

# Programação Front-end – AULA 01

Matheus Moresco

Engenharia de Software - 3º Período

2025/01

# Informações da Disciplina

- **Disciplina:** Programação Frontend
- **Curso:** Engenharia de Software
- **Carga Horária:** 80 horas
- **Aulas por Semana:** 4 (2 encontros de 1 hora e 40 minutos cada)
- **Professor:** Matheus Moresco
- **Semestre:** 2025/01

# Introdução a disciplina

- Esta disciplina tem como objetivo introduzir os conceitos e técnicas fundamentais para o desenvolvimento de interfaces web
- Estruturação do conteúdo com HTML
- Estilização com CSS
- Programação com JavaScript e TypeScript
- Aplicações web responsivas, acessíveis e interativas

# Importância do desenvolvimento front-end no mercado

- **Experiência do Usuário (UX):** Um design bem elaborado melhora a interação e satisfação do usuário.
- **Acessibilidade:** Permite que diferentes públicos, incluindo pessoas com deficiência, possam acessar conteúdos online.
- **Performance e SEO:** Páginas bem otimizadas carregam mais rápido e melhoram o ranqueamento nos buscadores.
- **Crescimento da Web e Mobile:** Com a popularização dos dispositivos móveis, a demanda por aplicações responsivas cresceu significativamente.
- **Alta demanda por profissionais:** Empresas de todos os setores buscam desenvolvedores front-end qualificados.

# Materiais

- **Linguagens:** HTML, CSS, JavaScript
- **IDE:** VS Code
- **Frameworks:** ReactJS
- **Ferramentas:** Git, Docker
- **Bibliografia:**
  - ALVES, William Pereira. HTML & CSS : aprenda como construir páginas web / 2021 São Paulo: Expressa, 2021.
  - UNICESUMAR; TOKUMOTO, Ronie Cesar. Programação Front End / 2016 Maringá: Unicesumar - Centro Universitário de Maringá, 2016.
  - MILETTO, Evandro Manara; BERTAGNOLLI, Silvia de Castro (null). Desenvolvimento de software ii : introdução ao desenvolvimento web com html, css, javascript e php / 2014 Porto Alegre: Bookman, 2014

# Programa

1. Introdução ao Desenvolvimento Web
2. Padrões de Design e Arquitetura
3. HTML e Estruturação de Conteúdo
4. CSS para Design Visual
5. JavaScript e Programação Dinâmica
6. Aplicações de Página Única (SPA)
7. Desenvolvimento com Frameworks e Bibliotecas
8. Integração com APIs e Backend
9. Ferramentas de Desenvolvimento e Testes

# Metodologia

- Aulas teóricas expositivas
- Exercícios práticos e mini-projetos
- Aulas práticas no Laboratório de programação
- Projeto Final

# Estrutura do curso e forma de avaliação

- Exercícios Práticos/ Projeto Final: 80%
- Atividade de Estudo Programada(AEP): 10%
- Prova integrada: 10%



# A História da Internet

- A internet surgiu de fins militares na época da guerra fria.
- Em 1969, o Ministério da Defesa dos Estados Unidos cria a ARPANET.
- O objetivo era interligar computadores utilizados em centros de pesquisa, com fins militares.
- A primeira conexão em rede foi estabelecida em 29/10/1969, entre a **UCLA** e o **STANFORD RESEARCH INSTITUTE**.

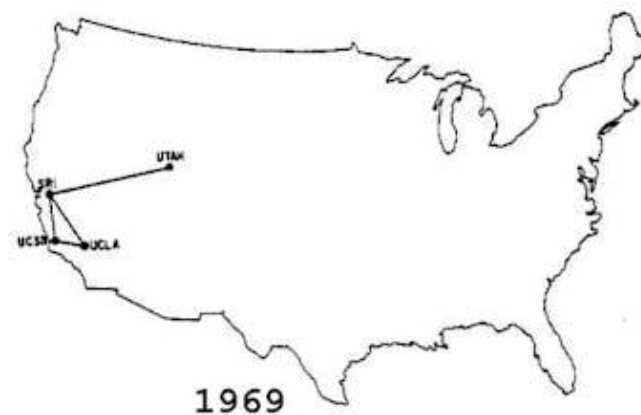
# A História da Internet

- Com o tempo a ARPANET foi se expandindo e foram conectadas novas instituições de pesquisa.

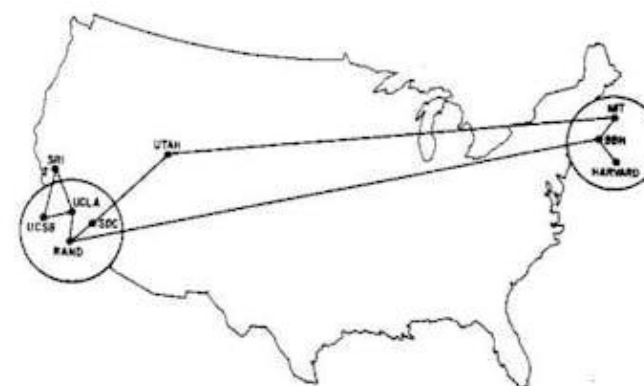


The ARPANET in December 1969

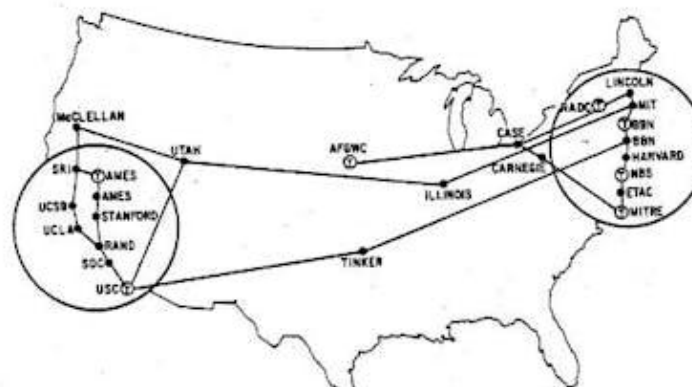
# A História da Internet



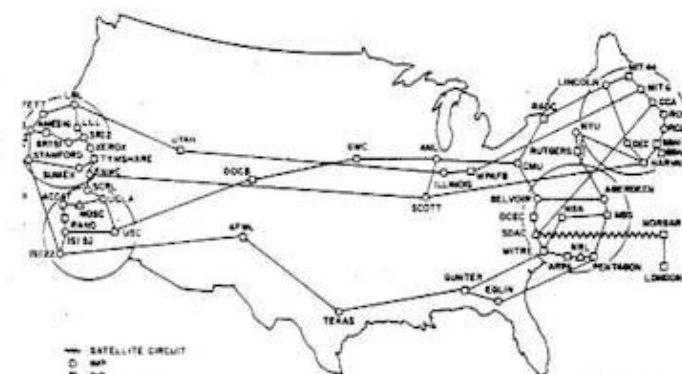
1969



1970



1972



1977

☒ SATELLITE CIRCUIT  
☐ MAP  
☐ TIP  
☐ PLUMBUS IMP

(NOTE: THIS MAP DOES NOT SHOW ARPAS (EXPERIMENTAL SATELLITE CONNECTIONS))  
 NAMES SHOWN ARE ARPANETS, NOT (NECESSARILY) HOST NAMES

# A História da Internet

- 1972 - Criação do **Network Control Protocol (NCP)**.
- Primeira conexão transatlântica entre **ARPANET** e **NOSAR**.
- Depois disso começou a ser pautada a necessidade de conectar a rede de computadores em um contexto global.
- Criação do protocolo TCP/IP como protocolo de conexão global.
- Nos anos 1980, a internet começa a expandir e várias redes fechadas começam a se interconectar.

# O Início da Web

- **World Wide Web(WWW)** foi criada por Tim Berners-Lee em 1989 no CERN.
- Primeiro site lançado em 1991 ([info.cern.ch](http://info.cern.ch)).
- Conceitos fundamentais: HTML, HTTP e URLs.
- Objetivo inicial: Compartilhamento de informações acadêmicas.

# Web 1.0 (1990 – 2000)

- Características principais:
  - Web estática: páginas fixas sem interação do usuário.
  - Uso predominante de HTML com pouca estilização.
  - Sites serviam apenas para leitura (read-only).
  - Primeiros sites institucionais, diretórios de links.
- Limitações:
  - Falta de interatividade e personalização.
  - Necessidade de atualizações manuais pelo desenvolvedor.

## Web 2.0 (2000 – 2010)

- Principais avanços:
  - Interatividade e colaboração dos usuários.
  - Introdução de CSS para design e layout mais avançado.
  - Uso de JavaScript para tornar as páginas mais dinâmicas.
  - Redes sociais, blogs e plataformas colaborativas surgem (Facebook, YouTube, Wikipedia).
  - APIs permitem comunicação entre sistemas (exemplo: Google Maps integrado a sites).
- Impacto:
  - Surgimento do conceito de experiência do usuário (UX).
  - Explosão do comércio eletrônico e serviços online

# Web 3.0 (2010 – Presente)

- Principais avanços:
  - Web semântica: Conteúdos mais organizados e compreensíveis por máquinas.
  - Inteligência Artificial integrada à experiência web.
  - Aplicações responsivas e progressivas (PWA).
  - Blockchain e descentralização (exemplo: criptomoedas e Web3).
  - Assistentes virtuais e busca por voz (exemplo: Alexa, Google Assistant).
- Impacto:
  - Maior personalização de conteúdo.
  - Privacidade e segurança como foco (GDPR, LGPD).



# Tecnologias que Moldaram a Web

- HTML → Melhor suporte para multimídia e interatividade.
- CSS → Animações, Flexbox, Grid Layout.
- JavaScript → Recursos modernos para aplicações interativas.
- Frameworks e Bibliotecas → React, Angular, Vue.js.
- APIs e Microservices → Integração eficiente com backends.

# Conceitos da Programação Web

- A web conecta bilhões de usuários e dispositivos globalmente.
- O desenvolvimento front-end depende de entender seu funcionamento básico.
- Vamos explorar os conceitos essenciais:
  1. **Cliente-Servidor**
  2. **HTTP**
  3. **URLs**
  4. **Navegadores**
  5. **Tipos de Aplicações Web.**

# Como a Web Funciona?

- **A Web é um sistema distribuído baseado no modelo Cliente-Servidor.**
  - **Fluxo básico de uma requisição web:**
    - O usuário digita uma URL no navegador.
    - O navegador envia uma **requisição HTTP** para o servidor.
    - O servidor processa a requisição e retorna uma **resposta** (HTML, CSS, JavaScript).
    - O navegador renderiza a página para o usuário.
- Compreender esse fluxo ajuda a desenvolver aplicações web eficientes.

# Modelo Cliente-Servidor

**Cliente** → Dispositivo do usuário (navegador, app).

**Servidor** → Computador que hospeda o site/aplicação e responde às requisições.

- **Principais tipos de servidores:**

- Servidor Web (ex: Apache, Nginx).
- Servidor de Aplicação (ex: Node.js, Express).
- Servidor de Banco de Dados (ex: MySQL, MongoDB).

- **Exemplo:**

O Google Drive armazena arquivos em servidores, mas o usuário acessa via um navegador (cliente).

# O que é HTTP e HTTPS?

**HTTP (Hypertext Transfer Protocol)** → Protocolo de comunicação entre cliente e servidor.

**HTTPS** → Versão segura com criptografia via **SSL/TLS**.

- **Principais Métodos HTTP:**

- **GET** → Solicita dados (Ex: acessar uma página).
- **POST** → Envia dados ao servidor (Ex: login, formulários).
- **PUT** → Atualiza um recurso.
- **DELETE** → Remove um recurso.

- **Importância do HTTPS:**

- Protege informações do usuário.
- Essencial para sites que lidam com pagamentos e dados sensíveis.

# O que é uma URL?

URL (Uniform Resource Locator) → Endereço que identifica um recurso na web.

Componentes de uma URL:

<https://www.exemplo.com:8080/pagina?busca=teste#secao1>

- **Protocolo** → https://
- **Domínio** → www.exemplo.com
- **Porta** (opcional) → :8080
- **Caminho** → /pagina
- **Parâmetro (Query String)** → ?busca=teste
- **Fragmento** → #secao1

# Navegadores e Motores de Renderização

- **O que são navegadores?**
  - Programas que interpretam código web e exibem páginas (ex: Chrome, Firefox, Edge).
- **Motores de Renderização:**
  - **Blink** (Chrome, Edge, Opera).
  - **Gecko** (Firefox).
  - **WebKit** (Safari).
- **Como um navegador processa uma página?**
  - Baixa o HTML, CSS e JavaScript.
  - Constrói a **DOM (Document Object Model)**.
  - Aplica estilos CSS.
  - Executa JavaScript e renderiza a página.

# Sites Estáticos vs. Dinâmicos

- **Sites Estáticos**

- Criados apenas com HTML e CSS.
- Exemplo: Portfólio, landing pages simples.

- **Sites Dinâmicos**

- Utilizam JavaScript, bancos de dados e servidores.
- Exemplo: Redes sociais, lojas virtuais, dashboards.



# Tipos de Aplicações Web

- A Web evoluiu e, com isso, surgiram diferentes tipos de aplicações que atendem a diversas necessidades dos usuários e empresas.
- Vamos explorar os principais tipos:
  - Websites
  - Web Apps
  - PWAs

# Websites (Sites Estáticos e Dinâmicos)

- **O que são?**
  - São páginas acessadas via navegador, que podem conter informações, imagens, vídeos e links.
  - Podem ser **estáticos** (conteúdo fixo) ou **dinâmicos** (conteúdo gerado a partir de um banco de dados).
- **Características:**
  - Estruturados com **HTML, CSS e JavaScript**.
  - Objetivo principal: Apresentação de informações.
  - Não exigem interações complexas do usuário.
- **Exemplos:** Blogs, sites institucionais, portfólios e notícias.

# Web Apps (Aplicações Web Dinâmicas)

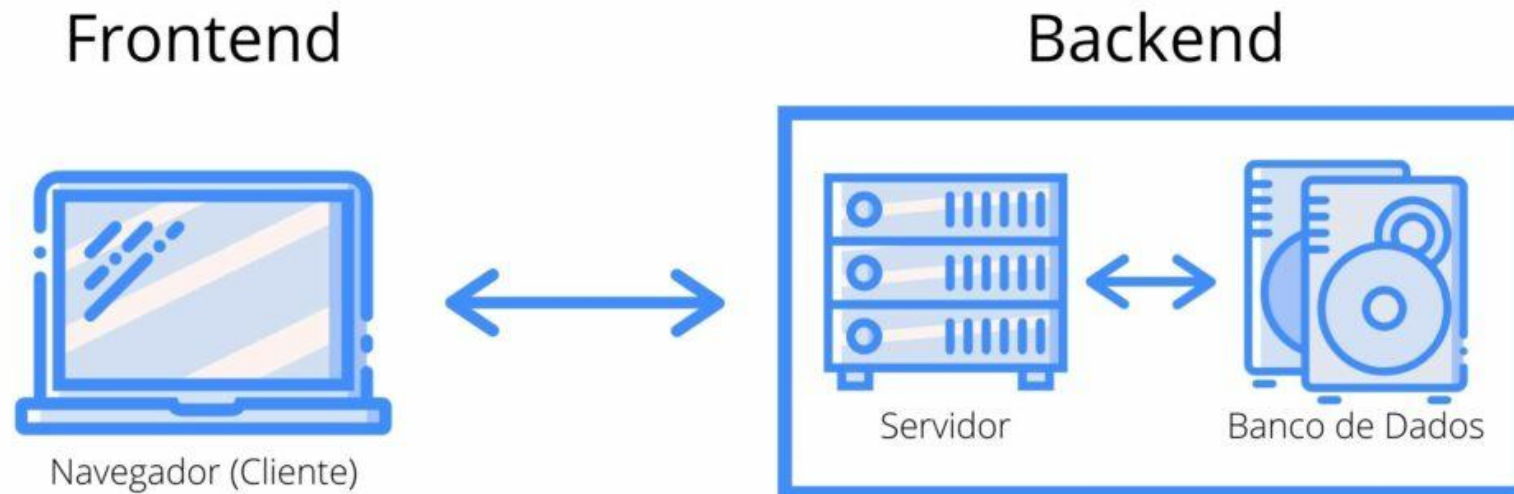
- **O que são?**
  - São sistemas interativos acessados pelo navegador, que funcionam como aplicativos.
  - Diferente de um website comum, permite que os usuários realizem ações, preencham formulários, façam login e interajam de maneira avançada.
- **Características:**
  - Interação dinâmica, geralmente construída com **JavaScript, React, Angular, Vue.js**.
  - Usa **APIs e Banco de Dados** para gerenciar informações.
  - Pode oferecer funcionalidades semelhantes a aplicativos móveis.
- **Exemplos:** Gmail, Trello, Facebook, Google Docs, sistemas de e-commerce.

# PWA (Progressive Web Apps – Aplicações Web Progressivas)

- **O que são?**
  - Aplicações web que combinam recursos de sites e aplicativos móveis, permitindo acesso offline, notificações push e instalação no dispositivo sem precisar de uma loja de apps.
- **Características:**
  - **Responsivos** e adaptáveis a diferentes dispositivos.
  - Utilizam **Service Workers** para permitir funcionamento offline.
  - Possuem **Performance aprimorada** e podem ser instalados no celular.
  - São mais leves do que aplicativos nativos.
- **Exemplos:** Twitter Lite, Uber Web App.

# Front-end e Back-end o que são?

- Front-end e back-end são partes fundamentais de uma aplicação, que se diferenciam pela interação com o usuário e pelo processamento de dados.



# Front-end

- É a parte da aplicação que os usuários veem e interagem
- Inclui elementos visuais como botões, caixas de seleção, gráficos e mensagens de texto
- É responsável por criar a interface de um site
- Utiliza tecnologias como HTML, JavaScript e CSS

# Back-end

- É a parte da aplicação que lida com a lógica de negócios, o processamento de dados e a comunicação com o banco de dados
- Armazena e processa dados da aplicação para os usuários
- É responsável por garantir que os dados sejam transmitidos de forma segura
- Utiliza linguagens como Java, Python ou o próprio Javascript

# Front-end vs Back-end

| Característica         | Front end  | Back end   |
|------------------------|--|--|
| O que é?               | Parte visual e interativa de um site ou aplicação.       | Parte lógica e funcional que processa dados e interações.              |
| Responsabilidade       | Criar a interface e experiência do usuário.              | Gerenciar dados, regras de negócio e comunicação com o banco de dados. |
| Principais Tecnologias | HTML, CSS, JavaScript, React, Angular, Vue.js.           | Node.js, Python (Django, Flask), Java (Spring), PHP, Ruby on Rails.    |
| Execução               | No navegador do usuário.                                 | No servidor remoto.  |
| Exemplo de Tarefa      | Criar um botão estilizado e responsivo.                  | Processar um pedido de compra em um e-commerce.                        |
| Interação              | Envia requisições para o backend via APIs.               | Responde às requisições do front end com dados e processamento.        |
| Desempenho             | Depende da otimização de código e tempo de carregamento. | Depende da eficiência do servidor e do banco de dados.                 |



# Configuração do ambiente de desenvolvimento

- Instalação do VS Code
- Introdução ao Visual Studio Code
  - Criar novo projeto
  - Adicionar extensões para programação front-end