Aplicações para Internet: Plataforma Cliente-Servidor

Humberto Patrick

Internet como Plataforma de Aplicações Cliente-Servidor

Conceito básico:

Cliente: solicita serviços (ex: navegadores, apps móveis).

Servidor: responde às solicitações (ex: Apache, Nginx).

Funcionamento:

Comunicação via protocolos (HTTP/HTTPS).

Troca de dados estruturados (ex: HTML, JSON).

Tecnologias de Desenvolvimento

Servidores

Função: Hospedar aplicações e processar requisições.

Exemplos: Apache, Nginx, servidores em PHP (PHP-FPM).

Browsers

Função: Interpretar e exibir conteúdo (HTML, CSS, JavaScript).

Exemplos: Chrome, Firefox, Edge.

Middleware

Função: Intermediar comunicação entre cliente e servidor.

Exemplos: APIs REST, sistemas de autenticação, frameworks PHP (Laravel, Symfony).

Servidores

Servidores web:

Executam scripts PHP no backend.

Gerenciam conexões simultâneas.

PHP como linguagem server-side:

Processa dados, acessa bancos de dados, gera HTML dinâmico.

Browsers

Interface do Cliente

Responsabilidades:

- Renderizar interfaces (HTML/CSS).
- Executar JavaScript para interatividade.

Integração com PHP:

- Dados enviados via formulários (POST/GET).
- Consumo de APIs PHP via AJAX/Fetch.

Middleware

Conectando Cliente e Servidor

O que é:

- Camada intermediária para processamento adicional.
- Gerencia tarefas como autenticação, logs, cache e roteamento.
- Exemplos: autenticação, logging, tratamento de rotas.

Middleware em PHP:

- Frameworks como Laravel (Rotas, Middlewares de segurança,).
- APIs RESTful para integração com frontend (React, Angular).

O que é API?

Application Programming Interface - API

Conjunto de regras e protocolos que permitem que diferentes sistemas ou componentes *se comuniquem*.

Exemplo: "Garçom" entre cliente e cozinha (você pede um prato, ele entrega).

Características Principais - API

Padronização:

- Usa métodos HTTP (GET, POST, PUT, DELETE).
- Formatos de dados comuns: JSON, XML.

Desacoplamento:

- Cliente e servidor evoluem independentemente.

Segurança:

- Autenticação via tokens (ex: JWT, OAuth).

Por que APIs são Importantes?

Facilitam a integração entre sistemas distintos.

Permitem escalabilidade e reutilização de código.

Base para arquiteturas modernas (mobile apps, IoT).

Comparação Cliente vs. Servidor

Cliente (Frontend)	**Servidor (Backend/PHP)**
Interface do usuário	Lógica de negócio
JavaScript	PHP
Consome APIs	Fornece APIs