CENTRO UNIVERSITÁRIO
FMU
LEVEL UP

DESENVOLVIMENTO FRONT-END UX/UI

https://portal.fmu.br

DESENVOLVIMENTO FRONT-END

Público-alvo: Estudantes do Centro Universitário FMU.

Objetivo: Esta apostila tem como objetivo unificar os fundamentos teóricos da **Interação Humano-Computador (IHC)** e do design de **Experiência do Usuário (UX/UI)** com as competências práticas da **Programação Front-End**, capacitando o aluno a analisar, projetar e implementar sistemas computacionais centrados no usuário. Através de uma abordagem que integra pensamento lógico e visão estratégica, o material busca desenvolver a habilidade de transformar requisitos complexos em interfaces eficazes, acessíveis e comunicáveis, utilizando as linguagens HTML, CSS e JavaScript como ferramentas de operacionalização da profissão.

O foco pedagógico está em fornecer uma base sólida sobre os fatores humanos, ergonomia e processos de design (ciclos de vida), permitindo que os estudantes não apenas construam páginas web, mas também avaliem a usabilidade de sistemas sob uma ótica técnica e emocional. Ao concluir este material, o aluno terá o domínio de sistemas de informação necessário para criar protótipos de baixa a alta fidelidade e realizar a manutenção de sistemas com foco na qualidade da interação, garantindo que a tecnologia atue como uma alavanca de eficiência e vantagem competitiva no mercado de trabalho.

Professor: Robson Carmo, mestre pelo programa de mestrado profissional em Gestão e Tecnologia em Sistemas Produtivos do Centro Paula Souza, concluiu 3 MBAs (Inovação e Transformação Digital, Gestão de Empresarial, Engenharia de Software). Instrutor de certificações da SAFE, auto e coautor de vários livros e ebooks.

Lattes: http://lattes.cnpq.br/8753112683856964

Orcid: https://orcid.org/0009-0002-7102-4993

LLM utilizada: Gemini 3 – Fevereiro de 2026.

Módulos e Conteúdo

Unidade 1: Fundamentos de IHC e Estruturação Web

Foco na compreensão da interação e na base semântica da web.

- **Introdução à IHC e Benefícios:** Conceitos fundamentais, áreas relacionadas e a importância do design centrado no usuário.
- **Interface, Interação e Affordance:** Definição de tipos de interface e como o design "convida" a ação.
- **Fatores Humanos e Ergonomia:** Dimensões sociais, emocionais e físicas que influenciam o desempenho e conforto na interação.
- **Introdução ao HTML5:** Declarações de documentos, tags de texto, imagens e semântica.
- **Organização com Listas e Tabelas:** Estruturação de dados e informações hierárquicas.

Unidade 2: Estética, Cognição e Bootstrap

Conectando processos mentais à construção visual e ao uso de frameworks de mercado.

- **Cognição e Frameworks:** Processos cognitivos (percepção, memória, atenção) e frameworks teóricos de IHC.
- **CSS3 - Folhas de Estilo:** Sintaxe, seletores, cores, tipografia e o conceito de *Box Model*.
- **Layout e Posicionamento:** Domínio de estruturas visuais e alinhamento de elementos.
- **Bootstrap Framework:**
    - **Sistema de Grid:** Criação de layouts responsivos utilizando colunas e containers.
    - **Componentes:** Uso de botões, cards, barras de navegação (navbars) e modais pré-estilizados.
    - **Utilidades:** Classes auxiliares para espaçamento (margin/padding) e tipografia rápida.

Unidade 3: Design de Experiência e Prototipação

Planejamento estratégico e validação de requisitos antes da implementação.

- **Processos de Design e Ciclos de Vida:** Metodologias para o desenvolvimento de IHC.
- **Análise de Requisitos e Coleta de Dados:** Identificação de necessidades e metas de usabilidade.
- **Modelagem e Prototipação:** Design conceitual vs. físico, storyboarding e protótipos de baixa, média e alta fidelidade.
- **Introdução ao JavaScript:** Sintaxe, variáveis, operadores e desenvolvimento de códigos de interação.
- **Manipulação do DOM:** Inclusão de scripts para tornar as páginas dinâmicas e interativas.

Unidade 4: Comportamento e Interatividade

Implementação de lógica e coleta de dados dinâmica.

- **Links, Âncoras e Navegação:** Fluxos de navegação e usabilidade para internet.
- **Formulários:** Criação de componentes de entrada de dados e validação de requisitos.
- **Interação Social e Emocional:** Como o design influencia a adoção de tecnologias e a experiência emocional.

Unidade 5: Organização de Código e Arquitetura Clean No Front-End

Boas práticas para criação de uma solução web.

- **Sintaxe Semântica e Convenções de Nomenclatura:** Uso avançado de metodologias para CSS (como o padrão BEM - Block, Element, Modifier) e padronização de nomenclatura de identificadores e classes no HTML/Bootstrap.
- **Componentização e Reutilização de Código:** Como quebrar layouts complexos em pequenos blocos isolados e autossuficientes. Boas práticas na segregação de responsabilidades (HTML para estrutura, CSS para apresentação e JavaScript puramente para comportamento).
- **Versionamento e Governança de Código:** Introdução prática ao fluxo de trabalho com Git (padrões de Conventional CLinks, Âncoras e Navegação: Fluxos de navegação e usabilidade para internet.

Unidade 1: Fundamentos de IHC e Estruturação Web

1.1. Introdução à IHC: A Ciência por Trás da Interface

A **Interação Humano-Computador (IHC)** é uma área multidisciplinar que estuda como as pessoas interagem com sistemas computacionais e como projetar tecnologias que sejam eficazes, seguras e satisfatórias. No contexto de **Tecnologia da Informação (TI)**, não basta que um algoritmo seja eficiente; sua interface deve permitir que o usuário opere o sistema sem erros cognitivos.

IHC como Alavanca Estratégica

Em organizações orientadas a dados, a IHC não é apenas "estética", mas uma alavanca central de crescimento e eficiência operacional. O objetivo profissional de um executivo de tecnologia (CTO/CIO) é liderar estratégias de plataformas digitais que tratem a tecnologia como vantagem competitiva. No desenvolvimento de sistemas modernos, a IHC garante que a complexidade técnica de um software se traduza em valor estratégico para o negócio.

Design como Processo de Comunicação

De acordo com **Geest (2001)**, o design de interfaces web deve ser compreendido fundamentalmente como **design de comunicação**. O site não é um objeto estático; é um canal onde o desenvolvedor comunica intenções e funcionalidades ao usuário. Se a estrutura do código (HTML) não reflete essa mensagem de forma clara, a comunicação falha, resultando em baixa usabilidade.

1.2. Interface, Interação e o Poder do Affordance

Para o recém-ingresso na área de **TI**, é vital distinguir entre a "ferramenta" e o "uso".

- **Interface**: É a superfície de contato, o conjunto de elementos que permite a interação entre o usuário e o sistema.
- **Interação**: É o processo de diálogo e troca entre o humano e a máquina.
- **Affordance**: Este termo, essencial na obra de **Leung (2008)**, refere-se às propriedades de um objeto digital que **"convidam"** o usuário a agir.

Principais Características da Afordância

- **Intuitividade:** A forma do objeto indica sua função (ex: uma maçaneta redonda sugere "girar", uma maçaneta em alavanca sugere "abaixar").
- **Relação Agente-Ambiente:** Não é apenas uma propriedade do objeto, mas a relação entre as propriedades físicas do objeto e as habilidades de quem o usa.
- **Percepção:** No design, o foco está na "afordância percebida" – o que o usuário acha que pode fazer, baseado na aparência.

Exemplos de Afordância

- **Físico**: Uma cadeira oferece a possibilidade de sentar; um botão físico oferece a possibilidade de ser pressionado.
- **Digital (UI/UX)**: Um botão com sombra (estilo skeuomorphic) sugere que ele pode ser clicado por parecer elevado. Um ícone de "lixeira" sugere exclusão.
- **Afordância Falsa**: Quando um objeto sugere uma ação que não existe (ex: um botão falso que não funciona ou um elemento de texto que parece um link, mas não é).

Por que a interface importa?

Muitas vezes, o programador foca apenas no código, mas esquece de quem vai usar o sistema. A **Interação Humano-Computador (IHC)** estuda como facilitar essa relação.

- **Affordance (afordância)**: É a característica de um objeto que indica **como deve ser usado**. Exemplo: Um botão em um site que tem uma sombra "parece" clicável. Se ele parecer apenas um texto plano, o usuário pode não saber que deve clicar ali.
- **Usabilidade**: O quão fácil e eficiente é para o usuário realizar uma tarefa. Exemplo: Se para fazer um PIX você precisar clicar em 15 telas diferentes, a usabilidade é ruim.
- **Acessibilidade**: Garantir que o site possa ser usado por todos, incluindo pessoas com deficiência visual ou motora.

Exemplo Prático de Affordance: Em um sistema de larga escala, como os utilizados no setor financeiro, um botão que parece “pressionável” reduz o erro operacional e aumenta a eficiência. Se um elemento clicável parece apenas um texto estático, a "comunicabilidade" do sistema falhou.

1.3. O Esqueleto da Web: Introdução ao HTML

Iniciamos agora a parte técnica tendo como foco simplicidade e prática imediata para quem está começando em TI.

O que é HTML?

HTML significa **HyperText Markup Language** (Linguagem de Marcação de Hipertexto).

- É a linguagem de marcação padrão para criar páginas Web.
- HTML descreve a **estrutura** de uma página Web.
- Consiste em uma série de **elementos** (tags) que dizem ao navegador como exibir o conteúdo.

Um Documento HTML Simples

Todo projeto profissional de Front-End começa com uma estrutura padrão. Vamos analisar cada linha:

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Olá, Mundo! Sou aluno da FMU.</h1>
    <p>Este é o meu primeiro passo no Front-End.</p>
</body>
</html>
```

exercicio_01.html

Explicação Detalhada:

<!DOCTYPE html>: Declaração que define que este documento é HTML5.

<html>: O elemento raiz de uma página HTML.

<head>: Contém informações meta sobre o documento que não aparecem na página.

<title>: Especifica o título exibido na aba do navegador.

<body>: Define o corpo do documento e contém todo o conteúdo visível.

Uma forma prática para testar códigos HTML é usando um editor online da W3School: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default

1.4. Títulos e Parágrafos: A Hierarquia da Informação

Como destacado por **Dell (2015)**, o design de interfaces digitais depende da hierarquia visual para guiar a atenção do usuário.

Títulos (Headings)

Os títulos HTML são definidos com as tags <h1> a <h6>.

<h1> define o título mais importante.

<h6> define o título menos importante.

Parágrafos (Paragraphs)

O elemento <p> define um parágrafo. O navegador adiciona automaticamente algum espaço (margem) antes e depois de cada elemento <p>.

1.5. Atributos HTML e Acessibilidade

Atributos fornecem informações adicionais sobre os elementos.

- Todos os elementos HTML podem ter atributos.
- Atributos são sempre especificados na **tag de abertura**.

O Atributo alt em Imagens: Para garantir a **acessibilidade** (um pilar de IHC), toda imagem deve ter um atributo alt. Se o usuário for deficiente visual, o leitor de tela usará esse texto para descrever a imagem.

1.6. Formatação e Ênfase: Comunicando Importância

No design de comunicação, conforme defendido por **Geest (2001)**, a forma como destacamos certas palavras altera a percepção da mensagem pelo usuário. Em HTML, utilizamos elementos específicos para dar ênfase visual e semântica aos dados.

A Tag (Importância Forte)

A tag é usada para definir um texto com forte importância.

- **Significado Semântico**: Ela indica ao navegador e aos leitores de tela que o conteúdo é importante.
- **Visual**: Por padrão, os navegadores exibem o texto dentro de em **negrito**.
- **Uso em TI**: É ideal para destacar alertas, palavras-chave em requisitos de sistemas ou termos críticos em contratos digitais.

A Tag (Ênfase)

A tag é usada para enfatizar um texto.

- **Significado Semântico**: Indica uma ênfase verbal ou idiomática.
- **Visual**: Por padrão, os navegadores exibem o texto em *itálico*.
- **Uso em TI**: Útil para termos técnicos em estrangeirismo ou citações em manuais de usuário.

Nota de IHC: Segundo **Dell (2015)**, o uso excessivo de negrito ou itálico pode poluir a interface e aumentar a carga cognitiva, dificultando a escaneabilidade da página. Use com moderação para manter a eficácia da comunicação.

1.7. Imagens e a Acessibilidade na Web

A inclusão de elementos visuais é parte essencial do design de interfaces modernas. No entanto, para alunos de **TI**, a implementação deve seguir padrões técnicos rigorosos.

A Tag <img>

A tag <img> é usada para incorporar uma imagem em uma página HTML. Ela é uma "tag vazia", o que significa que não possui uma tag de fechamento (como </img>).

- **Atributo src (source)**: Especifica o caminho (URL) para a imagem.
- **Atributo alt (alternate text)**: Fornece um texto alternativo para a imagem.

A Importância do alt para a Acessibilidade

Conforme os objetivos de aprendizagem da disciplina, o aluno deve ser capaz de criar interfaces acessíveis. O atributo alt é vital para:

1. **Leitores de Tela**: Usuários com deficiência visual dependem do alt para entender o que a imagem representa.
2. **Falha de Carregamento**: Se a internet estiver lenta ou o link quebrar, o texto do alt aparecerá no lugar da imagem.

1.8. Listas: Organizando a Lógica e os Processos

A organização de informações em listas é uma técnica poderosa de **UX (User Experience)** para facilitar a leitura rápida.

Listas Não Ordenadas (<ul>)

Usadas quando a ordem dos itens não importa (ex: uma lista de tecnologias utilizadas em um projeto).

- Utiliza a tag <ul> (Unordered List) e itens <li> (List Item).

Listas Ordenadas (<ol>)

Usadas para sequências lógicas, como algoritmos ou passos de instalação de um software.

- Utiliza a tag <ol> (Ordered List) e itens <li>.

1.9. Tags Semânticas de Estrutura

Diferente das tags genéricas, as **tags semânticas** descrevem seu significado tanto para o desenvolvedor quanto para a máquina.

- **<nav>**: Define um conjunto de links de navegação. É essencial para a **usabilidade**, indicando ao usuário onde ele pode transitar no sistema.
- **<header>**: Define um cabeçalho para um documento ou seção. Geralmente contém o logotipo, nome do site e títulos principais.
- **<main>:** Representa o conteúdo central de uma página web, englobando informações diretamente relacionadas ao tema principal ou à funcionalidade da aplicação. As <scection> podem ser criadas dentro destes elementos.
- **<section>**: Define uma seção em um documento. É usada para agrupar conteúdos relacionados tematicamente (ex: "Sobre", "Serviços").
- **<footer>**: Define um rodapé para um documento ou seção. Costuma conter informações de copyright, links de contato e mapas do site.

Exemplo de currículo em HTML:

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Currículo Profissional</title>
</head>
<body>
    <nav>
        <ul>
            <li><a href="#objetivo">Objetivo</a></li>
            <li><a href="#habilidades">Habilidades</a></li>
            <li><a href="#identificacao">Identificação</a></li>
            <li><a href="https://fmu.com.br" target="_blank">Site FMU</a></li>
        </ul>
    </nav>

    <header>
        <h1>[SEU NOME COMPLETO]</h1>
        <p>E-mail: <a href="mailto:seuemail@exemplo.com">Mande um e-mail</a></p>
        <p>Faculdade: <a href="https://fmu.com.br" target="_blank">Visitar FMU</a></p>
        <p>São Paulo/SP | Contato: [SEU E-MAIL]</p>
    </header>

    <hr>

    <section id="objetivo">
        <h2>OBJETIVO</h2>
        <p>
            Atuar como Desenvolvedor de Software, focando em
            <strong>Análise e Desenvolvimento de Sistemas</strong>.
        </p>
    </section>

    <section id="habilidades">
        <h3>Habilidades Técnicas</h3>
        <ul>
            <li>HTML5 Estrutural</li>
            <li>Lógica de Programação</li>
            <li>Ambiente Windows e VS Code</li>
        </ul>
    </section>

    <section id="identificacao">
        <h2>IDENTIFICAÇÃO VISUAL</h2>
        <img src="foto.webp" alt="Foto de perfil do aluno de ADS" width="150">
        <p><em>Legenda: Estudante de tecnologia em formação pela FMU.</em></p>
    </section>

    <footer>
        <hr>
        <p>&copy; 2026 - Desenvolvido na aula de Front-End</p>
    </footer>

</body>
</html>
```

exercicio_2.html

1.10. Tabelas: Estruturando Dados de Sistemas

As tabelas são fundamentais no domínio de sistemas de informação para exibir dados comparativos. Embora a mineração de dados utilize algoritmos complexos, o resultado para o usuário final é frequentemente uma tabela.

<table>: Define a tabela.

<tr>: Define uma linha (Table Row).

<th>: Define um cabeçalho (Table Header).

<td>: Define um dado (Table Data).

1.11. Semântica em Tabelas: Organizando Dados Complexos

Embora as tabelas básicas organizem dados, sistemas complexos — como os que lidam com **Big Data** (MORAIS et al., 2018) ou resultados de **Mineração de Dados** (CASTRO; FERRARI, 2016) — exigem uma estrutura mais robusta para facilitar a análise de requisitos.

<thead> (Cabeçalho da Tabela)

- **Conceito**: Agrupa o conteúdo do cabeçalho de uma tabela.
- **Uso**: Dentro dele, colocamos as linhas (<tr>) que contêm os títulos das colunas (<th>). Isso ajuda o navegador a entender o que são os rótulos dos dados.

<tbody> (Corpo da Tabela)

- **Conceito**: Agrupa o conteúdo principal ou o "corpo" da tabela.
- **Uso**: É onde inserimos os dados reais (<td>) que serão processados ou visualizados pelo usuário. Em sistemas de gestão como o **ISY.ONE**, é aqui que listamos centenas de notas fiscais ou produtos.

<tfoot> (Rodapé da Tabela)

- **Conceito**: Agrupa o conteúdo do rodapé em uma tabela.
- **Uso**: Geralmente utilizado para exibir totais, médias ou resumos dos dados apresentados no <tbody>.

Visão Estratégica: De acordo com **Dell (2015)**, organizar tabelas com essas tags melhora a **usabilidade**, pois permite que navegadores modernos mantenham o cabeçalho visível enquanto o usuário rola por longas listas de dados.

Exemplo 1: Tabela Simples (Sem Semântica Avançada)

Este modelo é funcional, mas menos legível para sistemas de grande escala.

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>FMU</title>
</head>
<body>
    <table border="1">
        <tr>
            <th>Produto</th>
            <th>Quantidade</th>
        </tr>
        <tr>
            <td>Licença Odoo</td>
            <td>10</td>
        </tr>
        <tr>
            <td>Total</td>
            <td>10</td>
        </tr>
    </table>
</body>
</html>
```

exercício_03.html

Exemplo 2: Tabela Estruturada (Com thead, tbody e tfoot)

Este é o padrão exigido em projetos de TI para garantir manutenção de sistemas e acessibilidade.

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>FMU</title>
</head>
<body>
    <table border="1">
        <thead>
            <tr>
                <th>Produto</th>
                <th>Quantidade</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Licença Odoo</td>
                <td>10</td>
            </tr>
            <tr>
                <td>Consultoria Python</td>
                <td>5</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td>Total de Itens</td>
                <td>15</td>
            </tr>
        </tfoot>
    </table>
</body>
</html>
```

exercio_04.html

A W3Schools disponibiliza um poderoso Tutorial de HTML:
https://www.w3schools.com/html/

Revisão do Conteúdo

I. Questões de Múltipla Escolha

1. Qual o significado da sigla HTML?

   a) Hyper Tool Markup Language

   b) HyperText Markup Language

   c) High Tech Modern Language

   d) Hyperlinks and Text Management Language

2. Qual elemento é o "raiz" de um documento HTML?

   a) <body>

   b) <head>

   c) <html>

   d) <!DOCTYPE>

3. Segundo Geest (2001), o design de um site é um processo de:

   a) Programação pura.

   b) Comunicação.

   c) Armazenamento de dados.

   d) Criptografia.

4. Qual tag define o título de maior importância?

   a) <head>

   b) <title>

   c) <h1>

   d) <h6>

5. Qual atributo define o destino de um link na tag <a>?

   a) src

   b) link

   c) href

   d) target

6. O que define "Affordance" em IHC?

   a) O custo da interface.

   b) A característica que sugere o uso do objeto.

   c) A velocidade do algoritmo.

   d) A memória da página

7. Onde ficam as informações meta (não visíveis)?

   a) No <body>

   b) No <head>

   c) No <footer>

   d) No <h1>

8. Para que serve a tag <p>?

   a) Para pontos de interrogação.

   b) Para definir um parágrafo.

   c) Para inserir imagens.

   d) Para definir o peso da fonte.

9. Qual a função de <!DOCTYPE html>?

   a) Exibir o nome do autor.

   b) Informar a versão do HTML.

   c) Criar um cabeçalho.

   d) Inserir comentários.

10. **Atributos devem ser colocados na:**

    a) Tag de fechamento.

    b) Entre as tags.

    c) Tag de abertura.

    d) No arquivo CSS apenas.

11. **Qual a função da tag <nav> em um projeto de TI?**

    a) Armazenar o banco de dados.

    b) Agrupar links de navegação principal.

    c) Definir o rodapé da página.

    d) Inserir scripts de JavaScript.

12. **Para que serve a tag <tbody> em uma tabela?**

    a) Para definir o título da tabela.

    b) Para agrupar as linhas que contêm os dados principais.

    c) Para criar uma linha de cabeçalho.

    d) Para definir a cor de fundo da tabela.

II. Perguntas Reflexivas

1. **Cenário de Negócios (SaaS):** Como o uso de **Affordance** (Leung, 2008) e a **Comunicação** (Geest, 2001) podem reduzir o custo de suporte técnico ao tornar os botões de ação autoexplicativos?

2. **Cenário de Governança:** Como a correta aplicação da **hierarquia de títulos (H1-H6)** e da **acessibilidade** (Dell, 2015) pode mitigar riscos de exclusão digital e melhorar o posicionamento estratégico de uma empresa de **TI** no mercado internacional?

III. Exercícios Práticos

1. **Estrutura Semântica Profissional:** No VS Code, crie um arquivo exercicio_05.html. Construa a estrutura básica do HTML5 e crie um cabeçalho para uma empresa de consultoria de **TI**. Utilize um <h1> para o nome da empresa e <h2> para os serviços oferecidos. Adicione um parágrafo <p> explicando a missão da empresa usando as tags para dar importância a palavras-chave.

2. **Affordance e Links:** Crie uma seção de "Contato" no seu HTML. Insira um link (<a>) para o seu e-mail e outro para o portal da faculdade. Garanta que o link externo abra em uma nova aba usando o atributo adequado. Verifique se, visualmente, os links possuem affordance (cor e sublinhado) que os diferencie do texto comum.

3. **Estrutura de Consultoria de TI com Ênfase Semântica:** No **Visual Studio Code**, crie um arquivo chamado exercício_06.html.

   - Crie um título <h1> com o nome "TechSolution Consultoria".
   - Crie uma seção <section> com um <h2> chamado "Nossa Missão".
   - Escreva um parágrafo <p> explicando que a empresa foca em **estratégia de tecnologia** e **segurança de dados**. Utilize a tag nas palavras "estratégia" e "segurança" para destacar a importância desses pilares para o negócio.
   - Adicione uma imagem com um ícone de tecnologia e não esqueça do atributo alt.

4. **Tabela de Requisitos:** Crie um novo arquivo chamado exercicio_7.html.

   - Utilize o conceito de listas ordenadas (<ol>) para descrever os 3 passos para configurar um ambiente de desenvolvimento (ex: 1. Baixar VS Code; 2. Instalar extensões; 3. Criar pasta do projeto).
   - Abaixo, crie uma tabela (<table>) com as colunas "Componente" e "Versão Mínima", listando ao menos dois softwares (ex: Chrome e VS Code) e suas respectivas versões. Use <th> para o cabeçalho.

5. Layout Semântico de Portal de TI:

   Crie um arquivo exercicio_8.html. Estruture a página utilizando as tags <header>, <nav>, <section> e <footer>. No <nav>, crie uma lista com links para "Início", "Tecnologia" e "Contato". Na <section>, escreva sobre a importância da IA e Big Data conforme mencionado por Silva (2019) e Morais (2018).

6. Tabela de Inventário de Hardware:

   Crie um arquivo exercicio_9.html. Desenvolva uma tabela utilizando as tags <thead>, <tbody> e <tfoot>. No cabeçalho, coloque "Equipamento" e "Status". No corpo, liste 3 itens (ex: Servidor, Roteador, Workstation). No rodapé, coloque uma nota informando a data da última verificação, reforçando a competência de manutenção de sistemas.

2.1. Cognição e Frameworks de IHC: O Usuário por Dentro

2.1.1. Breve História da Cognição na TI

A relação entre a mente humana e a máquina não começou com o HTML, mas sim com a necessidade de processamento de informações durante a Segunda Guerra Mundial e o surgimento dos primeiros computadores de grande porte (Mainframes). No início, a interação era puramente textual e técnica; o usuário precisava se adaptar à máquina.

A partir da década de 1970, com o surgimento da **Psicologia Cognitiva**, os cientistas da computação perceberam que, para a tecnologia ser adotada em larga escala, a máquina deveria se adaptar ao funcionamento do cérebro. Surgia a **Interação Humano-Computador (IHC)**. Autores como **Dell (2015)** destacam que a transição do "usuário técnico" para o "usuário comum" exigiu que as interfaces passassem a respeitar limites biológicos, como a velocidade de processamento visual e a retenção de memória.

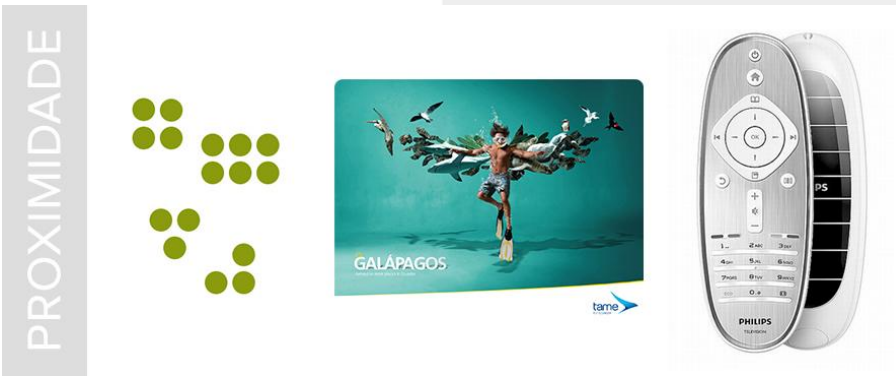
2.1.2. Processos Cognitivos: O "Hardware" Humano

Como profissionais de TI, devemos tratar o cérebro do usuário como um sistema de processamento que possui latência, memória cache e barramentos de entrada/saída.

A. Percepção e a Teoria da Gestalt

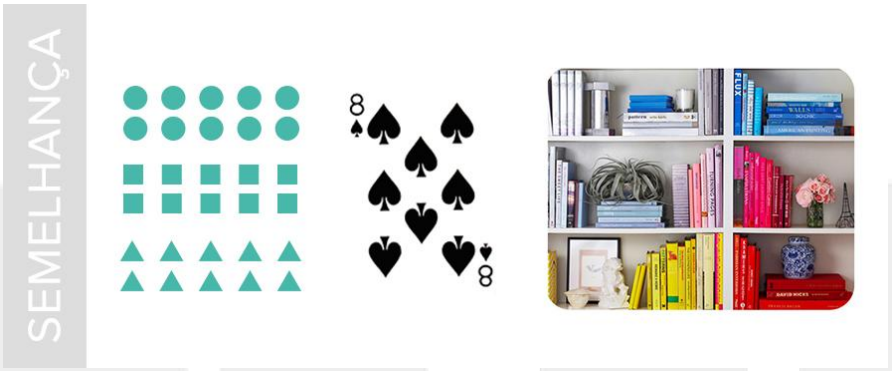
A percepção é o processo pelo qual organizamos e interpretamos estímulos sensoriais para entender o ambiente. Em IHC, a visão é o canal principal.

- **Leis da Gestalt:** São princípios que descrevem como o cérebro agrupa elementos visuais.
    - **Proximidade:** Itens próximos são percebidos como parte de um mesmo grupo. Se você coloca o rótulo de um formulário muito longe do campo de entrada, o usuário levará mais milissegundos para processar a relação entre eles.



Fonte: https://blog.render.com.br/design-grafico/aumente-a-compreensao-de-seu-trabalho-com-a-gestalt/

- **Semelhança:** Elementos com a mesma cor ou forma são percebidos como tendo a mesma função.



Fonte: https://blog.render.com.br/design-grafico/aumente-a-compreensao-de-seu-trabalho-com-a-gestalt/

- **Exemplo de Negócio:** Em sistemas de **Mineração de Dados (Castro & Ferrari, 2016)**, a percepção de padrões em gráficos de dispersão depende inteiramente de como a interface aplica as cores e formas para diferenciar clusters de dados.

B. Atenção e Carga Cognitiva

A atenção é um recurso mental limitado e seletivo.

- **Carga Cognitiva:** É o esforço mental total utilizado na memória de trabalho. Se uma interface de um sistema de **Internet das Coisas (IoT)**, como discutido por **Morais (2018)**, exibe centenas de alertas de sensores ao mesmo tempo, o usuário sofre de **sobrecarga cognitiva**, o que leva à paralisia de decisão ou erro operacional.
- **Fatores de Distração:** Pop-ups, cores excessivamente vibrantes e animações sem propósito competem pela atenção do usuário, desviando-o da tarefa principal.

C. Memória: Reconhecimento vs. Evocação

A memória humana é dividida em Curto Prazo (trabalho) e Longo Prazo.

- **Reconhecimento:** Ocorre quando o usuário vê um ícone (ex: uma lixeira) e reconhece sua função. É um processo rápido e de baixo custo energético para o cérebro.

- **Evocação:** Ocorre quando o usuário precisa lembrar um comando (ex: digitar rm -rf no terminal). É um processo lento e propenso a erros.

- **Estratégia de TI:** Designers de interfaces modernas (UX/UI) priorizam menus visuais e ícones universais para reduzir a necessidade de memorização, um conceito fortalecido por **Leung (2008)** na construção de experiências digitais fluidas.

2.1.3. Frameworks Teóricos de IHC

Um framework é uma estrutura de suporte que guia o desenvolvimento. Em IHC, eles servem para garantir que a interface seja uma ponte eficiente, não uma barreira.

A. O Framework de Comunicação de Geest (2001)

Geest argumenta que "Web Site Design is Communication Design". Para ele, a interface é uma conversa assíncrona entre o desenvolvedor e o usuário.

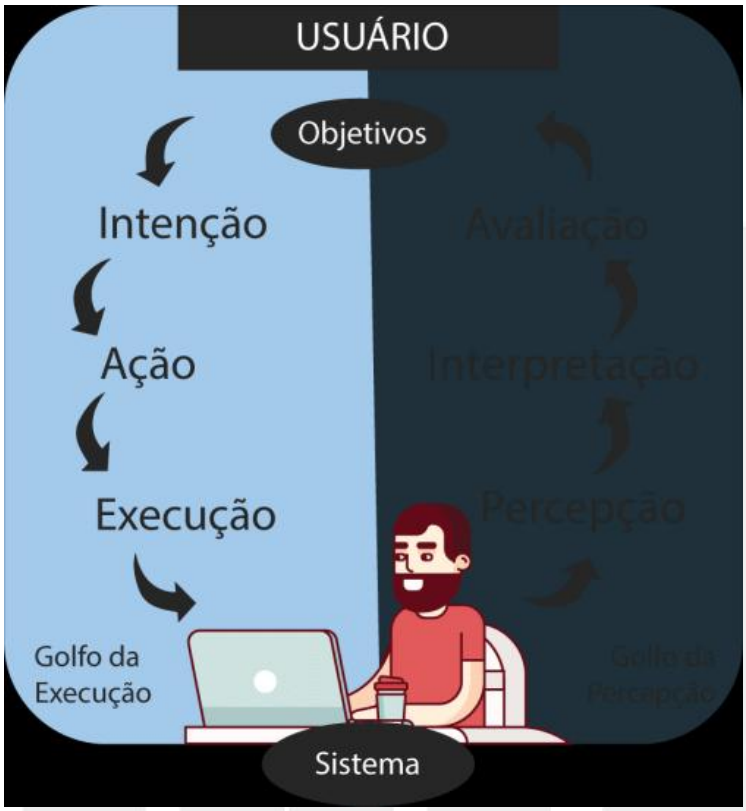
- **Clareza de Intenção:** O código HTML e o estilo CSS devem comunicar ao usuário o que ele pode fazer em cada tela. Se um link não parece um link (falta de **Affordance**), a comunicação falhou.
- **Feedback de Interação:** Para cada ação do usuário, o sistema deve fornecer uma resposta (ex: um botão que muda de cor ao ser clicado ou um "spinner" de carregamento). Sem feedback, o usuário entra em estado de incerteza cognitiva.

B. Ciclo de Interação de Norman (1998)

Donald Norman, um dos pais da usabilidade, propôs um ciclo que explica como o usuário interage com sistemas:

1. **Formação da Intenção:** O usuário quer realizar uma tarefa.
2. **Especificação da Ação:** O usuário decide qual botão clicar.
3. **Execução:** O clique acontece.
4. **Percepção do Estado:** O usuário vê o que aconteceu.

5. **Avaliação:** O usuário decide se o resultado foi o esperado.



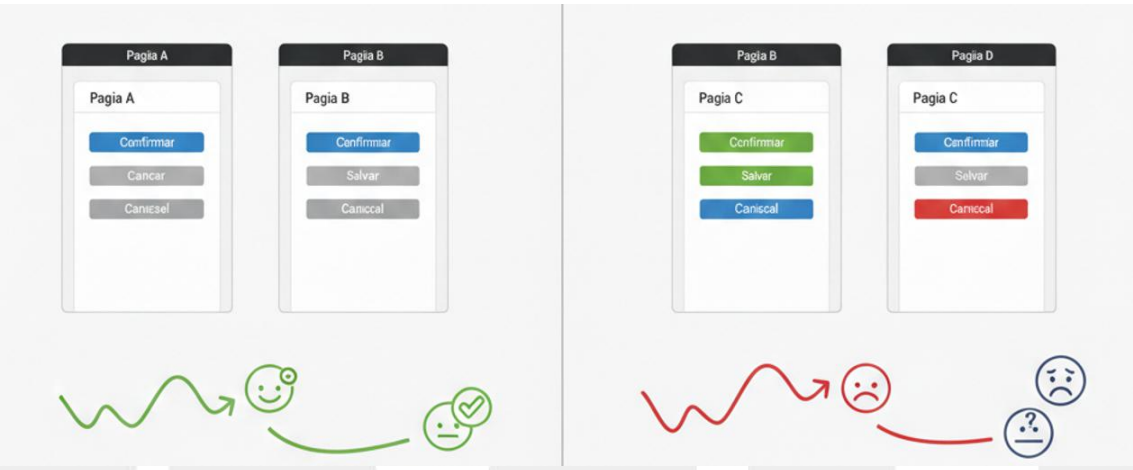
Fonte: https://www.researchgate.net/figure/Figura-4-Os-sete-estagios-da-teoria-da-acao-de-Norman-Para-Norman-1998-um-bom-design_fig4_322147999

- **Exemplo em TI:** Se um usuário de um sistema de **Inteligência Artificial (Silva, 2019)** solicita uma análise e a tela fica estática por 10 segundos, ele interrompe o ciclo de avaliação, achando que o sistema travou, o que gera frustração emocional.

2.1.4. Ergonomia Cognitiva e Interação Emocional

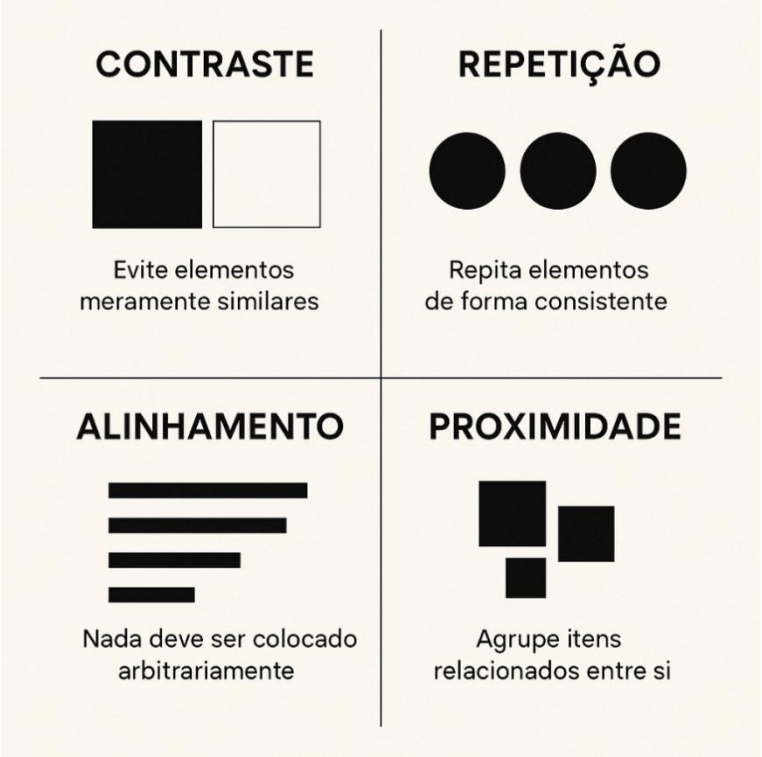
A **Ergonomia Cognitiva** foca na adequação das máquinas às capacidades mentais.

- **Consistência:** Um sistema que usa o botão **"Confirmar" em azul na página A** e em **verde na página B** quebra a consistência cognitiva, forçando o cérebro a reaprender a interface a cada tela.



- **Design Emocional:** Interfaces que transmitem confiança e clareza geram uma experiência positiva. Em sistemas financeiros complexos, o design limpo reduz a ansiedade do usuário ao lidar com grandes quantias de dinheiro.

Robin Williams, apresenta quatro fundamentos essenciais para qualquer conteúdo: contraste, repetição, alinhamento e proximidade.



Fonte: https://www.adtail.ag/post/design-grafico-como-influencia-no-negocio

2.1.5. Exemplo Prático de Aplicação: O Dashboard de IA

Imagine um software de monitoramento de servidores que utiliza **Inteligência Artificial** para prever falhas.

- **Sem IHC (Foco apenas em TI):** O sistema exibe logs de texto puro rolando na tela. Carga cognitiva altíssima.
- **Com IHC e Cognição:** O sistema usa o **Box Model do CSS** para criar cards organizados. Usa cores (Semântica) para destacar problemas (Vermelho = Crítico). Aplica a **Lei da Proximidade** para colocar o gráfico de performance ao lado do botão de "Reiniciar Servidor".

Dessa forma, a tecnologia deixa de ser um obstáculo e passa a ser uma ferramenta de **Visão Estratégica**, permitindo que o gestor de TI tome decisões rápidas com base em dados processados visualmente.

2.2. CSS3 - Folhas de Estilo

CSS significa **Cascading Style Sheets** (Folhas de Estilo em Cascata). Ele economiza muito trabalho, pois pode controlar o layout de várias páginas web de uma só vez.

A W3Schools disponibiliza um poderoso Tutorial de CSS:

https://www.w3schools.com/css/

2.2.1. Exemplo de Página HTML com CSS Interno

Para explicar os conceitos de sintaxe e seletores, utilizaremos o código abaixo. Nele, o CSS é inserido dentro da tag <style>, localizada no <head>.

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
<style>
  /* Isso é um comentário em CSS */
  body {
    background-color: lightblue; /* Cor de fundo da página */
  }

  h1 {
    color: white;             /* Cor do texto */
    text-align: center;       /* Alinhamento */
  }

  p {
    font-family: verdana;     /* Tipo de fonte */
    font-size: 20px;          /* Tamanho da fonte */
  }
</style>
</head>
<body>

    <h1>Minha Primeira Página Estilizada</h1>
    <p>Este é um parágrafo estilizado com CSS seguindo os padrões de TI.</p>

</body>
</html>
```

exercicio_10.html

2.2.2. Sintaxe do CSS

Uma regra CSS consiste em um **seletor** e um **bloco de declaração**.

- **Seletor:** Aponta para o elemento HTML que você deseja estilizar (ex: h1).
- **Bloco de Declaração:** Contém uma ou mais declarações separadas por ponto e vírgula ;. Cada declaração inclui um nome de **propriedade** e um **valor**, separados por dois pontos :.

2.2.3. Seletores CSS (CSS Selectors)

Os seletores são usados para "encontrar" ou selecionar os elementos que você deseja estilizar.

A. Seletor de Elemento

Seleciona elementos com base no nome da tag.

```
<style>
    /* Seleciona todos os <h2> da página */
    h2 {
    text-align: left;
    color: #2c3e50;
    }
</style>
```

B. Seletor de ID

Usa o atributo id de um elemento HTML para selecionar um elemento específico. O ID deve ser **único** na página. Para selecionar por ID, use o caractere #.

```
<style>
    /* Seleciona o elemento com id="cabecalho-unico" */
    #cabecalho-unico {
    background-color: yellow;
    }
</style>
```

C. Seletor de Classe

Seleciona elementos com um atributo class específico. Para selecionar por classe, use o caractere . (ponto).

```
<style>
    /* Disponibiliza uma classe para ser utilizada nos elementos do HTML" */
    .minha-classe {
    background-color: yellow;
    }
</style>
```

2.2.4. Cores no CSS (CSS Colors)

As cores no CSS são especificadas usando nomes de cores predefinidos ou valores RGB, HEX, HSL, RGBA ou HSLA.

- **Nomes:** red, blue, green, orange.
- **HEX (Hexadecimal):** #ff5733 (Muito usado em TI por sua precisão).
- **RGB:** rgb(255, 99, 71) (Vermelho, Verde, Azul).

```
<style>
    h1 {
    background-color: #663399; /* RebeccaPurple */
    color: rgb(255, 255, 255); /* Branco */
    }
</style>
```

2.2.5. Tipografia (CSS Fonts)

A escolha da fonte correta tem um enorme impacto na **Ergonomia Cognitiva** (Dell, 2015).

- **font-family:** Define a fonte (ex: "Arial", "Times New Roman"). Sempre forneça uma "família de fallback" caso a primeira não carregue.
- **font-size:** Define o tamanho. Em TI, usamos **px** para tamanhos fixos ou **rem** para designs responsivos e acessíveis.
- **Root EM (rem):** É uma unidade relativa baseada no tamanho da fonte definido no elemento <html> do documento. Se um usuário aumentar o tamanho da fonte padrão do navegador, todo o seu site baseado em rem aumentará proporcionalmente.

```
<style>
    p {
    font-family: "Lucida Console", "Courier New", monospace;
    font-size: 1.2rem;
    }
</style>
```

2.2.6. O Conceito de Box Model (Modelo de Caixa)

Este é o conceito mais importante do CSS. Em TI, todo elemento HTML é visto como uma caixa retangular. O **Box Model** permite que o desenvolvedor crie espaço ao redor dos elementos e defina bordas.

As Camadas do Box Model:

1. **Content (Conteúdo):** O conteúdo da caixa, onde texto e imagens aparecem.
2. **Padding (Preenchimento):** Limpa uma área ao redor do conteúdo. O preenchimento é **transparente** e fica **dentro** da borda.
3. **Border (Borda):** Uma borda que contorna o preenchimento e o conteúdo.
4. **Margin (Margem):** Limpa uma área **fora** da borda. A margem separa um elemento de outro.

```
<style>
    div {
    width: 300px;
    border: 15px solid green;
    padding: 50px;
    margin: 20px;
    }
</style>
```

Exemplo Prático: Engenharia do Box Model

Copie o código abaixo em um arquivo chamado exercicio_11.html e abra-o no seu navegador para ver a anatomia de um elemento.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Exemplo de Box Model - TI</title>
    <style>
        /* Estilizando o corpo para melhor visualização */
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            padding-top: 50px;
        }

        /* A "Caixa" principal que demonstra o conceito */
        .caixa-exemplo {
            /* 1. CONTENT: Definimos a largura do conteúdo interno */
            width: 300px;
            background-color: white; /* Cor do fundo do conteúdo */

            /* 2. PADDING: Espaço interno entre o conteúdo e a borda */
```

```
            /* Isso evita que o texto "sufoque" nas bordas */
            padding: 40px;

            /* 3. BORDER: A linha que envolve o preenchimento */
            border: 10px solid #2980b9; /* Borda azul sólida */

            /* 4. MARGIN: Espaço externo que afasta esta caixa de outras */
            margin: 20px;

            /* Estética adicional para clareza */
            text-align: center;
            box-shadow: 10px 10px 20px rgba(0,0,0,0.1);
        }

        /* Classe auxiliar para destacar o texto interno */
        .conteudo-texto {
            background-color: #ecf0f1;
            padding: 10px;
            border: 1px dashed #7f8c8d;
        }
    </style>
</head>
<body>

    <div class="caixa-exemplo">
        <div class="conteudo-texto">
            <strong>ÁREA DE CONTEÚDO</strong>
            <p>Aqui é onde os dados (texto/imagem) residem. O espaço ao redor deste
bloco cinza é o <strong>Padding</strong>.</p>
        </div>
    </div>

</body>
</html>
```

exercicio_11.html

Explicação

Ao inspecionar este elemento no navegador (F12 > Elementos), o aluno verá que, embora tenhamos definido width: 300px, o espaço real ocupado na tela é maior.

O cálculo da largura total neste exemplo é:

- **Largura do Conteúdo:** 300px
- **Padding (Esquerda + Direita):** 40px + 40px = 80px
- **Borda (Esquerda + Direita):** 10px + 10px = 20px
- **Margem (Esquerda + Direita):** 20px + 20px = 40px
- **LARGURA TOTAL NA TELA:** 300 + 80 + 20 + 40 = 440px

Em sistemas complexos de TI, entender este cálculo é fundamental para evitar que elementos "quebrem" o layout ou fiquem desalinhados em diferentes resoluções.

2.3. Layout e Posicionamento: Domínio de Estruturas Visuais

Para um profissional de **Tecnologia da Informação**, posicionar elementos não é apenas uma questão de estética, mas de **arquitetura de informação**. Se o Box Model define como cada "caixa" se comporta individualmente, o **Layout** define como essas caixas interagem e se organizam no espaço disponível.

Tradicionalmente, a Web utilizava técnicas limitadas (como tabelas ou *floats*) para criar colunas. Hoje, o CSS3 oferece duas ferramentas poderosas e nativas que você deve dominar: **Flexbox** e **CSS Grid**.

2.3.1. Flexbox (O Modelo de Caixa Flexível)

O Flexbox foi projetado para layouts unidimensionais. Isso significa que ele é excelente para alinhar itens em uma única direção: ou em **linha** (horizontal) ou em **coluna** (vertical).

Conceitos Fundamentais:

- **Flex Container:** O elemento pai onde você aplica display: flex;.
- **Flex Items:** Os elementos filhos diretos que se tornam flexíveis.

Propriedades de Alinhamento:

1. **justify-content:** Alinha os itens no eixo principal (horizontal, por padrão). Ex: center, space-between.
2. **align-items:** Alinha os itens no eixo transversal (vertical, por padrão). Ex: center, stretch.

Este exemplo demonstra como o Flexbox é ideal para criar barras de navegação ou alinhar itens em uma linha de forma proporcional.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Exemplo Flexbox - TI</title>
    <style>
        .flex-container {
            display: flex; /* Ativa o modelo Flexbox */
            background-color: #2c3e50;
            padding: 10px;
            justify-content: space-between; /* Distribui itens com espaço entre eles */
            align-items: center; /* Centraliza verticalmente */
        }

        .flex-item {
            background-color: #ecf0f1;
            padding: 20px;
            margin: 10px;
            border-radius: 5px;
            flex: 1; /* Faz com que todos os itens cresçam por igual */
            text-align: center;
        }
```

```
    </style>
</head>
<body>
    <h2 style="text-align:center">Layout com Flexbox (Unidimensional)</h2>
    <div class="flex-container">
        <div class="flex-item">Módulo 1: IHC</div>
        <div class="flex-item">Módulo 2: CSS3</div>
        <div class="flex-item">Módulo 3: UX/UI</div>
    </div>
</body>
</html>
```

exercicio_12.html

2.3.2. CSS Grid Layout

Enquanto o Flexbox é unidimensional, o **Grid** é bidimensional. Ele permite trabalhar com linhas e colunas ao mesmo tempo, sendo ideal para layouts de páginas inteiras (dashboards, portais de notícias).

Como funciona: Você define uma grade com a propriedade **display: grid**; e especifica o tamanho das colunas e linhas.

Este exemplo mostra a criação de um Dashboard clássico de TI, com cabeçalho, barra lateral, conteúdo principal e rodapé, tudo controlado por uma única grade.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Exemplo CSS Grid - TI</title>
    <style>
        .grid-wrapper {
            display: grid;
            /* Define 2 colunas (lateral fixa e central flexível) e 3 linhas */
            /* A primeira coluna tem exatos 200px (fixa)
               e a segunda ocupa TODO o resto do espaço disponível (flexível). */
            grid-template-columns: 200px 1fr;
            /* A primeira linha tem  altura fixa de 80px. Ideal para logotipos e menus.
               A segunda linha tem altura de 1fr (uma fração). Ela ocupará TODO o espaço
vertical restante da tela de forma flexível.
               A terceira linha tem altura fixa de 60px. Ideal para rodapés e copyright.*/
            grid-template-rows: 80px 1fr 60px;
            /* Cria um "corredor/espaçamento" de 10px entre as colunas e linhas */
            gap: 10px;
            /* O painel ocupará 90% da altura da tela visível
            1vh é igual a 1%*/
            height: 90vh;
        }

        /* Definindo as áreas do Grid */
        header { grid-column: 1 / span 2; background: #34495e; color: white; padding: 20px; }
        nav { background: #95a5a6; padding: 20px; }
        main { background: #ffffff; border: 1px solid #ddd; padding: 20px; }
        footer { grid-column: 1 / span 2; background: #2c3e50; color: white; text-align: center;
padding: 15px; }
    </style>
</head>
<body>
    <div class="grid-wrapper">
        <header><h1>Dashboard de Monitoramento</h1></header>
        <nav>Menu Lateral</nav>
        <main>Conteúdo Principal: Gráficos de Redes e IA</main>
        <footer>© 2026 Tecnologia da Informação</footer>
    </div>
</body>
</html>
```

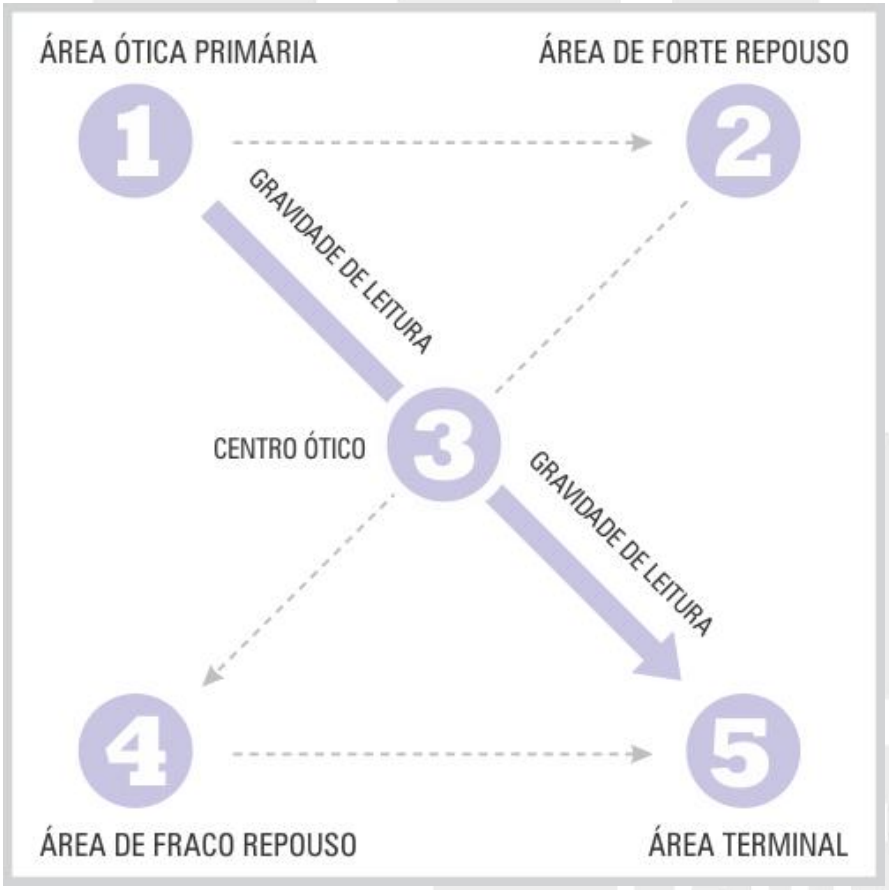
exercicio_13.html

2.3.3. A Visão de IHC no Posicionamento

De acordo com **Dell (2015)** e **Geest (2001)**, o posicionamento deve respeitar o **fluxo de leitura** (da esquerda para a direita, de cima para baixo na cultura ocidental).

- **Consistência:** Elementos críticos, como o botão de "Busca" ou "Login", devem ocupar posições previsíveis (geralmente no topo à direita).
- **Hierarquia:** Itens mais importantes devem ocupar áreas maiores ou posições superiores no layout, reduzindo o tempo de busca visual e a carga cognitiva do usuário.

O diagrama de Gutenberg divide visualmente o espaço em **cinco áreas principais**. Essas áreas guiam os olhos do leitor pela página em uma ordem decrescente de importância.



Fonte: https://brasil.uxdesign.cc/padroes-de-leitura-na-web-a-experiencia-de-escanear-informacao-76d0a846c35a

Conforme o olhar percorre o **caminho em Z**, o usuário identifica áreas de importância decrescente, elementos secundários devem são posicionados nos quadrantes, 2, 3, 4 e 5. A leitura de esquerda para direita é um hábito aprendido desde a infância na maioria das **culturas ocidentais**.

O comportamento de **varredura em F** surge da maneira como lemos na realidade. Nosso campo de visão é mais amplo horizontalmente, facilitando a leitura de linhas de texto. Este comportamento foi identificado nos primórdios da web, nos anos 1980.

Os usuários tendem a fazer uma leitura horizontal, seguida por uma varredura vertical descendente pelo lado esquerdo da página.

Fonte: https://brasil.uxdesign.cc/padroes-de-leitura-na-web-a-experiencia-de-escanear-informacao-76d0a846c35a

2.4. Bootstrap Framework: O Padrão Industrial

Agora que entendemos como o CSS posiciona as caixas, entramos no **Bootstrap**. Como discutido na Unidade 1, em um ambiente de desenvolvimento ágil de TI, ganhar tempo é estratégico. O Bootstrap é um framework que já traz o Flexbox e o Grid "mastigados" em classes prontas.

2.4.1. História e Evolução

O **Bootstrap** foi desenvolvido originalmente por Mark Otto e Jacob Thornton no **Twitter** em meados de 2010, como uma solução interna para resolver a inconsistência entre as ferramentas da equipe. Foi lançado como código aberto em agosto de 2011.

Atualmente, o framework está na **versão 5 (Bootstrap 5)**. Diferente das versões anteriores, a versão 5 removeu a dependência do *jQuery* (tornando-o mais leve e rápido) e encerrou o suporte ao Internet Explorer, focando em tecnologias modernas de CSS como as CSS Variables e o Grid nativo.

A W3Schools disponibiliza um poderoso Tutorial de BOOTSTRAP:

https://www.w3schools.com/bootstrap/

2.4.2. Primeiros Passos

Para utilizar o Bootstrap, a forma mais rápida é via **CDN** (Content Delivery Network). Basta incluir o link do CSS no <head> da sua página:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
```

2.4.3. O Sistema de Grid

O Bootstrap utiliza um sistema de **12 colunas**. Por que 12? Porque é um número altamente divisível (por 2, 3, 4 e 6), permitindo layouts de 2, 3, 4 ou 6 colunas de forma simples.

Como funciona:

1. **Container:** É o elemento raiz que define a largura do layout.
    - .container: Largura fixa responsiva (com margens nas laterais).
    - .container-fluid: Ocupa 100% da largura da tela.
2. **Row:** Uma linha que agrupa colunas. Deve estar dentro de um container.
3. **Col:** Os blocos de conteúdo. A soma das colunas em uma linha deve ser 12.

Exemplo Didático: Grid Responsivo

Neste exemplo, as colunas ficarão lado a lado no Desktop e se empilharão verticalmente no Celular.

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <title>Bootstrap Grid - TI</title>
</head>
<body>

<div class="container border p-3">
  <h3>Exemplo de Grid Responsivo</h3>
  <div class="row">
    <div class="col-sm-4 bg-primary text-white p-3">Coluna A (33.3%)</div>
    <div class="col-sm-4 bg-dark text-white p-3">Coluna B (33.3%)</div>
    <div class="col-sm-4 bg-primary text-white p-3">Coluna C (33.3%)</div>
  </div>
</div>

</body>
</html>
```

exercicio_14.html

As grids do Bootstrap possuem quatro classes:

- xs (for phones - screens less than 768px wide)
- sm (for tablets - screens equal to or greater than 768px wide)
- md (for small laptops - screens equal to or greater than 992px wide)
- lg (for laptops and desktops - screens equal to or greater than 1200px wide)

2.4.4. Tipografia (Typography)

O Bootstrap utiliza uma configuração de tipografia padrão que otimiza a legibilidade. Ele define fontes ***sans-serif*** modernas, alturas de linha suaves e margens padronizadas.

Elementos de Título (<h1> - <h6>)

Todos os títulos HTML são estilizados pelo Bootstrap para serem visualmente consistentes.

- **Classes de Display:** Quando você precisa de um título que se destaque mais que o normal (tamanhos gigantes), usa as classes **.display-1** até **.display-6**.

Classes Utilitárias de Texto

O Bootstrap fornece classes para alterar o alinhamento, o peso e a cor sem escrever CSS:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <title>Bootstrap Tipografia</title>
</head>
<body>

<div class="container mt-3">
  <p class="h1">Título estilizado como H1</p>
  <p class="text-start">Texto alinhado à esquerda (padrão).</p>
  <p class="text-center">Texto centralizado para destaque.</p>
  <p class="text-end">Texto alinhado à direita.</p>
  <p class="text-uppercase">texto transformado em maiúsculas.</p>
  <p class="fw-bold">Texto em negrito (Font-Weight Bold).</p>
  <p class="fst-italic">Texto em itálico.</p>
  <p class="text-muted">Texto com cor suave (cinza), usado para legendas.</p>
</div>

</body>
</html>
```

exercício_15.html

2.4.5. Cores e Fundos (Contextual Colors)

O Bootstrap usa cores semânticas para transmitir significado, o que reforça o conceito de **IHC** e **Comunicabilidade** (Geest, 2001).

Classe de Texto	Classe de Fundo	Significado em TI
.text-primary	.bg-primary	Cor principal / Ação importante
.text-success	.bg-success	Sucesso / Operação concluída
.text-danger	.bg-danger	Erro / Perigo / Interrupção
.text-warning	.bg-warning	Alerta / Atenção necessária
.text-info	.bg-info	Informação / Dica

Exemplo de Aplicação Semântica:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <title>Bootstrap Cores e Fundos</title>
</head>
<body>

<div class="p-3 mb-2 bg-danger text-white">
  ERRO CRÍTICO: O servidor de banco de dados não responde.
</div>

</body>
</html>
```

exercicio_16.html

2.4.6. Componentes Essenciais: Tabelas

As tabelas no Bootstrap são fundamentais para exibir logs de sistemas, informações financeiras, inventários de hardware e outros.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <title>Bootstrap Tabela</title>
</head>
<body>
```

```
<div class="container mt-3">
  <h2>Tabela de Inventário</h2>
  <table class="table table-striped table-hover">
    <thead class="table-dark">
      <tr>
        <th>Equipamento</th>
        <th>Status</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Roteador Central</td>
        <td><span class="badge bg-success">Online</span></td>
      </tr>
      <tr>
        <td>Servidor de Backup</td>
        <td><span class="badge bg-danger">Offline</span></td>
      </tr>
    </tbody>
  </table>
</div>

</body>
</html>
```

exercicio_17.html

2.4.7. Componentes e Utilidades

O Bootstrap vai além do grid, oferecendo componentes de interface testados para usabilidade:

- **Navbar:** Menu responsivo que vira um ícone "hambúrguer" no celular.
- **Cards:** Containers para conteúdo misto (imagem, título, texto).
- **Modais:** Janelas sobrepostas para interações importantes (ex: confirmação de exclusão de dados).
- **Utilidades de Espaçamento:** Em vez de escrever CSS, usamos:
  - m- (margin), p- (padding).
  - t, b, l, r (top, bottom, left, right).
  - Ex: **mt-3** adiciona uma margem no topo de nível 3.

Exercícios Práticos

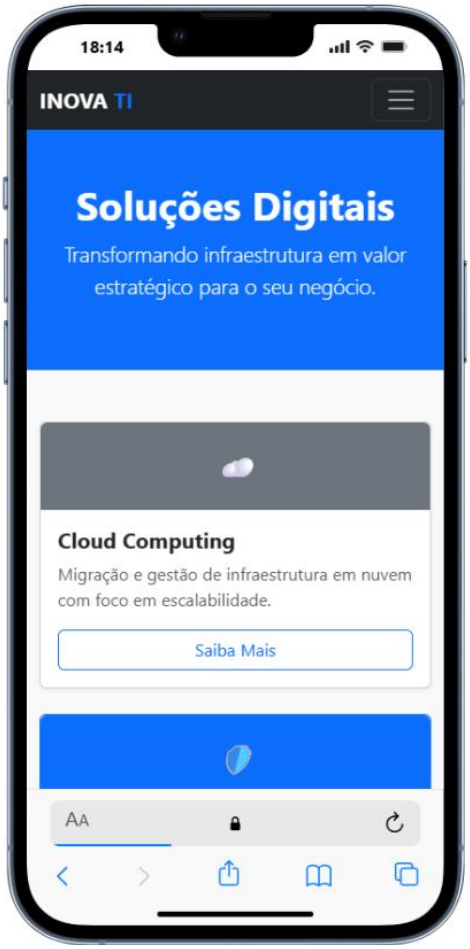
1. Objetivo: Criar um "Centro de Notificações" responsivo.

1. Crie um arquivo HTML e importe o Bootstrap 5.
2. Use um .container e uma .row.
3. Crie duas colunas (.col-md-6).
4. Na coluna da esquerda, coloque um título com a classe .display-4 escrito "Status do Sistema".
5. Na coluna da direita, crie três parágrafos usando as classes de cores:
    - Um .text-success para "Internet ativa".
    - Um .text-warning para "Espaço em disco baixo (85%)".
    - Um .text-danger para "Tentativa de acesso não autorizada".

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exercício 2.4 - Centro de Notificações TI</title>

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">

    <style>
        /* CSS adicional apenas para destacar a grid*/
        .debug-col {
            border: 1px dashed #ccc;
            padding: 20px;
        }
    </style>
</head>
<body class="bg-light">

    <div class="container mt-5 shadow-sm p-4 bg-white rounded">

        <div class="row align-items-center">

            <div class="col-md-6 debug-col">
                <h1 class="display-4 fw-bold">Status do Sistema</h1>
                <p class="text-muted">Monitoramento em tempo real - Unidade 2</p>
            </div>

            <div class="col-md-6 debug-col">
                <h3 class="mb-4">Alertas Recentes</h3>

                <p class="text-success fw-bold">
                    ● Internet ativa e estável.
                </p>

                <p class="text-warning fw-bold">
                    ● Espaço em disco baixo (85%).
                </p>

                <p class="text-danger fw-bold">
                    ● Tentativa de acesso não autorizada detectada.
                </p>
            </div>

        </div> <div class="row mt-4">
            <div class="col-12 text-center border-top pt-3">
                <small class="text-secondary">Desenvolvido para fins didáticos -
Módulo de Bootstrap.</small>
            </div>
        </div>

    </div>
</body>
</html>
```

exercicio_18.html

2. **Objetivo:** Criar uma página institucional de “Soluções Digitais” responsiva (computador, tablet, celular).



Código disponível no arquivo: exercicio_19.html

Pontos Chave:

1. **Componente Navbar:** Observe o uso da classe **.sticky-top**. Em termos de **Cognição**, manter o menu fixo no topo enquanto o usuário rola a página reduz a desorientação espacial.
2. **Componente Card:** Os cartões utilizam a classe **.shadow-sm** para criar uma leve sombra. Isso aplica o conceito de **Affordance**, sugerindo que o elemento é um objeto clicável "acima" do fundo.
3. **Utilitários de Alinhamento:** Usamos **.ms-auto** (Margin Start Auto) na lista da **Navbar** para empurrar os links para a direita, uma prática comum de design que melhora o balanço visual.
4. **Interatividade (JS):** O arquivo inclui o **bootstrap.bundle.min.js.** Sem esse script, o botão de menu (hambúrguer) no celular não abriria ao ser clicado.

Revisão do Conteúdo

1. No CSS3, qual propriedade é utilizada para criar espaço entre o conteúdo de um elemento e sua borda?

a) margin

b) spacing

c) padding

d) border-gap

2. O conceito de "Box Model" no CSS dita que todo elemento HTML é visto como uma caixa. Quais são os quatro componentes dessa caixa, de dentro para fora?

a) Border, Margin, Padding, Content

b) Content, Padding, Border, Margin

c) Margin, Border, Padding, Content

d) Content, Border, Padding, Margin

3. Sobre o framework Bootstrap 5, qual das alternativas abaixo é verdadeira quanto à sua história e evolução?

a) Foi criado pela Google para unificar o design do Android.

b) Foi desenvolvido originalmente no Twitter para resolver inconsistências internas.

c) Sua versão 5 passou a depender obrigatoriamente do jQuery.

d) Foi lançado como código aberto apenas em 2020.

4. No sistema de grid do Bootstrap, em quantas colunas virtuais uma linha (row) é dividida?

a) 10

b) 8

c) 12

d) 16

5. Qual classe do Bootstrap 5 deve ser usada para criar uma coluna que ocupa metade da largura da tela em dispositivos de tamanho médio (tablets/laptops)?

a) .col-sm-6

b) .col-lg-4

c) .col-md-6

d) .col-half

6. A unidade de medida "1fr" no CSS Grid Layout representa:

a) Um pixel fixo baseado no tamanho da fonte.

b) 1% da altura da tela (viewport height).

c) Uma fração do espaço livre disponível no contêiner.

d) O tamanho fixo do primeiro elemento da lista.

7. De acordo com os fundamentos de IHC e Cognição, por que o uso de cores semânticas (como .text-danger para erros) é importante no Bootstrap?

a) Para tornar o código CSS mais curto.

b) Para reduzir a carga cognitiva do usuário através de associações mentais pré-existentes.

c) Apenas para fins estéticos e decorativos.

d) Para garantir que o sistema funcione em navegadores antigos.

8. O que a instrução CSS "height: 90vh;" define para um elemento?

a) Que ele terá 90 pixels de altura.

b) Que ele ocupará 90% da altura total da janela de visualização do navegador.

c) Que ele terá uma altura proporcional a 90% da largura.

d) Que ele será visível apenas em 90% dos dispositivos.

9. Qual seletor CSS é utilizado para estilizar um elemento que possui um atributo id="main-header"?

a) .main-header

b) *main-header

c) #main-header

d) @main-header

10. No Bootstrap, qual classe é responsável por criar um container que ocupa 100% da largura da tela em todos os tamanhos de dispositivo?

a) .container

b) .container-full

) .container-fluid

d) .container-100

11. Qual das seguintes propriedades CSS é fundamental para o uso do Flexbox?

a) display: grid;

b) display: flex;

c) position: absolute;

d) float: left;

12. Na tipografia do Bootstrap, as classes ".display-1" até ".display-6" são utilizadas para:

a) Esconder elementos em diferentes telas.

b) Criar títulos de grande impacto visual, maiores que os títulos H1-H6 padrão.

c) Alterar a cor de fundo de parágrafos.

d) Definir a ordem de exibição de colunas..

II. Perguntas Reflexivas

1. Cenário de Negócio: Uma empresa de sistemas financeiros percebeu que seus usuários cometiam muitos erros ao preencher formulários de transferência, confundindo o botão "Cancelar" com "Confirmar" porque ambos eram azuis e estavam muito próximos. Como os conceitos de Leis da Gestalt (Proximidade e Semelhança) e as Cores Semânticas do Bootstrap poderiam ser aplicados para resolver esse problema de usabilidade?

2. Evolução Tecnológica: O Bootstrap 5 removeu a dependência do jQuery e focou em CSS moderno (como CSS Variables). De que forma essa decisão técnica impacta o desempenho de sistemas de grande escala e a experiência do desenvolvedor front-end moderno?

III. Exercícios de Revisão

1. Sintaxe CSS: Crie uma folha de estilo interna que altere a cor de fundo de todos os parágrafos para cinza claro e defina o tamanho da fonte para 18px.

2. Box Model: Crie uma com largura de 200px, borda sólida de 5px, padding de 20px e margin de 10px. Calcule e escreva no conteúdo da div qual a largura total que ela ocupa na tela.

3. Flexbox: Crie uma barra de navegação simples com 3 links ("Home", "Produtos", "Contato") usando display: flex e garanta que haja um espaço igual entre eles usando justify-content.

4. **Bootstrap Grid:** Monte uma estrutura onde, no desktop (md), existam 4 colunas iguais, mas no celular elas se empilhem ocupando a largura total.

5. **Componentes:** Crie um "Card" do Bootstrap que contenha uma imagem (ou ícone), um título "Servidor Cloud" e um botão azul escrito "Verificar Status".

6. **Tipografia e Cores**: Crie um alerta de erro usando as classes do Bootstrap que exiba em negrito a mensagem "ACESSO NEGADO" com o fundo vermelho e o texto branco.

Exercício de Fixação – Unidade 1 e 2

Exercício 1: A "Caixa" Perfeita (Box Model e Semântica)

Objetivo: Compreender na prática como o navegador calcula o espaço de um elemento e a importância da hierarquia visual.

Tarefa: No VS Code, crie um arquivo box-model.html.

1. Estruture uma seção (<section>) contendo um título (<h1>) e um parágrafo (<p>).
2. Crie uma folha de estilo interna (<style>) e configure a seção para ter:
    - Largura de 400px.
    - Fundo cinza claro (#f0f0f0).
    - Borda sólida de 10px na cor azul.
    - Padding de 30px.
    - Margin de 50px.
3. **Desafio:** Calcule manualmente qual a largura total que esse elemento ocupa na tela e escreva o resultado dentro do parágrafo.

Exercício 2: Portal de Chamados Técnico (Bootstrap 5 - Grid e Tipografia)

Objetivo: Utilizar o framework padrão de mercado para criar uma interface responsiva e comunicável.

Tarefa: Crie um arquivo chamados.html e importe o Bootstrap via CDN.

1. Crie uma **Navbar** escura com o nome "Sistema de Chamados".
2. Use o sistema de **Grid do Bootstrap** para criar uma linha com 3 colunas (.col-md-4).
3. Dentro de cada coluna, insira um **Card** descrevendo um chamado técnico:
    - Card 1 (Sucesso): "Servidor Online" (Use a classe .text-success).
    - Card 2 (Alerta): "Backup Pendente" (Use a classe .text-warning).
    - Card 3 (Erro): "Falha de Segurança" (Use a classe .text-danger).
4. Verifique se, ao diminuir a tela, os cards se empilham verticalmente.

Exercício 3: Relatório de Acessibilidade e IHC

Objetivo: Aplicar conceitos de acessibilidade e leis da Gestalt.

Tarefa: No arquivo do Exercício 3, realize as seguintes melhorias:

1. Adicione uma imagem de um gráfico de performance. É obrigatório o uso do atributo alt com uma descrição técnica clara para leitores de tela.
2. Certifique-se de que os botões de "Aprovar" e "Excluir" tenham cores contrastantes (ex: .btn-primary e .btn-danger) e um espaçamento adequado entre eles, aplicando a **Lei da Proximidade** para evitar erros de clique.
3. Utilize a classe .display-4 no título principal para garantir uma **Hierarquia Visual** forte, guiando o olhar do usuário.

Unidade 3: Design de Experiência e Prototipação

Nesta unidade, avançamos para além da codificação básica. Compreenderemos que o desenvolvimento front-end eficaz nasce de um planejamento centrado no ser humano, onde a tecnologia serve ao propósito da experiência.

3.1. Fundamentos de User Experience (UX)

3.1.1. O que é UX: Diferença entre UX (Experiência) e UI (Interface)

É comum a confusão entre os termos, mas para um desenvolvedor, a distinção é vital para a qualidade do produto final:

- **UI (User Interface):** Refere-se à **camada física e visual** do sistema. É o que o usuário vê e toca (botões, cores, tipografia, espaçamentos). O objetivo da UI é criar interfaces que sejam visualmente atraentes e logicamente estruturadas.
- **UX (User Experience):** É a **percepção subjetiva** e a resposta emocional que o usuário tem ao interagir com o sistema. Envolve utilidade, facilidade de uso e eficiência.

Analogia técnica: Se um site fosse um carro, a UI seria o design do painel, o estofado e a cor da lataria. A UX seria a sensação de dirigir, a facilidade de encontrar as marchas e o conforto durante a viagem. Uma UI bonita pode ter uma UX terrível se o sistema for lento ou confuso.

3.1.2. O Ciclo de Design Centrado no Usuário

O design não é um evento único, mas um processo iterativo. O ciclo de vida clássico (ISO 9241-210) divide-se em quatro etapas principais:

1. **Investigação (Análise):** Entender o contexto de uso e os requisitos do usuário. O que eles precisam resolver? Quais são suas necessidades/dores?
2. **Ideação (Projeto):** Criação de soluções conceituais. É o momento de gerar alternativas de design que atendam aos requisitos levantados.
3. **Prototipação:** Construção de representações da solução (desde rascunhos em papel até protótipos de alta fidelidade).
4. **Avaliação:** Testar o protótipo com usuários reais para identificar falhas de usabilidade. Os resultados desta fase retroalimentam o ciclo, gerando melhorias.

** **ISO 9241-210 (ou ABNT NBR ISO 9241-210)** é uma norma internacional que define requisitos e diretrizes para o design centrado no ser humano (Human-Centered Design - HCD) em sistemas interativos.

3.1.3. Personas e Jornada do Usuário

Antes de escrever a primeira linha de HTML, precisamos saber **para quem** estamos construindo.

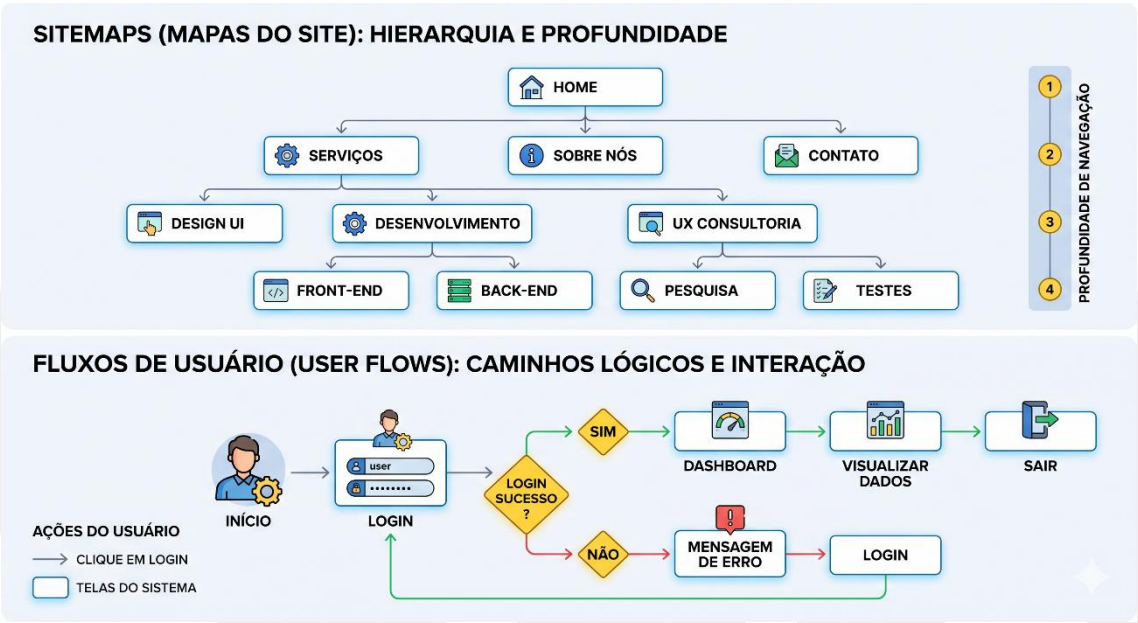
- **Personas:** São personagens fictícios criados com base em dados reais para representar os diferentes tipos de usuários dentro de um público-alvo. Elas ajudam a equipe de TI a manter o foco nas necessidades humanas em vez de apenas em requisitos técnicos.

- **Jornada do Usuário:** É um mapeamento visual de todos os passos que o usuário percorre para realizar uma tarefa (ex: "comprar um produto" ou "abrir um chamado"). Isso permite identificar "pontos de atrito" onde o front-end precisa ser mais intuitivo.

3.1.4. Arquitetura de Informação: Organização de fluxos e sitemaps

Segundo **Camargo & Vidotti (2011)**, a Arquitetura de Informação (AI) é a arte de organizar espaços de informação para ajudar as pessoas a encontrar o que precisam com o menor esforço cognitivo possível.

- **Sitemaps (Mapas do Site):** Representação visual da hierarquia de páginas do sistema. Define a profundidade da navegação.
- **Fluxos de Usuário (User Flows):** Diagramas que mostram os caminhos lógicos que o usuário segue. Se ele clicar em "X", vai para a tela "Y".



Uma boa Arquitetura da Informação (AI) garante que a navegação seja previsível, aplicando a **Ergonomia Cognitiva** discutida na Unidade 2, facilitando a localização de conteúdos e funções críticas.

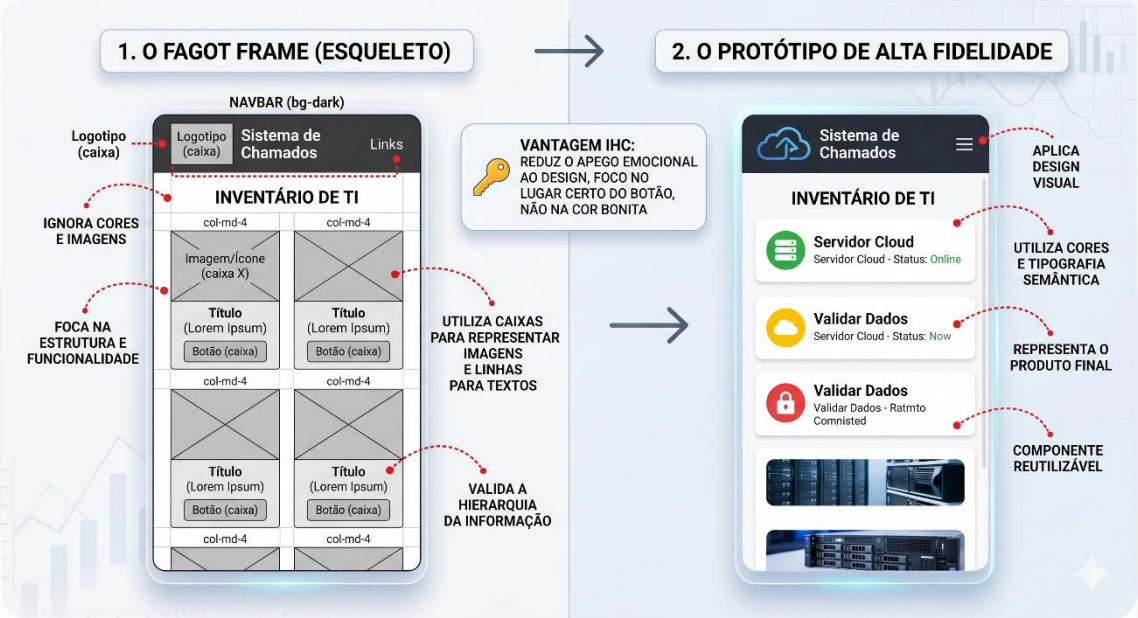
3.2. User Interface (UI) e Prototipação

A prototipação é o momento em que as ideias ganham forma física (ou digital). No mercado de tecnologia, existem várias soluções colaborativas e baseada em nuvem, como **Figma, Miro e Whimsical,** que permite aos designers e desenvolvedores front-end trabalhar em sintonia.

3.2.1. Wireframes de Baixa Fidelidade: O esboço lógico do sistema

O wireframe é o "esqueleto" da interface. Nesta etapa, ignoramos cores, imagens ou tipografias complexas para focar exclusivamente na **estrutura e na funcionalidade**.

- **Objetivo:** Validar a hierarquia da informação e o fluxo de navegação antes de investir tempo no design visual.
- **Características:** Geralmente feitos em tons de cinza, utilizam caixas para representar imagens e linhas para textos (conhecido como *lorem ipsum* ou *placeholder*).
- **Vantagem IHC:** Reduz o apego emocional ao design, permitindo que a equipe foque em saber se o botão está no lugar certo para o usuário, e não se a cor é bonita.



3.2.2. Protótipos de Alta Fidelidade: Design visual, cores, tipografia e componentes

Aqui, aplicamos a "pele" ao esqueleto. O protótipo de alta fidelidade deve ser o mais próximo possível do produto final.

- **Cores e Tipografia:** Aplicamos os conceitos de semântica e legibilidade discutidos na Unidade 2.
- **Componentização:** Componentes, como botões ou headers, podem ser reutilizados em várias telas. Se alterarmos o componente principal, todas as instâncias são atualizadas. Isso simula a lógica de **reutilização de código** que veremos no CSS e JavaScript.
- **Imagens e Ícones:** Substituímos os placeholders por ativos reais, testando o peso visual e a carga cognitiva de cada elemento.

3.2.3. Prototipação Interativa: Criando fluxos clicáveis

Um protótipo de alta fidelidade não é apenas uma imagem estática, é uma poderosa ferramenta interativa que auxilia na validação de hipóteses iniciais para novas soluções.

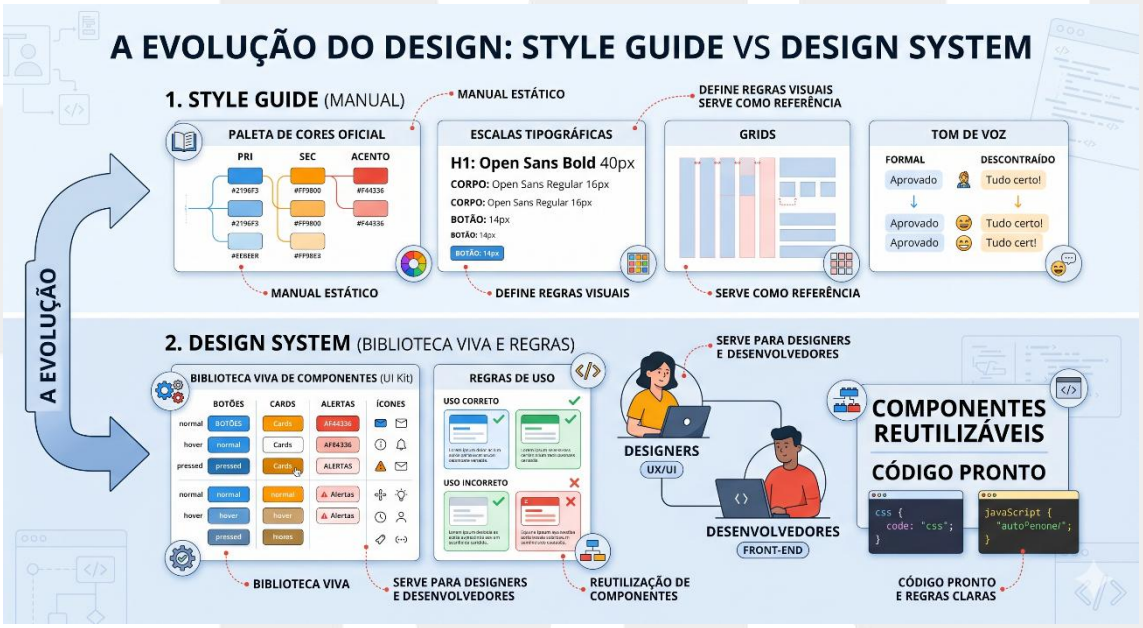
- **Interações:** Definimos o que acontece ao clicar em um botão (ex: abrir um menu lateral ou navegar para outra página).
- **Testes de Usabilidade:** Protótipos interativos permitem realizar testes com usuários reais sem escrever uma única linha de código. Isso economiza custos de desenvolvimento, pois os erros de lógica são detectados e corrigidos ainda na fase de design.

- **Transições:** Podemos configurar animações (dissolver, deslizar) que ajudam o usuário a entender a continuidade espacial do sistema.

3.2.4. Design System e Style Guides: Mantendo a consistência

Soluções digitais não podem ter botões diferentes em cada página. Para isso, utilizamos:

- **Style Guide:** Um manual que define a paleta de cores oficial, as escalas tipográficas, os grids e o tom de voz.
- **Design System:** É a evolução do Style Guide. É uma biblioteca viva de componentes (UI Kit) e regras de uso que serve tanto para designers quanto para desenvolvedores.



- **Impacto na Manutenção:** Garante a **Consistência** (um dos pilares de IHC), reduzindo o tempo de aprendizado do usuário e facilitando a manutenção do código front-end a longo prazo.

Exercício Prático: Do Requisito ao Protótipo (Portal de Notícias Tech)

Cenário: Uma startup de tecnologia deseja lançar o "TechPulse", um portal de notícias focado em Inteligência Artificial para profissionais de TI. O objetivo é ser um site rápido, limpo e que facilite a leitura diária.

Ferramenta: Whimsical.com (utilize as funções de *Board*, *Wireframe* e *Flowchart*).

Vídeos Explicativos Adicionais:

Levantamento de Requisitos - https://youtu.be/xEdGAC0qzgY?si=B6DNZdK_I2LpBGub

MVP de Produtos - https://youtu.be/4PMeG1G2oGM?si=yB5YvWbsCVRNo5bS

Como criar Wireframes (protótipo) - https://www.youtube.com/watch?v=-khZLAsG3d4

Etapa 1: Levantamento de Requisitos (Mínimo 5)

Com base nos vídeos assistidos, liste 5 requisitos funcionais que o sistema deve ter.

- *Exemplo:* "O sistema deve permitir filtrar notícias por categoria (IA, Cloud, Dev)."

Etapa 2: Identificação de Personas (Mínimo 2)

Crie dois perfis de usuários que utilizarão o portal. Use o modelo:

- **Nome e Cargo:** (ex: Pedro, Desenvolvedor Júnior)
- **Necessidade:** (O que ele busca no portal?)
- **Ponto de Atrito:** (O que o irrita em outros sites de notícias?)

Etapa 3: Jornada do Usuário (Mapa de Fluxo)

No Whimsical, utilize a ferramenta de **Flowchart** para desenhar o caminho do usuário desde o momento em que ele acessa a Home até o momento em que termina de ler uma notícia e decide compartilhar.

- **Caminho Feliz:** Home -> Escolha da Categoria -> Clique na Notícia -> Leitura -> Compartilhamento.

Etapa 4: Protótipo de Baixa Fidelidade (Wireframe)

Utilizando a ferramenta de **Wireframe** do Whimsical, desenhe a estrutura da página principal (Home) e da página de leitura de notícia.

- **Regra:** Não use cores ou imagens reais. Use apenas o "esqueleto" (caixas com "X" para imagens e linhas para texto).
- **Elementos obrigatórios:** Navbar, Área de Destaque, Lista de Notícias Recentes e Rodapé.

3.3. Introdução à Interatividade com JavaScript

3.3.1. O Papel do JavaScript no Front-End: Comportamento e dinamismo

O JavaScript (JS) é uma linguagem de programação de alto nível que permite implementar itens complexos em páginas web. Enquanto o HTML e o CSS fornecem uma experiência estática, o JS permite:

- Atualizar conteúdos dinamicamente (sem recarregar a página).
- Validar dados em formulários antes do envio.
- Criar animações e elementos interativos (menus, modais, carrosséis).
- Controlar multimídia e lidar com respostas de servidores (APIs).

A W3Schools disponibiliza um poderoso Tutorial de JAVSCRIPT:

https://www.w3schools.com/js/

Visão de IHC: O JavaScript é a ferramenta que operacionaliza o **feedback imediato**. De acordo com os critérios de usabilidade, o sistema deve informar ao usuário o que está acontecendo (ex: uma barra de carregamento ou uma mensagem de erro instantânea), e o JS é o responsável por essa comunicação em tempo real.

3.3.2. Sintaxe Básica: Variáveis, Tipos de Dados e Operadores

Para programar, precisamos armazenar e manipular informações:

- **Variáveis (let e const):**
  - **let:** Usada para valores que podem mudar durante a execução.
  - **const:** Usada para valores constantes (que não mudam).
- **Tipos de Dados:**
  - **String:** Textos (ex: "FMU").
  - **Number:** Números inteiros ou decimais (ex: 20 ou 19.90).
  - **Boolean:** Valores lógicos (true ou false).
- **Operadores:**
  - **Matemáticos** (+, -, *, /) .
  - **Comparação** (==, ===, !=, >, <).

3.3.3. Estruturas de Controle: Condicionais e Laços

As estruturas de controle permitem que o software "tome decisões":

- Condicionais (if/else):
    - Executa um bloco de código apenas se uma condição for verdadeira.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Portal de Notícias - Unidade 3.3</title>
    <!-- Bootstrap 5 CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

    <!-- Navbar -->
    <nav class="navbar navbar-dark bg-primary mb-4">
        <div class="container">
            <span class="navbar-brand mb-0 h1">TechNews FMU</span>
        </div>
    </nav>

    <div class="container">
        <header class="mb-4">
            <h1 class="display-5">Exemplo de JavaScript</h1>
            <p class="text-muted">Geração de um alert.</p>
        </header>
    </div>

    <!-- JavaScript Integrado -->
    <script>
        idade = 20;
        if (idade >= 18) {
            alert("Acesso Permitido");
            console.log("Acesso Permitido");
        }
    </script>
</body>
</html>
```

exercicio_20.html

- Laços de Repetição (for/while):
    - Repetem uma ação várias vezes. Útil para percorrer listas de produtos ou usuários vindos de um banco de dados.
    - *Exemplo:* Repetir a criação de um "Card" do Bootstrap para cada item de um inventário de TI.

A estrutura `for`

- O `for` é ideal quando sabemos exatamente quantas vezes queremos repetir uma ação. No exemplo abaixo, utilizaremos o laço para exibir um alerta de "Contagem Regressiva" para o lançamento de um sistema.

```
<script>
    // Exemplo de for: Contagem de 1 a 3
    for (let i = 1; i <= 3; i++) {
        alert("Preparando sistema... Passo " + i);
        console.log("Executando passo: " + i);
    }
</script>
```

A estrutura `while`

- O `while` é utilizado quando a repetição depende de uma condição que pode mudar. É muito útil para processar dados enquanto uma variável não atinge um limite.

```
<script>
    let tentativas = 1;
    const maxTentativas = 3;

    // Exemplo de while: Tentativas de conexão
    while (tentativas <= maxTentativas) {
        console.log("Tentativa de conexão ao servidor: " + tentativas);
        tentativas++; // Incremento necessário para não criar um loop infinito
    }
    alert("Verificação concluído com " + (tentativas - 1) + " tentativas.");
</script>
```

Exemplo de **for** e **while**

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Portal de Notícias - Unidade 3.3</title>
    <link          href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

    <nav class="navbar navbar-dark bg-primary mb-4">
        <div class="container">
            <span class="navbar-brand mb-0 h1">TechNews FMU</span>
        </div>
    </nav>

    <div class="container">
        <header class="mb-4">
            <h1 class="display-5">Exemplo de JavaScript</h1>
            <p class="text-muted">Estruturas de Repetição: For e While.</p>
        </header>
    </div>

    <script>
```

```
        // 1. Uso do FOR: Simulando carregamento de 3 notícias
        console.log("--- Iniciando Laço FOR ---");
        for (let i = 1; i <= 3; i++) {
            console.log("Notícia " + i + " carregada com sucesso.");
        }

        // 2. Uso do WHILE: Verificação de segurança
        console.log("--- Iniciando Laço WHILE ---");
        let contador = 1;
        while (contador <= 2) {
            alert("Verificação de segurança " + contador + " em curso...");
            console.log("Check de segurança: " + contador);
            contador++;
        }

        // Mantendo a lógica de condicional do seu exemplo original
        let idade = 20;
        if (idade >= 18) {
            console.log("Status Final: Acesso Permitido");
        }
    </script>
</body>
</html>
```

exercico_21.html

3.3.4. Manipulação do DOM (Document Object Model)

O **DOM** é a representação que o navegador cria do seu HTML. O JavaScript utiliza o DOM para "conversar" com a página:

- **Como funciona:** O JS pode selecionar um elemento (ex: um botão), ouvir quando ele é clicado e alterar o conteúdo de outro elemento (ex: mudar a cor de um texto ou exibir um alerta).
- **Métodos Principais:**
    - **document.getElementById()** ou **document.querySelector()**:
        - Para selecionar elementos.
    - **.innerHTML:**
        - Para mudar o texto ou HTML interno.
    - **.innerText:**
        - Para mudar o texto de um elemento interno.
    - **.style:**
        - Para alterar propriedades CSS dinamicamente.
    - **classList:**
        - Fornece uma forma simples de acessar e manipular a lista de classes CSS de um elemento

- replace:
  - O método permite trocar um item ou texto por outro, quando utilizado com classList permite trocar a classe CSS existente por uma nova em uma única operação.
- appendChild:
  - É um método fundamental para a manipulação dinâmica do DOM, utilizado para inserir um novo "nó" (como uma div, um h1 ou um card) como o último filho de um elemento pai selecionado.

Exemplo prático: Ao clicar em um botão de "DarkMode", o JavaScript acessa o DOM, identifica o corpo da página e altera a classe CSS de bg-light para bg-dark do Bootstrap.

Exemplo - Portal com Botão Dark Mode:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo DarkMode Corrigido - Unidade 3.3</title>
    <link          href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            transition: background-color 0.3s ease, color 0.3s ease;
        }
        /* Garantindo que o link da marca também tenha transição suave */
        .navbar-brand {
            transition: color 0.3s ease;
        }
```

```
    </style>
</head>
<body class="bg-light text-dark">

    <nav class="navbar navbar-expand-lg border-bottom mb-5">
        <div class="container">
            <span class="navbar-brand text-dark" id="logo-portal">Portal Tech</span>

            <button id="btn-dark-mode" class="btn btn-outline-primary">
                Alternar Dark Mode
            </button>
        </div>
    </nav>

    <div class="container text-center">
        <h1 id="titulo">Modo Claro Ativo</h1>
        <p class="lead">Agora, todos os elementos, incluindo o "Portal Tech", mudarão de cor
simultaneamente.</p>
    </div>

    <script>
        const botao = document.getElementById('btn-dark-mode');
        const corpoPagina = document.body;
        const titulo = document.getElementById('titulo');
        const logo = document.getElementById('logo-portal'); // Seleção do elemento que faltava

        botao.addEventListener('click', () => {

            if (corpoPagina.classList.contains('bg-light')) {
                // MUDANÇA PARA MODO ESCURO
                corpoPagina.classList.replace('bg-light', 'bg-dark');
                corpoPagina.classList.replace('text-dark', 'text-white');

                // Ajustando o "Portal Tech"
                logo.classList.replace('text-dark', 'text-white');

                titulo.innerText = "Modo Escuro Ativo";
                botao.innerText = "Mudar para Modo Claro";
                botao.classList.replace('btn-outline-primary', 'btn-outline-warning');
            } else {
                // VOLTA PARA MODO CLARO
                corpoPagina.classList.replace('bg-dark', 'bg-light');
                corpoPagina.classList.replace('text-white', 'text-dark');

                // Ajustando o "Portal Tech" de volta
                logo.classList.replace('text-white', 'text-dark');

                titulo.innerText = "Modo Claro Ativo";
                botao.innerText = "Mudar para Modo Escuro";
                botao.classList.replace('btn-outline-warning', 'btn-outline-primary');
            }
        });
    </script>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

exercicio_22.html

3.4. Eventos e Lógica de Interface

3.4.1. Escutadores de Eventos (EventListeners)

Na programação front-end, um "evento" é qualquer ação que ocorra no navegador, iniciado pelo usuário ou pelo próprio sistema. Para que o JavaScript reaja a essas ações, utilizamos os **EventListeners**.

- **O que são:** São "vigias" que ficam aguardando uma ação específica em um elemento do DOM para disparar uma função.
- **Principais Eventos de Interface:**
    - **click:** Disparado quando o usuário clica em um elemento (botão, link, imagem).
    - **keydown / keyup:** Capturam quando uma tecla é pressionada ou solta (essencial para acessibilidade e atalhos).
    - **mouseover / mouseout:** Detectam o movimento do mouse sobre elementos (usado para efeitos visuais e *tooltips*).
    - **submit:** Disparado quando um formulário é enviado.

Exemplo técnico:

```
const botao = document.querySelector('#btn-salvar');
botao.addEventListener('click', () => {
    alert('Dados enviados com sucesso!');
});
```

3.4.2. Validação de Formulários e Feedback Imediato

A validação de formulários é um dos pilares da **UX (User Experience)**. Em vez de permitir que o usuário envie dados errados para o servidor e receba um erro segundos depois, o JavaScript permite validar as informações **no momento da digitação**.

- **Feedback Imediato:** Se um campo de "E-mail" não contém um "@", o sistema deve exibir uma mensagem de erro instantânea (ex: usando a classe .text-danger do Bootstrap).
- **Vantagem de IHC:** Reduz a frustração do usuário e previne erros de entrada, aplicando o princípio da **Prevenção de Erros** de Nielsen.
- **Lógica de Validação:** Utilizamos condicionais (if/else) para verificar o value dos campos antes de processar a lógica de negócio.

3.4.3. Consumo de APIs Básicas (Fetch API)

Um sistema de TI moderno raramente trabalha isolado, ele consome dados de outros sistemas através de **APIs (Application Programming Interfaces)**.

- **Conceito de Fetch:** A função fetch() permite que o JavaScript faça uma requisição para uma URL externa (um servidor) e receba dados de volta, geralmente no formato **JSON**.

- **await fetch:** O fetch é uma API moderna para realizar requisições assíncronas (AJAX), e a palavra-chave await faz com que o JavaScript espere a resposta do servidor antes de prosseguir com a execução do código. Essa combinação é essencial para buscar dados externos, como arquivos JSON de notícias ou informações de uma base de dados, permitindo que a interface continue funcional enquanto os dados são carregados em segundo plano.
- **Exemplo de uso:** Buscar o endereço automaticamente ao digitar um CEP, ou carregar o inventário de hardware de um banco de dados na nuvem.
- **Assincronismo:** Como a internet pode ser lenta, essas requisições são **assíncronas**. O sistema continua funcionando enquanto espera a resposta do servidor, garantindo que a interface não "trave".

Exemplo - Página de Notícias

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Portal de Notícias - Unidade 3.3 (AJAX)</title>
    <link          href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body { background-color: #f8f9fa; }
        .news-card { transition: transform 0.2s; cursor: pointer; }
        .news-card:hover { transform: scale(1.02); }
        #loading-spinner { display: none; }
    </style>
</head>
<body>

    <nav class="navbar navbar-dark bg-primary mb-4">
        <div class="container">
            <span class="navbar-brand mb-0 h1">TechNews FMU</span>
            <button    class="btn    btn-outline-light    btn-sm"    id="btn-load">Atualizar    via
AJAX</button>
        </div>
    </nav>

    <div class="container">
        <header class="mb-4">
            <h1 class="display-5">Últimas Notícias</h1>
            <p    class="text-muted">Dados    carregados    via    <strong>Fetch    API    (AJAX
moderno)</strong>.</p>
        </header>

        <div id="loading-spinner" class="text-center my-5">
            <div class="spinner-border text-primary" role="status"></div>
            <p class="mt-2">Requisitando JSON ao servidor...</p>
        </div>

        <div class="row" id="news-container"></div>
    </div>

    <script>
        const newsContainer = document.getElementById('news-container');
        const btnLoad = document.getElementById('btn-load');
        const spinner = document.getElementById('loading-spinner');

        // 3.4.3. Consumo de APIs via AJAX (Fetch)
        async function fetchNews() {
```

```
            spinner.style.display = 'block';
            newsContainer.innerHTML = '';

            try {
                response = await fetch('https://robson.isy.one/noticias/');

                if (!response.ok) {
                    throw new Error("Não foi possível buscar notícias.");                }

                data = await response.json(); // Converte a resposta em objeto JS
                renderNews(data);
                spinner.style.display = 'none';

            } catch (error) {
                spinner.style.display = 'none';
                newsContainer.innerHTML = `
                    <div class="alert alert-danger">
                        <strong>Erro de Requisição:</strong> ${error.message} <br>
                        <small>Certifique-se de que o arquivo 'noticias.json' existe ou que você
está usando um servidor local (Live Server).</small>
                    </div>`;
            }
        }

        function renderNews(newsList) {
            newsList.forEach(item => {
                const card = document.createElement('div');
                card.className = 'col-md-4 mb-4';
                card.innerHTML = `
                    <div class="card h-100 shadow-sm news-card">
                        <div class="card-body">
                            <span class="badge bg-primary mb-2">${item.categoria}</span>
                            <h5 class="card-title">${item.titulo}</h5>
                            <p class="card-text text-muted">${item.resumo}</p>
                        </div>
                        <div class="card-footer bg-transparent border-top-0">
                            <small class="text-muted">Escrito por: ${item.autor}</small>
                        </div>
                    </div>
                `;
                newsContainer.appendChild(card);
            });
        }

        btnLoad.addEventListener('click', fetchNews);
        window.onload = fetchNews;
    </script>
</body>
</html>
```

exercicio_23.html

Questões de Múltipla Escolha

Questão 1 - No contexto do Design Centrado no Usuário (DCU), a etapa de Prototipação desempenha um papel fundamental na mitigação de riscos e na redução de custos de desenvolvimento. Sobre a diferença entre wireframes e protótipos de alta fidelidade, analise as asserções a seguir:

I. O wireframe foca na arquitetura de informação e na hierarquia funcional, ignorando elementos estéticos para evitar o apego emocional precoce ao design.

II. O protótipo de alta fidelidade é uma representação estática que serve apenas para a aprovação da paleta de cores pelo cliente.

III. A interatividade em protótipos permite validar fluxos de navegação e identificar gargalos de usabilidade antes da escrita do código final.

É correto o que se afirma em:

A) I apenas.

B) II apenas.

C) I e III apenas.

D) II e III apenas.

E) I, II e III.

Questão 2 - Ao implementar um sistema de Dark Mode utilizando JavaScript e o framework Bootstrap, um desenvolvedor decide utilizar o método classList.replace('bg-light', 'bg-dark'). Considerando a manipulação do DOM (Document Object Model), qual é a principal vantagem técnica dessa abordagem?

A) O método altera diretamente o arquivo CSS externo salvo no servidor.

B) Ele permite a substituição atômica de uma classe por outra, garantindo que o estado visual seja atualizado sem a necessidade de recarregar a página.

C) O método replace é o único capaz de ler variáveis JSON enviadas via AJAX.

D) Ele elimina a necessidade de utilizar o comando document.getElementById.

E) Garante que o navegador ignore as regras de especificidade do CSS.

Questão 3 - Segundo Camargo & Vidotti (2011), a Arquitetura de Informação é essencial para a organização de espaços de informação. Em um projeto de portal de notícias, a criação de um Sitemap tem como objetivo principal:

A) Definir a sintaxe das variáveis JavaScript que serão usadas no carregamento de dados.

B) Mapear a hierarquia das páginas e a profundidade da navegação para facilitar a localização de conteúdos pelo usuário.

C) Substituir a necessidade de testes de usabilidade com personas.

D) Gerar automaticamente o código HTML das listas de navegação.

E) Configurar as rotas de acesso ao banco de dados via Fetch API.

Questão 4 - A utilização de Personas e o mapeamento da Jornada do Usuário são ferramentas estratégicas no Front-End. Qual das alternativas melhor descreve a função da Jornada do Usuário no design de interfaces?

A) É um documento técnico que descreve o banco de dados do sistema.

B) Serve para cronometrar quanto tempo um desenvolvedor leva para escrever um componente.

C) Identifica os pontos de atrito e as necessidades emocionais do usuário em cada etapa da interação com o produto.

D) É uma lista de todos os navegadores onde o site deve ser testado.

E) Define quais APIs externas serão consumidas pelo JavaScript.

Questão 5 - Considere o seguinte trecho de código JavaScript:

```
const btn = document.querySelector('#ativar');

btn.addEventListener('click', () => { console.log('Evento
disparado'); });
```

Este padrão de projeto é conhecido como:

A) Estrutura de repetição condicional.

B) Manipulação estática de tipos.

C) Escutador de eventos (Event Listener) para tratamento de interatividade.

D) Declaração de constante assíncrona.

E) Validação de formulário via regex.

Questão 6 - Sobre a Fetch API e o conceito de AJAX moderno, é correto afirmar que:

A) As requisições são síncronas, travando a interface do usuário até que o servidor responda.

B) O uso do await permite que o código espere a resposta de uma URL externa de forma organizada, sem bloquear o funcionamento global da página.

C) O Fetch só é capaz de ler arquivos XML, não sendo compatível com JSON.

D) É uma ferramenta de design usada exclusivamente dentro do Figma.

E) O navegador não exige permissões de CORS para acessar dados de domínios diferentes.

Questões Práticas

Questão 1 - No desenvolvimento de interfaces com Bootstrap 5, a interatividade é frequentemente adicionada através da manipulação de classes utilitárias. Considere o código abaixo:

```
<div id="alerta" class="alert alert-success d-none">Processamento concluído!</div>
```

Qual instrução **JavaScript** deve ser utilizada para que esse alerta torne-se visível após o clique de um usuário?

A) document.getElementById('alerta').style.visibility = 'visible';

B) document.getElementById('alerta').classList.remove('d-none');

C) document.querySelector('#alerta').innerHTML = 'd-block';

D) document.getElementById('alerta').appendChild('d-block');

E) document.classList.replace('d-none', 'alert-success');

Questão 2 - Ao trabalhar com formulários dinâmicos, é comum a necessidade de capturar o valor de um campo de entrada (input) para validação. Se um aluno possui o elemento **<input type="text" id="usuario_nome" class="form-control">**, como ele deve capturar o texto digitado via JavaScript?

A) const nome = document.getElementById('usuario_nome').innerText;

B) const nome = document.getElementById('usuario_nome').innerHTML;

C) const nome = document.getElementById('usuario_nome').value;

D) const nome = document.querySelector('.form-control').textContent;

E) const nome = document.getElementById('usuario_nome').getAttribute('class');

Questão 3 - Um desenvolvedor Front-End precisa criar uma lista de itens dinamicamente a partir de um array de dados. Ele decide utilizar o método appendChild. Analise o fluxo lógico abaixo:

1. Criar um elemento <li> usando document.createElement.
2. Definir o conteúdo de texto do <li>.
3. Adicionar a classe list-group-item do Bootstrap ao <li>.
4. Anexar o <li> a uma <ul> existente no HTML.

Qual sequência de comandos reflete corretamente esse fluxo?

A) li.create(); li.text = "Item"; ul.add(li);

B) const li = document.createElement('li'); li.textContent = "Item"; li.classList.add('list-group-item'); ul.appendChild(li);

C) const li = "
Item
"; ul.innerHTML += li;

D) document.appendChild('li').innerText = "Item";

E) li.classList.replace('list-group-item'); document.createElement('li');

Questão 4 - A utilização de Template Literals (uso de crases `) no JavaScript facilita a construção de interfaces com Bootstrap. Qual é a principal utilidade técnica dessa funcionalidade ao renderizar componentes?

A) Impedir que o CSS do Bootstrap seja sobrescrito pelo desenvolvedor.

B) Permitir a interpolação de variáveis diretamente dentro de blocos de HTML, facilitando a criação de "cards" ou "tabelas" dinâmicas.

C) Aumentar a velocidade de download do arquivo .js pelo navegador.

D) Converter automaticamente arquivos PHP em arquivos HTML.

E) Substituir o uso de IDs no HTML por seletores de classe.

Questão 5 - Em uma aplicação que consome dados de uma API externa de clima, o desenvolvedor utiliza o seguinte código:

```
const dados = await fetch('api.exemplo.com/clima').then(res => res.json());
```

O que ocorre se a requisição falhar (ex: servidor fora do ar) e não houver um bloco try/catch envolvendo esse código?

A) O navegador ignora o erro e exibe a página normalmente com dados em branco.

B) O JavaScript interrompe a execução do script no ponto do erro, podendo travar funcionalidades da interface que dependem daquela resposta.

C) O Bootstrap assume o controle e exibe um alerta automático de erro.

D) O método fetch tenta realizar a requisição infinitamente até obter sucesso.

E) Os dados são recuperados automaticamente do cache do Google Search.

Questão 6 - O uso da palavra-chave const para declarar seletores do DOM (como botões e containers) é uma boa prática recomendada na Unidade 3. Por que const é preferível a let nesses casos específicos?

A) Porque o const ocupa menos memória no computador do usuário.

B) Porque elementos selecionados do DOM raramente precisam ter sua referência reatribuída, aumentando a segurança e previsibilidade do código.

C) Porque o const permite que o elemento HTML seja deletado automaticamente após o uso.

D) Porque apenas variáveis const podem receber eventos de clique (click).

E) Porque o Bootstrap não reconhece variáveis declaradas com let.

Questão 7 - Considere um botão HTML definido como: <button id="btn-acao" class="btn btn-primary">Enviar</button>.

Qual a forma correta de alterar a cor desse botão para vermelho (classe btn-danger) e desativá-lo após o clique?

A) btn.style.color = 'red'; btn.disabled = true;

B) btn.classList.replace('btn-primary', 'btn-danger'); btn.disabled = true;

C) btn.innerHTML = 'btn-danger'; btn.stop();

D) document.querySelector('btn-primary').remove();

E) btn.setAttribute('bootstrap', 'danger');

Unidade 4: Comportamento e Interatividade

Nesta unidade final, consolidamos a fusão entre a arquitetura de informação, o design emocional e a engenharia de software front-end. O foco é transformar interfaces estáticas em fluxos transacionais seguros, onde a coleta de dados respeita os critérios de usabilidade e o dinamismo da web moderna.

4.1. Links, Âncoras e Navegação: Fluxos de navegação e usabilidade para internet

A navegação é o sistema de sinalização de uma interface web. Sem uma estrutura de caminhos clara, o usuário experimenta o fenômeno da desorientação espacial digital.

4.1.1. Links Coesos e Acessibilidade

Os links **(<a>)** e as âncoras não servem apenas para conectar páginas, mas para guiar o fluxo cognitivo. Sob a ótica de IHC, os links devem possuir affordance claro (o usuário deve bater o olho e saber que o elemento é clicável) e textos descritivos. Evite o uso de expressões como **"Clique aqui"** ou **"Saiba mais"**; prefira "**Acesse o Relatório de TI"** ou **"Visualize seu Inventário"**.

4.1.2. Navegação Estruturada com Bootstrap 5

O Bootstrap resolve o desafio da navegação consistente através de componentes padronizados como a **.navbar**, **.nav-tabs** (abas) e as **breadcrumbs** (trilhas de navegação).

- **Breadcrumbs (Migalhas de Pão):** Indicam a posição exata do usuário na hierarquia do sistema (ex: Home > Inventário > Servidores). Isso reduz a carga de memória de trabalho do usuário, permitindo que ele retorne a níveis anteriores com apenas um clique.

4.2. Formulários: Criação de componentes de entrada de dados e validação de requisitos

Os formulários constituem o principal ponto de convergência transacional em sistemas de informação na web. Sob a ótica da Interação Humano-Computador (IHC), eles representam o canal de diálogo direto onde o usuário deixa de ser um espectador passivo e passa a fornecer dados para o sistema. Compreender a anatomia de um formulário HTML e sua estilização estrutural com o Bootstrap 5 é o primeiro passo para construir experiências digitais eficientes e livres de frustração.

4.2.1. Conceitos Básicos de Formulários HTML

Para que a coleta de dados ocorra de forma estruturada, o HTML fornece um conjunto de tags semânticas específicas. Cada elemento possui um papel bem definido na arquitetura da interface:

<form>: É o container principal que delimita a área do formulário. Ela possui dois atributos fundamentais para a comunicação com o servidor:

> **action:** Define o destino (URL ou endpoint da API) para onde os dados coletados serão enviados.
>
> **method:** Especifica o método HTTP de envio. Os mais comuns são o GET (que anexa os dados explicitamente na URL, indicado para buscas) e o POST (que envia os dados ocultos no corpo da requisição, indicado para cadastros e envio de informações sensíveis).

<label>: Representa o rótulo descritivo de um campo. Sua existência é mandatória para a acessibilidade (leitores de tela) e usabilidade. Através do atributo **for** (que deve apontar exatamente para o id do input correspondente), o **<label>** expande a área clicável do campo, permitindo que o usuário foque no input ao clicar no texto do rótulo.

<input>: É o componente mais versátil de entrada de dados. Ela não possui tag de fechamento e altera completamente seu comportamento e validação nativa dependendo do atributo type. Os tipos fundamentais incluem:

> **type="text":** Caixa de texto simples para dados alfanuméricos gerais.
>
> **type="email":** Valida nativamente se o texto inserido possui a estrutura de um endereço eletrônico válido (presença de "@" e domínio).
>
> **type="password":** Mascara os caracteres digitados por questões de privacidade e segurança visual.

type="submit": Define o botão que engatilha o disparo e o envio do formulário.

name: Enquanto o id é utilizado para manipulação no JavaScript e CSS local, o atributo name é a identidade do campo para o servidor. Sem o atributo name, os dados do input não são empacotados na requisição de envio, tornando o campo inútil para o processamento back-end.

<select> É utilizado para criar uma caixa de seleção (uma lista suspensa ou dropdown), permitindo que o usuário escolha uma ou mais opções de uma lista predefinida. É um componente clássico em formulários, ideal para economizar espaço na tela quando há várias alternativas disponíveis. Os atributos incluem:

value: É o dado real que será enviado para o servidor ou capturado pelo JavaScript. O texto que fica entre <option> e </option> é apenas o rótulo visual que o usuário lê. No exemplo acima, se o usuário escolher "Design UX/UI", o back-end receberá apenas a string "ux".

name: Atua como a chave/identificador do campo ao enviar o formulário.

disabled: Pode ser aplicado tanto no <select> inteiro (para congelar o campo) quanto em uma <option> específica (como a opção inicial "Selecione uma opção...", forçando o usuário a escolher outra válida).

selected: Define qual opção virá marcada por padrão quando a página for carregada.

multiple: Transforma a lista suspensa em uma caixa de rolagem fixa, permitindo que o usuário selecione múltiplas opções simultaneamente (segurando Ctrl ou Cmd).

size: Define quantas opções ficarão visíveis ao mesmo tempo na tela antes que o usuário precise rolar a lista.

4.2.2. Componentização e Estilização com Bootstrap 5

O HTML nativo renderiza componentes visuais heterogêneos e esteticamente datados, variando de acordo com o sistema operacional do usuário. O Bootstrap 5 padroniza e moderniza essa interface através de classes utilitárias que garantem responsividade e consistência visual:

.form-control: Classe aplicada a elementos **<input>** e **<textarea>**. Ela remove as bordas nativas pesadas, estiliza o foco com um efeito suave de sombreamento azul, padroniza o espaçamento interno (padding) e força o elemento a ocupar 100% da largura do seu container pai, adaptando-se perfeitamente a telas mobile.

.form-label: Aplicada à tag <label>, ajusta o alinhamento, a margem inferior e o peso da fonte para garantir o correto distanciamento em relação ao campo de entrada.

.mb-3 (Margin Bottom 3): Classe de espaçamento usada para agrupar conjuntos de label e input em blocos isolados, impedindo que os campos fiquem visualmente colados uns nos outros e reduzindo a carga cognitiva durante a leitura do formulário.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cadastro de Notícias - Unidade 4</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">

    <div class="container my-5">
        <div class="row justify-content-center">
            <div class="col-md-6 bg-white p-4 rounded shadow-sm">
                <h3 class="mb-4">Publicar Nova Matéria</h3>

                <form id="form-noticia" novalidate>
                    <div class="mb-3">
                        <label for="titulo" class="form-label">Título da Notícia</label>
                        <input id="titulo" type="text" class="form-control" required>
                        <div class="invalid-feedback">Por favor, insira um título válido.</div>
                    </div>
                    <div class="mb-3">
                        <label for="categoria" class="form-label">Categoria</label>
                        <select id="categoria" class="form-select" required>
                            <option value="">Selecione...</option>
                            <option value="Tecnologia">Tecnologia</option>
                            <option value="Desenvolvimento">Desenvolvimento</option>
                            <option value="Design UX">Design UX</option>
                        </select>
                        <div class="invalid-feedback">Selecione uma categoria técnica.</div>
                    </div>
                    <div class="mb-3">
                        <label for="resumo" class="form-label">Resumo</label>
                        <textarea id="resumo" class="form-control" rows="3" required></textarea>
                        <div class="invalid-feedback">O resumo é obrigatório.</div>
                    </div>
                    <button type="submit" class="btn btn-primary w-100" id="btn-enviar">
                        Enviar para o Servidor
                    </button>
                </form>
                <div id="status-envio" class="mt-3"></div>
            </div>
        </div>
    </div>
</body>
</html>
```

exercicio_24.html

4.2.3. O Fluxo de Envio Assíncrono com JavaScript (Fetch API)

O comportamento padrão de um formulário HTML ao disparar o evento submit é recarregar a página inteira para enviar os dados. No desenvolvimento front-end

moderno, esse comportamento quebra a fluidez da aplicação. Utilizando AJAX através da Fetch API, interceptamos esse envio para processar os dados em segundo plano:

Intercepção do Evento: O JavaScript monitora o formulário através do escutador addEventListener('submit', ...).

Prevenção do Padrão (preventDefault()): A primeira instrução da função de envio deve ser event.preventDefault(). Isso cancela o refresh automático do navegador, mantendo o usuário na mesma página.

Captura de Valores: Através da propriedade .value de cada input selecionado, o JavaScript extrai as strings digitadas em tempo real.

Disparo Assíncrono (fetch): O método fetch() envia uma requisição assíncrona para o script do servidor (como um arquivo .php externo), utilizando o método POST e convertendo os dados capturados em uma string estruturada no formato JSON via JSON.stringify().

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cadastro de Notícias - Unidade 4</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">

    <div class="container my-5">
        <div class="row justify-content-center">
            <div class="col-md-6 bg-white p-4 rounded shadow-sm">
                <h3 class="mb-4">Publicar Nova Matéria</h3>

                <form id="form-noticia" novalidate>
                    <div class="mb-3">
                        <label for="titulo" class="form-label">Título da Notícia</label>
                        <input id="titulo" type="text" class="form-control" required>
                        <div class="invalid-feedback">Por favor, insira um título válido.</div>
                    </div>
                    <div class="mb-3">
                        <label for="categoria" class="form-label">Categoria</label>
                        <select id="categoria" class="form-select" required>
                            <option value="">Selecione...</option>
                            <option value="Tecnologia">Tecnologia</option>
                            <option value="Desenvolvimento">Desenvolvimento</option>
                            <option value="Design UX">Design UX</option>
                        </select>
                        <div class="invalid-feedback">Selecione uma categoria técnica.</div>
                    </div>
                    <div class="mb-3">
                        <label for="resumo" class="form-label">Resumo</label>
                        <textarea id="resumo" class="form-control" rows="3" required></textarea>
```

```
                    <div class="invalid-feedback">O resumo é obrigatório.</div>
                </div>
                <button type="submit" class="btn btn-primary w-100" id="btn-enviar">
                    Enviar para o Servidor
                </button>
            </form>
            <div id="status-envio" class="mt-3"></div>
        </div>
    </div>
</div>
<script>
    const form = document.getElementById('form-noticia');
    const statusEnvio = document.getElementById('status-envio');
    form.addEventListener('submit', async (event) => {
        // 1. Evita que a página dê refresh
        event.preventDefault();
        // 2. Validação visual do Bootstrap 5
        if (!form.checkValidity()) {
            event.stopPropagation();
            form.classList.add('was-validated');
            return; // Interrompe a execução se houver erros
        }
        form.classList.add('was-validated');
        statusEnvio.innerHTML = `<div class="alert alert-info">Enviando dados
assincronamente...</div>`;
        // 3. Construção do Objeto JSON com os dados do Formulário
        const dadosFormulario = {
            titulo: document.getElementById('titulo').value,
            categoria: document.getElementById('categoria').value,
            resumo: document.getElementById('resumo').value,
            autor: "Prof. Robson Carmo" // Dado estático inserido pelo sistema
        };
        try {
            // 4. Envio dos dados para a API Externa via método POST
            const response = await fetch('URL', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(dadosFormulario) // Converte o objeto JS para texto
JSON
            });
            if (response.ok) {
                statusEnvio.innerHTML = `
                    <div class="alert alert-success">
                        <strong>Sucesso!</strong> A notícia foi enviada à API via AJAX.
                    </div>`;
                form.reset(); // Limpa os campos
                form.classList.remove('was-validated'); // Limpa os estados visuais
            } else {
                throw new Error("O servidor retornou um status de erro.");
            }
        } catch (error) {
            statusEnvio.innerHTML = `
                <div class="alert alert-danger">
                    <strong>Falha no Envio:</strong> Não foi possível conectar à API. erro:
${error.message}
                </div>`;
        }
    });
</script>
</body>
</html>
```

exercicio_25.html

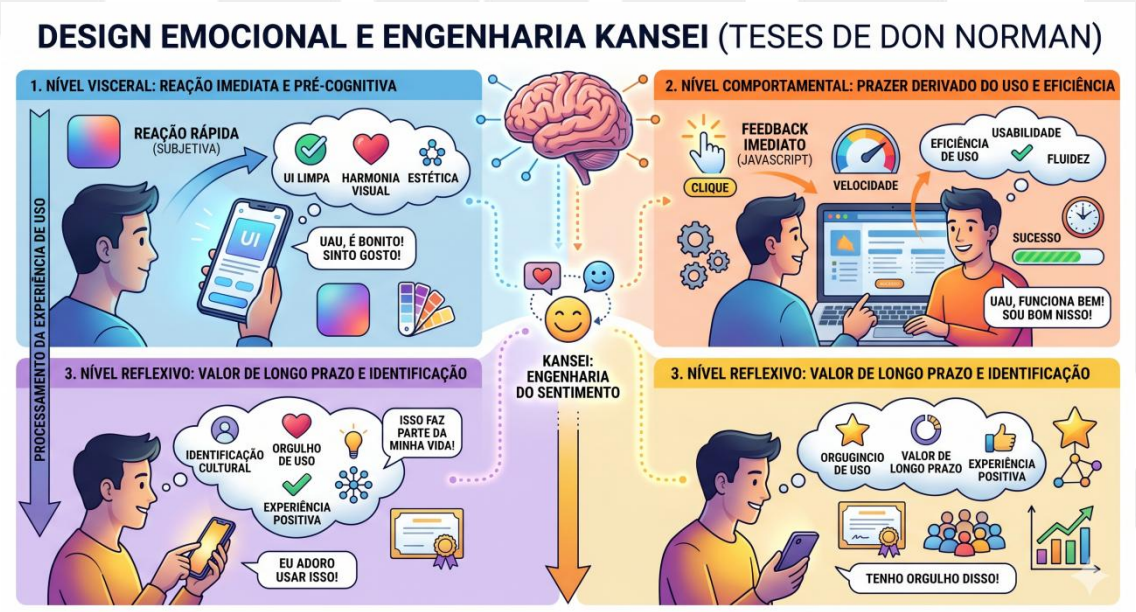
4.3. Interação Social e Emocional: Como o design influencia a adoção de tecnologias e a experiência emocional

A tecnologia não opera em um vácuo puramente racional. Interfaces evocam respostas emocionais nos usuários que determinam o sucesso ou o fracasso comercial de um software.

4.3.1. Design Emocional e Engenharia Kansei

Seguindo as teses de Don Norman (Design Emocional), a experiência de uso é processada em três níveis pelo cérebro humano:

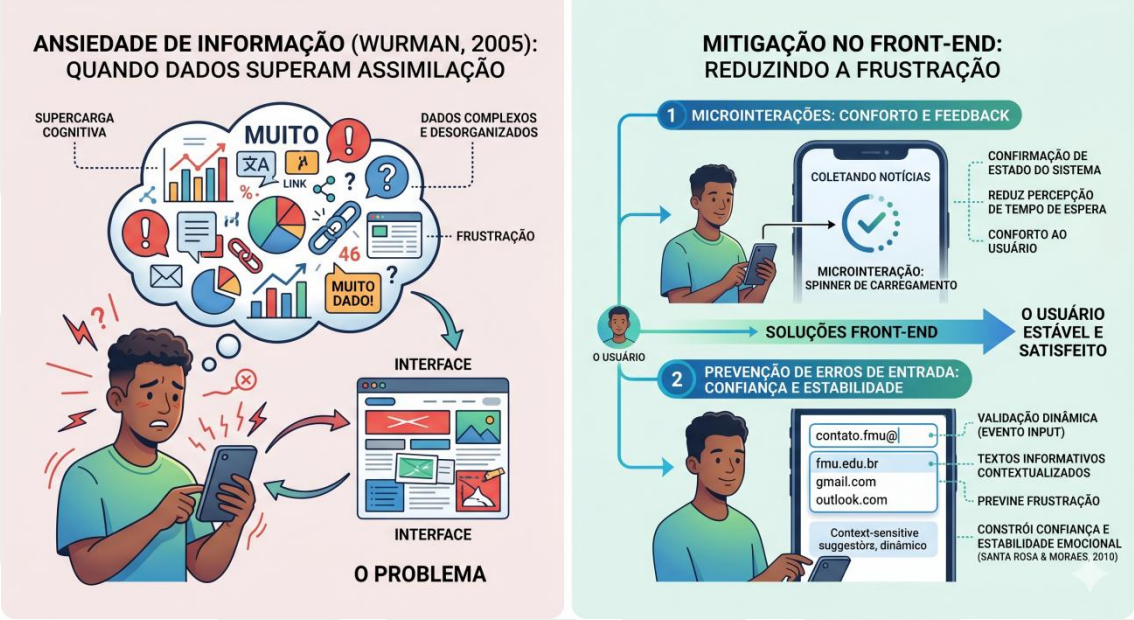
- **Visceral:** A reação imediata e pré-cognitiva à estética visual do sistema (UI limpa, cores harmoniosas).
- **Comportamental:** O prazer derivado da eficiência de uso, usabilidade e feedback imediato (o quão rápido o JavaScript responde ao clique).
- **Reflexivo:** O valor de longo prazo, a identificação cultural e o orgulho gerado pelo uso do produto.



4.3.2. Ansiedade de Informação e Mitigação de Erros

De acordo com Wurman (2005), a Ansiedade de Informação ocorre quando a quantidade de dados gerados pela interface supera a capacidade do usuário de assimilá-la. No front-end, mitigamos esse problema aplicando:

- **Microinterações:** Pequenas animações que indicam estados do sistema (ex: o spinner de carregamento). Elas confortam o usuário, confirmando que a requisição de rede está ativa e reduzindo a percepção do tempo de espera.
- **Prevenção de Erros de Entrada:** Quando validamos um e-mail no ato da digitação com o evento input ou exibimos textos informativos contextualizados, prevenimos a frustração e construímos uma relação de confiança e estabilidade emocional entre o homem e a máquina (Santa Rosa & Moraes, 2010).



Questões de Múltipla Escolha

Questão 1 - No desenvolvimento de sistemas web, a escolha entre métodos de envio de dados através da tag <form> impacta diretamente a segurança e a arquitetura da aplicação. Um tecnólogo em Design de Interfaces precisa projetar a tela de alteração de senha de um usuário. Consideranto as boas práticas de desenvolvimento e os conceitos de IHC, quais atributos da tag <form> devem ser utilizados?

A) method="GET" e action apontando para uma URL pública com os dados na barra de endereço.

B) method="POST" para garantir que os dados sensíveis sejam enviados mascarados no corpo da requisição, mantendo a privacidade.

C) method="PUT" associado à classe .form-control do Bootstrap para bloquear o botão de envio.

D) method="FETCH" para forçar o navegador a recarregar a página automaticamente.

E) Apenas o atributo id, dispensando os atributos method ou action na camada cliente.

Questão 2 - O uso de componentes de navegação padronizados do Bootstrap 5, como as Breadcrumbs (Migalhas de Pão), resolve desafios complexos de usabilidade descritos na literatura de Interação Humano-Computador. Sob o ponto de vista cognitivo do usuário, a principal vantagem de aplicar Breadcrumbs em um sistema de grande escala é:

A) Aumentar a velocidade de download de scripts assíncronos armazenados no cache.

B) Substituir a necessidade de utilizar links tradicionais (<a>) e barras de menu (.navbar).

C) Reduzir a carga de memória de trabalho do usuário, situando-lo na hierarquia do sistema e permitindo o retorno a níveis anteriores com um único clique.

D) Forçar o navegador a renderizar os elementos visuais em Modo Escuro de forma nativa.

E) Validar o atributo name dos inputs em tempo de execução.

Questão 3 - Considere o seguinte fluxo lógico implementado em um script de front-end:

JavaScript

```
form.addEventListener('submit', (event) => {
event.preventDefault();
if (!form.checkValidity()) {
event.stopPropagation();
form.classList.add('was-validated');
return;
}
// lógica de envio...
});
```

Considerando a integração entre JavaScript e a validação nativa de formulários do Bootstrap 5, a instrução event.preventDefault() cumpre o papel de:

A) Validar se o e-mail digitado possui o caractere "@".

B) Interromper a execução de estilos CSS customizados.

C) Injetar a classe .is-invalid em todos os inputs de texto simultaneamente.

D) Cancelar o comportamento padrão de envio do HTML, impedindo o recarregamento total da página (refresh) para permitir o processamento assíncrono.

E) Enviar os dados automaticamente para uma API externa via método POST.

Questão 4 - Donald Norman, em sua obra Design Emocional, argumenta que as interfaces evocam respostas psicológicas estruturadas em três níveis de processamento cerebral. Quando um desenvolvedor front-end foca em garantir que o JavaScript responda de maneira imediata ao clique do usuário, reduzindo o tempo de espera através de microinterações eficientes, ele está atuando diretamente no nível:

A) Visceral.

B) Comportamental.

C) Reflexivo.

D) Semântico.

E) Sintático.

Questão 5 - De acordo com Wurman (2005), a "Ansiedade de Informação" é uma reação gerada quando o volume de dados expostos por uma interface ultrapassa a capacidade humana de processamento e assimilação. No desenvolvimento de formulários complexos, qual técnica de front-end mitiga diretamente esse problema?

A) Utilizar estilos inline (style="...") com cores altamente contrastantes em todos os elementos.

B) Remover todos os rótulos (<label>) para deixar a interface visualmente mais limpa.

C) Agrupar opções de um elemento <select> e fornecer feedbacks visuais contextualizados passo a passo.

D) Forçar o usuário a digitar todos os dados em um único campo de texto sem divisão de blocos.

E) Configurar as requisições assíncronas para operar de forma síncrona, travando a tela.

Questão 6 - Analise o trecho de código JavaScript abaixo, projetado para capturar a escolha de um usuário em um formulário:

JavaScript

```
const seletor = document.getElementById('categoria');
seletor.addEventListener('change', () => {
alert("Opção selecionada: " + seletor.value);
});
```

Considerando que o elemento selecionado é uma tag <select> estilizada com a classe .form-select do Bootstrap, o evento 'change' será disparado quando:

A) O usuário passar o ponteiro do mouse por cima da caixa de seleção.

B) A página HTML terminar de ser carregada completamente pelo navegador.

C) O usuário submeter o formulário clicando em um botão do tipo submit.

D) Houver a alteração do estado da seleção, ou seja, quando o usuário escolher uma opção diferente da atual no menu suspenso.

E) Ocorrer uma falha na requisição assíncrona enviada à API externa.

Questões Discursivas

Questão 1 - No desenvolvimento de aplicações Web modernas, a transição do modelo tradicional de submissão de formulários para o modelo assíncrono (AJAX/Fetch API) alterou significativamente a experiência do usuário (UX).

Com base nos conceitos de Interação Humano-Computador (IHC) e no código estudado na Unidade 4, responda aos itens a seguir:

1 - Explique a diferença de comportamento prático na interface do usuário entre um formulário enviado de forma tradicional (síncrona) e um formulário enviado via Fetch API (assíncrona).

2 - Sob a ótica do Design Emocional, justifique a importância de se utilizar elementos de feedback visual (como spinners de carregamento e classes de validação do Bootstrap) durante o fluxo de uma requisição assíncrona.

Questão 2 - Considere o cenário em que um desenvolvedor front-end cria uma página de cadastro sem utilizar a tag <label>, inserindo apenas os elementos <input> isolados e contando exclusivamente com o atributo placeholder="..." para indicar o que deve ser digitado.

A partir dos conceitos de Acessibilidade Digital e Semântica HTML, faça o que se pede:

1- Demonstre o impacto negativo dessa abordagem na usabilidade para usuários que dependem de tecnologias assistivas (como leitores de tela).

2- Apresente a correção estrutural em código HTML e Bootstrap 5 para um campo de entrada de "E-mail", demonstrando a associação correta entre o rótulo e o campo de texto.

HTML, CSS, JAVASCRIPT e BOOTSTRAP

A criação de um sistema web não se resume a fazer a interface funcionar; é fundamental que o código seja legível, organizado e fácil de manter ao longo do tempo. Em projetos profissionais, o desenvolvimento é realizado em equipe, o que exige a adoção de padrões que permitam a qualquer desenvolvedor entender, modificar ou corrigir o sistema sem dificuldades. Esta unidade aborda as principais boas práticas para estruturar seus projetos front-end de forma profissional.

5.1. Boas Práticas de Organização e Estruturação de Arquivos

Uma arquitetura limpa (Clean) começa pela separação física e lógica das responsabilidades de cada tecnologia:

HTML: Cuida estritamente da estrutura e do significado (semântica).

CSS: Cuida estritamente da apresentação e da estética visual.

JavaScript: Cuida estritamente do comportamento e da lógica de interatividade.

5.1.1. Estrutura de Pastas de um Projeto Profissional

Nunca misture todos os arquivos na raiz do projeto. A convenção de mercado dita que os recursos devem ser categorizados em subdiretórios claros:

```
meu-projeto/
│
├── index.html                  # Página principal (raiz)
├── cadastro.html               # Outra página estrutural
│
├── assets/css/                 # Arquivos de estilo
│       └── estilos.css         # CSS customizado do projeto
│
├── assets/js/                  # Scripts de comportamento
│       └── main.js             # Lógica global/DarkMode/Validações
│
└── assets/images/              # Mídias e ícones estáticos
        └── logo.png
```

5.1.2. Boas Práticas na Criação de Arquivos .css

1. Não duplique o Bootstrap: O Bootstrap já fornece a base de layout. Seu arquivo estilos.css deve servir exclusivamente para customizações pontuais (como a identidade visual da sua marca ou regras que o framework não cobre).

2. Ordem de Importação no HTML: Sempre importe o CSS do Bootstrap antes do seu arquivo customizado. Isso garante que as suas regras tenham prioridade e consigam sobrescrever o framework quando necessário.
3. Uso de Classes em vez de IDs: Utilize seletores de classe (.minha-classe) para aplicar estilos no CSS. IDs (#meu-id) possuem uma especificidade muito alta no navegador, o que dificulta a reutilização do estilo em outros elementos.

5.1.3. Boas Práticas na Criação de Arquivos .js

1. **Escopo Isolado:** Utilize const e let para declarar variáveis, evitando o escopo global.
2. **Funções com Responsabilidade Única:** Cada função deve fazer apenas uma coisa. Por exemplo: uma função seleciona os dados, outra valida e uma terceira envia para a API.
3. **Um arquivo para cada contexto:** Se a lógica de validação de formulários crescer, separe-a em um arquivo validacoes.js e mantenha a lógica da interface geral em um main.js.

5.2. Componentização e Reutilização de Código

Componentizar significa quebrar a interface em pedaços menores, independentes e reutilizáveis (como blocos de montar).

5.2.1. Como Reutilizar Componentes com Bootstrap

O Bootstrap facilita a componentização porque ele fornece padrões visuais prontos. Para reutilizar um componente de forma limpa:

1. **Crie uma estrutura HTML padrão:** Se você precisa de vários cartões de notícias, defina uma estrutura de classe única baseada nos cards do Bootstrap.
2. **Injete dados dinamicamente com JavaScript:** Em vez de copiar e colar o código HTML dez vezes, use o JavaScript para ler uma lista de dados (como um array ou JSON) e gerar os componentes dinamicamente em loop usando Template Literals e o método appendChild().

5.3. O Que Não Deve Ser Feito: Más Práticas a Evitar

Para garantir que o sistema não se torne um problema insolúvel no futuro, os alunos devem evitar os seguintes padrões de código:

1. Estilos Inline (style="..." no HTML)

O erro: Escrever regras de CSS diretamente dentro das tags HTML (ex:).

Por que evitar: Isso quebra totalmente a separação de responsabilidades, infla o tamanho do arquivo HTML e torna impossível fazer uma alteração visual global no sistema no futuro, pois o desenvolvedor terá que caçar tag por tag.

2. JavaScript Inline ou Atributos de Evento Legados (onclick="...")

O erro: Chamar funções diretamente no HTML (ex: <button onclick="enviarDados()">).

Por que evitar: Dificulta a manutenção e testes do código. O correto sob a ótica de IHC e arquitetura moderna é manter o HTML limpo e gerenciar o comportamento exclusivamente dentro do arquivo .js utilizando o método addEventListener('click', ...).

3. Código Espaguete e Loops de Seleção Redundantes

O erro: Ficar fazendo buscas no DOM desnecessariamente toda vez que um bloco é executado, ou criar funções gigantescas misturando validação, animação visual e requisições de rede no mesmo lugar.

Por que evitar: Reduz drasticamente a performance do navegador. O correto é selecionar o elemento uma única vez no início do script (armazenando-o em uma const) e reutilizar essa referência.

4. Ignorar o Tratamento de Erros em Requisições

O erro: Fazer um fetch() assumindo que a internet nunca vai oscilar ou que o servidor PHP nunca vai cair, deixando o código sem uma estrutura de proteção.

Por que evitar: Se a API falhar, o JavaScript simplesmente para de funcionar silenciosamente, quebrando a experiência do usuário. O uso do bloco try/catch é obrigatório para que o sistema consiga falhar de forma elegante, exibindo um alerta visual compreensível (como um componente .alert-danger do Bootstrap) para o usuário.

Exemplo de Código HTML bem Estruturado

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>FMU Tech</title>
    <link rel="icon" href="assets/images/fmu.png"/>
    <!-- Utilizando o Bootstrap -->
    <link href="assets/css/bootstrap.min.css" rel="stylesheet">
    <!-- CSS customizado da minha página -->
    <link rel="stylesheet" href="assets/css/main.css" />
</head>
<body>
    <!-- Start Nav -->
    <nav class="navbar navbar-expand-md navbar-dark bg-dark mb-4">
        <div class="container">
            <span class="navbar-brand mb-0 h1">
                <img src="assets/images/fmu.png" alt="FMU" /> | Tech
            </span>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item"><a class="nav-link" href="#">Menu 1</a></li>
                    <li class="nav-item"><a class="nav-link" href="#">Menu 2</a></li>
                    <li class="nav-item"><a class="nav-link" href="#">Menu 3</a></li>
```

```
                </ul>
            </div>
        </div>
    </nav>
    <!-- End Nav -->
    <!-- Start Header -->
    <header class="container mb-4">
        <h1 class="display-6 fw-bold">Título Principal da Página</h1>
        <p class="text-muted">Subtítulo da página</p>
    </header>
    <!-- End Header -->
    <!-- Start Main -->
    <main class="container">
        <section id="first-section" class="section-1">
            <div class="row m-3 text-white">O conteúdo da primeira sessão entra aqui.</div>
        </section>
        <section id="secondy-section" class="section-2">
            <div class="row m-3 text-white">O conteúdo da segunda sessão entra aqui.</div>
        </section>
        <section id="third-section" class="section-3">
            <div class="row m-3 text-white">O conteúdo da terceira sessão entra aqui.</div>
        </section>
    </main>
    <!-- End Main -->
    <!-- Start Footer -->
    <footer class="bg-dark text-white text-center py-3 mt-5">
        <p class="mb-0">© 2026 FMU - Rodapé</p>
    </footer>
    <!-- End Footer -->
    <!-- JS Files -->
    <script src="assets/js/bootstrap.bundle.min.js"></script>
    <script src="assets/js/main.js"></script>
    <!-- End JS Files -->
</body>
</html>
```

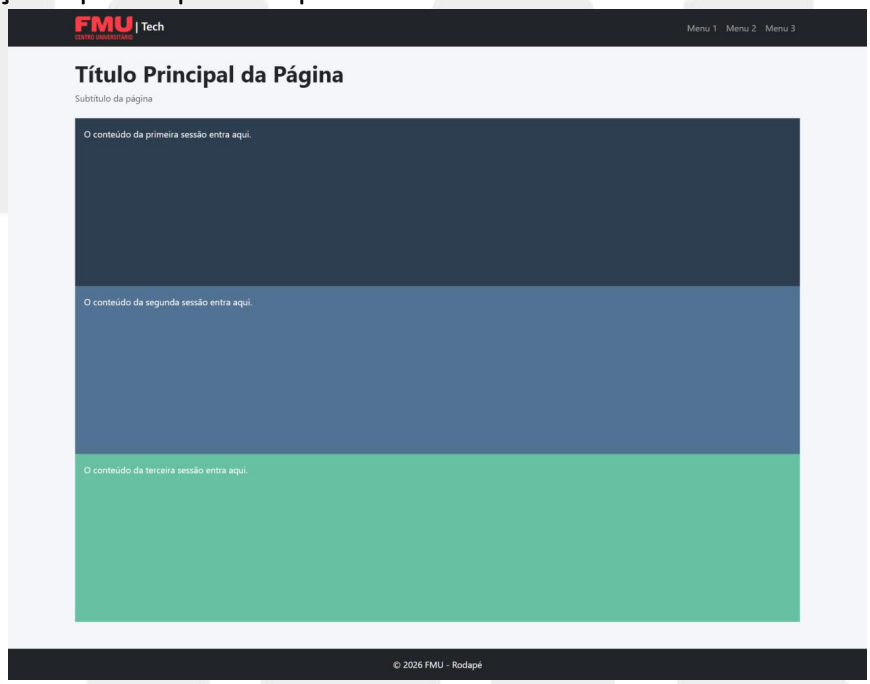
exercicio_26.html

```
body {
    height: 100%;
    margin: 0; /* Remove margens padrão do navegador */
    background-color: #f4f6f9;
    transition: background-color 0.3s ease;
}

footer {
    background: #111a2d;
    color: white;
    text-align: center;
    padding-top:20px
}

main {
    min-height: 100vh; /* Ocupa pelo menos 100% da altura da tela */
    box-sizing: border-box; /* Inclui paddings e bordas no cálculo de tamanho */
}

.section-1 {
    display: flex; /* Ativa o modelo Flexbox */
    background-color: #2c3e50;
    padding: 0px;
    justify-content: space-between; /* Distribui itens com espaço entre eles */
    align-items:flex-start;
    min-height: 300px;
}

.section-2 {
    display: flex; /* Ativa o modelo Flexbox */
    background-color: #527293;
    padding: 0px;
    justify-content: space-between; /* Distribui itens com espaço entre eles */
    align-items:flex-start;
    min-height: 300px;
}
```

```css
.section-3 {
    display: flex; /* Ativa o modelo Flexbox */
    background-color: #67c0a1;
    padding: 0px;
    justify-content: space-between; /* Distribui itens com espaço entre eles */
    align-items:flex-start;
    min-height: 300px;
}
```

assets\main.css

Renderização esperada para computador



Renderização esperada para celular

Projeto Prático

📋 Cenário do Case

Você foi contratado por uma startup para desenvolver a interface Web de uma plataforma SaaS voltada para a gestão de infraestrutura técnica de pequenas e médias empresas. O gerente de produto (PO) entregou a estrutura básica em formato HTML estático (o template fornecido).

Sua missão é dar vida à interface dividindo os requisitos entre as três seções estruturais predefinidas no arquivo, integrando lógica de programação e aplicação de estados de IHC de forma puramente local.

🛠️ Instruções e Requisitos do Exercício

Utilizando como template de código HTML o arquivo exercicio_26.html, você deverá cumprir as seguintes etapas de engenharia front-end:

1. Customização da Identidade Visual e Navegação (Header & Nav)

Requisito 1.1: Altere os textos genéricos do menu de navegação para links semânticos e focados no negócio da plataforma. O "Menu 1" deverá se chamado por “Cadastrar Ativo”, o "Menu 2" por “Lista de Servidores” e o "Menu 3" por “Ajustes Visuais”.

Requisito 1.2: Modifique o título principal (<h1>) para "Painel de Controle" e o subtítulo para "Gerenciamento de Ativos e Customização de Interface".

2. Implementação do Conteúdo por Seções (<main>)

Sessão 1 (#first-section): O Formulário de Cadastro

Substitua o texto padrão por um formulário estruturado do Bootstrap 5 para o cadastramento de um novo ativo de TI (servidor).

O formulário deve conter os seguintes componentes de entrada básicos:

- Um campo de texto para o **Hostname do Servidor** (classe .form-control).
- Um campo de texto para o **IP do Servidor** (classe .form-control).
- Um menu de seleção (<select> com a classe .form-select) contendo pelo menos 3 **Categorias** (ex: NexCloud, BD Postgreee, Aplicação Python).
- Um botão de **Salvar** estilizado (type="submit").

Lógica JavaScript: Crie um escutador de eventos para o submit deste formulário no arquivo assets/js/main.js. Use o método adequado para evitar o recarregamento

automático da página, capture os valores digitados e exiba um alerta visual de sucesso na

Sessão 2 (#secondy-section): A Tabela Servidores

Esta seção representará a listagem estruturada dos ativos de TI. O HTML deverá ter uma tabela estática do Bootstrap 5 (<table class="table table-striped table-hover">), contendo o cabeçalho (<thead>) com as colunas: ID, Hostname do Servidor, IP do Servidor, Categoria e Status.

Exemplo de dados para a tabela:

ID	Hostname	IP	Categoria	Status
1	server-01	201.43.199.138	BD Postgreee	Ativo
2	server-02	201.43.199.139	NexCloud	Ativo
3	server-03	201.43.199.140	Aplicação Python	Ativo
4	server-04	201.43.199.141	Aplicação Python	Ativo
5	server-05	201.43.199.142	Aplicação Python	Desligado

Sessão 3 (#third-section): Customização de Acessibilidade (Filtro de Contraste Local)

Esta seção demonstrará o controle de propriedades visuais para ergonomia de IHC (Interação Humano-Computador). Crie um botão nesta seção chamado "Alternar Alto Contraste" (utilizando a classe .btn .btn-warning).

Lógica JavaScript: Adicione um escutador de eventos para o clique desse botão. Ao ser acionado, o script deve acessar o DOM local e alterar dinamicamente propriedades CSS específicas (como mudar o tamanho da fonte da tabela da Seção 2 ou alterar a cor de fundo das seções para preto e o texto para branco), simulando um modo de acessibilidade visual para usuários com baixa visão. Você poderá utilizar o arquivo exercicio_22.html como exemplo.

Revisão Para Prova

UNIDADE 1: Fundamentos de IHC e Estrutura Web

Questão 1 - A Interação Humano-Computador (IHC) é uma disciplina preocupada com o design, avaliação e implementação de sistemas computacionais interativos para uso humano. De acordo com a literatura da área, qual é o objetivo primordial ao se projetar uma interface centrada no usuário?

A) Maximizar o uso de recursos de hardware do servidor de banco de dados.

B) Garantir a eficiência, a segurança, a usabilidade e a satisfação do usuário durante a execução de suas tarefas.

C) Eliminar a necessidade de estilização visual através de arquivos CSS externos.

D) Substituir o uso de linguagens de programação por protótipos estáticos de baixa fidelidade.

E) Forçar o usuário a memorizar comandos complexos para aumentar sua velocidade cognitiva.

Questão 2 - O HTML (HyperText Markup Language) é a linguagem de marcação fundamental da web. Sob a ótica da semântica estrutural, qual é a função correta das tags no desenvolvimento de uma página?

A) Definir o comportamento assíncrono e a lógica de negócios em bancos de dados.

B) Configurar o layout responsivo através de regras de transição animada.

C) Atribuir significado e estrutura ao conteúdo, separando logicamente elementos como cabeçalhos, parágrafos e seções.

D) Compilar o código JavaScript diretamente na memória do navegador do usuário.

E) Substituir as requisições de rede executadas pela Fetch API.

Questão 3 - Considere o uso das tags semânticas <header>, <main> e <footer> na estruturação de um documento HTML5. Assinale a alternativa que descreve corretamente a boa prática de arquitetura dessas tags:

A) A tag <footer> deve ser inserida obrigatoriamente dentro da tag <header>.

B) A tag <main> serve para abrigar o conteúdo principal e exclusivo da página, não devendo ser duplicada no mesmo documento.

C) Todas as seções de texto da página devem ser escritas sem o uso de containers, utilizando apenas estilos inline.

D) O uso dessas tags é opcional e não influencia a indexação por motores de busca ou leitores de tela.

E) A tag <header> serve exclusivamente para armazenar scripts de conexão com APIs externas.

Questão 4 - Na sintaxe do HTML, atributos são utilizados para fornecer informações adicionais sobre os elementos. No caso da tag de hiperlink Fale Conosco, o atributo href cumpre o papel de:

A) Definir a cor do texto do link utilizando o padrão hexadecimal do Bootstrap.

B) Especificar o destino do link, ou seja, o endereço da página para onde o usuário será direcionado.

C) Ativar um escutador de eventos do tipo submit na camada cliente.

D) Indicar ao leitor de tela que o elemento deve ser ocultado por questões de acessibilidade.

E) Executar uma conversão automática do documento para o formato JSON.

Questão 5 - A folha de estilo em cascata (CSS) permite a separação entre a estrutura do documento e sua apresentação visual. Quando um desenvolvedor insere regras de estilo dentro de um arquivo independente com a extensão .css e o vincula ao HTML através da tag <link>, ele está utilizando o método:

A) CSS Inline.

B) CSS Incorporado (Internal).

C) CSS Externo (External).

D) CSS Assíncrono.

E) CSS Semântico.

Questão 6 - Os seletores CSS determinam quais elementos HTML receberão as regras de estilo. Se um desenvolvedor aplica a regra .texto-destaque { color: blue; }, a propriedade color afetará:

A) Todos os elementos que possuírem o atributo id="texto-destaque".

B) Exclusivamente a tag <body> da página web.

C) Todos os elementos que possuírem o atributo class="texto-destaque".

D) Apenas o primeiro parágrafo localizado dentro da tag <main>.

E) Os links que realizarem requisições assíncronas via Fetch API.

Questão 7 - A propriedade box-sizing: border-box; é amplamente utilizada no design de layouts CSS modernos. Qual é o seu efeito prático no comportamento dos elementos da interface?

A) Ela faz com que o espaçamento interno (padding) e as bordas sejam incluídos no cálculo da largura e altura totais do elemento.

B) Ela força o elemento a se comportar como uma tabela estática invisível.

C) Ela remove completamente a margem externa (margin) de todas as tags do documento.

D) Ela impede que o arquivo CSS consiga sobrescrever as classes nativas do Bootstrap.

E) Ela converte elementos do tipo block em elementos do tipo inline automaticamente.

UNIDADE 2: Estilização Avançada e Responsividade com Bootstrap

Questão 8 - O Bootstrap é um dos frameworks front-end mais utilizados no mundo. Qual é a principal vantagem de sua adoção no desenvolvimento de interfaces digitais?

A) Ele elimina a necessidade de escrever qualquer código HTML ou JavaScript no projeto.

B) Ele fornece um ecossistema de componentes pré-estilizados e um sistema de grid responsivo que acelera o design de layouts adaptáveis.

C) Ele substitui o banco de dados relacional na validação de credenciais de segurança.

D) Ele desativa nativamente o processamento de animações no navegador para economizar memória.

E) Ele restringe o acesso ao site, permitindo a visualização apenas em computadores desktop.

Questão 9 - O sistema de Grid do Bootstrap é baseado no módulo Flexbox do CSS e divide a largura da tela em um número máximo de colunas. Qual é esse número padrão de colunas utilizado para organizar o layout?

A) 6 colunas.

B) 10 colunas.

C) 11 colunas.

D) 12 colunas.

E) 16 colunas.

Questão 10 - No Bootstrap 5, o gerenciamento de layouts responsivos é feito por meio de classes utilitárias associadas a pontos de quebra (breakpoints). Considere a classe col-md-4 aplicada a uma div. O que essa configuração representa na estrutura da página?

A) A div ocupará 4 pixels de largura fixa em qualquer tipo de monitor.

B) A div ocupará um terço da largura total da linha (4 de 12 colunas) em telas de tamanho médio (medium) ou superior.

C) O elemento será duplicado 4 vezes consecutivas na primeira seção do documento.

D) A div será ocultada da tela quando o usuário acessar o site por um dispositivo mobile.

E) Ativa um espaçamento de margem interna correspondente a 40% do container pai.

Questão 11 - Para alinhar elementos de forma flexível dentro de uma linha utilizando o Bootstrap, os desenvolvedores recorrem a classes utilitárias de alinhamento. A classe justify-content-between aplicada a um container Flexbox tem como função:

A) Distribuir os itens uniformemente, colocando o primeiro item no início e o último item no fim da linha, maximizando o espaço entre eles.

B) Centralizar todos os itens filhos verticalmente dentro do espaço disponível.

C) Forçar os elementos a ocuparem linhas diferentes, quebrando a estrutura de grid.

D) Esconder os elementos excedentes utilizando a propriedade display: none;.

E) Alterar a cor de fundo dos componentes para a cor amarela de alerta (bg-warning).

Questão 12 - Os componentes do tipo Card (Cartões) do Bootstrap são estruturas versáteis para exibição de conteúdos fragmentados, como produtos ou notícias. Qual classe deve ser aplicada ao container interno para delimitar a área de texto principal e aplicar o espaçamento interno padrão do cartão?

A) .card-header

B) .card-footer

C) .card-body

D) .card-title

E) .card-grid

Questão 13 - O Bootstrap utiliza classes de cores semânticas para comunicar estados e hierarquias visuais. Se um desenvolvedor deseja criar um botão de confirmação com a cor verde associada a "sucesso", qual classe ele deve adicionar ao elemento <button>?

A) .btn-primary

B) .btn-danger

C) .btn-warning

D) .btn-success

E) .btn-dark

Questão 14 - As classes de espaçamento do Bootstrap seguem uma nomenclatura mnemônica para facilitar o desenvolvimento. A classe .mb-4 representa:

A) Uma margem interna (padding) na lateral esquerda de nível 4.

B) Uma margem externa inferior (margin-bottom) de nível 4.

C) Um alinhamento centralizado do texto em telas pequenas.

D) Uma borda sólida de 4 pixels de espessura na cor escura.

E) O redimensionamento automático de imagens para exibição em dispositivos móveis.

UNIDADE 3: Programação JavaScript e Manipulação do DOM

Questão 15 - O JavaScript desempenha um papel crucial no desenvolvimento front-end, atuando na camada de comportamento. Qual é a principal função do JavaScript em uma página web moderna?

A) Definir as tags semânticas de cabeçalho e rodapé do documento HTML.

B) Adicionar dinamismo, interatividade, manipulação lógica de dados e controle sobre os elementos da interface em tempo de execução.

C) Armazenar os arquivos de imagens e mídias estáticas diretamente no servidor local.

D) Configurar as regras de transição responsiva do framework Bootstrap de forma puramente estática.

E) Substituir a folha de estilos externa para acelerar a renderização visual do layout.

Questão 16 - Na declaração de variáveis no JavaScript moderno, recomenda-se a substituição do antigo escopo do var pelas palavras-chave const e let. Sobre a diferença entre elas, assinale a alternativa correta:

A) Variáveis declaradas com const podem ter seu valor reatribuído a qualquer momento do script.

B) O let é utilizado exclusivamente para armazenar arrays de objetos JSON obtidos via AJAX.

C) O const declara uma referência imutável, impossibilitando que a variável seja reatribuída ao longo da execução do código naquele escopo.

D) Ambas as palavras-chave são ignoradas pelo navegador caso o site utilize o Bootstrap 5.

E) O let impede que o elemento HTML selecionado sofra manipulação no DOM.

Questão 17 - O Document Object Model (DOM) é a interface de programação para documentos HTML. Quando o navegador carrega uma página, ele cria uma árvore de objetos que representa a estrutura do documento. Qual instrução JavaScript é utilizada para selecionar um elemento específico do HTML com base no seu atributo identificador exclusivo?

A) document.getElementById('id-do-elemento')

B) document.getElementsByTagName('div')

C) document.classList.add('classe-nova')

D) document.createElement('table')

E) document.innerHTML = 'conteudo'

Questão 18 - Para alterar dinamicamente o texto ou o HTML interno contido entre as tags de abertura e fechamento de um elemento selecionado no DOM, o desenvolvedor deve acessar a propriedade:

A) .value

B) .classList

C) .innerHTML

D) .style

E) .addEventListener

Questão 19 - A propriedade classList do JavaScript expõe métodos eficientes para a manipulação de classes CSS de um elemento. Se um desenvolvedor deseja substituir uma classe já existente por outra, ele deve utilizar o método:

A) classList.remove()

B) classList.toggle()

C) classList.add()

D) classList.replace()

E) classList.clear()

Questão 20 - Considere o seguinte comando JavaScript: objeto.classList.replace('bg-light', 'bg-dark');. Qual é o resultado prático dessa instrução na interface do usuário?

A) O arquivo CSS externo será deletado do servidor local da aplicação.

B) A classe 'bg-light' será removida e, simultaneamente, a classe 'bg-dark' será adicionada ao elemento, atualizando o estado visual instantaneamente.

C) O formulário HTML será disparado assincronamente para uma API externa via método POST.

D) O navegador recarregará a página inteira para processar a mudança de cor de fundo.

E) O elemento correspondente será excluído permanentemente da árvore do DOM.

Questão 21 - Os escutadores de eventos (Event Listeners) são essenciais para capturar as interações do usuário. Qual é a sintaxe correta para monitorar o clique de um usuário em um botão armazenado na constante btnSalvar?

A) btnSalvar.onclick(funcao);

B) btnSalvar.addEventListener('click', () => { // lógica });

C) btnSalvar.captureEvent('submit');

D) btnSalvar.innerHTML = 'click';

E) btnSalvar.classList.add('click');

Questão 22 - No desenvolvimento de interfaces dinâmicas, o método document.createElement() e o método appendChild() trabalham em conjunto para:

A) Validar se o endereço IP inserido no formulário é estruturalmente válido.

B) Cancelar o comportamento padrão de envio assíncrono disparado pelo formulário.

C) Sobrescrever o sistema de grid do Bootstrap, travando o layout em telas mobile.

D) Modificar os atributos action e method de uma tag <form> estática.

E) Criar um novo nó de elemento HTML na memória e inseri-lo na árvore do DOM como o último filho de um container pai selecionado.

UNIDADE 4: Comportamento, Formulários e Interatividade Avançada

Questão 23 - Os formulários representam o ponto central de coleta de dados e transações na web. Na sintaxe do HTML, o contêiner <form> agrupa os campos de entrada. Os atributos action e method servem, respectivamente, para definir:

A) O destino (URL ou endpoint da API) para onde os dados serão enviados e o método HTTP de envio (como GET ou POST).

B) A estilização visual do botão e a classe de responsividade do Bootstrap.

C) O identificador exclusivo do script JavaScript e a ordem de tabulação para acessibilidade.

D) O texto explicativo do rótulo e o mascaramento de caracteres de senhas.

E) A limpeza dos campos de texto e a exibição imediata do spinner de carregamento.

Questão 24 - No contexto de IHC e usabilidade de formulários, o uso correto da tag <label> associada a um elemento <input> por meio do atributo for cumpre um papel de extrema importância porque:

A) Altera automaticamente o método HTTP de envio de GET para POST.

B) Permite que os dados viajem criptografados no corpo da requisição de rede.

C) Garante a acessibilidade (leitores de tela correlacionam o rótulo ao campo) e expande a área clicável do input para o usuário.

D) Substitui a necessidade de declarar o atributo name no campo de entrada de dados.

E) Força a aplicação a utilizar estilos inline do tipo anti-design.

Questão 25 - Ao gerenciar o envio de um formulário via JavaScript para implementar um fluxo assíncrono (AJAX), qual é a primeira instrução obrigatória a ser executada dentro do escutador de eventos do tipo 'submit'?

A) form.reset();

B) event.preventDefault();

C) form.classList.add('was-validated');

D) fetch('api_noticias.php');

E) event.stopPropagation();

Questão 26 - Quando coletamos dados de entrada de um formulário utilizando o JavaScript para posterior tratamento ou envio, como extraímos a string de texto digitada pelo usuário dentro de um elemento <input type="text" id="hostname">?

A) Acessando a propriedade document.getElementById('hostname').innerHTML

B) Acessando a propriedade document.getElementById('hostname').value

C) Acessando a propriedade document.getElementById('hostname').classList

D) Chamando o método document.getElementById('hostname').createElement()

E) Verificando o atributo document.getElementById('hostname').getAttribute('type')

Questão 27 - A Fetch API é a ferramenta moderna do JavaScript nativo para a execução de requisições assíncronas. Quando estruturamos o envio de dados via método POST, as informações do formulário coletadas no front-end devem ser transmitidas no corpo da mensagem. Qual instrução converte o objeto de dados do JavaScript em uma string de texto formatada em JSON para essa transmissão?

A) JSON.stringify(dados)

B) JSON.parse(dados)

C) dados.toString()

D) fetch.toJSON(dados)

E) response.json()

Questão 28 - Em sistemas que realizam operações de rede assíncronas (async/await), o uso do bloco estrutural try/catch é uma diretriz de IHC indispensável. A presença do bloco catch tem como objetivo:

A) Impedir que o usuário clique no botão de atualizar a página antes do carregamento do CSS.

B) Validar se os campos obrigatórios do formulário do Bootstrap foram preenchidos pelo usuário.

C) Duplicar automaticamente as linhas estruturais de tabelas estáticas presentes no DOM.

D) Capturar e tratar erros de forma elegante (como queda de conexão ou servidor fora do ar), permitindo exibir um feedback compreensível ao usuário em vez de travar o sistema.

E) Alterar as cores dos botões de alerta de verde para vermelho sem intervenção de classes.

Questão 29 - De acordo com Wurman (2005), a "Ansiedade de Informação" ocorre quando a interface expõe um volume de dados desorganizados que supera a capacidade humana de processamento. No desenvolvimento front-end, a aplicação de microinterações (como a exibição de um Spinner de carregamento do Bootstrap durante uma requisição de rede) mitiga esse problema porque:

A) Reduz a largura de banda consumida pelo arquivo JavaScript do projeto.

B) Altera as propriedades de alto contraste do container pai em tempo de execução.

C) Conforta o usuário ao confirmar visualmente que o sistema está operando ativamente na sua solicitação, reduzindo a percepção do tempo de espera.

D) Cancela a obrigatoriedade de se fazer a validação de dados na camada do servidor.

E) Exclui as dependências de roteamento e navegação de menus secundários.

Questão 30 - Don Norman, em suas teses sobre o Design Emocional, aponta que as experiências humanas com produtos digitais ocorrem em três níveis de processamento. O prazer derivado da eficiência de uso, da usabilidade fluida e da resposta interativa imediata de um script JavaScript a uma ação do usuário corresponde ao nível:

A) Comportamental

B) Visceral

C) Reflexivo

D) Semântico

E) Estático

Bibliografia Básica

DELL, J. Digital Interface Design and Application. [s.l.]: Incorporated, 2015. LEUNG, L. Digital Experience Design: Ideas, Industries, Interaction. [s.l.]: Intellect Books, 2008.

GEEST, T. Web Site Design is Communication Design. [s.l.]: John Benjamins Publishing Company, 2001.

CASTRO, Leandro Nunes de; FERRARI, Daniel Gomes. Introdução à mineração de dados: conceitos básicos, algoritmos e aplicações. São Paulo: Saraiva, 2016. Disponível em: https://integrada.minhabiblioteca.com.br/books/978-85-472-0100-5.

MORAIS, Izabelly Soares de et al. Introdução a big data e internet das coisas (IOT). Porto Alegre: SAGAH, 2018. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788595027640.

SILVA, Fabrício Machado da et al. Inteligência artificial. Porto Alegre: SAGAH, 2019. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788595029392.

Bibliografia Complementar:

AGNER, Luiz. Ergodesign e arquitetura de informação: trabalhando com o usuário. Rio de Janeiro: Quartet, 2009.

CAMARGO, Liriane Soares de. VIDOTTI, Silvana Aparecida Gregório. Arquitetura da Informação - Uma Abordagem Prática. [s.l.]: LTC, 2011.

HARRIS, D.; HARRIS, S. Digital Design and Computer Architecture: From Gates to Processors. [s.l.]: Elsevier Science, 2007.

INFODESIGN. Revista Brasileira de Design da Informação. Disponível em: <https://infodesign.org.br/ infodesign.

SANTA ROSA, José Guilherme; MORAES, Anamaria de. Avaliação e projeto no design de interfaces. Teresópolis: 2AB Editora, 2010.

UXDESIGN. Usabilidade, User Experience, Design de Interação e Tecnologia. Disponível em: <https://brasil.uxdesign.cc

WURMAN, Richard Saul; FREIRE, Virgílio. Ansiedade de informação: como transformar informação em compreensão. São Paulo: Cultura Editores Associados, 2003.

Periódicos Científicos:

Advances in Software Engineering - ISSN 1687-8655

Computación y Sistemas - ISSN 1405-5546

Computational & Applied Mathematics - ISSN 1807-0302

Journal of the Brazilian Computer Society - ISSN 0104-6500