Relatório Teórico do Projeto

Personalização e Integração Web: Café Aurora

Disciplina: Padrões Web **Instituição:** UniFECAF **Aluno:** Gabriel Falcão **Data:** Dezembro/2025

🔗 Link do Site Publicado

Acesse: https://projeto-unifecaf-rocketseat-dezembro-2025.vercel.app

---

1. Explicação do Projeto

1.1 Finalidade

O projeto **Café Aurora** é um site institucional desenvolvido para uma cafeteria artesanal fictícia localizada em São Paulo. A proposta surgiu da necessidade de criar uma presença digital profissional para pequenos negócios locais que não possuem orçamento para contratar programadores.

O site tem como objetivo principal apresentar a cafeteria aos potenciais clientes, oferecendo informações sobre:

- A história e valores da empresa
- O cardápio com preços e descrições
- Galeria de fotos do ambiente
- Horários de funcionamento e localização
- Canal de contato direto via formulário

1.2 Público-Alvo

O público-alvo da aplicação são **pessoas de 25 a 50 anos** que moram ou trabalham na região de São Paulo, especialmente no bairro Vila Mariana. São pessoas que:

- Valorizam experiências gastronômicas diferenciadas
- Buscam ambientes aconchegantes para trabalhar ou socializar
- Utilizam smartphones e computadores para pesquisar estabelecimentos
- Podem ter diferentes níveis de familiaridade com tecnologia

Por isso, priorizei uma interface limpa, intuitiva e que funcione bem em qualquer dispositivo.

1.3 Estrutura do Site

O site foi estruturado como uma **Single Page Application (SPA)** com as seguintes seções:

Seção	Descrição
Header	Logo, navegação principal e menu mobile
Hero	Apresentação visual com call-to-action
Sobre	História da cafeteria e diferenciais
Cardápio	Produtos organizados com filtros por categoria
Galeria	Fotos do ambiente em grid responsivo

Informações	Horários, localização e redes sociais
Contato	Formulário funcional com validação
Footer	Links rápidos e créditos

---

2. Justificativa da Ferramenta Escolhida

2.1 Escolha Estratégica: Código Puro como Base dos Padrões Web

A escolha por desenvolver o projeto utilizando **código puro** (HTML, CSS, JavaScript) foi uma decisão **deliberada e pedagógica**, alinhada com o objetivo central da disciplina de Padrões Web: **compreender profundamente os fundamentos que sustentam TODAS as ferramentas visuais**.

Ferramentas no-code como Webflow, Softr e Glide são construídas sobre HTML, CSS e JavaScript. Ao dominar esses padrões diretamente, adquiri a capacidade de:

1. **Customizar qualquer plataforma visual** — Entendo o código que essas ferramentas geram
2. **Debugar problemas** — Sei identificar e corrigir issues no código exportado
3. **Otimizar performance** — Reconheço código desnecessário e sei refatorar
4. **Superar limitações** — Posso adicionar funcionalidades que a ferramenta não oferece nativamente

2.2 Comparativo: Código Puro vs No-Code

Aspecto	Código Puro	Plataformas No-Code
Aprendizado	Profundo — entende os padrões	Superficial — aprende a ferramenta
Controle	Total sobre cada elemento	Limitado às opções da interface
Custo	Gratuito (hospedagem inclusa)	Planos pagos para funcionalidades
Portabilidade	Funciona em qualquer servidor	Dependente da plataforma
Customização	Ilimitada	Restrita ao que a ferramenta permite
Performance	Otimizada (sem overhead)	Pode ter código desnecessário

2.3 Por que esta abordagem atende ao desafio?

O desafio proposto tem como objetivo que o estudante entenda a **aplicabilidade dos padrões web na realidade do mercado**. Ao desenvolver com código puro:

1. **Demonstrei domínio dos padrões fundamentais** — HTML5 semântico, CSS3 moderno (Custom Properties, Grid, Flexbox), JavaScript ES6+ (Fetch API, IntersectionObserver, async/await)

2. **Apliquei as mesmas técnicas usadas internamente pelas ferramentas no-code** — Qualquer customização em Webflow ou Softr usa exatamente os mesmos conceitos (CSS embed, HTML embed, JavaScript customizado)

3. **Criei uma solução real e funcional** — O site está integrado com API externa (Formspree), é responsivo e acessível

4. **Adquiri conhecimento transferível** — Posso agora usar qualquer ferramenta visual com consciência do que acontece "por baixo dos panos"

2.4 Análise das Plataformas No-Code

Pesquisei as ferramentas sugeridas antes de decidir:

Plataforma	Limitação Encontrada
Webflow	Plano gratuito: 2 projetos, domínio .webflow.io, sem exportar código
Softr	Focado em apps com Airtable, não em sites institucionais estáticos
Glide	Voltado para PWAs mobile, não sites tradicionais
Bubble	Curva de aprendizado alta, plano gratuito muito limitado

Para o escopo do projeto (site institucional com formulário de contato), código puro com deploy gratuito no Vercel/Netlify oferecia a melhor relação custo-benefício e o aprendizado mais profundo.

2.5 Conclusão sobre a Escolha

> *"Entender os padrões web fundamentais é como aprender matemática antes de usar calculadora. As ferramentas no-code são calculadoras poderosas, mas quem entende a matemática por trás pode resolver qualquer problema — inclusive os que a calculadora não resolve."*

Esta abordagem me preparou para o mercado de trabalho onde, frequentemente, é necessário **customizar, debugar e otimizar** soluções criadas em ferramentas visuais. O conhecimento adquirido é permanente e aplicável a qualquer tecnologia futura.

---

3. Aplicação dos Padrões Web

3.1 HTML5 Semântico

Utilizei as tags semânticas do HTML5 para estruturar o conteúdo de forma significativa:

```
<header role="banner">        <!-- Cabeçalho do site -->
<nav role="navigation">       <!-- Navegação principal -->
<main id="main-content">      <!-- Conteúdo principal -->
<section aria-labelledby="">  <!-- Seções temáticas -->
<article>                     <!-- Conteúdo independente -->
<figure> e <figcaption>       <!-- Imagens com legenda -->
<address>                     <!-- Informações de contato -->
<footer role="contentinfo">   <!-- Rodapé -->
```

Benefícios: Melhor SEO, acessibilidade para leitores de tela, código mais legível e manutenível.

3.2 CSS3 Moderno

O CSS foi desenvolvido seguindo práticas modernas:

Custom Properties (Variáveis CSS):

```
:root {
    --color-primary: #8B4513;
    --font-heading: 'Playfair Display', serif;
    --space-md: 1rem;
    --radius-md: 8px;
}
```

Isso permite alterar toda a identidade visual mudando poucos valores.

Layout com Flexbox e Grid:

- **Flexbox** para alinhamentos e distribuição de espaço (header, botões, cards)
- **CSS Grid** para layouts bidimensionais (galeria, grid do cardápio, formulário)

Responsividade com Media Queries:

```
@media (max-width: 768px) {
    .main-nav { position: fixed; right: -100%; }
    .cardapio-grid { grid-template-columns: 1fr; }
}
```

3.3 JavaScript ES6+

O JavaScript foi escrito com funcionalidades modernas:

- **Arrow Functions** para código mais conciso
- **Template Literals** para strings dinâmicas
- **Async/Await** para chamadas de API
- **Destructuring** para extrair valores de objetos
- **IntersectionObserver** para detectar elementos visíveis
- **Fetch API** para envio do formulário

Exemplo de código moderno utilizado:

```
const response = await fetch(form.action, {
    method: 'POST',
    body: formData,
    headers: { 'Accept': 'application/json' }
});
```

---

4. Elementos Customizados com Código

4.1 Sistema de Filtros do Cardápio

Criei um sistema de filtragem dinâmica que permite ao usuário visualizar apenas os produtos de uma categoria específica:

```
filterButtons.forEach(button => {
    button.addEventListener('click', () => {
        const filter = button.dataset.filter;
        items.forEach(item => {
```

```
            const category = item.dataset.category;
            item.classList.toggle('hidden',
                filter !== 'todos' && category !== filter
            );
        });
    });
});
```

Valor agregado: Melhora a experiência do usuário ao permitir navegação rápida pelo cardápio sem recarregar a página.

4.2 Validação de Formulário em Tempo Real

Implementei validação client-side com feedback imediato:

```
const fields = {
    email: {
        validate: (value) => {
            const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            return regex.test(value) ? '' : 'E-mail inválido';
        }
    }
};
```

Valor agregado: Previne envios incorretos, melhora a experiência do usuário e reduz carga no servidor.

4.3 Máscara de Telefone

Desenvolvi uma máscara que formata automaticamente o número:

```
phoneInput.addEventListener('input', (e) => {
    let value = e.target.value.replace(/\D/g, '');
    // Aplica formatação (XX) XXXXX-XXXX
});
```

Valor agregado: Padroniza os dados recebidos e melhora a usabilidade.

4.4 Navegação com Scroll Spy

Implementei detecção automática da seção visível para destacar o link correspondente:

```
const observer = new IntersectionObserver((entries) => {
    entries.forEach(entry => {
        if (entry.isIntersecting) {
            // Atualiza link ativo na navegação
        }
    });
}, { rootMargin: '-20% 0px -80% 0px' });
```

Valor agregado: Orientação visual para o usuário saber em qual seção está.

---

5. Responsividade e Acessibilidade

5.1 Cuidados com Responsividade

O design foi desenvolvido com abordagem **Mobile First**, garantindo:

Recurso	Implementação
Breakpoints	768px (tablet) e 992px (desktop)
Unidades relativas	rem, em, %, vw, vh
Imagens flexíveis	max-width: 100% e object-fit: cover
Menu adaptável	Hambúrguer em mobile, horizontal em desktop
Grid fluido	auto-fill e minmax() para adaptar colunas

5.2 Cuidados com Acessibilidade

Segui as diretrizes WCAG 2.1 implementando:

Skip Link:

```
<a href="#main-content" class="skip-link">
    Pular para o conteúdo principal
</a>
```

ARIA Labels:

```
<button aria-label="Abrir menu de navegação"
        aria-expanded="false"
        aria-controls="main-nav">
```

Alt Text Descritivo:

```
<img alt="Interior aconchegante do Café Aurora
          com mesas de madeira e iluminação quente">
```

Foco Visível:

```
:focus-visible {
    outline: 3px solid var(--color-primary);
    outline-offset: 2px;
}
```

Respeito a Preferências:

```
@media (prefers-reduced-motion: reduce) {
    * { animation-duration: 0.01ms !important; }
```

```
}
```

---

6. Integração com API Externa

6.1 Formspree

O formulário de contato está integrado com o serviço **Formspree**, que permite receber mensagens via e-mail sem necessidade de backend próprio.

Funcionamento:

1. Formulário HTML aponta para endpoint do Formspree
2. JavaScript intercepta o submit e envia via Fetch API
3. Formspree processa e envia e-mail para o proprietário
4. Usuário recebe feedback visual de sucesso/erro

Código da integração:

```
const response = await fetch(form.action, {
    method: 'POST',
    body: formData,
    headers: { 'Accept': 'application/json' }
});

if (response.ok) {
    // Mostra mensagem de sucesso
} else {
    // Mostra mensagem de erro
}
```

Por que Formspree?

- Gratuito para até 50 envios/mês
- Não requer backend próprio
- API REST simples
- Proteção anti-spam incluída

---

7. Aprendizados Obtidos

Desenvolver este projeto me proporcionou diversos aprendizados sobre padrões web:

7.1 HTML Semântico Importa

Aprendi que usar as tags corretas não é apenas "boas práticas" — impacta diretamente em SEO, acessibilidade e manutenibilidade. Um `<article>` comunica muito mais que uma `<div>`.

7.2 CSS Moderno é Poderoso

Custom Properties, Grid e Flexbox eliminaram a necessidade de frameworks CSS. Com código puro consegui criar layouts complexos e responsivos.

7.3 JavaScript Deve ser Progressivo

Implementei funcionalidades que melhoram a experiência, mas o site continua funcional mesmo com JavaScript desabilitado (formulário ainda envia via POST normal).

7.4 Acessibilidade Desde o Início

É muito mais fácil desenvolver acessível desde o começo do que adaptar depois. ARIA labels, contraste e navegação por teclado devem fazer parte do fluxo de desenvolvimento.

7.5 Performance é UX

Lazy loading de imagens, passive event listeners e requestAnimationFrame fazem diferença perceptível na fluidez do site.

7.6 APIs Simplificam

Serviços como Formspree permitem adicionar funcionalidades backend sem escrever código server-side, democratizando o desenvolvimento web.

---

8. Conclusão

O projeto Café Aurora demonstra que é possível criar aplicações web profissionais e acessíveis utilizando apenas os padrões fundamentais da web (HTML, CSS, JavaScript), sem depender de frameworks pesados ou plataformas pagas.

O conhecimento adquirido neste projeto é transferível para qualquer ferramenta no-code/low-code futura, pois entender os padrões subjacentes permite customizar, debugar e otimizar qualquer solução web.

O site está funcional, responsivo, acessível e pronto para publicação, atendendo a todos os requisitos propostos pelo desafio.

Gabriel Falcão Dezembro/2025 UniFECAF - Análise e Desenvolvimento de Sistemas