

WebMAC

Com FastAPI e HTMX

Aula 9

Integração de Front e Back-end

Semanas do Curso

1. HTML, CSS e Responsividade

2. Javascript

3. FastAPI

4. SQL

5. HTMX e UI

6. Docker

Objetivos das semanas

1. Criar páginas web
2. Tornar páginas dinâmicas
3. Criar um back-end
4. Gerenciar dados
- 5. Criar interfaces modernas**
6. Rodar com containers

Canal do Codelab no Youtube



CodeLab

@CodeLabBR · 1,47 mil inscritos · 138 vídeos

O CodeLab é um grupo de extensão inter-universitário que tem como objetivo estimular a ...mais

uclab.xyz/site e mais 5 links

Inscrever-se

Início **Vídeos** Shorts Ao vivo Playlists

Mais recentes

Em alta

Mais antigos



Deployment - Aula5 (Webdev 2025)

56 visualizações · há 3 meses



HTMX - Aula4 (Webdev 2025)

65 visualizações · há 3 meses



SQL e Django Models - Aula3 (Webdev 2025)

78 visualizações · há 4 meses



Django - Aula2 (Webdev 2025)

242 visualizações · há 5 meses



JavaScript - Aula1 (Webdev 2025)

234 visualizações · há 5 meses



HTML e CSS - Aula0 (Webdev 2025)

156 visualizações · há 5 meses



Promises - Aula 6 - WEBDEV 23

103 visualizações · há 2 anos



Objetos e Arrays - Aula 5 - WEBDEV 23

89 visualizações · há 2 anos

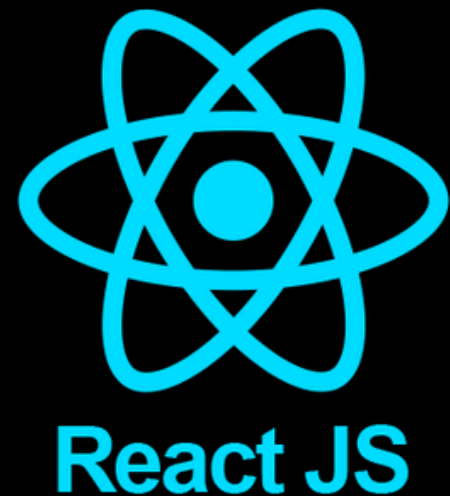
Resumo da Aula:

Tópicos:

- Frameworks de Front-end;
- Integração de BD com front usando HTMX;
- UI e Animações;

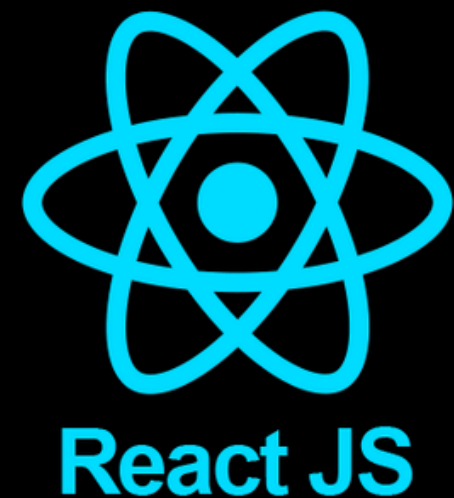
Frameworks de Front-end

- Voltados para o cliente ao invés do servidor
- Atualização da página em tempo real
- Client-side Rendering
- Muito JS para que o próprio browser construa a página



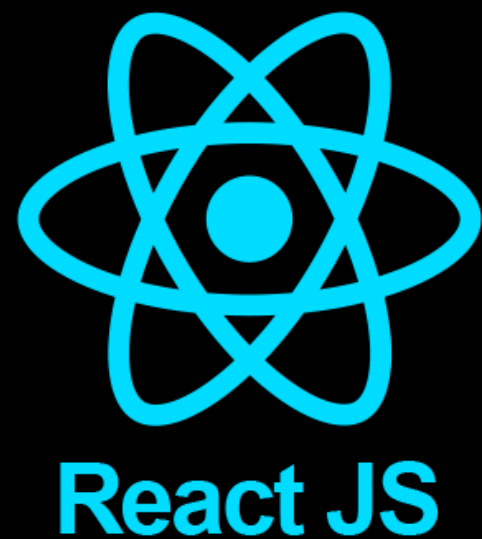
Frameworks de Front-end

Os benefícios de Frameworks de Front-end se assemelham aos de SPAs quando comparados aos de Back-end

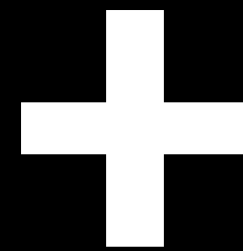


Frameworks de Front-end

É comum o uso de ambos os tipos de frameworks no mesmo projeto!



(UI)



(Dados)

Frameworks de Front-end

**Então, por que não ensinamos
nenhum Framework Front-end?**

Frameworks de Front-end

- Exemplo da aula passada em REACT

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

const ExemploRequisicao = () => {
  const [conteudo, setConteudo] = useState(null);
  const [carregando, setCarregando] = useState(false);

  const buscarDados = async (idPagina) => {
    setCarregando(true);
    try {
      const response = await fetch(`https://localhost:8000/home/${idPagina}`);
      const data = await response.json();
      setConteudo(data);
    } catch (error) {
      console.error("Erro ao buscar dados:", error);
      setConteudo({ title: "Erro", body: "Não foi possível carregar a página." });
    } finally {
      setCarregando(false);
    }
  };

  useEffect(() => {
    buscarDados('pagina1');
  }, []);
};
```

```
return (
  <div>
    <h1>Meu Site</h1>
    <div>
      <button onClick={() => buscarDados('pagina1')}>Página 1</button>
      <button onClick={() => buscarDados('pagina2')}>Página 2</button>
    </div>
    <hr/>

    {carregando ? (
      <p>Carregando...</p>
    ) : conteudo ? (
      <div>
        <h2>{conteudo.title}</h2>
        <p>{conteudo.body}</p>
      </div>
    ) : (
      <p>Nenhum conteúdo disponível.</p>
    )}
  </div>
);

export default ExemploRequisicao;
```

Frameworks de Front-end

- Exemplo da aula passada com HTMX

```
<DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Meu Site</title>
    <style>...</style>
    <script src="HTMX"></script> <!-- Import do HTMX -->
  </head>
  <body>
    <h1>Meu Site</h1>
    <p>Bem vindo ao meu site!</p>
    <nav>
      <button hx-get="/home/pagina1"
              hx-target="main">
        Página 1
      </button>
      <button hx-get="/home/pagina2"
              hx-target="main">
        Página 2
      </button>
    </nav>
    <hr>
    <main>
      <h2>Pagina 1</h2>
      <p>Conteúdo da página 1</p>
    </main>
  </body>
</html>
```

HTMX

Alguns atributos

- `hx-get`
- `hx-post`
- `hx-put`
- `hx-delete`
- `hx-target`
- `hx-select`
- `hx-trigger`
- `hx-swap`
- `hx-confirm`
- `hx-push-url`
- `hx-indicator`
- ...