

# Revisão de conceitos WEB

# Como Funcionam os Sistemas Web?

O Que é um Sistema Web?



## **Definição:**

Um sistema web é uma aplicação acessada via navegador, que pode ter interação com banco de dados, APIs e diversas tecnologias no frontend e backend.



## **Exemplos:**

- Sites de e-commerce
- Aplicações bancárias
- Sistemas de gestão



# Arquitetura Cliente-Servidor



**Cliente:** Navegador que solicita dados.




**Servidor:** Processa requisições e retorna informações.



**Comunicação:** Via HTTP/HTTPS.



# Frontend vs Backend

 **Frontend:** O que o usuário vê e interage.

 **Backend:** Processa regras de negócio e gerencia o banco de dados.


 **Exemplo:**

- O frontend exibe um formulário de login.
- O backend verifica se o usuário e senha estão corretos.



# Tecnologias do Frontend - HTML, CSS e JavaScript

 **HTML:** Estrutura da página.

 **CSS:** Estilização e aparência.

 **JavaScript:** Interatividade e dinamismo.



# Frameworks e Bibliotecas do Frontend

## O que são frameworks?

Conjuntos de ferramentas para facilitar o desenvolvimento.

## Principais frameworks e bibliotecas:

- ♦ **React:** Biblioteca JavaScript para interfaces interativas.
- ♦ **Angular:** Framework completo para aplicações web.
- ♦ **Vue.js:** Alternativa leve para criação de interfaces.

## Vantagens:

- Componentização
- Melhor performance
- Facilidade de manutenção



# Tecnologias do Backend - O Que São APIs?



## **API (Application Programming Interface)**

Conjunto de regras que permitem a comunicação entre sistemas.



### **Tipos de APIs:**

- **REST:** Baseada em HTTP (GET, POST, PUT, DELETE).
- **GraphQL:** Consultas mais flexíveis para obter dados.



### **Exemplo:**

Um site de clima usa uma API para buscar informações meteorológicas.



# Frameworks Backend

## Principais tecnologias:

- Node.js (JavaScript)
- Django (Python)
- Spring Boot (Java)

## Benefícios:

- Agilidade no desenvolvimento
- Segurança
- Escalabilidade







# Estilização e Responsividade - CSS e Frameworks de Estilo

## CSS Puro vs Frameworks

- **CSS puro:** Escreve estilos manualmente.
- **Frameworks:** Facilitam a estilização com classes prontas.

## Principais frameworks:

 **Bootstrap** – Componentes prontos e responsivos.

 **Tailwind CSS** – Classes utilitárias para maior controle.




# Estilização e Responsividade - CSS e Frameworks de Estilo

## CSS Puro vs Frameworks

- **CSS puro:** Escreve estilos manualmente.
- **Frameworks:** Facilitam a estilização com classes prontas.

## Principais frameworks:

 **Bootstrap** – Componentes prontos e responsivos.

 **Tailwind CSS** – Classes utilitárias para maior controle.



# O que é APIs

API (**A**pplication **P**rogramming **I**nterface) é um conjunto de regras que permite que aplicações se comuniquem entre si. Em outras palavras, uma **API permite que um sistema envie e receba dados de outro sistema**.

## Exemplo:

- Quando um app de clima pede os dados de previsão do tempo, ele se comunica com uma **API de previsão do tempo** para buscar essas informações.
- Quando um site de e-commerce exibe produtos, ele pode buscar esses produtos de uma **API de produtos**.



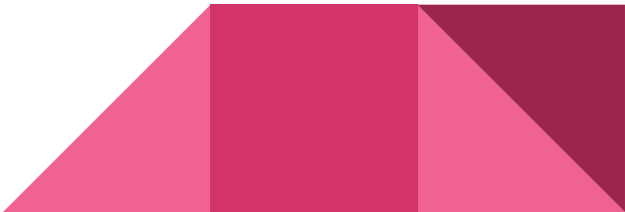
# O que é JSON?

JSON (**JavaScript Object Notation**) é um formato de troca de dados leve e fácil de ler. Ele é muito usado para enviar e receber dados entre servidores e aplicações.

## 📌 Exemplo de JSON:

Estrutura **chave-valor** ("**chave**": "**valor**")

```
{  
  "nome": "João",  
  "idade": 25,  
  "cidade": "São Paulo"  
}
```



# O que é JSON?

## Exemplo

EndPoint:

<https://jsonplaceholder.typicode.com/users>



# Decode em Python

```
import requests

# URL da API JSONPlaceholder

url = "https://jsonplaceholder.typicode.com/users"

# Fazendo a requisição GET para buscar os usuários

response = requests.get(url)

# Verificando se a requisição foi bem-sucedida (status code 200)

if response.status_code == 200:

    users = response.json() # Convertendo a resposta para JSON


    # Exibindo os usuários

    for user in users:

        print(f"Nome: {user['name']}, Email: {user['email']}")

else:

    print("Erro ao buscar dados:", response.status_code)
```



# Criando nosso primeiro projeto

```
npm create vite@latest my-app --template react  
cd my-app  
npm install  
npm run dev
```



# Estrutura básica de um projeto React + Vite

```
my-app/
├── node_modules/      # Dependências instaladas pelo npm
├── public/            # Arquivos estáticos (favicon, imagens, etc.)
│   ├── vite.svg      # Logo do Vite (pode ser removido)
├── src/               # Código-fonte do projeto
│   ├── assets/       # Arquivos estáticos usados no código (opcional)
│   ├── App.jsx       # Componente principal do React
│   ├── main.jsx      # Arquivo de entrada do React
│   └── index.css      # Estilos globais do projeto
├── .gitignore         # Arquivos que serão ignorados pelo Git
├── index.html         # Página principal do projeto
├── package.json       # Lista de dependências e scripts do projeto
├── vite.config.js     # Configurações do Vite
└── README.md         # Documentação do projeto
```



# O que é o Vite?

O **Vite** é uma ferramenta moderna para construção de aplicações **front-end**. Ele foi criado para ser **mais rápido** e **eficiente** do que ferramentas tradicionais como **Create React App (CRA)** e **Webpack**.

📌 Principais vantagens do Vite:

- ✓ Inicialização ultra rápida ⚡
- ✓ Hot Module Replacement (HMR) eficiente ↺
- ✓ Build otimizado com ES Modules 📦
- ✓ Suporte nativo a TypeScript, JSX e CSS moderno 🛠
- ✓ Compatível com React, Vue, Svelte e outros frameworks