



Universidade Federal  
de Campina Grande

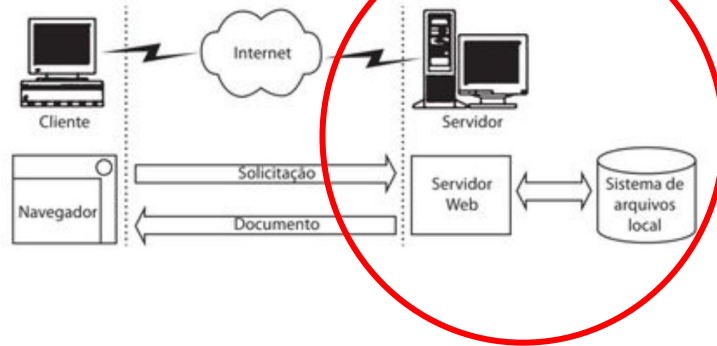
# Aplicações WEB - Frontend

José Glauber UFCG 2024.2

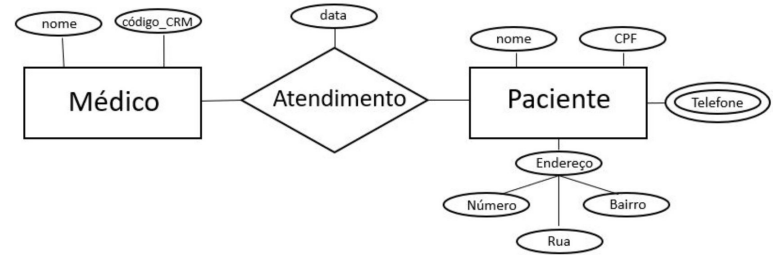
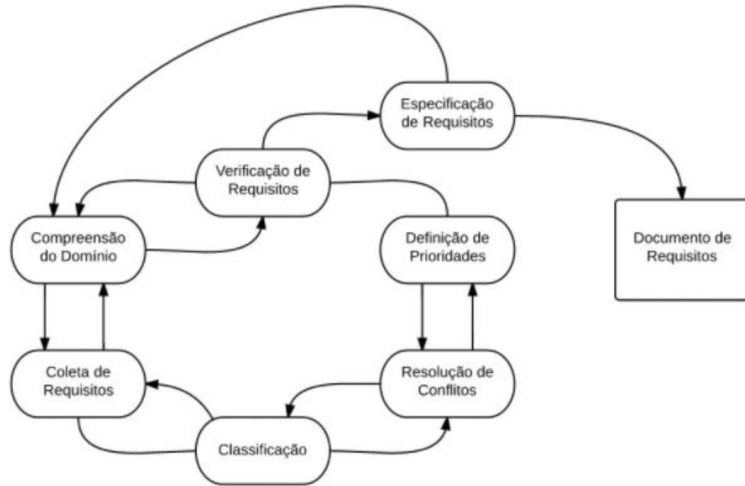
# Relembrando..

Protocolo      Domínio      Caminho do arquivo      Arquivo

`http://www.grupoa.com.br/tekne/livrodesenvolv2/material.html`



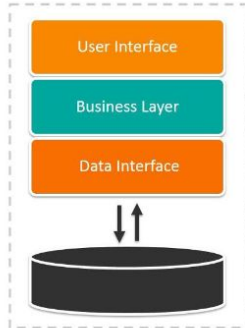
# Relembrando..



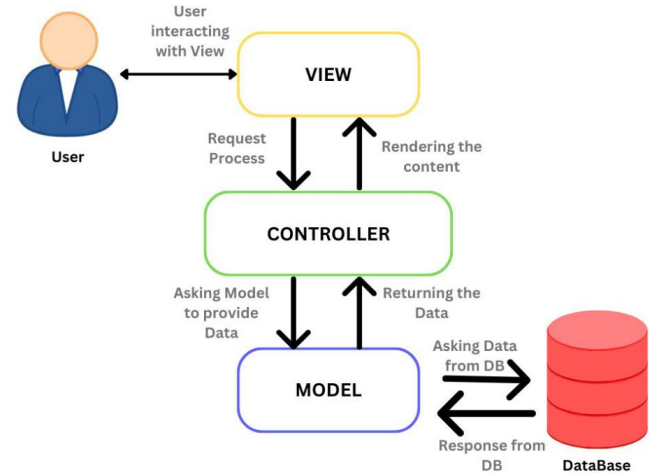
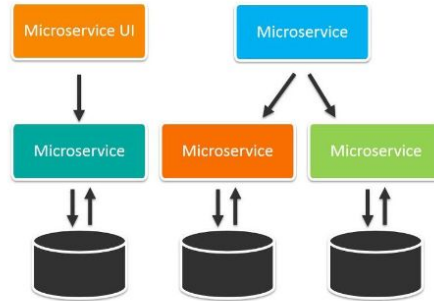
Como essas atividades estão relacionadas?

# Relembrando..

**Monolithic Architecture**



**Microservices Architecture**



# Até então evoluímos..

- Nos requisitos do sistema
- Na modelagem e relacionamentos das nossas entidades
- Na documentação da API
- Na modelagem e arquitetura da nossa aplicação

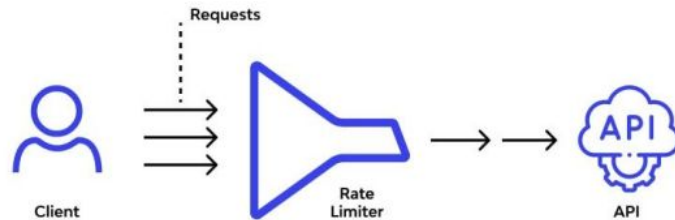
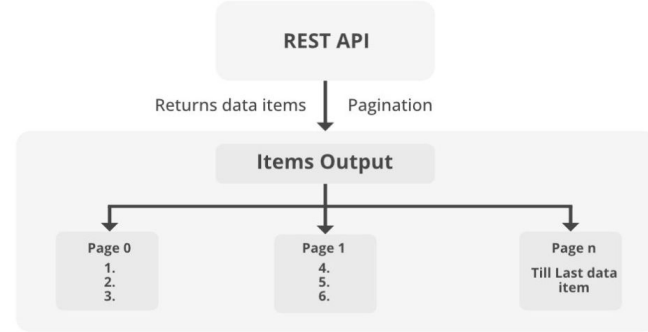
# 1ª entrega

Documentação + CRUDs + Banco de dados



mensagens HTTP, tratamento de erros, lógica de negócio...

# Funcionalidades avançadas API



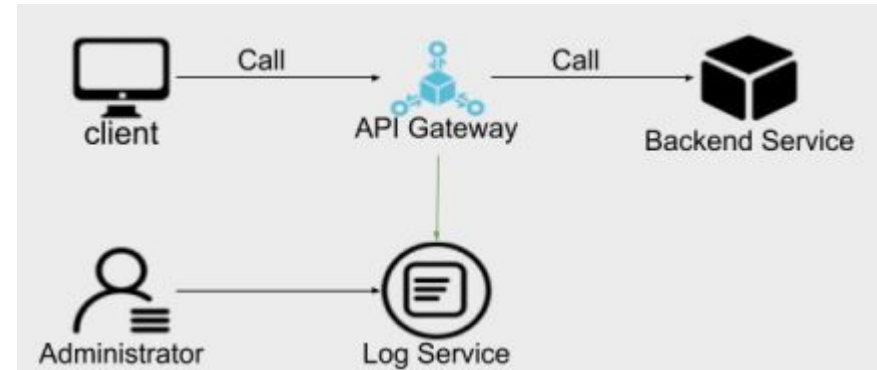
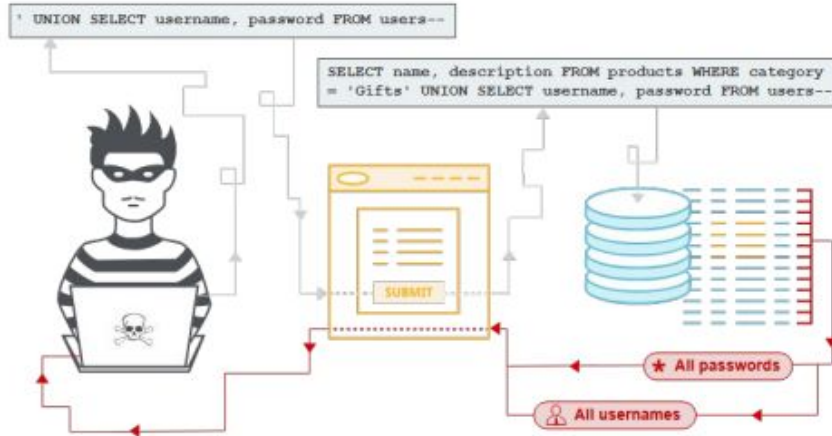
## CRIPTOGRAFIA



## HASH



# Funcionalidades avançadas API



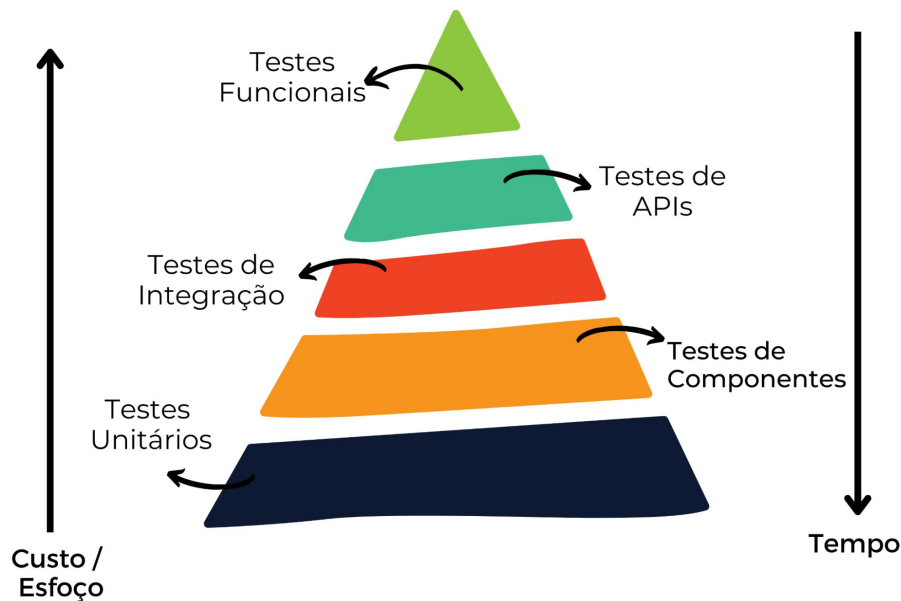


# Testando

Softwares **precisam** ser testados!

Identifique a complexidade da sua API e decida o tipo de teste a ser realizado (unidade ou API)

## Pirâmide de Testes



## 2ª entrega

API mais segura + boas práticas + Testes

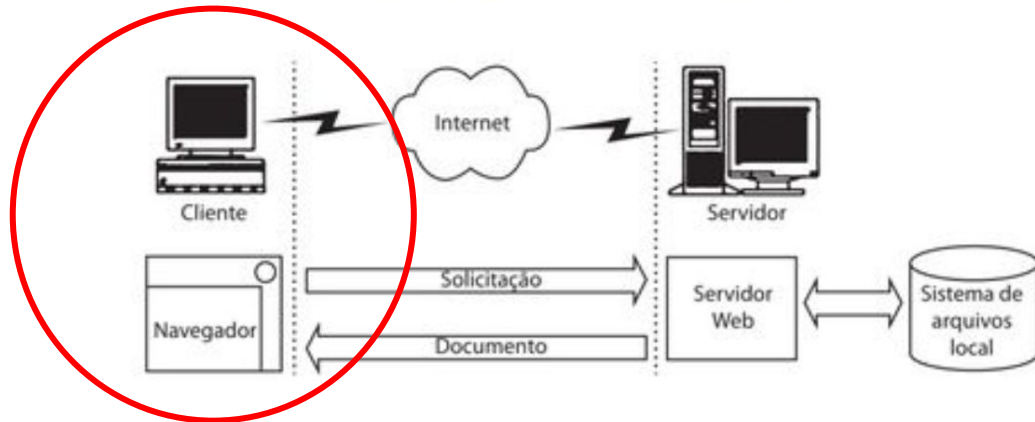


autenticação, tratamento de mensagens de erros,  
tratamento de dados, validações e testes.

# Frontend

Protocolo      Domínio      Caminho do arquivo      Arquivo

<http://www.grupoa.com.br/tekne/livrodesenvolv2/material.html>



# O que é?

Parte visual da nossa aplicação.



## Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation



## Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

É muito importante a experiência do usuário

frontend **!= design**

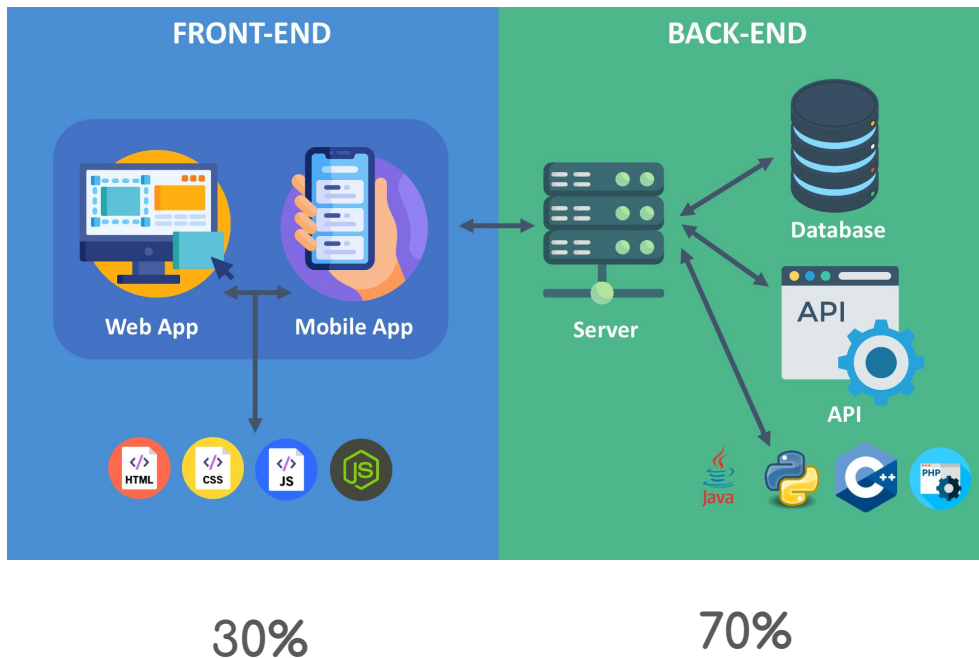
# Qual a importância do frontend?

O papel principal é facilitar a usabilidade e garantir que a ferramenta realmente funcione conforme o esperado.

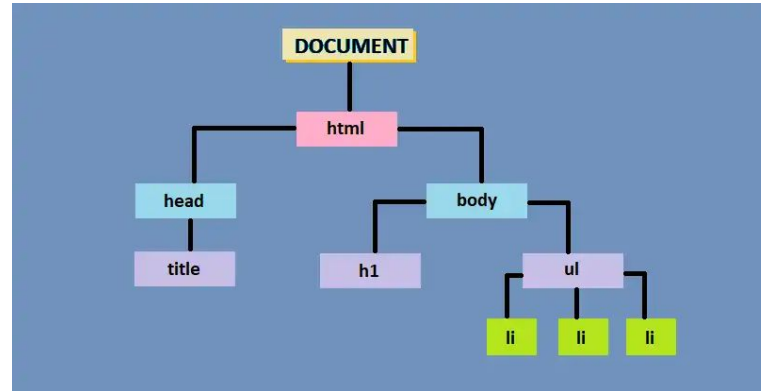
Um site com erros constantes ou lentidão torna a navegação quase impossível, o que gera **perda de tráfego**.

**API perde acesso e usabilidade.**

# O que desejamos..



# Iniciando...



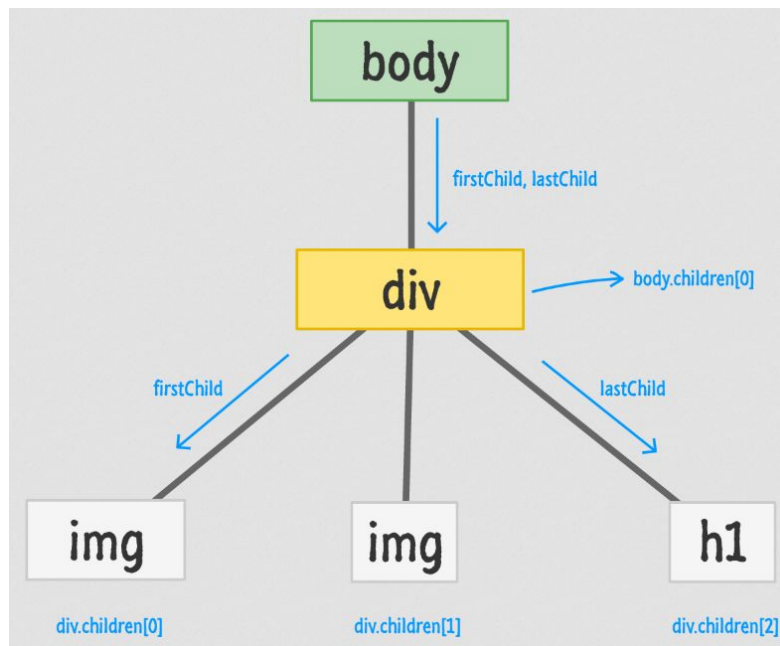
# Iniciando...





# Manipulação de elementos DOM

Document Object Model - É uma interface de programação que permite aos scripts (como o JavaScript) interagir e manipular os elementos de uma página HTML.



# Exemplos

## Selecionar Elementos no DOM

```
// Selecionar um elemento com id "meuTitulo"  
const titulo = document.getElementById('meuTitulo');  
console.log(titulo); // Mostra o elemento no console
```

```
// Selecionar todos os elementos <li>  
const itens = document.querySelectorAll('li');  
console.log(itens); // Mostra NodeList com todos os <li>
```

# Exemplos

## Manipulação de eventos

```
const botao = document.getElementById('meuBotao');  
botao.addEventListener('click', () => {  
  alert('Botão clicado!');  
});
```

```
const input = document.querySelector('input');  
input.addEventListener('input', (event) => {  
  console.log('Novo valor: ', event.target.value);  
});
```

# Exemplos

## Navegar entre os elementos

```
const filho = document.querySelector('.filho');  
const pai = filho.parentElement; // Acessa o elemento pai  
console.log(pai);
```

```
const item2 = document.querySelector('.item');  
const proximoItem = item2.nextElementSibling; // Próximo irmão  
const itemAnterior = item2.previousElementSibling; // Irmão anterior
```

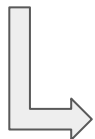
# Frameworks Frontend

Os frameworks automatizam, otimizam e tornam mais **eficientes** as interações com o DOM, permitindo que os desenvolvedores se concentrem mais na **lógica da aplicação** do que nas tarefas de manipulação manual do DOM.



# Manipulação DOM React

Virtual DOM: representação em memória do DOM real.



Trabalha com uma árvore de componentes que descrevem como a interface deve se parecer. O Virtual DOM é mantido atualizado com as alterações no estado da aplicação e, quando necessário, atualiza apenas as partes do DOM real que mudaram.

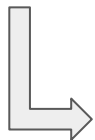
```
import React, { useState } from 'react';

function App() {
  const [contador, setContador] = useState(0);

  return (
    <div>
      <h1>Contador: {contador}</h1>
      <button onClick={() => setContador(contador + 1)}>
        Incrementar
      </button>
    </div>
  );
}
```

# Manipulação DOM Vue

Data binding: por meio de data binding e um sistema reativo.



É vinculado o estado dos seus componentes diretamente aos elementos do DOM e, assim como no React, o framework cuida da atualização eficiente do DOM sempre que o estado muda.

```
<div id="app">
  <p>Contador: {{ contador }}</p>
  <button @click="incrementar">Incrementar</button>
</div>

<script>
  new Vue({
    el: '#app',
    data: {
      contador: 0
    },
    methods: {
      incrementar() {
        this.contador++;
      }
    }
  });
</script>
```

# Por que Frameworks Automatizam a Manipulação do DOM?

Melhor performance

Código mais limpo

Sincronização automática de UI e Dados

Reatividade e responsividade



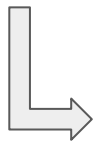
# Frontend projeto

Podem escolher qual possuem mais familiaridade e se adequa mais ao seu projeto...



# Frontend projeto

## Requisitos:



Todas as rotas devem ter telas que a utilizem, seja de listagem, cadastro, atualização ou até mesmo a remoção.



Deve obrigatoriamente ter uma tela de cadastro e login.



Criem um repositório separado, que rodando localmente consiga consumir os dados da sua API.

# 3º e última entrega

Documentação de entrega da API + Manual de  
utilização da API + Frontend



interface gráfica e como utilizar a API