CENTRO PAULA SOUZA

ESCOLA TÉCNICA PROFESSOR MASSUYUKI KAWANO

Técnico em Desenvolvimento de Sistemas

Gabriel Roberto Rodella de Assis

Matheus Mota Tonini

Nei Junio Nogueira Gomes

Weslley Katsube

URBAN ESTÉTICA AUTOMOTIVA

Tupã - SP

2024

Gabriel Roberto Rodella de Assis

Matheus Mota Tonini

Nei Junio Nogueira Gomes

Weslley Katsube

URBAN ESTÉTICA AUTOMOTIVA

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em 2024 da Etec Prof. Massuyuki Kawano orientado pelo Prof. Allan Carlos Oliveira Alves da Luz, como requisito parcial para a obtenção do título de técnico em Desenvolvimento de Sistemas

Menção do Trabalho: MB

Tupã - SP

2024

CENTRO PAULA SOUZA

ESCOLA TÉCNICA PROFESSOR MASSUYUKI KAWANO

Técnico em Desenvolvimento de Sistemas

Gabriel Roberto Rodella de Assis

Matheus Mota Tonini

Nei Junio Nogueira Gomes

Weslley Katsube

URBAN ESTÉTICA AUTOMOTIVA

Apresentação para a Banca em caráter de validação do título de Desenvolvimento de Sistemas.

BANCA EXAMINADORA:

Prof. Esp. Allan Carlos Oliveira Alves da Luz
Orientador

Prof. Esp. Bruno Leonardo Morette
Avaliador

Guilherme Henrique Sanches Anjolette
Avaliador

Eduardo Decioli
Avaliador

João Vitor de Oliveira Cara
Avaliador

Tupã, 3 de dezembro de 2024

RESUMO

O URBAN Estética Automotiva é um sistema web criado para otimizar o agendamento de serviços em empresas de estética automotiva. A plataforma permite que os clientes escolham e reservem dias e horários disponíveis, sem necessidade de contato direto com a empresa, facilitando o processo e garantindo uma gestão mais eficiente dos agendamentos.

Palavras-chave: Sistema de agendamento, JavaScript, Node, Next, MySQL.

ABSTRACT

URBAN Automotive Aesthetics is a web system designed to optimize service scheduling for automotive aesthetics businesses. The platform allows customers to select and book available dates and times without the need for direct contact with the company, streamlining the process and ensuring more efficient appointment management.

Keywords: Scheduling system, JavaScript, Node, Next, MySQL.

LISTA DE FIGURAS

Figura 1 - Exemplo do carousel....11
Figura 2 - Configurações da biblioteca do carousel....12
Figura 3 - Código do exemplo do SweetAlert2....12
Figura 4 - Exemplo do uso do SweetAlert2....13
Figura 5 - Código do exemplo da FullCalendar....14
Figura 6 - Exemplo do uso da FullCalendar....15
Figura 7 - Código exemplo da div....17
Figura 8 - Exemplo do uso da div....18
Figura 9 - Código exemplo do span....18
Figura 10 - Exemplo do uso do span....19
Figura 11 - Código exemplo do label e input....19
Figura 12 - Exemplo do uso de label e input....20
Figura 13 - Código exemplo do nav....20
Figura 14 - Exemplo do uso do nav....21
Figura 15 - Código do uso do botton com carousel....21
Figura 16 - Exemplo do uso do button com carousel....22
Figura 17 - Código responsivo para web....23
Figura 18 - Exemplo da responsividade web....23
Figura 19 - Código responsividade para tablet....24
Figura 20 - Exemplo da responsividade para tablet....25
Figura 21 - Código responsividade para mobile....26
Figura 22 - Exemplo da responsividade mobile....27
Figura 23 - Banco de dados (MER)....28
Figura 24 - Tabela dos usuários....28
Figura 25 - Tabela dos veículos....29
Figura 26 - Tabela das marcas....29
Figura 27 - Tabela das categorias....29
Figura 28 - Tabela onde é vinculado o veículo ao usuário....30
Figura 29 - Tabela de serviços....30
Figura 30 - Tabela de categoria dos serviços....31
Figura 31 - Tabelas de disponibilidade e indisponibilidade....31
Figura 32 - Tabelas de indisponibilidade....31
Figura 33 - Tabela da situação....31
Figura 34 - Tabela dos agendamentos....32
Figura 35 - Pasta usuários da API....32
Figura 36 - Pasta da API dos veículos....33
Figura 37 - Pasta da API dos veículos do usuário....33
Figura 38 - Pasta da API dos serviços....34
Figura 39 - Pasta da API da agenda....34
Figura 40 - Pasta da API do status....35
Figura 41 - Pasta da API das marcas....35
Figura 42 - Pasta da API dos modelos....36

Figura 43 - Pasta da API das categorias de veículos....36
Figura 44 - Pasta da API das categorias de serviços....37
Figura 45 - Tela de cadastro 38
Figura 46 - Tela de login 39
Figura 47 - Tela do administrador 39
Figura 48 - Tela de clientes do administrador 40
Figura 49 - Tela de veículos do adminstrador 40
Figura 50 - Tela de serviços do administrador 41
Figura 51 - Tela de agendamentos do administrador....41
Figura 52 - Tela de históricos do administrador 42
Figura 53 - Tela do cliente....42
Figura 54 - Tela dos dados do cliente 43
Figura 55 - Tela de veículos do cliente....43
Figura 56 - Tela de histórico do cliente 44
Figura 57 - Tela de agendamento do cliente....44

SUMÁRIO

1. INTRODUÇÃO 5
2. CRONOGRAMA 6
3. DADOS DO PROJETO 7
3.1. TIPO DE OBRA OU PROJETO 7
3.2. LOCALIZAÇÃO OU ÁREA DE ABRANGÊNCIA 7
3.3. ANÁLISE DO SISTEMA 7
3.4. ESTRUTURA DA SOLUÇÃO 7
3.5. DESENVOLVIMENTO DO EMPIRISMO 8
4. OBJETIVOS 9
4.1. OBJETIVO GERAL 9
4.2. OBJETIVOS ESPECÍFICOS 9
5. DESENVOLVIMENTO 10
5.1. EMBASAMENTO TEÓRICO 10
5.1.1. SOBRE O JAVASCRIPT 10
5.1.2. FERRAMENTAS E BIBLIOTECAS DO FRONT-END 10
5.1.2.1 NEXT.JS 10
5.1.2.2 NODE JS 11
5.1.2.3 REACT-RESPONSIVE-CAROUSEL 11
5.1.2.4 SWEETALERT2 12
5.1.2.5 FULLCALENDAR 13
5.1.3 LINGUAGENS DE MARCAÇÃO 15
5.1.3.1 HTML 15
5.1.3.2 CSS 16
5.1.4 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS 16
5.1.4.1 SOBRE O MYSQL 16
5.2. METODOLOGIA 17
5.2.1. PRINCIPAIS TAGS UTILIZADAS NO PROJETO 17
5.2.1.1 SOBRE A DIV 17
5.2.1.2 SOBRE O SPAN 18
5.2.1.3 SOBRE O INPUT E LABEL 19
5.2.1.4 SOBRE O NAV 20
5.2.1.6 SOBRE O BUTTON 21

5.2.2. RESPONSIVIDADE 22
5.2.2.1 EXEMPLO DE RESPONSIVIDADE WEB 22
5.2.2.2 EXEMPLO DE RESPONSIVIDADE TABLET 24
5.2.2.3 EXEMPLO DE RESPONSIVIDADE DO MOBILE 26
5.2.3. BANCO DE DADOS 28
5.2.4. ESTRUTURAÇÃO DA API 32
5.2.4.1 API PARA O USUÁRIO. 32
5.2.4.2 API PARA OS VEÍCULOS. 33
5.2.4.3 API PARA O VÍNCULO DO VEÍCULO COM O USUÁRIO. 33
5.2.4.4 API PARA OS SERVIÇOS. 34
5.2.4.5 API DA AGENDA. 34
5.2.4.5 API PARA DISPONIBILIDADE E INDISPONIBILIDADE. 35
5.2.4.6 API PARA MARCAS 35
5.2.4.7 API PARA MODELOS 36
5.2.4.8 API PARA CATEGORIAS DE VEÍCULOS 36
5.2.4.9 API PARA CATEGORIAS DE SERVIÇOS 37
5.3. UTILIZAÇÃO PRÁTICA 37
5.3.1. TELA DE CADASTRO DO USUÁRIO 38
5.3.2. TELA DE LOGIN 39
5.3.3. TELA DO PAINEL ADMINISTRATIVO 39
5.3.4. TELA DO PAINEL DO CLIENTE 42
6. CONSIDERAÇÕES FINAIS 45
6.1. RESULTADOS 45
6.2. DISCUSSÃO 45
6.3. IMPLEMENTAÇÕES FUTURAS 46
7. CONCLUSÃO 48
REFERÊNCIAS 49

1. INTRODUÇÃO

O uso de sistemas gerenciáveis tem se tornado cada vez mais frequente, especialmente entre pequenas empresas e profissionais autônomos. Esse avanço tecnológico tem proporcionado inúmeras vantagens para empreendedores, auxiliando na gestão e na organização de suas atividades diárias. Uma das principais dificuldades enfrentadas por empresas prestadoras de serviços é a gestão eficiente de agendamentos. À medida que a demanda por serviços personalizados cresce, torna-se essencial para essas empresas adaptarem-se às necessidades dos clientes, oferecendo soluções práticas e eficazes.

Nesse contexto, o desenvolvimento de um sistema gerenciável não é apenas uma tendência, mas uma necessidade imperativa. Esses sistemas oferecem um suporte fundamental para a organização interna das empresas, possibilitando um controle mais preciso dos agendamentos, gestão de recursos e comunicação com os clientes. Além disso, ao automatizar processos manuais, os sistemas gerenciáveis liberam tempo e recursos para que as empresas possam focar em outras áreas estratégicas do negócio.

Para os clientes, esses sistemas representam uma conveniência significativa. Eles podem realizar agendamentos de forma rápida e fácil, seja por meio de plataformas online, sem a necessidade de longas trocas de mensagens ou telefonemas. Isso não só melhora a experiência do cliente, como também fortalece o relacionamento entre a empresa e seus consumidores.

Assim, a adoção de sistemas gerenciáveis para a gestão de agendamentos e outras funções essenciais torna-se indispensável para empresas que desejam se destacar em um mercado competitivo, ao mesmo tempo em que aprimoram a eficiência operacional e a satisfação do cliente.

2. CRONOGRAMA

Quadro 1 - Atividades do 1º semestre de 2024

Tabela 1 - Cronograma

ATIVIDADE	FEV	MAR	ABR	MAI	JUN	JUL
Definição do Tema	x	x				
Problematização		x	x			
Identificação de fontes		x	x			
Análise do Sistema			x	x	x	x
Modelagem de dados					x	x
Criação do banco de dados					x	x

Fonte: Própria

Quadro 2 - Atividades 2º semestre 2024

Tabela 2 - Cronograma

ATIVIDADE	JUL	AGO	SET	OUT	NOV	DEZ
Desenvolvimento do Layout	x	x	x	x	x	
Programação do Software	x	x	x	x	x	
Documentação do Trabalho			x	x	x	
Preparo da Apresentação do Trabalho					x	
Apresentação do Trabalho						x
Teste e Validação de Erros			x	x	x	

Fonte: Própria

3. DADOS DO PROJETO

3.1. TIPO DE OBRA OU PROJETO

Projeto Técnico para o Desenvolvimento de um Programa Web voltado à gestão de serviços em empresas de estética automotiva.

3.2. LOCALIZAÇÃO OU ÁREA DE ABRANGÊNCIA

A solução será implementada em empresas de estética automotiva com alto volume de serviços diários, futuramente podendo ser adaptada para outros nichos de mercado.

3.3. ANÁLISE DO SISTEMA

O programa web visa solucionar problemas específicos enfrentados por empresas de estética automotiva:

Facilidade para o Cliente: Implementação de um sistema simples e eficiente para que clientes possam agendar serviços por conta própria e visualizar históricos de serviços realizados.

Gerenciamento Simplificado para Gerentes: Desenvolvimento de funcionalidades que permitam aos gerentes organizar e tratar os serviços de maneira eficiente e organizada, reduzindo a sobrecarga administrativa.

Controle de Volume e Tempo: Ferramentas que auxiliem no controle de serviços agendados e na gestão do tempo para otimizar os recursos da empresa.

3.4. ESTRUTURA DA SOLUÇÃO

O sistema será composto por:

Módulo de Agendamento: Interface amigável para clientes escolherem serviços e horários disponíveis.

Dashboard Gerencial: Painel para acompanhamento e organização dos serviços em andamento e registrados.

Interface de Histórico: Área dedicada para clientes e administradores acessarem os serviços já realizados, permitindo consultar detalhes e datas, promovendo transparência e fidelização.

Interface de Cadastro de Veículos: Uma seção específica para registrar os veículos de cada cliente, permitindo associar os serviços realizados a cada veículo. Essa funcionalidade organiza o histórico de manutenção, facilita o acesso a informações detalhadas e promove maior controle sobre a posse e os serviços atribuídos a cada veículo.

3.5. DESENVOLVIMENTO DO EMPIRISMO

O projeto será desenvolvido com base em uma abordagem empírica, utilizando protótipos e ciclos de validação contínuos para garantir que a solução atenda às necessidades tanto dos clientes quanto dos gestores.

4. OBJETIVOS

4.1. OBJETIVO GERAL

O projeto tem como principal objetivo facilitar os agendamentos de serviços de forma em que não seja necessário a utilização de ligações ou mensagens.

4.2. OBJETIVOS ESPECÍFICOS

- O sistema é prático e fácil de ser utilizado, podendo fazer agendamentos de qualquer lugar e de dispositivos diferentes por conta do uso via web.
- Redução de erros na hora do agendamento manual, evitando agendamentos errados por parte da empresa e dos clientes.
- O sistema traz uma agilidade nos agendamentos, não sendo necessário haver um responsável na empresa por agendar os serviços.
- Dentro do sistema é possível verificar o histórico de serviços executados em cada veículo.

5. DESENVOLVIMENTO

5.1. EMBASAMENTO TEÓRICO

A base do projeto foi desenvolvida com foco no JavaScript, uma das linguagens fundamentais da programação. Além disso, destaca-se pela ampla possibilidade de integração com bibliotecas, ferramentas e extensões que facilitam o processo de desenvolvimento.

5.1.1. SOBRE O JAVASCRIPT

JavaScript é uma linguagem de programação leve, interpretada e baseada em objetos, amplamente utilizada em páginas Web. É uma linguagem dinâmica, multi-paradigma e baseada em protótipos, suportando estilos de programação orientados a objetos, imperativos e declarativos, incluindo programação funcional (MOZILLA – JAVASCRIPT).

5.1.2. FERRAMENTAS E BIBLIOTECAS DO FRONT-END

No desenvolvimento do projeto, foram utilizadas ferramentas e bibliotecas voltadas à criação de interfaces front-end dinâmicas e eficientes, priorizando a experiência do usuário e o desempenho da aplicação.

5.1.2.1 NEXT.JS

O Next.js é uma plataforma que pode ser implementada junto ao React JS, juntos eles trazem uma experiência mais fluida para o usuário, deixando o site mais rápido, bonito e otimizado. Uma das suas principais características é permitir o uso e ter compatibilidade com HTML e CSS dando mais liberdade para o desenvolvedor (SILVA, 2024).

Além disso, o Next.js oferece funcionalidades avançadas como rotas automáticas, que simplificam a organização de páginas, e suporte nativo à API de dados para manipulação de conteúdo dinâmico. O framework também se destaca pela capacidade de realizar renderização híbrida, combinando estratégias de renderização

do lado do servidor (SSR) e geração de páginas estáticas (SSG), o que melhora a performance e contribui para a otimização SEO. Adicionalmente, o uso do Hot Module Replacement (HMR) facilita a atualização em tempo real durante o desenvolvimento, tornando o processo mais ágil e eficiente.

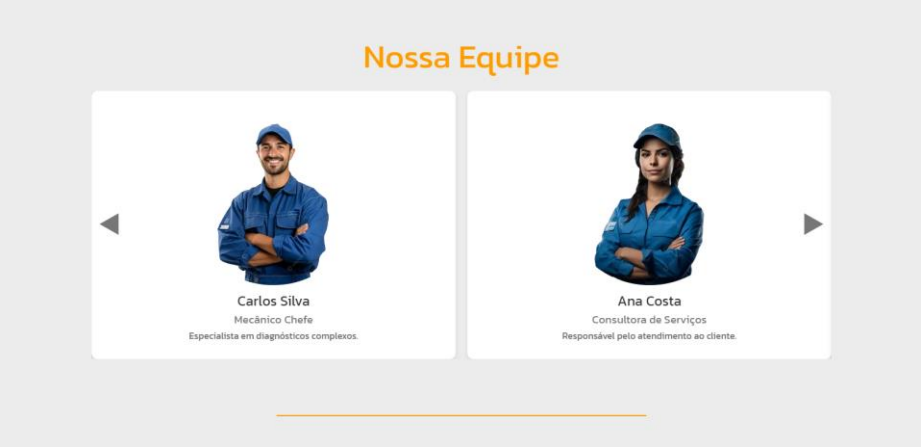
5.1.2.2 NODE JS

O Node.js é uma plataforma desenvolvida sobre a engine V8, a mesma utilizada pelo navegador Google Chrome para interpretar JavaScript de maneira rápida e eficiente. Amplamente utilizado no desenvolvimento back-end, o Node.js também pode ser integrado a projetos front-end, oferecendo um ecossistema rico em bibliotecas, ferramentas e APIs, acessíveis por meio do gerenciador de pacotes NPM (Node Package Manager).

5.1.2.3 REACT-RESPONSIVE-CAROUSEL

O componente <Carousel /> utilizado no projeto foi o react-responsive-carousel que é responsável pela funcionalidade do carrossel de imagens apresentado na tela inicial.

Figura 1 - Exemplo do carousel



Fonte: Própria

O componente disponibiliza diversas configurações para tornar o carrossel mais intuitivo e mais estilizável, como tempo de reprodução, tamanho da rolagem, pausas, repetições e outras funcionalidades.

Figura 2 - Configurações da biblioteca do carousel

```
<Carousel
    showArrows={true}
    showThumbs={false}
    infiniteLoop={true}
    autoPlay={true}
    interval={7000}
    showStatus={false}
    stopOnHover={true}
    className={styles.carousel}
    renderIndicator={false}
    renderArrowPrev={(clickHandler, hasPrev, labelPrev) =>
```

Fonte: Própria

5.1.2.4 SWEETALERT2

A biblioteca SweetAlert2 é utilizada para notificar o usuário por meio de modais interativos, exibindo mensagens de aviso, como login bem-sucedido, erro de credenciais ou outras notificações importantes.

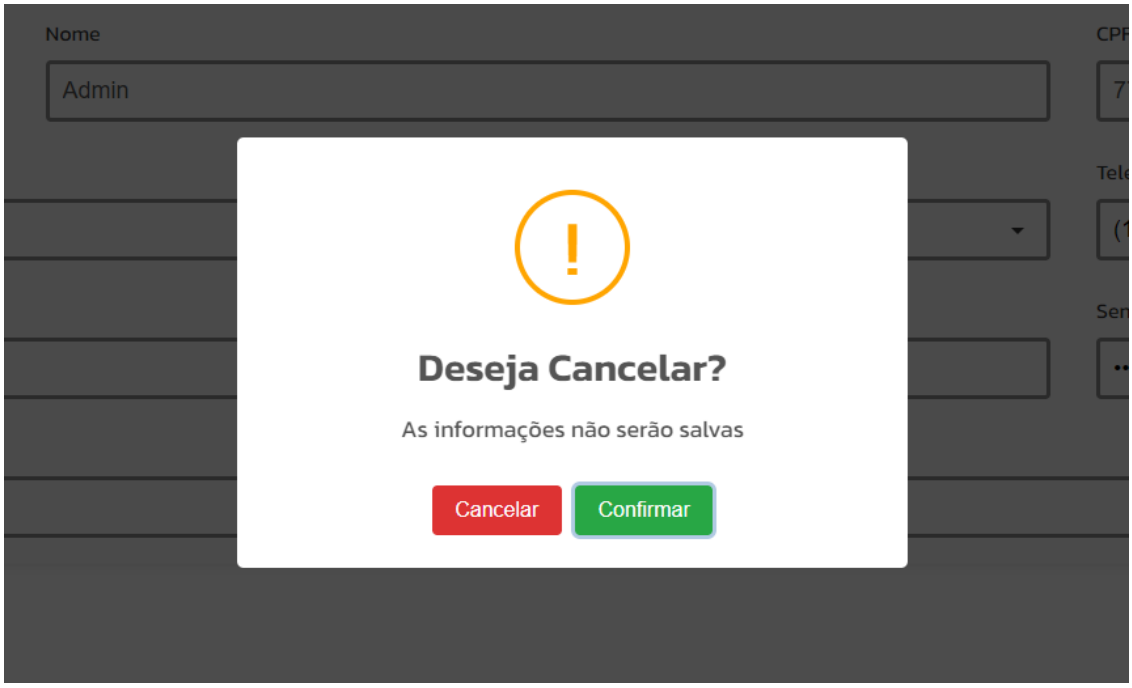
Na imagem a seguir, o SweetAlert2 é empregado na tela de login para informar ao usuário sobre e-mail ou senha incorretos, garantindo uma experiência mais clara e intuitiva.

Figura 3 - Código do exemplo do SweetAlert2

```
Swal.fire({
    title: "Deseja Cancelar?",
    text: "As informações não serão salvas",
    icon: "warning",
    iconColor: "orange",
    showCancelButton: true,
    cancelButtonColor: "#d33",
    confirmButtonColor: "rgb(40, 167, 69)",
    cancelButtonText: "Cancelar",
    confirmButtonText: "Confirmar",
    reverseButtons: true,
    backdrop: "rgba(0,0,0,0.7)",
```

Fonte: Própria

Figura 4 - Exemplo do uso do SweetAlert2



Fonte: Própria

5.1.2.5 FULLCALENDAR

A FullCalendar é uma biblioteca de JavaScript usada para criar calendários interativos e personalizáveis em aplicações web. Compatível com frameworks como React, Angular e Vue, ela oferece funcionalidades como visualizações mensais, semanais e diárias, além de suporte para arrastar, redimensionar e manipular eventos em tempo real. Sua interface pode ser adaptada com temas personalizados, garantindo flexibilidade para diferentes projetos.

A biblioteca também suporta idiomas, fusos horários e eventos recorrentes, sendo ideal para aplicações globais. Plugins adicionais permitem funcionalidades como grids de horários e listas de eventos. Com documentação detalhada e exemplos práticos, o FullCalendar é uma solução eficiente para sistemas de agendamento e gerenciamento de eventos.

Figura 5 - Código do exemplo da FullCalendar

```
useEffect(() => {
    if (currentMonth && currentYear) {
        const initialDate = `${currentYear}-${String(currentMonth).padStart(2, '0')}-01`;

        const calendar = new Calendar(calendarRef.current, {
            contentHeight: 600,
            selectable: true,
            locale: ptLocale,
            aspectRatio: 2,
            showNonCurrentDates: false,
            timeZone: 'local',
            eventOverlap: false,
            selectOverlap: false,
            expandRows: true,
            plugins: [dayGridPlugin, interactionPlugin, timeGridPlugin],
            initialView: 'dayGridMonth',
            headerToolbar: {
                left: 'dayGridMonth',
                center: 'title',
                right: 'today prev,next'
            },
            initialDate: initialDate,
            events: eventos,
            datesSet: handleDatesSet,
            dateClick: function (info) {
                const clickedDate = info.dateStr;
                if (info.view.type === "dayGridMonth") {
                    if (calendar) {
                        calendar.changeView('timeGridDay', clickedDate);
                    }
                } else {
                    handleDateClick(info);
                }
            },
            eventClick: handleEventClick,
            slotMinTime: '08:00:00',
            slotMaxTime: '18:00:00',
            eventTimeFormat: {
                hour: '2-digit',
                minute: '2-digit',
                meridiem: false
            },
            allDaySlot: false,
        });

        calendar.render();
        setCalendarApi(calendar);
    } else {
        console.error("currentMonth ou currentYear não estão definidos corretamente");
    }
}, [eventos, currentMonth, currentYear]);
```

Fonte: Própria

Figura 6 - Exemplo do uso da FullCalendar

Fonte: Própria

5.1.3 LINGUAGENS DE MARCAÇÃO

No desenvolvimento front-end, além do uso do JavaScript, suas bibliotecas e frameworks, também foram utilizados HTML e CSS. Essas linguagens de marcação complementaram o processo, permitindo a criação de interfaces mais completas e personalizadas. O Next.js, por sua flexibilidade, proporcionou a integração eficiente dessas tecnologias.

5.1.3.1 HTML

O HTML é a linguagem de marcação usada para estruturar e formatar a maioria das páginas da internet e aplicativos online. Diferente de uma linguagem de programação, o HTML não cria funcionalidades dinâmicas, mas permite aos desenvolvedores organizar e exibir informações como textos, hiperlinks e arquivos de mídia em navegadores web.

Os principais usos do HTML incluem o desenvolvimento web, onde ele define a apresentação de elementos em uma página, e a navegação na internet, facilitando a interconexão de páginas por meio de hiperlinks. Além disso, o HTML é utilizado na documentação, permitindo a organização e formatação de documentos de maneira semelhante a softwares como o Microsoft Word (ANDREI, 2023).

5.1.3.2 CSS

O CSS (Cascading Style Sheets) é crucial para a criação de páginas web modernas, pois transforma a estrutura básica fornecida pelo HTML em interfaces visualmente atraentes e funcionais. Enquanto o HTML define a estrutura e o conteúdo, o CSS lida com a apresentação visual, permitindo a personalização de cores, fontes, animações e layout. Essa separação entre conteúdo e estilo facilita a manutenção e atualização do código, além de melhorar a acessibilidade.

Com o crescente uso de diversos dispositivos para acessar a web, o design responsivo tornou-se essencial. O CSS é fundamental para criar layouts que se adaptam a diferentes tamanhos de tela, utilizando ferramentas como flexbox e frameworks como Bootstrap. Além disso, boas práticas na organização do código CSS ajudam a evitar duplicação e a manter o código limpo, facilitando a manutenção e a escalabilidade do projeto (PEREIRA, 2023).

5.1.4 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

Entre os diversos sistemas de gerenciamento de bancos de dados disponíveis no mercado, o MySQL foi o escolhido para este projeto. Nele, foram criadas as tabelas necessárias para armazenar e gerenciar as informações utilizadas no site e no aplicativo, garantindo a integridade e a eficiência do acesso aos dados.

5.1.4.1 SOBRE O MYSQL

O MySQL é um Sistema de Gerenciamento de Banco de Dados Relacional (SGBD) de código aberto, que utiliza a linguagem SQL para acessar, adicionar, modificar e deletar dados (CRUD).

Reconhecido por sua alta performance, confiabilidade e facilidade de uso, o MySQL é amplamente utilizado para gerenciar grandes volumes de dados de forma eficiente. Ele é otimizado para operações rápidas de leitura e escrita, suportando grandes bases de dados com agilidade. Sua confiabilidade é garantida por recursos robustos de transação. Além disso, o MySQL é fácil de instalar, configurar e manter, também é compatível com as principais plataformas e suportando diversas linguagens

de programação, o MySQL oferece flexibilidade e pode ser facilmente integrado a uma ampla variedade de aplicações e ambientes (MAGALHÃES; PORTUGAL, 2024).

5.2. METODOLOGIA

5.2.1. PRINCIPAIS TAGS UTILIZADAS NO PROJETO

No projeto, foi utilizado diversas tags para o desenvolvimento do front-end na parte de estilizações e em melhoras na funcionalidade do sistema web:

5.2.1.1 SOBRE A DIV

A tag é amplamente utilizada no desenvolvimento web por ser um elemento de bloco que permite organizar e separar conteúdos de forma estruturada.

Figura 7 - Código exemplo da div

```
<main className={styles.main}>
    <div className={styles.container}>
        <div className={styles.title}>Fale conosco</div>

        <div className={styles.info}>
            <div className={styles.card}>
                <div className={styles.infoDuvidas}>
                    <span className={styles.infoDuvidasTitulo}>Dúvidas</span>
                    <div className={styles.infos}>
                        <span className={styles.informacao}>Telefone: (XX)XXXX-XXXX</span>
                        <span className={styles.informacao}>WhatsApp: (XX)XXXX-XXXX</span>
                        <span className={styles.informacao}>Email: urban_ouvidoria@teste.com</span>
                    </div>
                </div>
                <div className={styles.infoContatos}>
                    <span className={styles.infoContatosTitulo}>Contatos</span>
                    <div className={styles.infos}>
                        <span className={styles.informacao}>Telefone: (XX)XXXX-XXXX</span>
                        <span className={styles.informacao}>WhatsApp: (XX)XXXX-XXXX</span>
                        <span className={styles.informacao}>Email: urban_atendimento@teste.com</span>
                    </div>
                </div>
                <div className={styles.infoParcerias}>
                    <span className={styles.infoParceriasTitulo}>Parcerias</span>
                    <div className={styles.infos}>
                        <span className={styles.informacao}>Email: urban_rh@teste.com</span>
                        <span className={styles.informacao}>Email: urban_rh@teste.com</span>
                        <span className={styles.informacao}>Obrigado pela preferência!</span>
                    </div>
                </div>
            </div>
        </div>
```

Fonte: Própria

Figura 8 - Exemplo do uso da div

Fonte: Própria

5.2.1.2 SOBRE O SPAN

A tag foi utilizada principalmente no footer para aplicar conteúdos de forma inline. Diferentemente da tag , que gera uma quebra de linha, permite a formatação e estilização sem alterar o fluxo do texto.

Figura 9 - Código exemplo do span

```
<div className={styles.footerIcons}>
    <span className={styles.footerLogo}></span>
    <div className={styles.footerSocialIcons}>
        <Link href={'https://instagram.com'}>
            <span className={styles.iconInstagram}></span>
        </Link>
        <Link href={'https://whatsapp.com'}>
            <span className={styles.iconWhatsapp}></span>
        </Link>
        <Link href={'https://facebook.com'}>
            <span className={styles.iconFacebook}></span>
        </Link>
        <Link href={'https://youtube.com'}>
            <span className={styles.iconYoutube}></span>
        </Link>
    </div>
</div>
```

Fonte: Própria

Figura 10 - Exemplo do uso do span

Fonte: Própria

5.2.1.3 SOBRE O INPUT E LABEL

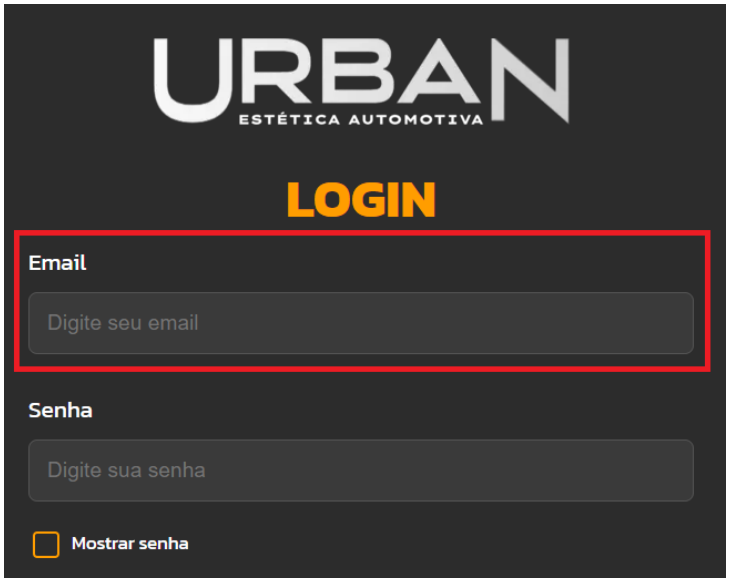
Duas tags utilizadas para campos de preenchimento foram o <label> junto com <input>, o <label> tem a função de criar uma legenda e o input é o campo de preenchimento, com o uso dos dois é criado um rótulo para um campo que receberá inserção de dados.

Figura 11 - Código exemplo do label e input

```
<div className={styles.inputGroup}>
    <label htmlFor="email" className={styles.labelLogin}>Email</label>
    <input
        type="email"
        id="email"
        name="email"
        className={styles.inputLogin}
        placeholder="Digite seu email"
        onChange={e => setEmail(e.target.value)}
        value={email}
        required
    />
</div>

<div className={styles.inputGroup}>
    <label htmlFor="password" className={styles.labelLogin}>Senha</label>
    <input
        type={showPassword ? "text" : "password"}
        id="password"
        name="password"
        className={styles.inputLogin}
        placeholder="Digite sua senha"
        onChange={e => setSenha(e.target.value)}
        value={senha}
        required
    />
</div>
```

Fonte: Própria

Figura 12 - Exemplo do uso de label e input



Fonte: Própria

5.2.1.4 SOBRE O NAV

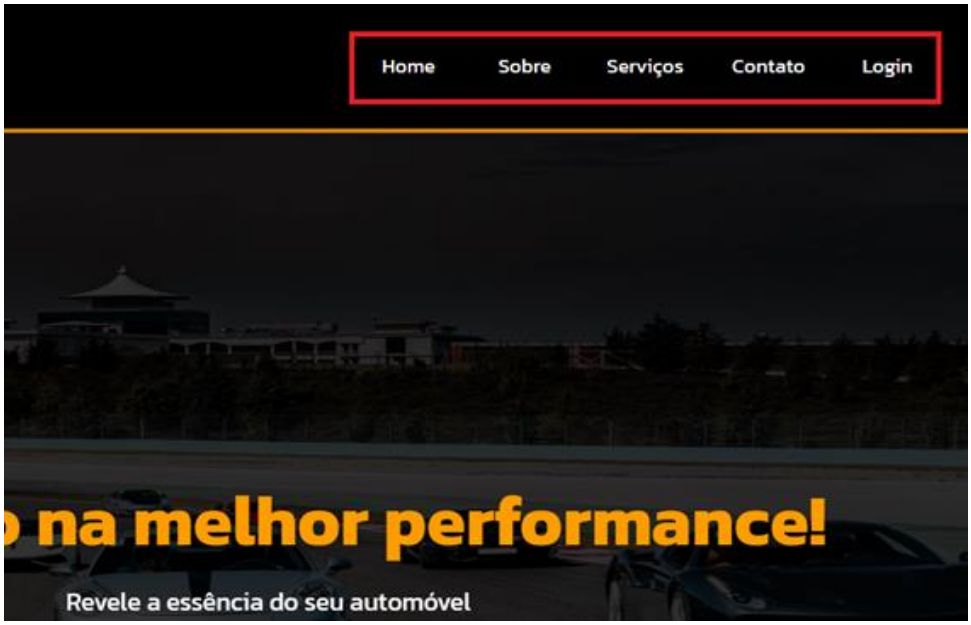
Na seção do header, foi utilizada a tag <nav>, responsável por criar um contêiner para os links de navegação, facilitando a organização e estruturação do menu de navegação na página.

Figura 13 - Código exemplo do nav

```
<nav className={styles.navbar}>
    <ul className={styles.navlist}>
        <li><Link href="/" className={styles.linkNav}>Home</Link></li>
        <li><Link href="/telas/sobre" className={styles.linkNav}>Sobre</Link></li>
        <li><Link href="/telas/servicos" className={styles.linkNav}>Serviços</Link></li>
        <li><Link href="/telas/contatos" className={styles.linkNav}>Contato</Link></li>
        <li><Link href="/telas/login" className={styles.linkNav}>Login</Link></li>
    </ul>
</nav>
```

Fonte: Própria

Figura 14 - Exemplo do uso do nav



Fonte: Própria

5.2.1.6 SOBRE O BUTTON

Junto ao carrossel é possível a utilização de <button> que torna o componente ainda mais intuitivo, fazendo com que você possa passar e voltar as imagens.

Figura 15 - Código do uso do botton com carousel

```
            <button
                type="button"
                onClick={clickHandler}
                className={styles.customPrevArrow}
                aria-label={labelPrev}
            >
                &#9664; {/* Setinha para esquerda */}
            </button>
        )
    }
    renderArrowNext={(clickHandler, hasNext, labelNext) =>
        hasNext && (
            <button
                type="button"
                onClick={clickHandler}
                className={styles.customNextArrow}
                aria-label={labelNext}
            >
                &#9654; {/* Setinha para direita */}
            </button>
```

Fonte: Própria

Figura 16 - Exemplo do uso do button com carousel

Fonte: Própria

5.2.2. RESPONSIVIDADE

O projeto foi desenvolvido pensado tanto nos usuários web quanto nos usuários mobile, dessa forma se adaptando de acordo com a tela utilizada. A forma encontrada para a funcionalidade da responsividades foi criando mais de uma estilização para cada item, dessa forma as estilizações são alteradas conforme o tamanho máximo ou mínimo definido.

5.2.2.1 EXEMPLO DE RESPONSIVIDADE WEB

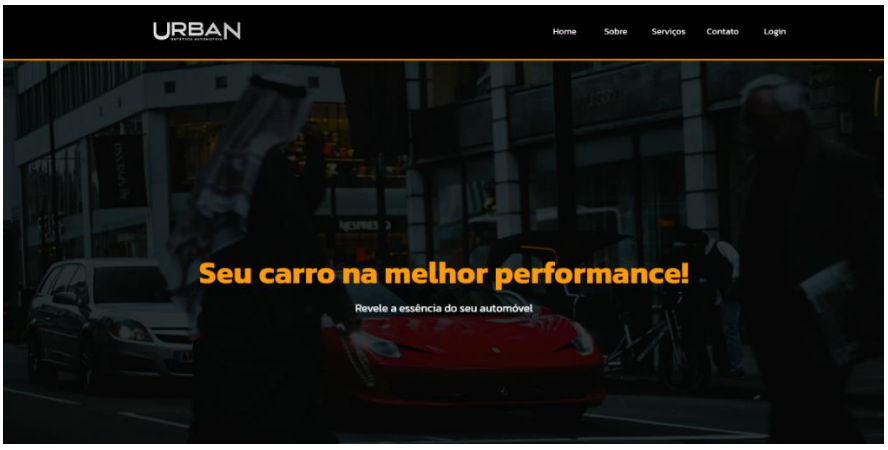
A responsividade no caso da web é feito para ser limitada em 1024px quando for acessado através de um computador, com essa resolução é possível atender a compatibilidade com a maioria dos monitores.

Figura 17 - Código responsivo para web

```css
@media (max-width: 1024px) {
    .main {
        padding: 0 5%;
    }

    .section2 {
        display: flex;
        flex-direction: column;
        height: 80vh;
    }

    .section2Image {
        width: 100%;
        height: 50%;
    }

    .section2Content {
        text-align: justify;
        width: 100%;
        height: auto;
        min-height: 50%;
        padding-left: 0;
    }

    .section2Title {
        font-size: 3rem;
    }

    .section2Description {
        font-size: 1.7rem;
    }

    .employeeContainer {
        grid-template-columns: repeat(2, 1fr);
    }
}
```

Fonte: Própria

Figura 18 - Exemplo da responsividade web



Fonte: Própria

5.2.2.2 EXEMPLO DE RESPONSIVIDADE TABLET

No caso da responsividade de um tablet o tamanho máximo é ainda menor, sendo 768px para poder atender diversos modelos de tablets e ipads.

Figura 19 - Código responsividade para tablet

```
@media (max-width: 768px) {

    .container_questions {
        width: 90%;
        padding: 0;
    }

    .section2 {
        height: 100vh;
    }

    .section2Button {
        width: 100%;
    }

    .employeeContainer {
        grid-template-columns: 1fr;
    }

    .divider {
        width: 70%;
    }

    .title_questions {
        font-size: 2.5rem;
    }

    .carrouselEquipe{
        display: none;

    }

    .carrouselEquipeMobile{
        display: flex;
        width: 100%;
    }

}
```

Fonte: Própria

Figura 20 - Exemplo da responsividade para tablet

Fonte: Própria

5.2.2.3 EXEMPLO DE RESPONSIVIDADE DO MOBILE

No caso mobile a resolução é bem menor que os demais, pois trabalhamos com telas menores e mais limitadas, sendo assim utilizamos uma proporção de 480px.

Figura 21 - Código responsividade para mobile

```css
@media (max-width: 480px) {

    .section2 {
        height: 110vh;
    }

    .container_questions {
        width: 100%;
    }

    .section2Title {
        font-size: 2.7rem;
    }

    .section2Description {
        font-size: 1.5rem;
    }

    .employeeCard {
        height: auto;
    }

    .divider {
        width: 90%;
    }

    .title_questions {
        font-size: 2rem;
    }

    .summary_content {
        font-size: 1.2rem;
        text-wrap: wrap;
    }
}
```

Fonte: Própria

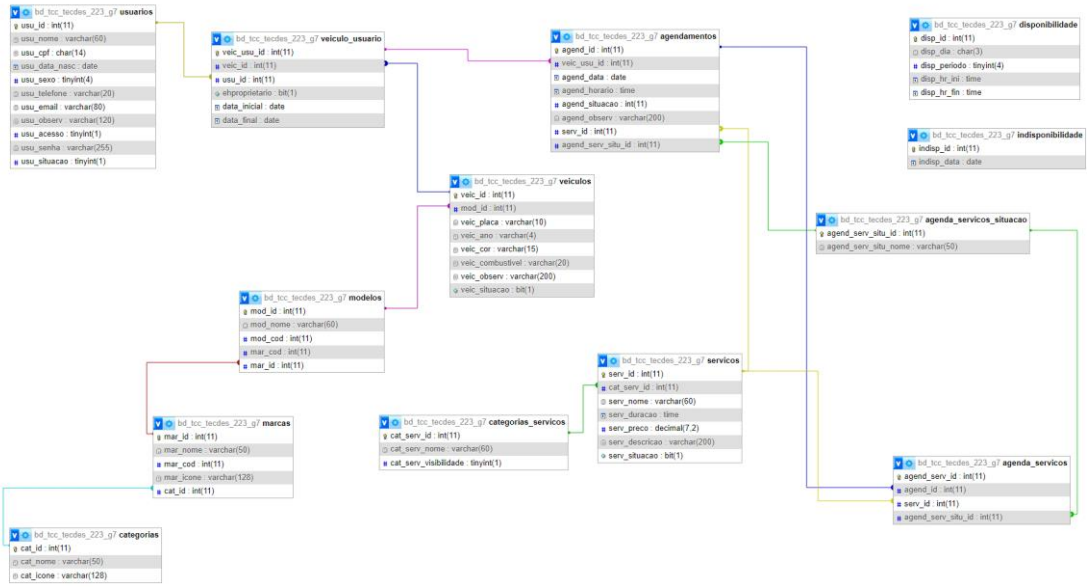
Figura 22 - Exemplo da responsividade mobile

Fonte: Própria

5.2.3. BANCO DE DADOS

No projeto foi utilizado o MySQL, onde foi desenvolvido diversas tabelas para o banco de dados, as tabelas são vinculadas conforme necessário para se complementarem.

Figura 23 - Banco de dados (MER)



Fonte: Própria

Figura 24 - Tabela dos usuários

Table: usuarios

Columns:

usu_id	int(11) AI PK
usu_nome	varchar(60)
usu_cpf	char(14)
usu_data_nasc	date
usu_sexo	tinyint(4)
usu_telefone	varchar(20)
usu_email	varchar(80)
usu_observ	varchar(255)
usu_acesso	tinyint(1)
usu_senha	varchar(256)
usu_situacao	bit(1)

Fonte: Própria

Figura 25 - Tabela dos veículos

Table: veiculos

Columns:

veic_id	int(11) AI PK
mod_id	int(11)
veic_placa	varchar(10)
veic_ano	varchar(4)
veic_cor	varchar(15)
veic_combustivel	varchar(20)
veic_observ	varchar(200)
veic_situacao	bit(1)

Fonte: Própria

Figura 26 - Tabela das marcas

Table: marcas

Columns:

mar_id	int(11) AI PK
mar_nome	varchar(50)
mar_cod	int(11)
mar_icone	varchar(128)
cat_id	int(11)

Fonte: Própria

Figura 27 - Tabela das categorias

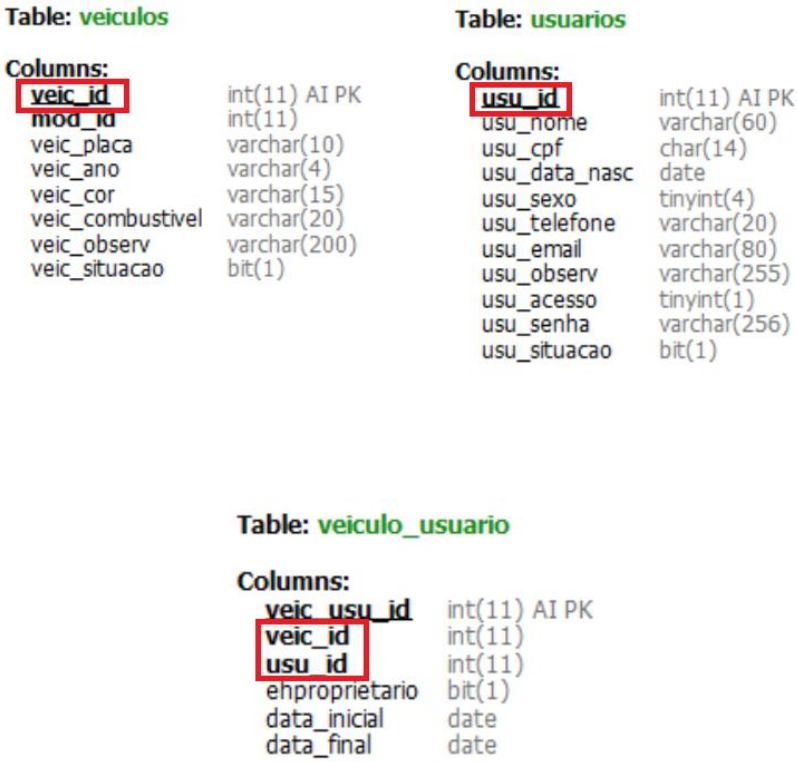
Table: categorias

Columns:

cat_id	int(11) AI PK
cat_nome	varchar(50)
cat_icone	varchar(128)

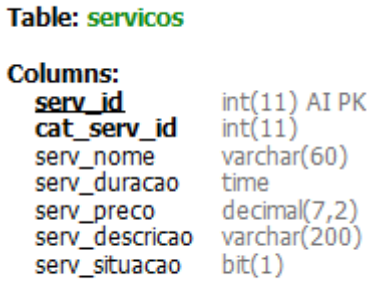
Fonte: Própria

Figura 28 - Tabela onde é vinculado o veículo ao usuário



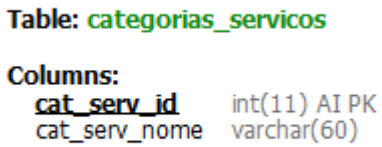
Fonte: Própria

Figura 29 - Tabela de serviços



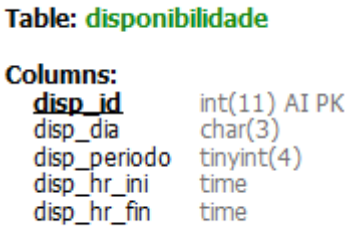
Fonte: Própria

Figura 30 - Tabela de categoria dos serviços



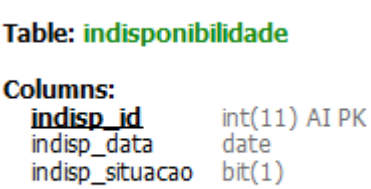
Fonte: Própria

Figura 31 - Tabelas de disponibilidade e indisponibilidade



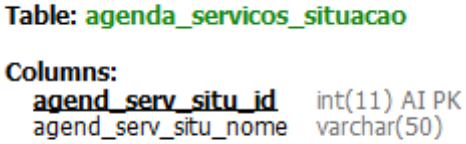
Fonte: Própria

Figura 32 - Tabelas de indisponibilidade



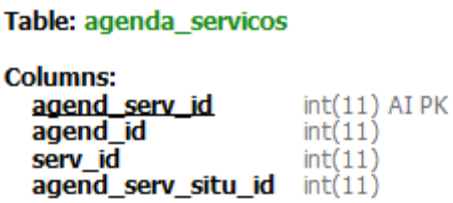
Fonte: Própria

Figura 33 - Tabela da situação



Fonte: Própria

Figura 34 - Tabela dos agendamentos



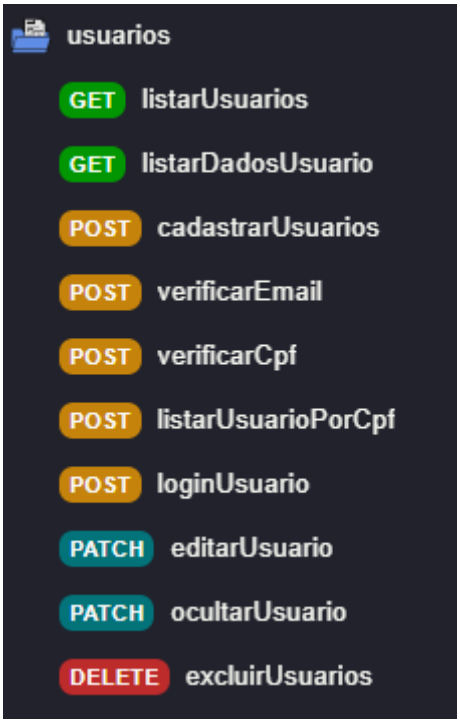
Fonte: Própria

5.2.4. ESTRUTURAÇÃO DA API

Na parte de API foi desenvolvido funções para a comunicação do site com o banco trazendo funções como post, get, delete e patch, dessa forma será possível acessar o banco podendo fazer consulta, cadastro e alterações.

5.2.4.1 API PARA O USUÁRIO.

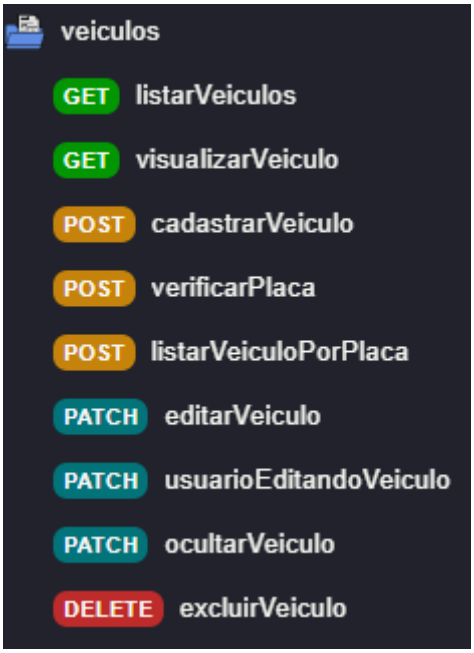
Figura 35 - Pasta usuários da API



Fonte: Própria

5.2.4.2 API PARA OS VEÍCULOS.

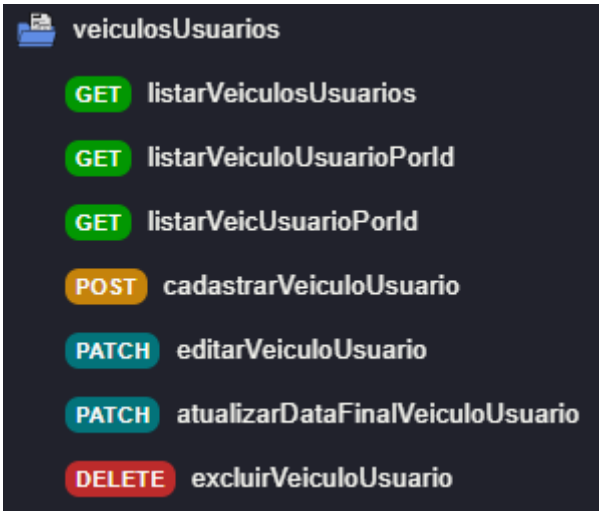
Figura 36 - Pasta da API dos veículos



Fonte: Própria

5.2.4.3 API PARA O VÍNCULO DO VEÍCULO COM O USUÁRIO.

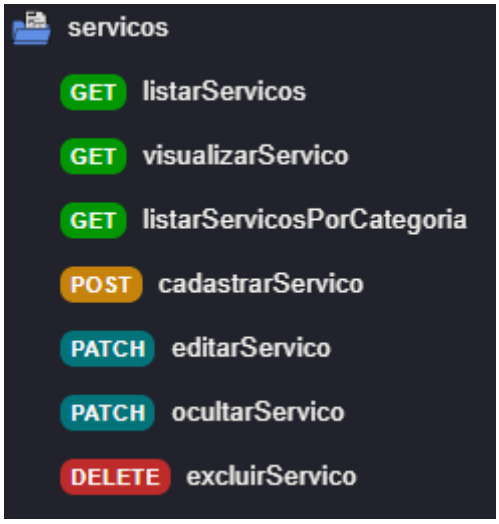
Figura 37 - Pasta da API dos veículos do usuário



Fonte: Própria

5.2.4.4 API PARA OS SERVIÇOS.

Figura 38 - Pasta da API dos serviços



Fonte: Própria

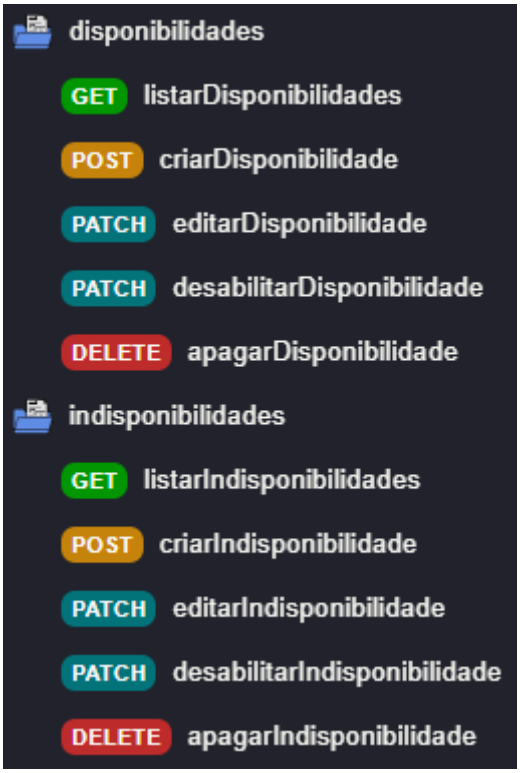
5.2.4.5 API DA AGENDA.

Figura 39 - Pasta da API da agenda

Fonte: Própria

5.2.4.5 API PARA DISPONIBILIDADE E INDISPONIBILIDADE.

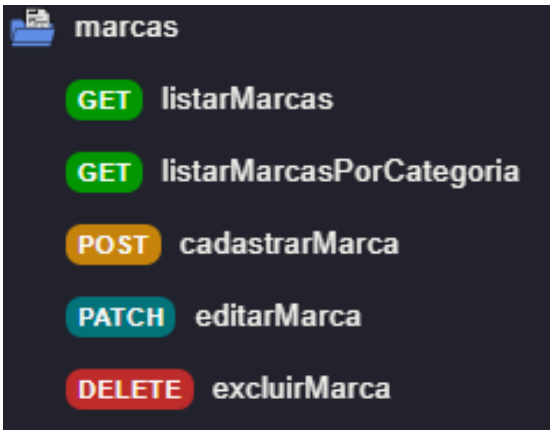
Figura 40 - Pasta da API do status



Fonte: Própria

5.2.4.6 API PARA MARCAS

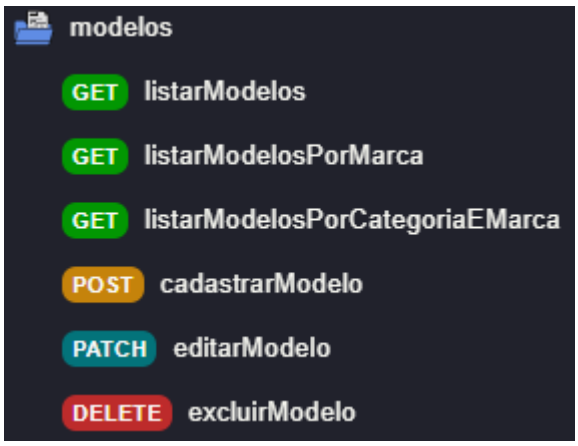
Figura 41 - Pasta da API das marcas



Fonte: Própria

5.2.4.7 API PARA MODELOS

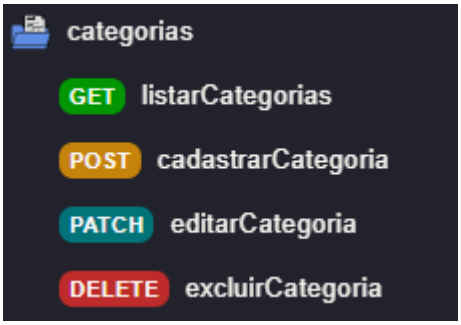
Figura 42 - Pasta da API dos modelos



Fonte: Própria

5.2.4.8 API PARA CATEGORIAS DE VEÍCULOS

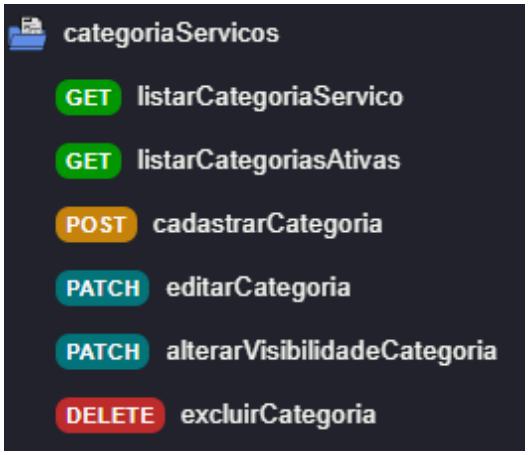
Figura 43 - Pasta da API das categorias de veículos



Fonte: Própria

5.2.4.9 API PARA CATEGORIAS DE SERVIÇOS

Figura 44 - Pasta da API das categorias de serviços



Fonte: Própria

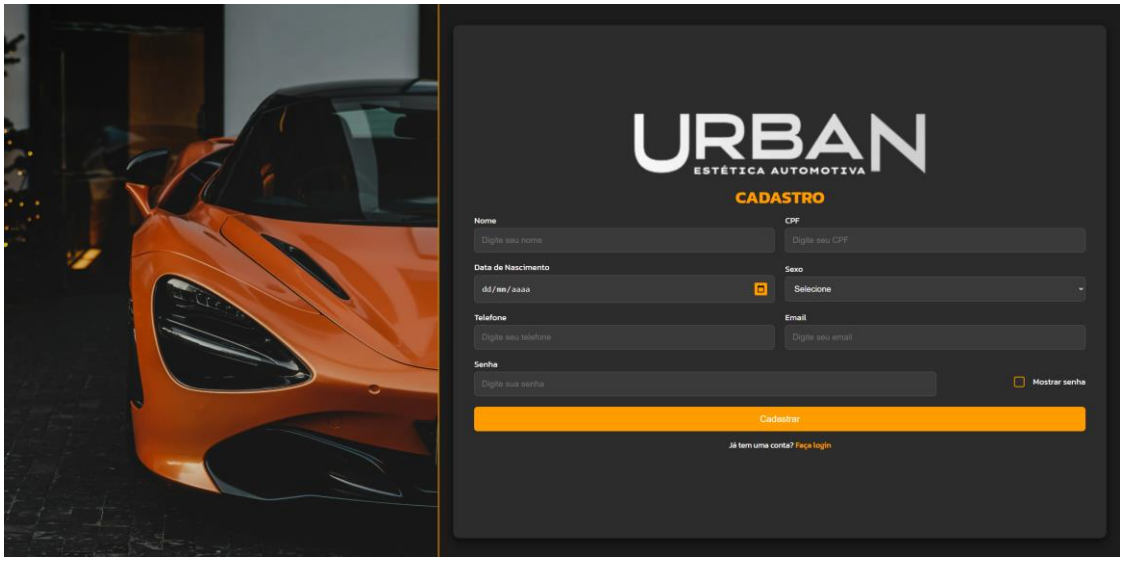
5.3. UTILIZAÇÃO PRÁTICA

No momento da utilização fazemos a requisição no front-end que comunica com a API que busca as informações diretamente no banco de dados.

5.3.1. TELA DE CADASTRO DO USUÁRIO

A tela de cadastro pode ser acessada tanto por meio de um atalho na tela de login inicial quanto diretamente no painel do administrador. Durante o processo de cadastro, o usuário deve fornecer informações básicas. Após o preenchimento, serão verificadas algumas informações, como o CPF, que será validado quanto à sua autenticidade e se já não está em uso, e o e-mail, que será verificado quanto ao formato correto e à disponibilidade. Após a validação, o cadastro será concluído e os dados serão enviados para a API, que realizará um POST no banco de dados, armazenando as informações e gerando uma ID exclusiva para o cliente.

Figura 45 - Tela de cadastro

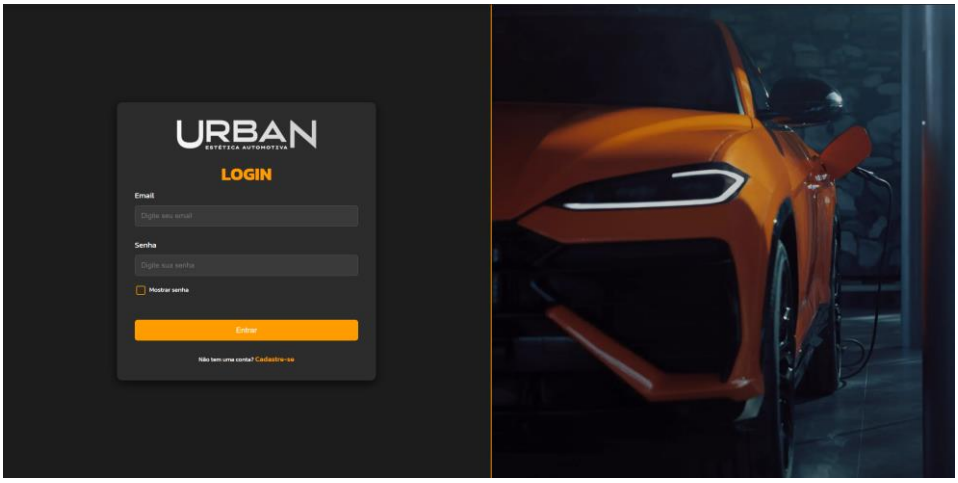


Fonte: Própria

5.3.2. TELA DE LOGIN

Na tela de login os dados preenchidos no e-mail e senha serão verificados pela API no banco fazendo a autenticação do usuário, e caso seja válido, o usuário será direcionado para o painel de acordo com o tipo de permissão que o usuário possui, podendo ser administrador ou cliente.

Figura 46 - Tela de login



Fonte: Própria

5.3.3. TELA DO PAINEL ADMINISTRATIVO

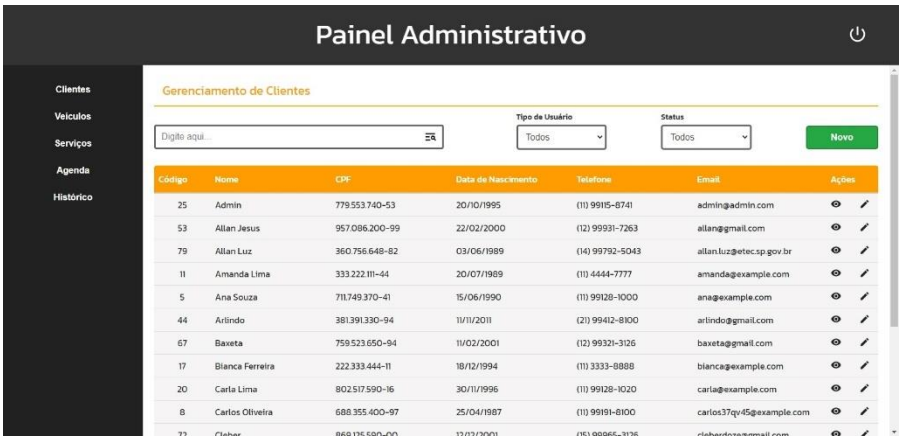
Dentro da tela inicial, os usuários com acesso administrativo poderão gerenciar usuários, veículos, serviços, agenda e histórico.

Figura 47 - Tela do administrador

Fonte: Própria

Na tela de clientes, é possível realizar a busca por todos os clientes, utilizando a barra de pesquisa por nome e aplicando filtros, como tipo de usuário (administrador ou usuário) e status (ativos ou inativos). Na tabela onde os usuários são listados, é possível alterar a ordem de exibição, organizando-os por código, CPF, data de nascimento, telefone, ou em ordem alfabética por nome, sexo e e-mail. Além disso, a tela oferece a funcionalidade de cadastrar um novo usuário e alterar os dados de qualquer outro usuário já registrado.

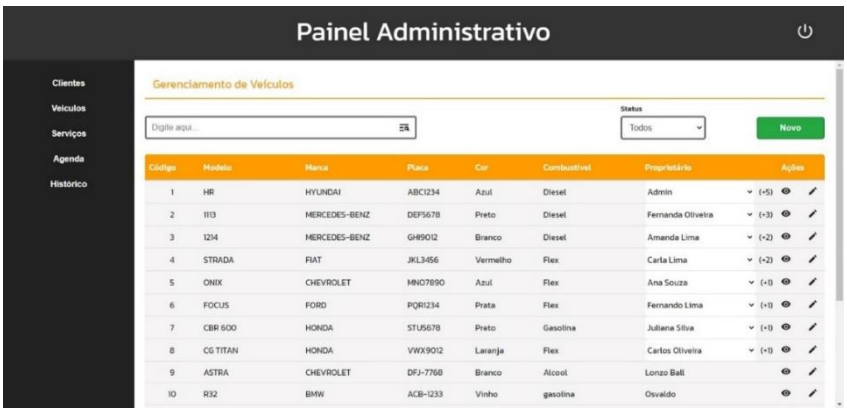
Figura 48 - Tela de clientes do administrador



Fonte: Própria

Na seção de veículos, o administrador pode visualizar, buscar e filtrar os veículos cadastrados por nome, status, código, modelo, marca, placa, cor, combustível e proprietário. Também é possível editar informações e cadastrar ou vincular veículos a clientes.

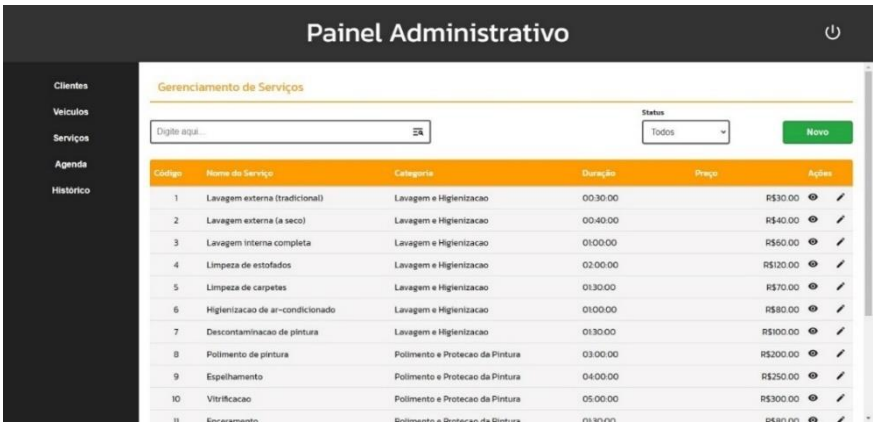
Figura 49 - Tela de veículos do adminstrador



Fonte: Própria

Dentro da seção de serviços, há uma aba onde são listados os serviços disponíveis para agendamento. É possível realizar buscas por nome ou status. Na tabela, os serviços podem ser ordenados por código, duração, preço ou em ordem alfabética, considerando o nome do serviço e sua categoria. O administrador tem a possibilidade de criar e alterar as informações dos serviços disponíveis.

Figura 50 - Tela de serviços do administrador



Fonte: Própria

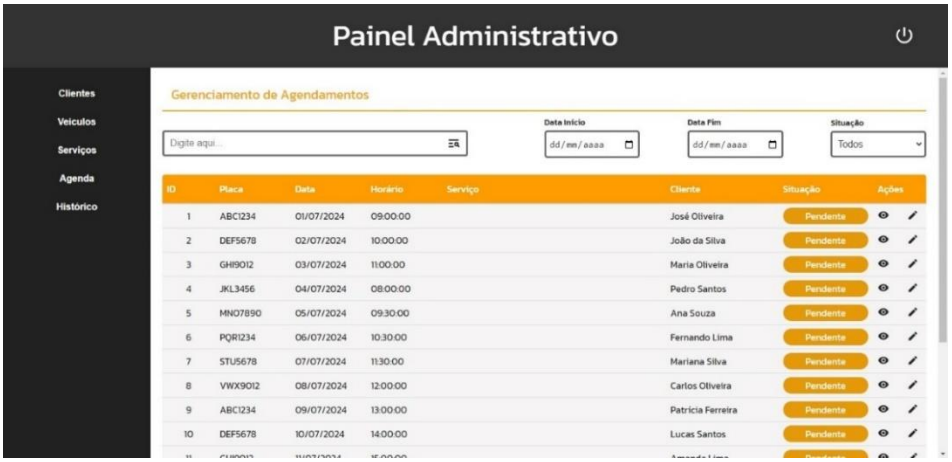
A tela da agenda exibe os agendamentos e a disponibilidade de dias e horários. Nela, é possível não apenas visualizar os agendamentos, mas também realizar o agendamento e o cancelamento de serviços. Para isso, foi utilizado o componente FullCalendar, um calendário interativo que pode ser customizado e adaptado conforme as necessidades específicas de cada tipo de utilização.

Figura 51 - Tela de agendamentos do administrador

Fonte: Própria

No histórico, é possível localizar serviços agendados, cancelados, em andamento e concluídos. A funcionalidade também inclui um mecanismo de busca por data de início e fim, situação e nome do cliente. Além disso, dentro do histórico, é possível realizar alterações nos agendamentos.

Figura 52 - Tela de históricos do administrador



Fonte: Própria

5.3.4. TELA DO PAINEL DO CLIENTE

Dentro do painel do cliente, é possível acessar as seguintes telas: "Meus Dados", "Meus Veículos", "Histórico" e "Agenda".

Figura 53 - Tela do cliente

Fonte: Própria

Na parte dos dados é possível ver os dados e altera-los se necessário.

Figura 54 - Tela dos dados do cliente

Fonte: Própria

A seção de veículos é onde estão listados os veículos cadastrados, permitindo ao usuário cadastrar um novo veículo, visualizar suas informações, editar ou excluir os veículos vinculados aos clientes.

Figura 55 - Tela de veículos do cliente

Fonte: Própria

O histórico permite visualizar todos os agendamentos realizados pelo cliente, incluindo o veículo agendado e o serviço executado, além de possibilitar a consulta aos detalhes de cada agendamento.

Figura 56 - Tela de histórico do cliente

Fonte: Própria

A seção da agenda permite que o cliente selecione o dia e o horário disponíveis no calendário. Em seguida, ele poderá escolher o veículo e o serviço desejados. Dentro da mesma área, também é possível cancelar o agendamento ou alterar o dia, horário, veículo e serviço.

Figura 57 - Tela de agendamento do cliente

Fonte: Própria

6. CONSIDERAÇÕES FINAIS

6.1. RESULTADOS

Com o desenvolvimento do sistema, todos os objetivos propostos foram plenamente alcançados, resultando em uma solução funcional e eficiente. O sistema conta com módulos de cadastro e login totalmente integrados ao banco de dados, garantindo a validação precisa das informações dos usuários.

No painel administrativo, é possível acessar e gerenciar todas as informações dos usuários e veículos cadastrados, além de realizar e modificar agendamentos de forma prática. Já na interface do cliente, o sistema permite a edição de dados pessoais e informações de seus veículos, assim como a realização de agendamentos.

A funcionalidade de agendamento destaca-se pela sua eficiência, oferecendo uma visão clara de dias e horários disponíveis e permitindo que os agendamentos sejam realizados em tempo real, de forma prática e intuitiva. Esses resultados refletem o cumprimento dos objetivos iniciais, entregando uma plataforma que atende tanto às necessidades dos clientes quanto à administração do serviço de forma moderna e funcional.

6.2. DISCUSSÃO

Os resultados obtidos com o desenvolvimento do projeto demonstram que o sistema de agendamento pode, de fato, trazer significativos benefícios tanto para os usuários quanto para os administradores. A agilidade e praticidade proporcionadas pela automação eliminam a necessidade de atendimento presencial ou por telefone, reduzindo erros e permitindo que agendamentos sejam realizados a qualquer momento e de qualquer lugar. Esses fatores tornam o sistema uma ferramenta eficiente e moderna, alinhada às necessidades dos usuários.

O principal benefício do sistema é a automatização completa do processo de agendamento. Isso eliminou a dependência de atendimentos manuais, minimizando erros e oferecendo uma experiência mais fluida para os clientes. Além disso, a possibilidade de acessar o sistema em qualquer dispositivo e em qualquer horário, tornando altamente acessível e conveniente, atendendo às demandas de usuários que valorizam a praticidade.

Durante o desenvolvimento, diversos desafios foram encontrados, mas todos foram superados com estratégias e ajustes adequados:

Foi necessário criar uma relação robusta entre as chaves primárias e garantir uma comunicação eficiente com a API. Isso foi resolvido por meio de revisões e aprimoramentos no modelo relacional e na integração.

A responsividade foi inicialmente um desafio, assim como a implementação de elementos visuais como o carrossel. Essas questões foram tratadas com a adoção de frameworks e bibliotecas específicas, além de ajustes manuais no CSS e no JavaScript.

Foram encontradas dificuldades na implementação de validações de dados para garantir segurança e precisão. O problema foi solucionado com o uso de bibliotecas de validação e testes rigorosos, aprimorando o fluxo de autenticação.

Durante o desenvolvimento da funcionalidade de agendamento, houve desafios na criação de modais para listar informações de usuários e exibir dias e horários já ocupados sem comprometer a privacidade de outros clientes. Isso foi resolvido utilizando filtragem e exibição dinâmica de dados, garantindo uma experiência de uso fluida e segura.

Essa etapa de resolução de problemas foi fundamental para o sucesso do projeto, garantindo que o sistema fosse finalizado de forma funcional, prática e alinhada às necessidades tanto do cliente quanto do administrador.

6.3. IMPLEMENTAÇÕES FUTURAS

O projeto foi finalizado dentro dos objetivos propostos; no entanto, alguns recursos foram selecionados para serem implementados futuramente:

- Recurso de alteração de tema, podendo alterar o modo de cor, deixando modo escuro ou claro, isso de forma automática de acordo com o tema atual do celular ou alterado manualmente.
- Desenvolvimento de um aplicativo exclusivamente mobile, para facilitar o uso do sistema para o usuário.
- Melhoras da interface e experiência do usuário, uma otimização para melhorar o uso do sistema.

- Otimização do processamento, melhorar a velocidade dos processos, evitando carregamentos ou travamentos.
- Uso de token para acesso.
- Servidor de e-mail para recuperação e autenticação de senha.
- Autenticação de 2 fatores.
- Notificação de alteração do status do serviço e notificações em geral.

7. CONCLUSÃO

O sistema foi desenvolvido e concluído com sucesso, incorporando todas as funcionalidades planejadas. Ele conta com uma página inicial informativa, que apresenta a empresa de forma profissional, e um módulo de cadastro e login completo, com validação eficiente de dados. Além disso, integra o gerenciamento de veículos e um sistema de agendamento robusto utilizando a biblioteca FullCalendar, que proporciona autonomia tanto para os clientes quanto para os administradores no agendamento e gerenciamento de compromissos. O painel administrativo permite cadastrar, editar e personalizar os serviços oferecidos, além de controle de usuários e veículos, trazendo flexibilidade e controle.

O projeto foi concebido com o objetivo principal de criar uma solução de agendamento que une rapidez, praticidade e intuitividade, atendendo às necessidades específicas da Urban Estética Automotiva. Toda a estrutura foi planejada de forma organizada, garantindo uma experiência fluida e eficiente para os usuários, tanto clientes quanto administradores. Com isso, elimina-se a necessidade de interações pessoais ou por telefone, promovendo uma abordagem mais moderna e eficiente no gerenciamento de agendamentos.

REFERÊNCIAS

ANDREI. **O que é HTML? Conceitos Básicos**. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-html-conceitos-basicos. Acesso em: 23 out. 2024.

BESSA, André. **Node.js**. Disponível em: https://www.alura.com.br/artigos/node-js. Acesso em: 23 out. 2024.

FULLCALENDAR. **FullCalendar**. Disponível em: https://fullcalendar.io/. Acesso em: 29 nov. 2024.

MAGALHÃES, Beatriz; PORTUGAL, Daniel. **MySQL no Mercado**. Disponível em: https://www.alura.com.br/artigos/mysql-no-mercado. Acesso em: 23 out. 2024.

MOZILLA. **JavaScript**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 23 out. 2024.

NEVES, Vinicios. **React.js**. Disponível em: https://www.alura.com.br/artigos/react-js. Acesso em: 21 out. 2024.

PEREIRA, Rafael. **Estilo e Estrutura: A Importância do CSS na Programação Web**. Disponível em: https://www.dio.me/articles/estilo-e-estrutura-a-importancia-do-css-na-programacao-web. Acesso em: 23 out. 2024.

SILVA, Patrícia. **Aplicações Fullstack**. Disponível em: https://www.alura.com.br/artigos/aplicacoes-fullstack. Acesso em: 21 out. 2024.

SOUTO, Mario. **Next.js: Iniciando com o Framework**. Disponível em: https://www.alura.com.br/conteudo/next-js-iniciando-framework. Acesso em: 21 out. 2024.