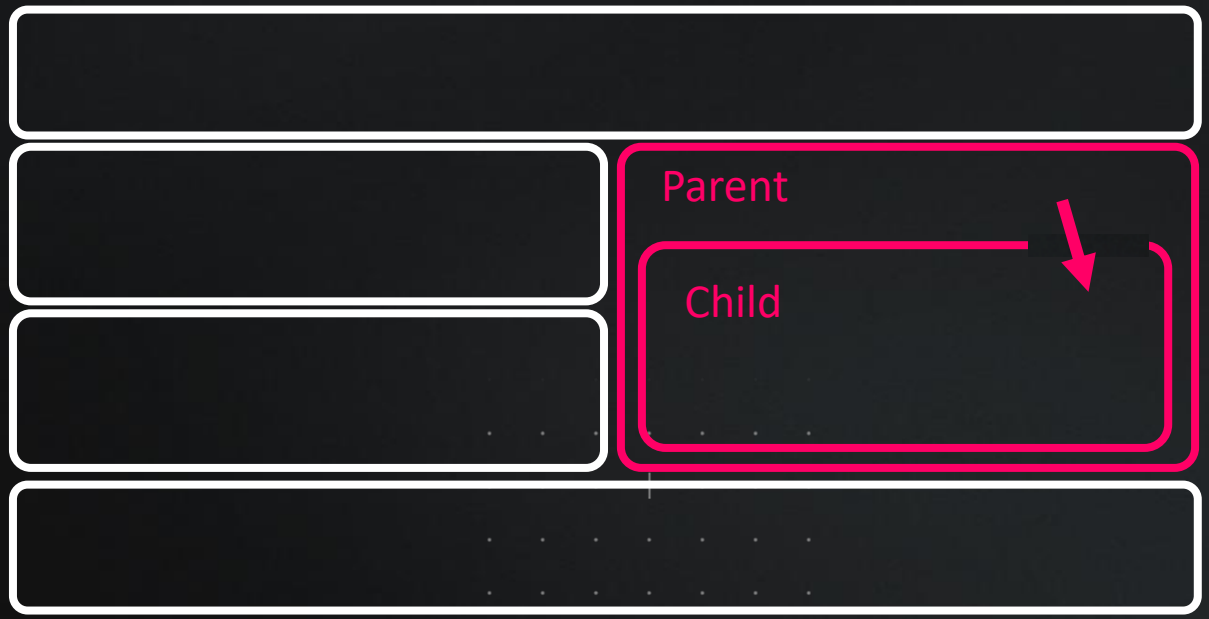


VITE + REACT

REACT

45697056

Ele trabalha com um fluxo unidirecional, em um único sentido, ou “One-way data flow”. As informações devem sempre vir do elemento pai para o elemento filho.



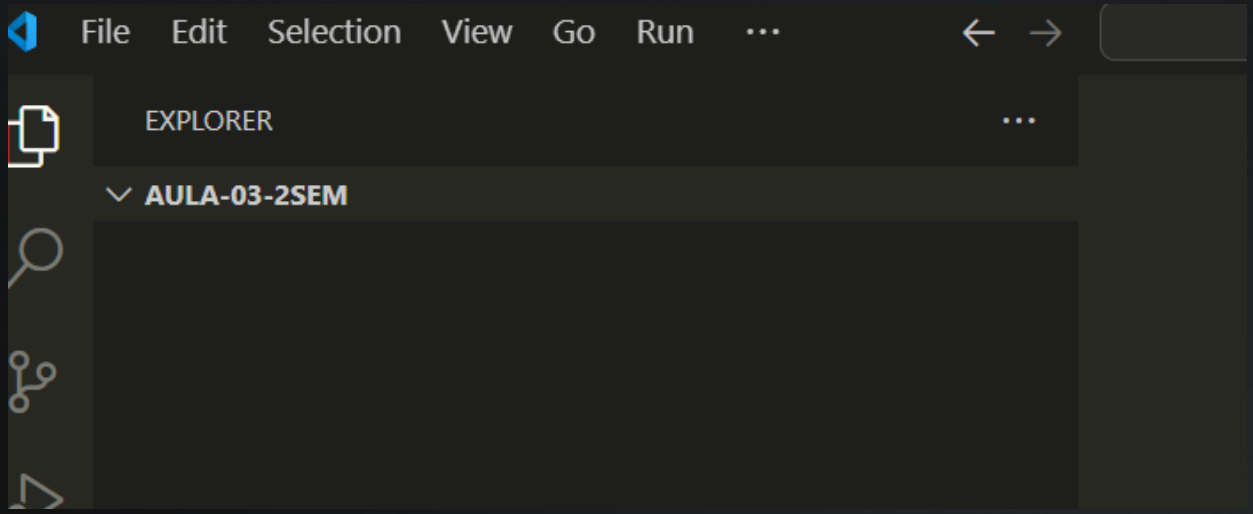
VITE + REACT

REACT

45697056

Na apostila passada, criamos o projeto base do vite+react, agora vamos criar um novo projeto, inicializando já o versionamento.

- Vamos abrir o VS Code e criar uma pasta na área de trabalho, chame de Aula0x.

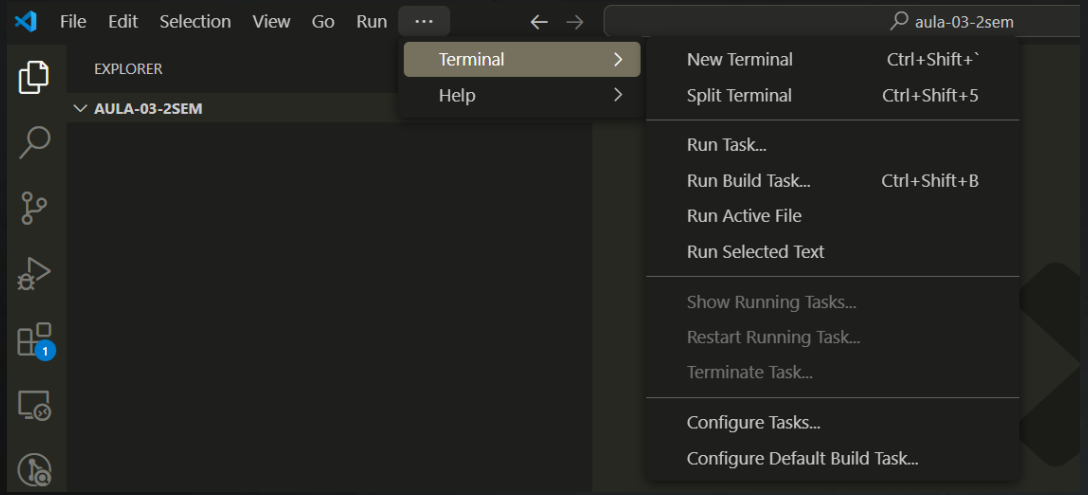


VITE + REACT

REACT

45697056

- Abra o terminal do VS Code, pode usar o atalho: crtl + shift + ‘ ou Menu Terminal -> Novo terminal.



- Obs. Se quiser também pode usar o prompt de comando, basta acessar a nossa pasta Aula0x.

REACT

45697056

- Agora vamos usar o npm, que é gerenciador de pacote do node:

```
PS D:\projetos\base> npm create vite@latest
? Project name: » vite-project
```

- Digite o comando acima e depois coloque o nome do projeto e pressione enter.

REACT

45697056

- Agora vamos escolher o template/framework:

```
PS D:\projetos\base> npm create vite@latest
√ Project name: ... my-app
? Select a framework: » - Use arrow-keys. Return to submit.
    Vanilla
    Vue
>   React
    Preact
    Lit
    Svelte
    Solid
    Qwik
    Others
```

- Utilize as setas do teclado e selecione React e pressione enter.

REACT

45697056

- Por último escolha a variante da linguagem. ***Typescript***

```
PS D:\projetos\base> npm create vite@latest
√ Project name: ... my-app
√ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
>   TypeScript
    TypeScript + SWC
    JavaScript
    JavaScript + SWC
    Remix ↗
```

- Selecione Typescript e pressione enter.

REACT

45697056

- Execute os comandos na ordem.

```
PS D:\projetos\base> npm create vite@latest
√ Project name: ... my-app
√ Select a framework: » React
√ Select a variant: » TypeScript

Scaffolding project in D:\projetos\base\my-app...

Done. Now run:

  cd my-app
  npm install
  npm run dev

PS D:\projetos\base>
```

abre a pasta do projeto

instala a pasta de dependencias node_modules.

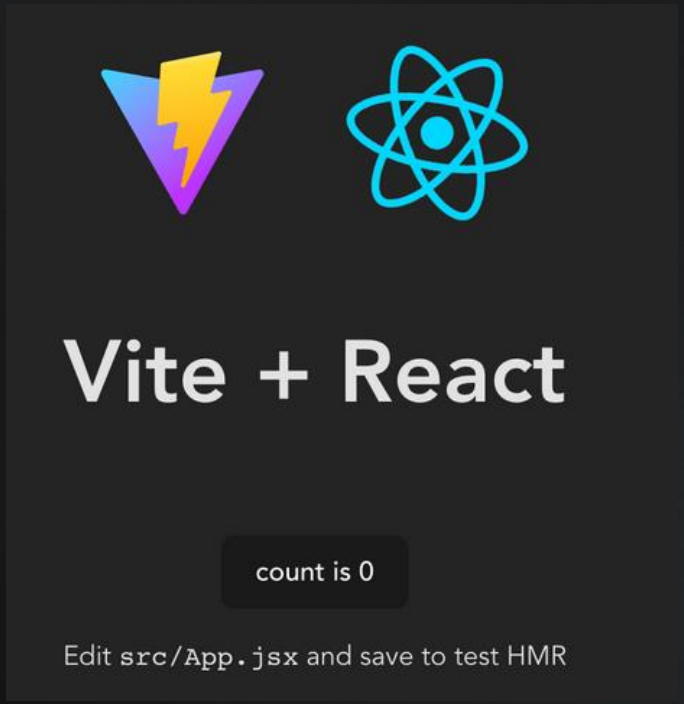
inicia o projeto(coloca no ar)

VITE + REACT

REACT

45697056

- Esta é a página inicial da nossa aplicação.



VITE + REACT

REACT

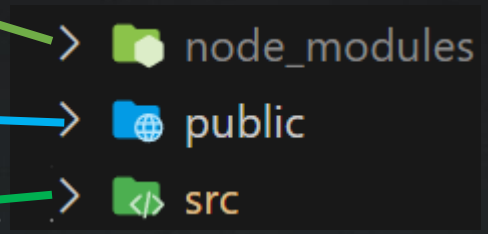
45697056

- Na pasta do nosso projeto temos 3 subpastas que foram criadas com o projeto:

Node Modules: são bibliotecas do Node que estão disponível para uso na aplicação

Public: contém os arquivos que são carregados no lado do cliente

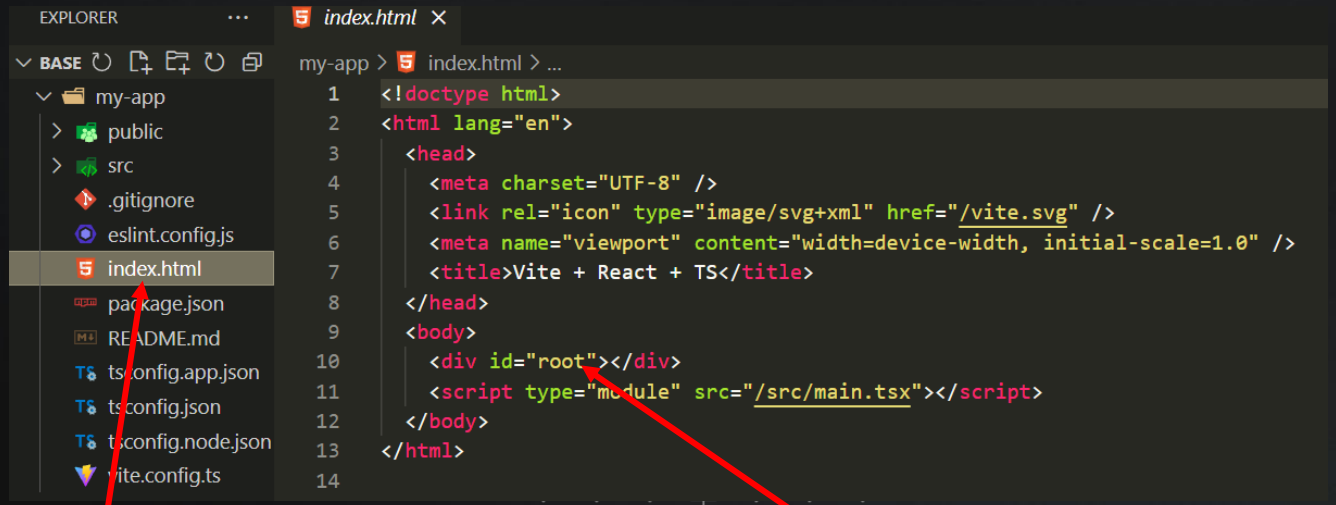
Src: contém os arquivos que estão do lado do servidor



REACT

45697056

- O React é carregado na página index.html, dentro da div com id “root” e é a partir dai que o javascript carrega todos os componentes da aplicação.



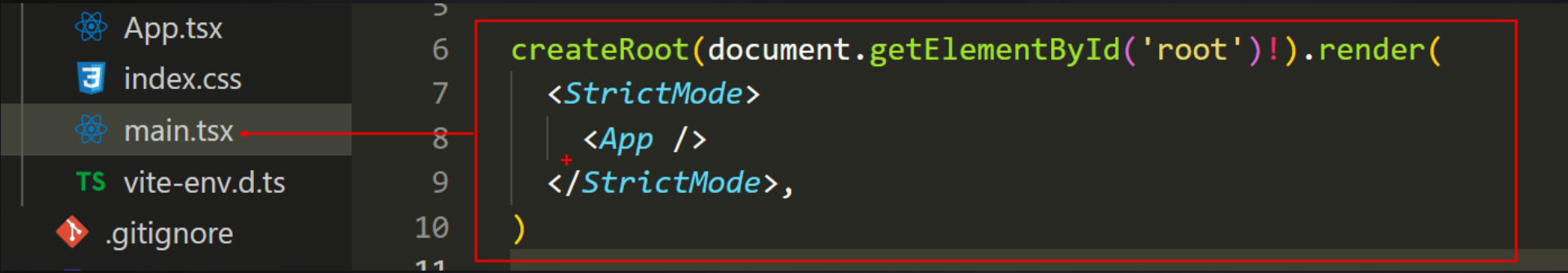
Arquivo index.html na pasta public

Div “root”, onde o javascript carrega toda a aplicação

REACT

45697056

- Já na pasta SRC, o ***main.tsx*** é o responsável por carregar toda a nossa aplicação na div de id “root” que está no index.html. Repare que ele está carregando o ***App.tsx*** nele.




```
createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

REACT

45697056

- O arquivo App.tsx é nosso arquivo que está dando origem ao componente principal da aplicação, repare que todos os componentes da tela estão sendo criados nele dentro de uma função que recebe o seu nome. Esta função retorna todos os elementos da tela que serão criados no html.
- Isso é possível por causa do jsx.

```jsx
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

VITE + REACT

REACT

45697056

Vamos limpar o conteúdo do arquivo App.tsx e vamos criar um novo conteúdo do zero.

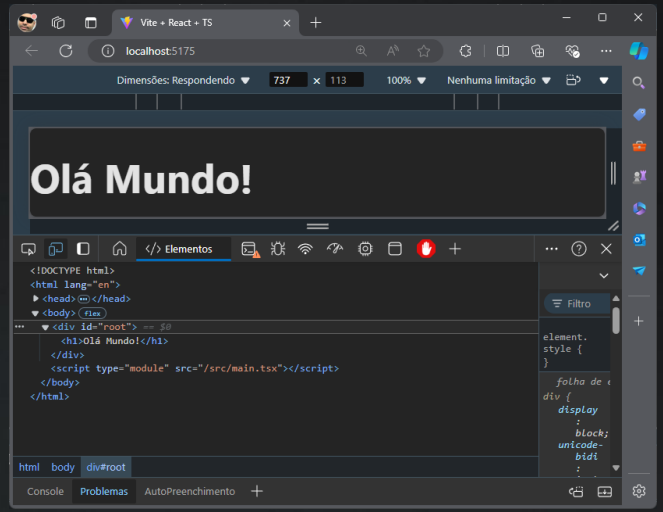
```tsx
function App() {
  return (
    <>
      <h1>Olá Mundo!</h1>
    </>
  );
}

export default App;
```

Função com o nome do arquivo.

A função deve retornar o conteúdo que será renderizado.

Instrução para chamar a função



VITE + REACT

REACT

45697056

Podemos ter quantos elementos for necessário, mas todos eles devem estar dentro de um elemento pai(elemento de bloco), caos contrários o compilador acusará um erro. Este elemento principal pode ser uma DIV por exemplo ou uma tag vazia simplesmente para marcar o conteúdo, como temos abaixo:

```
import React from 'react'

export default function App(){

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <a href="#">FIAP</a>
      <br/>
      <a href="#">GOOGLE</a>
    </>
  )
}
```

Podemos fazer a exportação da função na mesma linha

Tags self close precisam de barra de fechamento “obrigatório”

A tag vazia se chama FRAGMENT e não interfere no layout atual.

VITE + REACT

REACT

45697056

Toda a parte de código a ser processado que for usado no JSX, deve estar dentro de chaves duplas ou uma (expression "{ }"), para ser compilada.

```
import React from 'react'

export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
    </>
  )
}
```

Código javascript da função antes do return

Valores expressos pelas constantes entre chaves

EXPRESSION {...}

VITE + REACT

REACT

45697056

Para inserir elementos como imagens devemos importar antes de inserir no componente, lembrando sempre se usar as chaves:

Adicionando imagens

```
1  import React from 'react'
2  import Fiap1 from './img/fiap1.jpg'
3
4    export default function App(){
5
6      const aluno = 'Matheus Ramalho'
7      const curso = 'ADS'
8
9      return(
10       <>
11         <h1>TDS FIAP - RWD </h1>
12         <img src={Fiap1} width="250px"/>
13         <p>Alunos: {aluno}</p>
14         <p>Curso: {curso}</p>
15       </>
16     )
17   }
```

Import da imagem

Inserindo a imagem na tag img

OBS. Se a imagem estiver na pasta public, o mapeamento será do modo convencional

REACT

45697056

Como falamos a pouco, o React trabalha com componentes, então vamos criar alguns para ver como funciona. Dentro da pasta src crie uma pasta chamada componentes e dentro dela um arquivo chamado ListaAluno.js. Nela insira o seguinte código:

Criando componentes

```
import React from 'react'

export default function ListaAlunos(){

    return(
        <ul>
            <li>Huguinho</li>
            <li>Zézinho</li>
            <li>Luizinho</li>
        </ul>
    )
}
```

VITE + REACT

REACT

45697056

Agora para conseguirmos utilizar o arquivo ListaAlunos.js como um componente devemos importa-lo dentro de App.js e inserir ele no componente principal, conforme abaixo:

Importando componentes

```
import React from 'react'
import Fiap1 from './img/fiap1.jpg'
import Lista from './compomentes/ListaAlunos'

export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <img src={Fiap1} width="250px"/>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
      <Lista/>
    </>
  )
}
```

Importando o arquivo ListaAlunos.js

Inserindo no componente principal, criando uma tag com seu nome

Exercício

45697056

1. Crie uma nova aplicação chamada exercício;
2. Limpe o conteúdo do arquivo App.tsx
3. Crie um componente chamado Cabecalho.tsx e insira uma tag header com um h1 e um parágrafo;
4. Crie um componente chamado Carros.tsx e insira uma imagem de carro e uma lista com 5 modelos de carro.
5. Crie um componente chamado Parceiros.tsx e insira um h2 e 4 links.

VITE + REACT

REACT

45697056

Como comentamos no início os dados, se necessário, devem fluir do pai para seus filhos, este é o fluxo natural. Para conseguirmos passar estes dados vamos utilizar ‘props’. Ele é passado do pai para seus filhos como um objeto e dentro do filho podemos acessar seus valores.

Para enviar os valores para o filho devemos criar atributos em sua tag que está no componente pai. Vamos enviar os nomes dos alunos daquela lista que criamos a pouco:

PROPS

```
export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'
  const alunos = ['Luís','Alexandre','Allen']

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <img src={Fiap1} width="250px"/>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
      <Lista alunos={alunos} />
    </>
  )
}
```

Dados contidos no pai

Enviando os dados para o filho

REACT

45697056

Para o filho poder utilizar os dados enviados pelo pai devemos colocar o parâmetro props na função ListaAlunos e, como um objeto, chamar seus valores.

```jsx
import React from 'react'

export default function ListaAlunos(props){

    return(
        <ul>
            <li>{props.alunos[0]}</li>
            <li>{props.alunos[1]}</li>
            <li>{props.alunos[2]}</li>
        </ul>
    )
}
```

Parâmetro props

Utilização dos valores enviados pelo pai

VITE + REACT

REACT

45697056

Assim como passamos dados do pai para o filho, também podemos passar funções, vamos enviar uma função para ver as diferenças. Crie a função no componente pai conforme abaixo:

```
export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'
  const alunos = ['Luís','Alexandre','Allen']

  const novoAluno = () => 'Fernanda'

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <img src={Fiap1} width="250px"/>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
      <Lista alunos={alunos} maisAluno={novoAluno}/>
    </>
  )
}
```

Criação de uma função

Enviando a função para o filho

REACT

45697056

Para a utilização da função herdada do pai só não devemos esquecer dos parênteses, o resto é igual as variáveis:

```
import React from 'react'

export default function ListaAlunos(props){

    return(
        <ul>
            <li>{props.alunos[0]}</li>
            <li>{props.alunos[1]}</li>
            <li>{props.alunos[2]}</li>
            <li>{props.maisAluno()}</li>
        </ul>
    )
}
```

Utilizando a função

Exercício

45697056

1. Volte para a aplicação exercício que criamos a pouco;
2. Passe os modelos dos carros do componente pai para o componente filho.
3. Crie uma função multiplicando dois números no componente pai e passe para o componente Parceiros, executando em uma span logo abaixo do h2;

DUVIDAS

Prof. Alexandre Carlos
Prof. Luís Carlos
Prof. Wellington Cidade