Resumo de Código de Alta Performance - Web 🌎

📡 Unidade I – Fundamentos da Web

🔐 1. Protocolos de Comunicação na Web

- **Protocolo**: conjunto de regras para a comunicação entre dispositivos.
- **Modelo OSI (7 camadas)**:
    - Física, Enlace, Rede, Transporte, Sessão, Apresentação, Aplicação.
- **Modelo TCP/IP (4 camadas)**:
    - Aplicação, Transporte, Internet, Acesso à Rede.

Principais Protocolos:

Protocolo	Função	Exemplo
HTTP / HTTPS	Acessar websites	`https://google.com`
SMTP	Enviar e-mails	Servidores de e-mail
POP3 / IMAP	Receber e-mails	Outlook, Thunderbird
FTP / SFTP / FTPS	Transferência de arquivos	Upload de sites
DHCP	Atribuir IP automaticamente	Conexão Wi-Fi
SSH	Acesso remoto seguro	Linux remoto
ICMP	Diagnóstico de rede (ping)	`ping google.com`
XMPP	Mensagens em tempo real	WhatsApp, Telegram

🏗️ 2. Arquitetura de Aplicações Web

Tipos de Arquitetura:

- **Monolítica**: tudo em um único sistema.
    - Exemplo: Sistema legado com front e back-end juntos.
- **Microsserviços**: componentes independentes.
    - Exemplo: API de login separada do serviço de produtos.
- **Cliente-Servidor**: cliente solicita, servidor responde.
    - Exemplo: Site que acessa um banco de dados remoto.
- **3 Camadas**:
    - **Apresentação**: Interface com usuário.
    - **Negócio**: Regras de negócio.
    - **Dados**: Banco de dados.
- **Serverless**:
    - Código executado sob demanda, sem gerenciar servidores (Ex: AWS Lambda).

🎨 3. Web Design e UI/UX Design

- **Web Design**: aparência e estrutura do site (cores, fontes, layout).
- **UX Design (Experiência do Usuário)**:
    - Foco no uso, na navegação e na satisfação do usuário.
    - Exemplo: um app fácil de usar fideliza mais usuários.
- **UI Design (Interface do Usuário)**:
    - Criação de botões, menus, ícones e aparência visual.
    - Exemplo: Layouts bonitos e interativos.

Etapas de Criação de um Website:

1. Objetivos (briefing)
2. Wireframe (esqueleto visual)
3. Layout (visual final)
4. Programação (funcionalidades)

⚙️ 4. Frameworks Front-End

Frameworks JavaScript para construir **Single Page Applications (SPA)**:

Framework	Características	Destaque
Angular	Mantido pelo Google, usa TypeScript	Estrutura robusta
Vue.js	Simples, eficiente e progressivo	Leve e rápido
Ember.js	Arquitetura MVVM	Escalável, com Ember CLI
Svelte	Compilador, não usa virtual DOM	Código limpo e eficiente

Por que usar frameworks?

- Reutilização de componentes
- Segurança e performance
- Agilidade no desenvolvimento
- Melhor compatibilidade entre navegadores

🔁 5. Comunicação Cliente-Servidor

Modelos de comunicação entre navegador (cliente) e servidor:

Modelo	Características	Exemplos

HTTP / HTTPS	Requisição e resposta (stateless)	Sites e APIs
RPC	Chamada de funções remotas	Microsserviços
WebSockets	Comunicação em tempo real e bidirecional	Chats, games
SOAP	Baseado em XML, usado em empresas	Sistemas bancários
REST	Usa métodos HTTP (GET, POST, PUT, DELETE)	APIs RESTful

📌 **Diferença entre REST e SOAP**:

- REST: leve, usa JSON.
- SOAP: formal, usa XML.

💻 Unidade II – Programação Front-End

🧱 6. HTML (HyperText Markup Language)

- Linguagem de **marcação** (não é programação).
- Define **estrutura** e conteúdo da página.
- **Tags importantes**:
  - `<html>`, `<head>`, `<body>`
  - `<p>`, `<a href="">`, `<img src="">`, `<table>`, `<tr>`, `<td>`

🎨 7. CSS (Cascading Style Sheets)

- Define **estilo visual** da página.
- Controla cores, fontes, tamanhos, espaçamento.
- Pode ser:

- Inline: dentro da tag
- Interno: dentro do `<style>`
- Externo: arquivo `.css`

📜 8. JavaScript

- Linguagem de **programação do navegador**.
- Permite interações dinâmicas.
- Manipula eventos como cliques, teclas, etc.

🌳 DOM – Document Object Model

- Representa a página como uma **árvore de objetos**.
- Permite acesso e modificação dinâmica dos elementos HTML.
- **Eventos DOM**:
  - `click`, `mouseover`, `keydown`, etc.
  -
  - Usado com `addEventListener()`.

📌 Exemplo:

javascript

```
document.getElementById("botao").addEventListener("click", () => {
  alert("Você clicou!");
});
```