Biblioteca React

UFAM

Capacitação em Desenvolvimento Web Full Stack

Douglas Silva de Melo

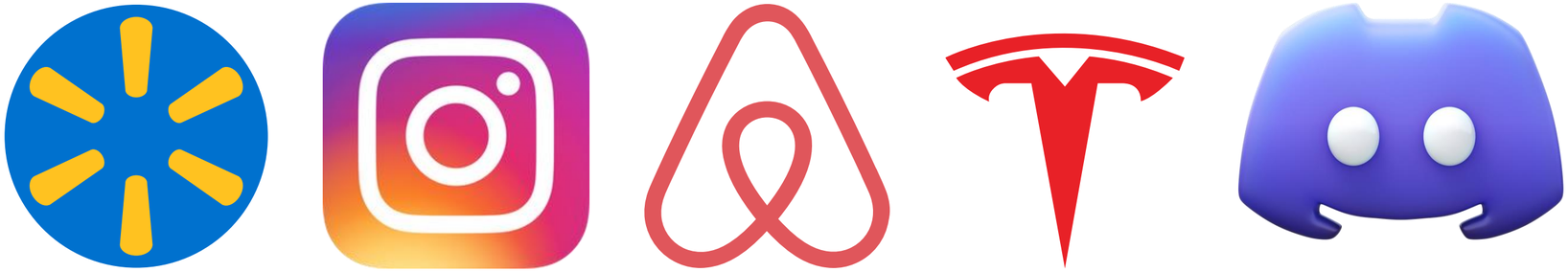
O QUE É O REACT?

- *React é uma biblioteca de código aberto.*
  - *HTML*
  - *CSS*
  - *Linguagem Script (Javascript e Typescript)*
- *É amplamente utilizada para criar interfaces de usuário (UI) interativas em aplicativos web.*
- *Utiliza o conceito de componentes, que são blocos de construção reutilizáveis para a interface do usuário.*
- *Introduz o uso do JSX (JavaScript XML) para criar elementos de interface no JavaScript.*
- *É conhecida por sua capacidade de atualização eficiente do DOM.*
- *É amplamente adotada pela indústria e é uma das principais escolhas para o desenvolvimento front-end de aplicativos web modernos.*

HISTÓRIA

- *Desenvolvido pelo Facebook em 2013 para melhorar o desempenho de suas aplicações web.*
- *Lançado como código-fonte aberto em maio de 2013.*
- *Introduziu o conceito de Virtual DOM para otimizar atualizações do DOM.*
- *Expandido para o desenvolvimento de aplicativos móveis com o React Native em 2015.*

QUEM USA O REACT?



POR QUE USAR REACT?

- **Componentização:** *Divide a interface do usuário em componentes reutilizáveis.*
- **Reatividade**: *Atualiza apenas partes relevantes da interface quando os dados mudam.*
- **Biblioteca, não framework:** *Flexibilidade na escolha de outras ferramentas.*
- **Ampla comunidade e ecossistema:** *Recursos e suporte abundantes.*
- **React Native**: *Desenvolvimento de aplicativos móveis nativos.*
- **Documentação detalhada:** *Facilita a aprendizagem e uso.*
- **Empresas de renome o utilizam:** *Validado pela indústria.*
- **Crescimento contínuo:** *Novos recursos e melhorias frequentes.*
- ***Possibilita o desenvolvimento de interfaces rapidamente***

FÁCIL GERENCIAMENTO DE INTERFACE

Javascript + HTML Puro

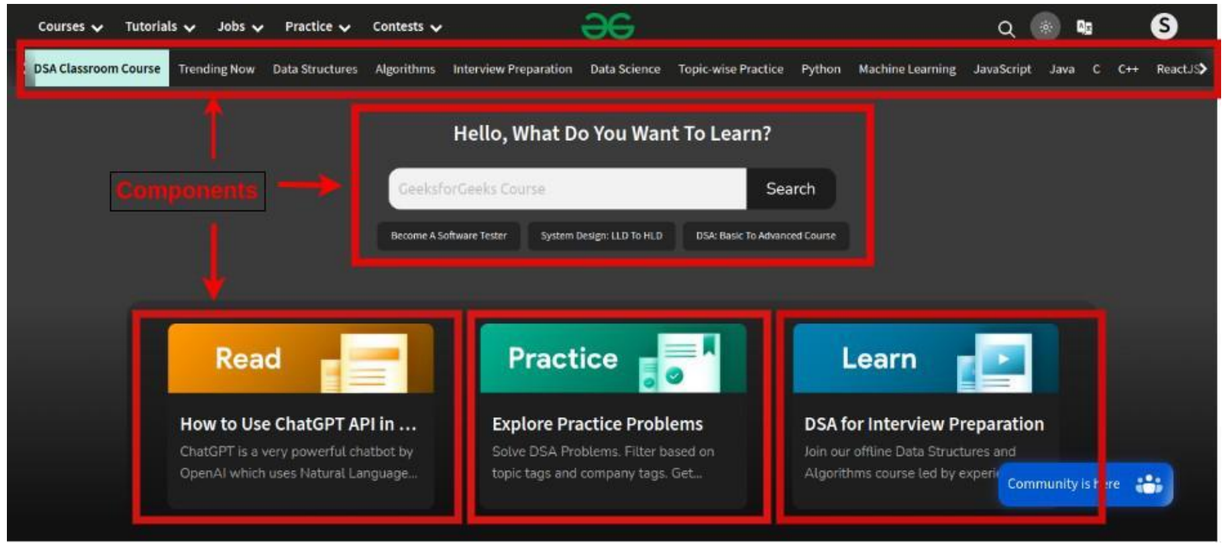
```html
<div>
  <h3>Número aleatório:</h3>
  <h1 id="numeroAleatorio"></h1>

  <button onClick="gerarNumero()">
    Gerar número
  </button>
</div>

<script>
  function gerarNumero() {
    const novoNumero = Math.floor(Math.random() * (100-1) + 1);

    // Seleciona o elemento
    const elementoH1 = document.getElementById("numeroAleatorio");

    // Altera o valor
    elementoH1.innerText = novoNumero;
  }
</script>
```

React

```jsx
import React, { useState } from 'react';

export default function PaginaInicial() {
  const [ numeroAleatorio, setNumeroAleatorio ] = useState(1);

  function gerarNumero() {
    const novoNumero = Math.floor(Math.random() * (100-1) + 1);

    // Apenas altera a variável
    setNumeroAleatorio(novoNumero);
  }

  return(
    <div className="conteudo-centralizado">
      <h3>Número aleatório:</h3>
      <h1>{numeroAleatorio}</h1>

      <button onClick={gerarNumero}>
        Gerar número
      </button>
    </div>
  );
}
```

ESTRUTURA (FLUXO) - COMPONENTS

- *Os desenvolvedores escreviam mais de milhares de linhas de código para desenvolver um aplicativo de página única, pois não conseguiam reaproveitar código.*
- *Todo o aplicativo é dividido em um pequeno grupo lógico de código, conhecido como componentes.*



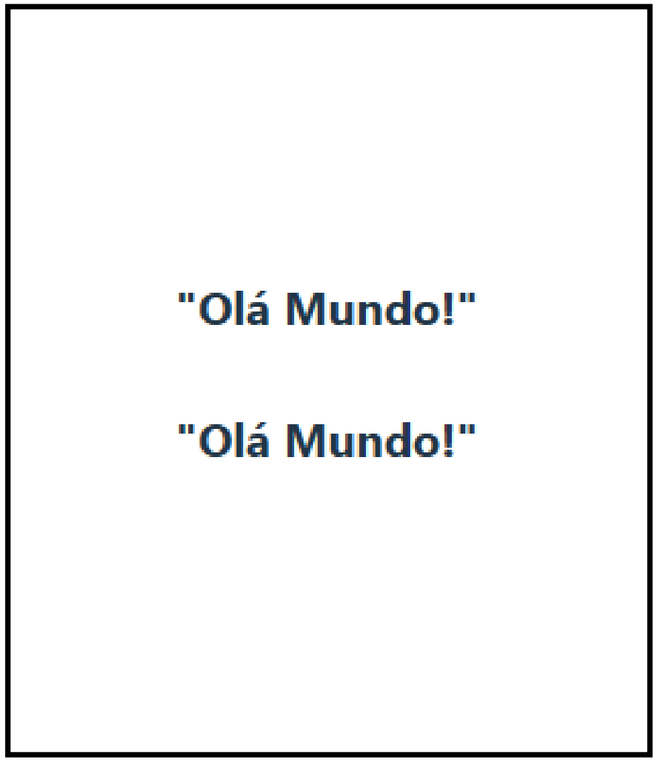
ESTRUTURA (FLUXO) - COMPONENTS

Componente

```jsx
import React from "react";

function MeuComponente() {
    return <h4>"Olá Mundo!"</h4>;
}

export default MeuComponente;
```

Componente Page

```jsx
import MeuComponente from "./components/cp1";

export default function App() {
  return (
    <div>
      <MeuComponente />
      <MeuComponente />
    </div>
  );
}
```



ESTRUTURA (FLUXO) - PROPS

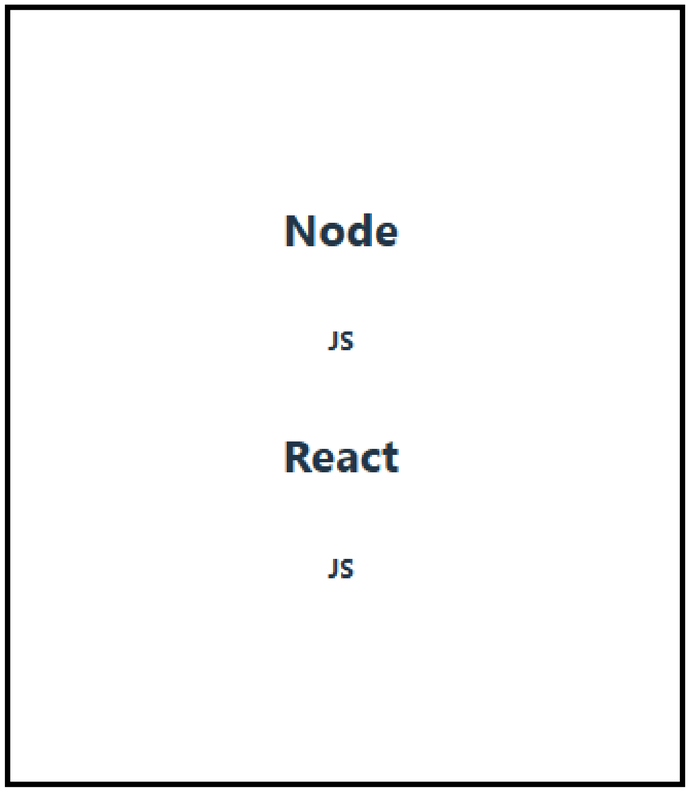
```
<input type="email" placeholder="Digite seu e-mail" />
```

Componente

```tsx
interface PropsMeuComponente {
  title: string;
  subtitle: string;
}

function MeuComponente(props: PropsMeuComponente) {
  return (
    <div>
      <h3>{props.title}</h3>
      <h6>{props.subtitle}</h6>
    </div>
  );
}

export default MeuComponente;
```

Componente Page

```tsx
import MeuComponente from "./components/cp1";

export default function App() {
  return (
    <div>
      <MeuComponente title="Node" subtitle="JS" />
      <MeuComponente title="React" subtitle="JS" />
    </div>
  );
}
```



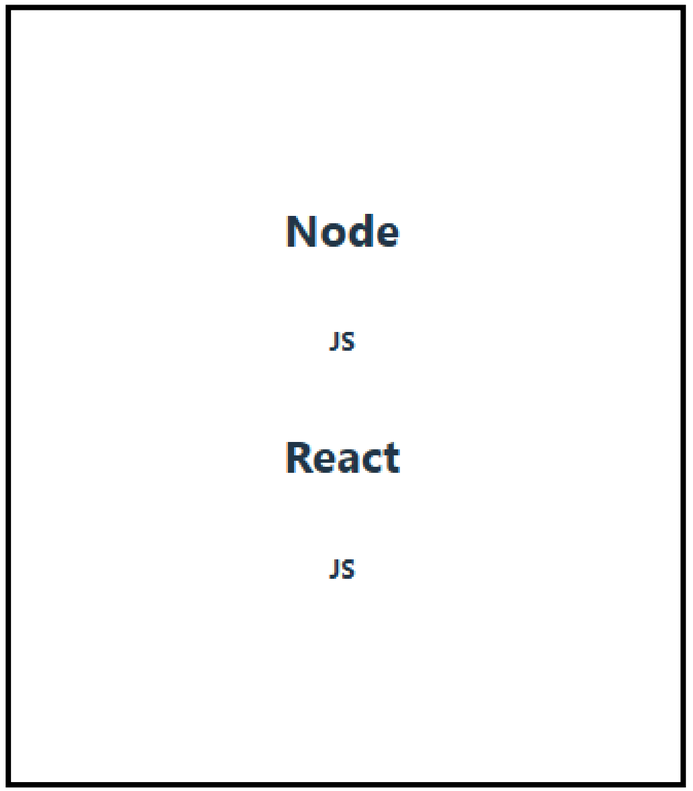
ESTRUTURA (FLUXO) - PROPS

Componente

```tsx
function MeuComponente({
  title,
  subtitle,
}: {
  title: string;
  subtitle: string;
}) {
  return (
    <div>
      <h3>{title}</h3>
      <h6>{subtitle}</h6>
    </div>
  );
}

export default MeuComponente;
```

Componente Page

```tsx
import MeuComponente from "./components/cp1";

export default function App() {
  return (
    <div>
      <MeuComponente title="Node" subtitle="JS" />
      <MeuComponente title="React" subtitle="JS" />
    </div>
  );
}
```



ESTRUTURA (FLUXO) - PROPS

Componente

```tsx
import { ReactNode } from "react";

interface PropsMeuComponente {
  title: string;
  children: ReactNode;
}

function MeuComponente(props: PropsMeuComponente) {
  return (
    <div>
      <h3>{props.title}</h3>
      {props.children}
    </div>
  );
}

export default MeuComponente;
```

```tsx
<div>
  <h1>Hello</h1>
  <h3>World</h3>
  <input />
</div>
```

Componente Page

```tsx
import MeuComponente from "./components/cp1";

export default function App() {
  return (
    <div>
      <MeuComponente title="Node">
        <ul>
          <li>Express</li>
          <li>API</li>
        </ul>
      </MeuComponente>

      <MeuComponente title="React">
        <ul>
          <li>Componentes</li>
          <li>Props</li>
        </ul>
      </MeuComponente>
    </div>
  );
}
```

Node

- Express
- API

React

- Componentes
- Props

ESTRUTURA (FLUXO) - STATE

- *Estados são como variáveis que estão totalmente associadas a interface.*
- *Quando alteramos o estado da variável, alteramos a interface.*
- *Podemos armazenas nesses estados/variáveis informações de input do usuários e demais informações.*
- *É uma parte crucial do desenvolvimento React.*
- *Utilização do **useState.***
- *Com useState, podemos declarar variáveis de estado, seu valor inicial e uma função que altera o seu valor.*

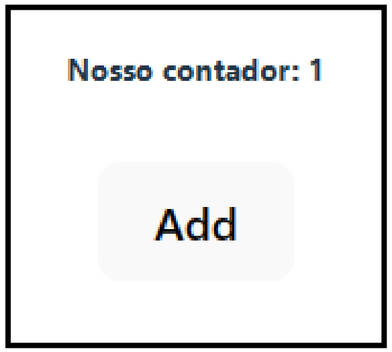
ESTRUTURA (FLUXO) - STATE

```jsx
import { useState } from 'react'
import './App.css'

function App() {
  const [count, SetCount] = useState(1);


  function Increment() {
    const sum = count + 1;
    SetCount(sum);
  }

  return (
    <div className="App">
      <h1>Nosso contador: {count}</h1>

      <button onClick={() => {
        Increment(); // Função
      }}>
        Add
      </button>
    </div>
  )
}

export default App
```

Importa o o useState()

define a função para atualizar o estado

Inicializa o useState()




```jsx
import { useState } from 'react'
import './App.css'

function App() {
  const [count, SetCount] = useState(1)


  function Increment() {
    //! TRECHO DE CÓDIGO ERRADO
    count = count + 1;
  }

  return (
    <div className="App">
      <h1>Nosso contador: {count}</h1>

      <button onClick={() => {
        Increment(); // Função
      }}>
        Add
      </button>
    </div>
  )
}

export default App
```

CRIANDO NOSSO PRIMEIRO PROJETO COM VITE

- *Inicialização rápida devido à compilação em tempo real.*
- *Ideal para desenvolvimento rápido e projetos menores.*
- *Evita configurações excessivamente complexas.*
- *Eficiência e agilidade são focos principais.*
- *Não é oficialmente mantido pelo React, mas suporta o React.*

Criar do Zero

```
npm create vite@latest
```

```
√ Project name: ... projeto-teste
√ Select a framework: » React
√ Select a variant: » TypeScript
```

```
cd frontend
npm install
npm run dev
```

Baixar

https://drive.google.com/file/d/1qsoucUS5-BH5C0HIFWffai5-5UeF4iA3/view?usp=sharing

Criar projeto configurado com Docker

```
-> Unzip
-> Entrar na pasta do projeto (Raiz do projeto)
cp .env.example .env
cp frontend/.env.example frontend/.env
cp backend/.env.example backend/.env
cd frontend && npm install && cd ..
cd backend && npm install && cd ..
docker compose up
docker exec -it loja_backend npx prisma migrate dev
docker exec -it loja_backend npx prisma db seed
```

Se estiver usando o sistema operacional Windows, utilize o comando 'copy' em vez de 'cp' e `\` ao invés de `/`