UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

Programação II:

Introdução às Aplicações Web

Giancarlo Salton, Neimar Marcos Assmann

Parte I: Introdução às Aplicações Web

O que são Aplicações Web?

- São soluções/programas acessíveis/executados através de um navegador/browser, e que utilizam tecnologias web padrão para funcionar.

Tem como principais características:

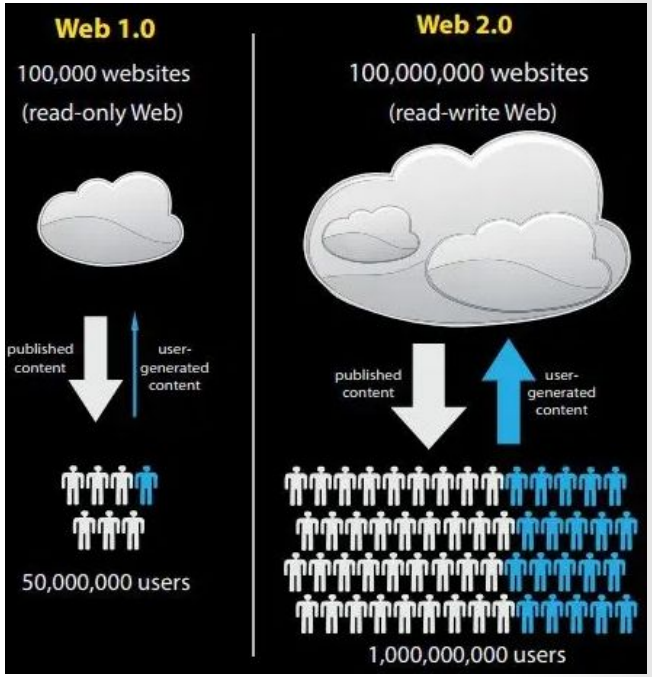
- Ser acessível via navegador;
- Ser hospedadas em servidores web;
- Não requerem instalações em computadores dos usuários

Exemplos:

Gmail, Facebook, Sigaa, uffs.edu.br

Sugestão de leitura:
https://tecnoblog.net/responde/o-que-e-o-w3c-world-wide-web-consortium/

Parte I: Introdução às Aplicações Web



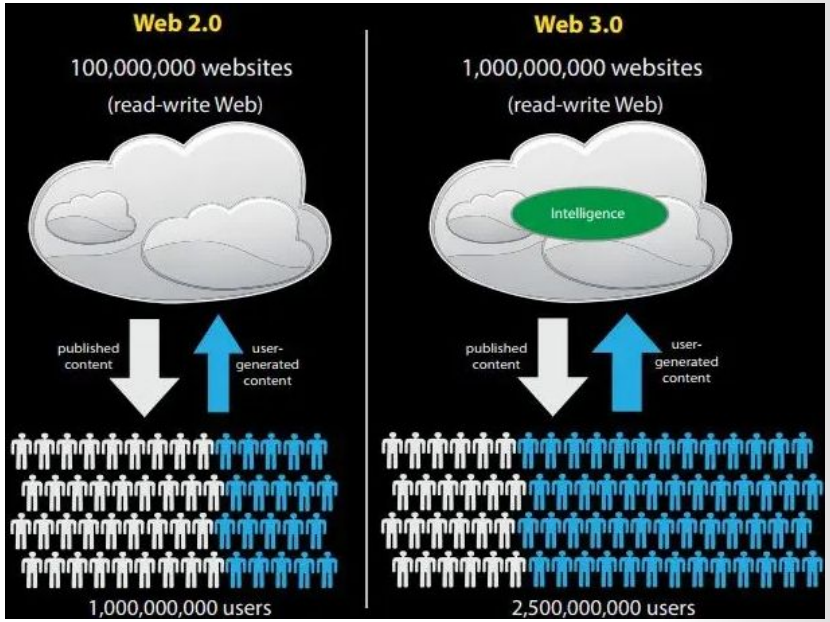
- Evolução das aplicações web
  - **Web 1.0** (primeira fase): Caracterizada por sites estáticos que oferecem informações aos usuários mas não permitem a interação ou colaboração. Tecnologia predominante HTML. Principal objetivo era sites para fornecer informações. Teve seu início em 1991 até meados de 2000.
  - **Web 2.0** (segunda fase): Caracterizada pela interatividade e colaboração entre usuários. Foco na comunicação e redes sociais. Os sites tornavam-se mais dinâmicos e interativos. De 2000 à aproximadamente 2010.

    Ex:
    - Surgimento de Blogs, Orkut, Facebook, Twitter, Youtube.

Parte I: Introdução às Aplicações Web

- Evolução das aplicações web
  - **Web 3.0** (terceira fase): objetivo é tornar a internet mais acessível e intuitiva para os usuários, com a ajuda de tecnologias como IA, Blockchain, realidade virtual, maior eficiência de tarefas anteriormente feitas manualmente com uso de tecnologias como Machine learning (Ex: algoritmos de recomendação de conteúdo), dados conectados, busca avançada. De 2010 em diante.

    Ex:
    - Assistentes virtuais (Alexa, Siri);
    - Motores de recomendação (Netflix, Amazon);
    - Desenvolvimento é o metaverso.



Sugestão de leitura sobre metaverso:
https://blog.culte.com.br/o-que-e-metaverso-venha-conhecer/

Parte I: Introdução às Aplicações Web

- Evolução das aplicações web
  - **Web 4.0**: Idealiza-se essa fase, onde baseia-se em:
    - Sistemas operacionais da Web,
    - Interação total entre o mundo físico e digital
    - Internet das Coisas (IoT) e computação em nuvem;

      Ex:
      - Evolução dos assistentes virtuais para se tornarem mais contextuais, autônomos e integrados a um ambiente inteligente e interconectado.
      - Tesla Autopilot
      - Smart homes
      - Sistema de Gerenciamento de Tráfego em Cidades Inteligentes
  -

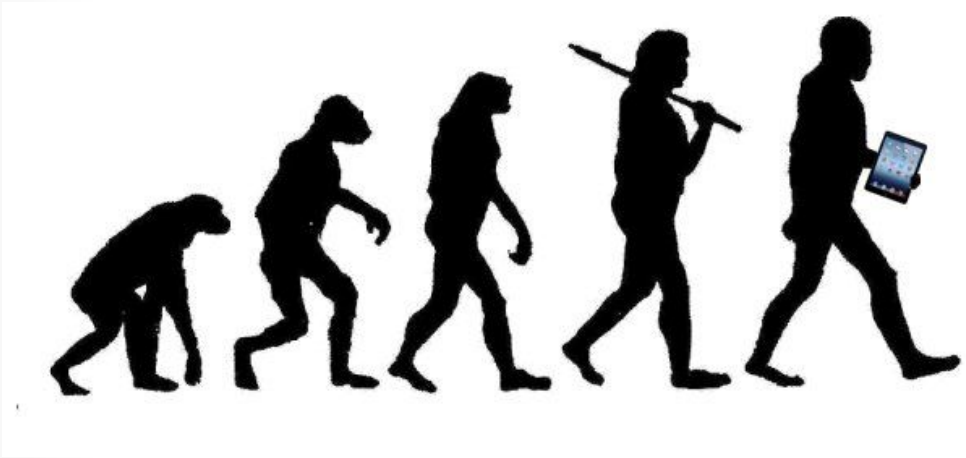
Parte I: Introdução às Aplicações Web

Fonte: Poder360: https://www.poder360.com.br/brasil/dispositivos-iot-devem-chegar-a-27-bilhoes-ate-2025/

Parte I: Introdução às Aplicações Web

- Discussão sobre aplicações modernas:
    - Funcionalidades e benefícios aos usuários?
    - Quais são as necessidades dos usuários?
    - Interação com o usuário de forma segura?

G-mail, Facebook, Spotfy, Netflix



Parte II: Arquitetura Cliente/Servidor e em Camadas

1) **Arquitetura Cliente/Servidor:** Definição e Funcionamento

- Definição:
  - É um modelo de sistema distribuído no qual um ou mais clientes (dispositivos ou aplicações) solicitam serviços e recursos a um ou mais servidores, que os fornecem.

- Funcionamento Básico:
  - O cliente envia uma solicitação ao servidor.
  - O servidor processa essa solicitação, realiza a tarefa solicitada (como acessar um banco de dados, processar dados, ou fornecer conteúdo).
  - O servidor envia uma resposta de volta ao cliente.

- Aplicações Práticas:
  - Navegação na Web: O navegador (cliente) solicita uma página ao servidor web, que retorna o HTML da página.

Ex:

Aplicações de E-mail: O cliente de e-mail solicita ao servidor de e-mail o envio ou recebimento de mensagens.

Parte II: Arquitetura Cliente/Servidor e em Camadas

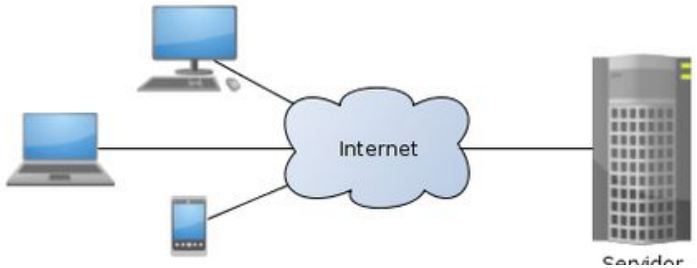
1) **Arquitetura Cliente/Servidor:** Componentes principais

- **Cliente**: dispositivo ou aplicação que faz solicitações ao servidor. Pode ser um navegador web, aplicativo de leitura de e-mail, aplicativo de dispositivo móvel.
  Função:
  - Enviar solicitações ao servidor;
  - Receber e processar as respostas do servidor;
  - Apresentar os dados ou resultados ao usuário final;

  Ex: Mozilla Thunderbird, Google Chrome, Outlook.

- **Servidor**: servidor ou sistema que fornece serviços ou recursos aos clientes.
  Função:
  Processar as solicitações recebidas dos clientes;
  Acessar recursos, como banco de dados, arquivos para processar as solicitações;
  Enviar as respostas ao cliente;

  Ex: Servidor Apache, Nginx.



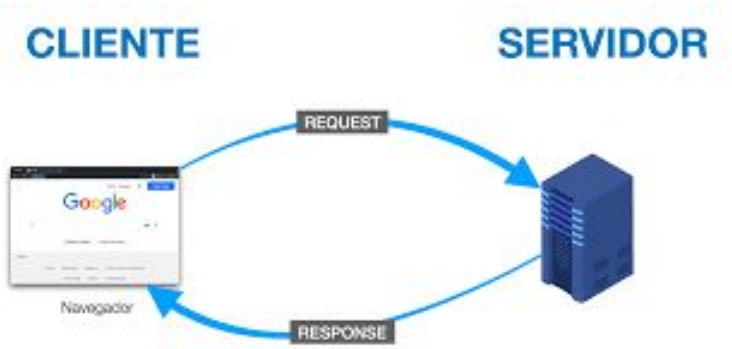
Parte II: Arquitetura Cliente/Servidor e em Camadas

1) **Arquitetura Cliente/Servidor:** Comunicação e troca de dados entre cliente e servidor.

A comunicação entre cliente e servidor geralmente é realizada por meio de protocolos como: HTTP/HTTPS, FTP, SMTP, etc.

Funcionamento:

- **Solicitação (Request)**: O cliente envia uma solicitação ao servidor usando um protocolo definido (por exemplo, uma requisição HTTP GET para acessar uma página web).
- **Resposta (Response):** O servidor processa a solicitação e envia uma resposta de volta ao cliente. Esta resposta pode incluir dados como arquivos HTML, JSON, XML, entre outros.



Parte II: Arquitetura Cliente/Servidor e em Camadas

1) **Arquitetura Cliente/Servidor**: Exemplo prático

Acesso ao SIGAA**:**

Passo 1: O cliente (navegador) envia uma solicitação HTTP GET ao servidor para acessar uma página.

Passo 2: Servidor da UFFS/O servidor web processa essa solicitação, acessa o conteúdo solicitado (HTML, CSS, JS) e envia de volta ao cliente.

Passo 3: O cliente (navegador) renderiza a página para exibição ao usuário.

:

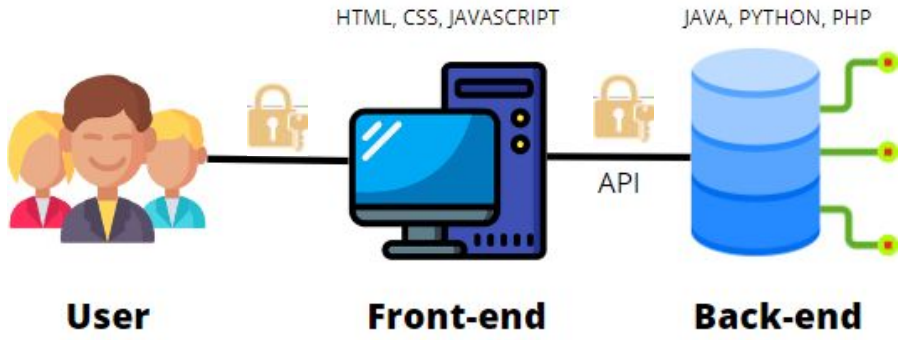
Parte II: Arquitetura Cliente/Servidor e em Camadas

2) **Arquitetura em Camadas:** Definição e funcionamento básico.

- É um modelo organizacional no qual o sistema é dividido em camadas ou níveis, cada uma com responsabilidades específicas.

- As camadas interagem entre si, e isso promove modularidade, permitindo que diferentes partes do sistema sejam desenvolvidas, testadas e modificadas independentemente.

- A ideia desse tipo de arquitetura é separar preocupações do sistema em diferentes níveis de abstração.



Parte II: Arquitetura Cliente/Servidor e em Camadas

2) **Arquitetura em Camadas:** Exemplo de funcionamento básico:

Sistema XYZ:

- Camada de Apresentação (UI): responsável pela interface do usuário, onde os dados são apresentados.

- Camada de Lógica de Negócios: destinada para regras de negócio. Processa ações do usuário e interage com a camada de dados.

- Camada de Dados: atua com a persistência de dados, como consultas, inserções e alterações em bases de dados.

Parte II: Arquitetura Cliente/Servidor e em Camadas

2) **Arquitetura em Camadas:** Exemplo de funcionamento básico:

Um sistema pode ter mais camadas?

Sim, pode ter outras camadas, como:

> Em um sistema de Gerenciamento de Cadastros de Usuários, onde pode ter uma Camada de Serviço, que fornece como um serviço, métodos de autenticação de usuários em outros sistemas/plataformas.
>
> Ex: Google Identity Platform

Parte III: Servidores Web

É um software, ou conjunto destes, que hospedam páginas e serviços para usuários.

Recebem solicitações via aplicativos do tipo “cliente”, como navegadores (Chrome, Firefox), e com as instruções da aplicação que está hospedada, processam as informações e devolvem uma resposta para o usuário.

Principais funções:

- Receber solicitações: via protocolos como: HTTP/HTTPS/SMTP;
- Resposta: responder solicitações, como: apresentação de conteúdos por meio de arquivos HTML, CSS, Javascript
- Gerenciamento de Conexões: estabelece e gerencia conexões como clientes, inclusive manutenção de sessões e controle de acesso;
- Execução de aplicações: executar scripts e aplicações, como por exemplo Scripts em PHP, Python, para gerar conteúdo dinâmico baseado nas solicitações

Parte III: Servidores Web

Exemplo de Servidores Web:

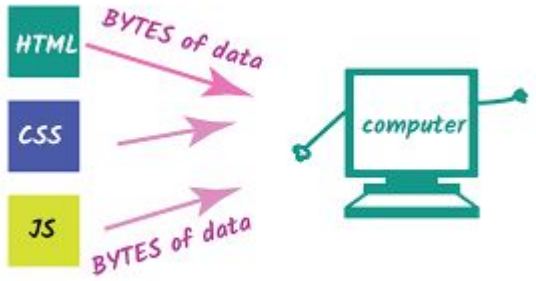
- Apache
- NGinx
- Microsoft Internet Information Services (IIS)
- Tomcat

Parte III: Servidores Web e Renderização de Conteúdo

O que é Renderização?

É um processo no qual o navegador interpreta e exibe o código HTML, CSS, Javascript…. de uma página web.

Ou seja, um processo de transformação de um código em uma representação visual para o usuário.



Parte III: Servidores Web e Renderização de Conteúdo

Exemplo de etapas de Renderização:

1) Carregamento do HTML:

   a) o navegador faz uma solicitação via HTTP para o servidor web e recebe o documento HTML.
   b) o HTML é lido e interpretado para construir o DOM (Documento Object Model), que é uma representação em árvore da estrutura do documento.

1) Carregamento do CSS:

   b) O Navegador carrega arquivos CSS referenciados no HTML
   c) Os arquivos CSS são usados para construir o CSSOM (CSS Object Model), que é uma representação em árvore das regras de estilo.
   d) O DOM e o CSSOM são combinados para criar o Render Tree.

Parte III: Servidores Web e Renderização de Conteúdo

3) Layout (ou Reflow):

   a) O navegador calcula as posições e tamanhos dos elementos da página com base no Render Tree, determinado a posição final dos elementos na página.

4) Pintura:

   a) O navegador desenha os elementos visuais na tela, aplicando cores, estilos, fontes.

5) Composição:

   a) A página é dividida em camadas (se necessário) que são compostas e exibidas no navegador.

6) Execução de Javascript:

   a) Scripts são executados, e que podem modificar o DOM, CSSOM, ou até causar novas renderizações ou atualizações da página.

Parte III: Servidores Web e Renderização de Conteúdo

Frameworks de renderização:

- São conjunto de ferramentas ou bibliotecas que auxiliam na construção e gerenciamento da renderização de interfaces para usuário.

- Fornecem estruturas e convenções para simplificar e otimizar o processo de criação de interfaces cada vez mais interativas, dinâmicas, e facilitando a construção e manutenção do código.

Parte IV: Linguagens Client-side e Server-side

- Linguagens Client-side:
    - São executadas no navegador do usuário e usadas para criar e gerenciar a interação com a interface do usuários e manipular elementos da página sem precisar de uma atualização da página ou comunicação com o servidor, ou seja, sem tráfego na rede.

        Ex: Javascript através da manipulação do DOM, validação de formulários, animações.

Parte IV: Linguagens Client-side e Server-side

- Linguagens Server-side:
    - São linguagens executadas no servidor, usadas para processar dados, interagir com banco de dados, gerenciar sessões e enviar respostas para o cliente. O código é executado no servidor e posteriormente enviado/renderizado no navegador.

      Ex: Python, PHP, Ruby, Java.

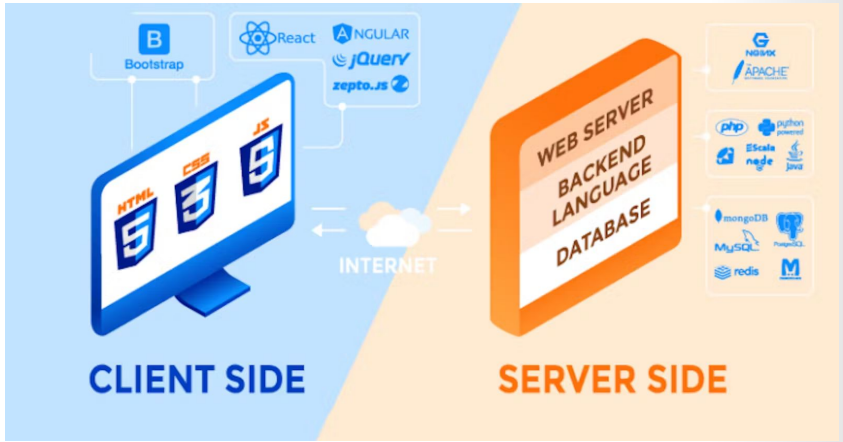
Parte IV: Linguagens Client-side e Server-side

- Client-side ***VS*** Server-side:

  Ambas têm papéis distintos e complementares no desenvolvimento web.

Discussão?!

A decisão de quando usar é conceitual, e depende de fatores que o desenvolvedor deverá definir e para isso, deverá considerar fatores como performance e interatividade.



Parte IV: Linguagens e Protocolo de Comunicação

- Protocolo **HTTP/HTTPS**
  - São protocolos de comunicação usados para transferir dados entre um cliente (navegador) e um servidor web.
  - Por padrão, operam nas portas 80 (HTTP) e 443 (HTTPS);
  - HTTPS é a versão segura do HTTP, pois utiliza criptografia para proteger a comunicação.
    - Usa SSL/TLS e com isso visa garantir a confidencialidade, integridade e autenticidade.
    - É ideal para proteger informações sensíveis, dados pessoais, senhas.

Parte IV: Linguagens e Protocolo de Comunicação

- Principais métodos do HTTP:
  - **Get**: recupera dados do servidor.
    É utilizado para obter páginas HTML, imagens, etc.
    Ex: Em um página de pedidos de um restaurante, solicitar a abertura do menu de pedidos.
  - **Post**: envia dados para o servidor, para criar/atualizar recursos. Comumente utilizado para submissão de formulários;
    Ex: Após escolher o prato do restaurante, você escolhe o item, uma pizza, e envia o pedido.
  - **Put**: Substitui um recurso no servidor ou cria um novo recurso se ele não existir.
    Ex: Você esqueceu um item no pedido, e decide adicionar outro item, uma coca!
  - **Delete**: remove um recurso do servidor.
    Ex: Antes que seu pedido seja preparado, você decide cancelar!
  - Outros métodos: Patch, Options, Head, Connect, Trace.

Parte V: Cookies e Sessões

Gerenciamento da **Sessão**:

- Como Funciona uma Sessão?

  - O servidor cria uma sessão (Session) e associa um identificador único ao usuário.
  - O identificador é armazenado no navegador como um cookie.
  - Dados da sessão são armazenados no servidor (memória, banco de dados).

- Persistência de Sessão:

  - Sessões expiram após um período de inatividade.
  - Persistência com cookies de sessão ou cookies persistentes.

Parte V: Cookies e Sessões

- Cookies e Sessões estão intimamente relacionados no contexto das aplicações web, desempenhando papéis complementares na manutenção do estado entre as interações do usuário com o servidor.

Parte V: Cookies e Sessões

	Cookie	Sessão
Armazenamento de dados	Informações armazenadas no navegador	Informações armazenadas no servidor
Segurança	Os dados podem ser acessados e manipulados pelo cliente, o que pode representar risco.	Geralmente é maior, pois dados sensíveis não ficam salvos no navegador do usuário, reduzindo riscos de exposição.
Vida útil dos dados	podem ter uma data de expiração configurada, e o navegador vai manter o cookie até que a data de expiração passe ou até que o usuário o exclua manualmente	Geralmente tem tempo limitado, geralmente pelo tempo em que o navegador está aberto ou quando o usuário fizer definir (logout).
Utilização	Ideal para armazenar preferências do usuário, e outras informações que precisam ser persistentes entre visitas, como temas de interface ou configurações de idioma.	Ideal para armazenar dados temporários e específicos para a duração da visita do usuário, como informações de autenticação ou detalhes do carrinho de compras

Parte VI: Autenticação *vs* Autorização

- **Autenticação**:

  - É o processo de provar que você é quem diz ser.
  - É obtido pela verificação da identidade de uma pessoa ou de um dispositivo.
  - Às vezes, a autenticação é abreviada para **AuthN.**

  Pode ser aprovada por credenciais de login, como:

  - Nomes de usuário e senhas
  - Tokens de acesso e códigos de uso único
  - Perguntas de segurança
  - Aplicativos de autenticação associados a um número de telefone ou e-mail

Parte VI: Autenticação *vs* Autorização

- **Autenticação**:

Parte VI: Autenticação *vs* Autorização

- **Autorização**:

  - É o ato de conceder a uma parte autenticada a permissão para fazer algo.
  - Ele especifica quais dados você tem permissão para acessar e o que pode fazer com esses dados.
  - Às vezes, a autorização é abreviada para **AuthZ**.

Parte VI: Autenticação *vs* Autorização

- **Autorização**: