1 Introdução

Bem-vindo ao módulo de Desenvolvimento Web! Preparamos uma jornada de aprendizado para que você domine os fundamentos e as habilidades práticas para criar sites e aplicações web do zero. Vamos navegar juntos pelos conceitos essenciais de design, linguagens de marcação, estilização, programação front-end e back-end. Também vamos equipá-lo com o conhecimento sobre os frameworks e as ferramentas mais utilizadas no mercado para acelerar o desenvolvimento.

Nosso material está dividido em duas grandes áreas. A primeira, que começaremos a explorar em breve, é o universo do front-end: tudo aquilo que o usuário vê e com que interage na tela. Em seguida, mergulharemos no back-end, o motor que funciona nos bastidores – no servidor – e que dá vida e inteligência à aplicação.

Para completar sua formação, após vencermos as etapas de apresentação do front-end e back-end, vamos dedicar um tempo valioso para conhecer alguns dos respectivos frameworks mais populares, que trazem produtividade e organização ao seu trabalho. Ao final deste módulo, com o material que selecionamos, você se sentirá seguro e preparado para começar sua trajetória. Deixe que sua curiosidade seja o combustível para se aprofundar ainda mais e mergulhar de cabeça no fascinante mundo do desenvolvimento web.

Minha sugestão é que você consulte esse material escrito sempre como uma complementação e reforço à aula que você assistiu sobre o tema. Se você se sentir inseguro sobre o aprendizado do conteúdo visto em aula, volte ao material escrito e releia. Combinado? Então, vamos lá!

1.1 A Conversa que Constrói a Web: Entendendo a Arquitetura Cliente-Servidor

Você já parou para pensar no que realmente acontece quando você digita um endereço em seu navegador e aperta Enter? Não é mágica, mas é um processo incrivelmente rápido e bem orquestrado. Esse processo constitui a base de boa parte das aplicações web e se chama **arquitetura cliente-servidor**. Vamos desvendar como essa conversa funciona.

Imagine que você está em um restaurante. Você (o **cliente**) quer fazer um pedido. O garçom e a cozinha (o **servidor**) estão sempre prontos para anotar seu pedido, prepará-lo e trazer até você assim que você desejar.

Na web, a lógica é a mesma:

- **Cliente**: É o seu navegador (Chrome, Firefox, Safari, etc.). É ele quem, operado por você, **inicia** a conversa, pedindo alguma coisa.
- **Servidor**: É um computador potente que fica ligado 24/7, em algum lugar do mundo, sempre atento e esperando por pedidos. Ele armazena os sites, aplicações e os arquivos que você irá requisitar.
- **Requisição (*Request*)**: É o pedido que o cliente faz. Quando você digita `google.com` e aperta Enter, seu navegador está enviando uma requisição para o servidor do Google, dizendo: "Olá, por favor, me envie a sua página principal!".
- **Resposta (*Response*)**: É o que o servidor entrega de volta ao cliente. Se tudo der certo, ele envia os arquivos da página solicitada. Se algo der errado (como uma página que não existe), ele envia um código de erro.

O ponto-chave é este: **o cliente sempre inicia a conversa, e o servidor está sempre ouvindo, pronto para responder.**

1.1.1 O Servidor Web: Um Grande Arquivista Digital

Na maioria das vezes, a função de um servidor web é surpreendentemente simples: ele é um grande organizador de arquivos. Pense nele como uma biblioteca com um sistema de endereçamento perfeito. Cada arquivo (uma imagem, um documento, uma página HTML) está guardado em uma pasta específica.

Quando seu navegador (o cliente) faz uma requisição por um endereço (URL), ele está, na verdade, pedindo um arquivo específico localizado na árvore de diretórios do servidor.

E qual é o idioma oficial dessa conversa? O **Protocolo de Transferência de Hipertexto**, ou simplesmente **HTTP**. Ele é o conjunto de regras que clientes e servidores usam para se entenderem perfeitamente durante as requisições e respostas.

Vamos Ver na Prática?

Para sentir como isso funciona, vamos propor quatro experimentos simples que você pode fazer agora mesmo em aula, usando arquivos hospedados em um servidor local ou em um serviço online. Atente-se às instruções complementares dos monitores.

Prática 1: O Pedido que Vira um Download

Tente acessar um arquivo que o navegador não foi projetado para exibir, como um `.docx` (documento do Word) ou um `.zip` (arquivo compactado).

- **O que acontece?**

  O navegador vai iniciar o download do arquivo.

- **Por quê?**

  O navegador fez a requisição HTTP, o servidor encontrou e enviou o arquivo. No entanto, o navegador não sabe o que fazer com ele, não tem a capacidade nativa de "renderizar"um documento do Word. Sua única ação possível é oferecê-lo para você salvar.

Prática 2: O Pedido que o Navegador Entende

Agora, tente acessar um arquivo de texto simples (`.txt`) ou um documento `.pdf`.

- **O que acontece?**

  Na maioria das vezes, o arquivo será exibido diretamente na aba do navegador.

- **Por quê?**

  O navegador recebeu o arquivo e, desta vez, ele *sabe* o que fazer com ele. Navegadores modernos têm leitores de PDF e visualizadores de texto embutidos, então eles podem renderizar o conteúdo para você.

Prática 3: A Mágica do HTML

Por fim, acesse um arquivo `.html` básico.

- **O que acontece?**

  Desta vez, você não verá o código `<p>Olá, mundo!</p>`. Em vez disso, você verá uma página web estruturada, com o texto "Olá, mundo!"formatado.

- **Por quê?**

  HTML é a língua nativa do navegador. Ao receber um arquivo `.html`, ele não o exibe como um simples texto. Ele o **interpreta**, entende as tags, a estrutura, e **renderiza** o resultado visual final. Esta é a base da construção de qualquer site.

Prática 4: Onde estão esses arquivos?

O servidor não expõe todos os seus arquivos para o mundo. Ele possui uma pasta especial, conhecida como **Document Root** (ou diretório raíz web), que é a única parte visível para a internet. A URL que você digita no navegador é um caminho a partir da entrada dessa pasta.

Observe junto com o monitor como os arquivos estão estruturados no computador servidor. Imagino que ele deve mostrar uma estrutura de arquivos no computador servidor semelhante à que segue:

```
/home/usuario/
|-- projeto_secreto/
|   `-- notas.txt
|
...

/var/www/html/  <-- ESTE É O DOCUMENT ROOT
|-- index.html
|-- sobre.html
|
|-- imagens/
|   `-- logo.png
|
```

```
‘-- css/
    ‘-- estilo.css
```

- **Acessando a raíz:** Para pegar o arquivo `index.html`, a URL seria simplesmente `http://meusite.com/index.html`. O servidor está configurado para que o domínio principal aponte diretamente para o conteúdo de `/var/www/html/`.
- **Acessando uma subpasta:** Para pegar a imagem `logo.png`, a URL precisa incluir o caminho a partir da raíz: `http://meusite.com/imagens/logo.png`. O navegador pede pelo recurso "imagens/logo.png", e o servidor o traduz para o caminho completo no disco.
- **O que é inacessível?** O arquivo `notas.txt`, localizado em `/home/usuario/projeto_secreto/`, está *fora* do *Document Root*. É impossível acessá-lo via navegador. Isso é uma medida de segurança essencial para proteger arquivos de configuração e dados sensíveis do servidor.

1.1.2 Além do Pedido Único: O "Hiper"do Protocolo HTTP

Agora vamos para a parte mais interessante. O "H"em HTTP vem de **Hipertexto**. Um documento de hipertexto não é apenas um texto isolado; ele é um texto que se conecta a vários outros recursos.

Uma única página HTML (`index.html`) raramente vem sozinha. Ela é como a planta de uma casa que diz onde tudo deve ir:

- "Use a folha de estilos `estilo.css` para definir as cores e as fontes."
- "Execute o script `animacoes.js` para criar interações."
- "Mostre a imagem `logo.png` no topo da página."
- "Incorpore o vídeo `tutorial.mp4` no meio do conteúdo."

Para que o navegador possa construir a página completa que você vê, ele precisa de todos esses "pedaços". E como ele os consegue? Fazendo **novas requisições HTTP para cada um deles!**

Espiando os Bastidores com as Ferramentas de Desenvolvedor

Esta é a hora de provar tudo isso na prática e se sentir um verdadeiro desenvolvedor web.

Sua Missão (Se Decidir Aceitá-la):

1. Abra seu navegador em um site movimentado, como um portal de notícias ou uma rede social.
2. Pressione a tecla **F12** para abrir as **Ferramentas de Desenvolvedor**.
3. Clique na aba **Network** (ou **Rede**).
4. Com a aba aberta, pressione **F5** ou **Ctrl+R** para recarregar a página.

5. **Observe a mágica!** Você verá uma cascata de arquivos aparecendo na lista. O primeiro geralmente é o documento HTML principal. Logo em seguida, você verá o navegador fazendo dezenas (ou centenas!) de outras requisições para buscar os arquivos `.css`, `.js`, `.jpg`, `.png`, fontes (`.woff2`) e muito mais.

Essa visão clara demonstra o conceito fundamental: uma única interação sua na barra de endereços desencadeia uma avalanche de conversas cliente-servidor, todas orquestradas pelo protocolo HTTP para montar o quebra-cabeça que chamamos de página web.

Aprofundamentos sobre o Protocolo HTTP

Já estabelecemos que o HTTP é o "idioma"da web, mas o que significa ser um protocolo na prática?

Significa que ele é um **conjunto de regras formais**. Dizer que seu navegador (cliente) e o servidor web (como o popular `Apache2`) suportam HTTP, é dizer que os programadores de ambos os softwares escreveram código que implementa essas regras à risca. É por isso que um navegador feito pela Google consegue conversar perfeitamente com um servidor web feito pela Microsoft ou por uma comunidade de software livre. Eles todos seguem o mesmo manual de instruções.

Quando você digita um endereço na barra do navegador, na verdade está montando uma requisição HTTP completa, mesmo que de forma simplificada. O endereço completo geralmente começa com `http://` ou `https://`.

- `http://` - Indica o uso do protocolo HTTP padrão. A comunicação é feita em texto plano, sem criptografia.
- `https://` - Indica o uso do **HTTPS (HTTP Secure)**. É uma camada de segurança (usando outro protocolo chamado SSL/TLS) que criptografa toda a conversa entre o cliente e o servidor. Hoje, é o padrão absoluto para qualquer site que lide com informações sensíveis, desde logins até transações financeiras.

Parte dessas "regras"do protocolo inclui uma série de códigos de status que o servidor envia na sua resposta. Eles são essenciais para sabermos o resultado de uma requisição. Você já deve ter esbarrado em alguns deles!

1.1.2.0.1 Tabela de Status HTTP Comuns:

Código	Nome	O que significa na prática?
200	OK	Deu tudo certo! O servidor encontrou o que você pediu e está te entregando. É o código do sucesso.
301	Moved Permanently	O recurso que você pediu mudou de endereço permanentemente. O navegador, de forma inteligente, já te redireciona para o novo local.
403	Forbidden	Você não tem permissão para ver isso. O servidor entendeu seu pedido, mas se recusa a entregar o recurso. É como tentar entrar em uma área restrita.
404	Not Found	O famoso "não encontrado". O servidor não conseguiu encontrar nada no endereço que você pediu. O link pode estar quebrado ou você digitou errado.
500	Internal Server Error	O problema é no servidor. Ele recebeu seu pedido, mas alguma coisa quebrou do lado de lá (um erro no script, um problema no banco de dados) e ele não conseguiu completar a resposta.

Outros Protocolos Essenciais na Arquitetura Cliente-Servidor

Apesar do HTTP ser a estrela do show, um desenvolvedor web utiliza vários outros protocolos baseados na arquitetura cliente-servidor para conseguir trabalhar. Conhecer o papel de cada um é fundamental. Vamos ver os principais:

FTP (File Transfer Protocol)

O FTP é o protocolo "pai"da transferência de arquivos. Antes de termos interfaces web complexas, era a principal forma de enviar e baixar arquivos de um servidor. Para um desenvolvedor, ele é a ferramenta para "subir"os arquivos do site (HTML, CSS, imagens) do seu computador para o servidor de hospedagem.

1.1.2.0.2 Prática Rápida com FTP *(Atente-se às instruções do monitor)*:

1. Baixe um cliente de FTP gráfico como o **FileZilla**.
2. Use as credenciais de FTP (host, usuário, senha) fornecidas pelo seu serviço de hospedagem para se conectar ao servidor.
3. Observe a interface: de um lado, os arquivos do seu computador; do outro, os arquivos no servidor. Arraste um arquivo de imagem do seu lado para o lado do servidor. Pronto, você acabou de fazer um upload via FTP!

SSH (Secure Shell)

Se o FTP é para transferir arquivos, o SSH é para **controlar o servidor**. Ele cria um túnel de comunicação seguro e criptografado que permite a você acessar a linha de comando do servidor como se estivesse sentado na frente dele. É usado para instalar programas, configurar o ambiente, gerenciar permissões e realizar tarefas administrativas.

Prática Rápida com SSH: ***(Atente-se às instruções do monitor)***

1. Abra o terminal do seu computador (no Windows, pode ser o PowerShell ou o WSL).
2. Digite o comando de conexão: `ssh seu_usuario@ip_do_servidor`.
3. Após digitar a senha, o prompt de comando mudará para indicar que você está conectado ao servidor remoto (ex: `seu_usuario@nome_do_servidor:~$`).
4. Digite um comando simples como `ls -la` e aperte Enter. A lista de arquivos que aparecerá é a que está *no servidor*, não no seu computador. Você está no controle!

DNS (Domain Name System)

O DNS é a "lista telefônica"da internet. Nenhum dos protocolos acima funcionaria de forma amigável sem ele. Computadores se encontram por números (endereços IP, como `172.217.25.142`), mas nós, humanos, usamos nomes (como `google.com`). O DNS é o sistema cliente-servidor distribuído que traduz os nomes de domínio que digitamos nos endereços IP que as máquinas entendem.

Prática Rápida com DNS: ***(Atente-se às instruções do monitor)***

1. Abra o terminal do seu computador.
2. Digite o comando `nslookup google.com` (ou `dig google.com` em sistemas Linux/-Mac).
3. O resultado mostrará o(s) endereço(s) IP associados àquele domínio. Você acabou de fazer uma consulta DNS manualmente, exatamente o que seu navegador faz secretamente toda vez que você acessa um site.

Espero que as informações aqui apresentadas ajudem a complementar o conteúdo visto na aula de hoje. Até a próxima!