CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ETEC SYLVIO DE MATTOS CARVALHO

Curso de Técnico em Informática para Internet Integrado ao Ensino Médio

Eric Wilson de Souza Alves

Felipe Henrique Savoini

Gabrielly Beatriz Silva

Jackson Silva Oliveira

FINDBUS:
website para auxiliar os alunos da Etec Sylvio de Mattos Carvalho no uso dos ônibus escolares

Matão, SP
2025

Eric Wilson de Souza Alves

Felipe Henrique Savoini

Gabrielly Beatriz Silva

Jackson Silva Oliveira

FINDBUS:
website para auxiliar os alunos da Etec Sylvio de Mattos Carvalho no uso dos ônibus escolares

Trabalho de Conclusão do Curso apresentado ao Curso de Técnico em Informática para Internet Integrado ao Ensino Médio (MTec) da Escola Técnica Estadual Sylvio de Mattos Carvalho, orientado pelo Prof. Dr. Carlos Alberto Diniz, como parte dos requisitos para a obtenção do título de Técnico em Informática para Internet.

Matão, SP
2025

RESUMO

O FindBu é uma plataforma web para informar alunos da Etec Sylvio de Mattos Carvalho sobre os trajetos, paradas e horários dos ônibus escolares. O objetivo principal é facilitar a locomoção desses estudantes, garantindo que possam planejar suas rotas com segurança e autonomia. Como objetivos específicos, buscou-se: (1) reunir todas as rotas e horários dos ônibus em uma interface única e de fácil acesso; (2) apresentar mapas interativos com o traçado exato dos pontos de parada; (3) implementar um sistema de autenticação de usuários (cadastro e login) para gerenciamento de perfil; e (4) oferecer uma barra de busca para localizar pontos de parada específicos. A abordagem metodológica partiu do estudo do cenário produtivo, englobando: definição do tema, revisão de literatura, aplicação de questionários, definição dos requisitos e estruturação do projeto. No plano de desenvolvimento, iniciou-se pelo layout e pela criação da identidade visual. Em seguida, foi feita a modelagem do banco de dados, com o DER representando entidades como Usuarios, Motoristas, Onibus, Rotas e Pontos, conectadas por tabelas associativas. O backend foi implementado em PHP, servindo como uma API para se comunicar com um banco de dados MySQL; o frontend em HTML, CSS (com framework TailwindCSS) e JavaScript, integrando a API do Google Maps. Realizaram-se testes manuais de usabilidade e correção de bugs. O sistema entregue permite ao aluno consultar horários por rota, visualizar o traçado dos pontos no mapa e buscar por paradas específicas. Usuários cadastrados podem fazer login e gerenciar seus perfis. Conclui-se que o FindBus contribui para a inclusão digital e auxilia os alunos na tomada de decisões sobre transporte, oferecendo agilidade, confiabilidade e autonomia.

Palavras-chave: Transporte escolar. Aplicativo web. Mapas interativos. Etec Sylvio de Mattos Carvalho.

SUMÁRIO

1. INTRODUÇÃO

A modernização dos serviços informacionais no ambiente escolar tem se tornado cada vez mais necessária, especialmente em instituições públicas que atendem a uma grande quantidade de estudantes. Na Etec Sylvio de Mattos Carvalho, observou-se que muitos alunos, especialmente os ingressantes, enfrentam dificuldades para obter informações claras e precisas sobre o funcionamento do transporte escolar, como os pontos de embarque, rotas e horários dos ônibus disponibilizados. Esta dificuldade, além de gerar confusões e atrasos, compromete a plena utilização de um direito garantido pela Lei nº 9.394, de 20 de dezembro de 1996 (BRASIL, 1996), que define as diretrizes e bases da educação nacional e pela Lei nº 10.880, de 9 de junho de 2004, que institui o Programa Nacional de Apoio ao Transporte do Escolar (PNATE) (BRASIL, 2004).

Diante deste cenário, a escolha do tema surgiu da necessidade prática de proporcionar aos alunos uma ferramenta eficiente e de fácil acesso que centralize as informações do transporte escolar realizado no município de Matão, realizado pela empresa Via Paraty[1]. Assim, este projeto propõe o desenvolvimento do FindBus, um website destinado a auxiliar os estudantes da Etec Sylvio de Mattos Carvalho na utilização do transporte escolar, evitando equívocos quanto aos pontos e rotas dos ônibus e facilitando o planejamento dos horários diários dos alunos. Como objetivos específicos, destacamos: disponibilizar os horários atualizados dos ônibus; permitir aos estudantes a consulta rápida aos pontos de embarque e desembarque; reduzir a desinformação e os erros de locomoção; e ampliar o conhecimento dos alunos sobre o transporte escolar público disponível.

Ao adotar recursos digitais acessíveis, o projeto visa não apenas otimizar o acesso às informações sobre o transporte escolar, mas também contribuir para a eficiência administrativa da Etec, ao diminuir a sobrecarga de informações que, atualmente, precisam ser repassadas individualmente pela equipe escolar. Além disso, o desenvolvimento do FindBus representa uma aplicação concreta dos conhecimentos adquiridos no curso técnico de Informática para Internet,

[1] Sobre a empresa, vide: https://paratymobilidade.com.br/viacao_paraty.php. Acesso feito em 03 out. 2025.

demonstrando a capacidade dos estudantes em propor soluções tecnológicas para demandas reais da comunidade escolar.

A metodologia aplicada neste projeto partiu do estudo do cenário produtivo que nos ajudou na escolha do tema e, consequentemente, realizamos uma revisão bibliográfica sobre legislação educacional e transporte escolar. Na sequência, foram desenvolvidas as etapas técnicas, que incluíram: a) criação do Diagrama de Entidade-Relacionamento (DER); b) escolha das ferramentas de desenvolvimento (HTML, CSS, PHP, MySQL, JavaScript, Visual Studio Code, Canva); c) criação do logotipo; d) desenvolvimento do backend e frontend; e) realização de testes de navegabilidade, acessibilidade e usabilidade; f) hospedagem do sistema em servidor, e; d) registro das atividades em diário de bordo e relatório técnico.

Dito isto, este relatório está estruturado em três seções principais, a saber: na primeira parte, apresentamos o Diagrama de Entidade-Relacionamento e as ferramentas utilizadas no desenvolvimento do projeto; na segunda parte, apresentamos todo o processo de implementação do FindBus, inclusive o código-fonte; por fim, na terceira parte, analisamos os resultados obtidos com o projeto concluído, bem como os impactos esperados, como a melhoria na organização dos estudantes e o aumento do acesso ao transporte escolar, sintetizando assim as principais contribuições do projeto, suas limitações e as possibilidades de aprimoramentos futuros.

2. DIAGRAMA DE ENTIDADE-RELACIONAMENTO E FERRAMENTAS UTILIZADAS NO DESENVOLVIMENTO DO PROJETO FINDBUS

Nessa seção apresentamos o Diagrama de Entidade-Relacionamento (DER) do projeto FindBus, bem como as ferramentas utilizadas no seu desenvolvimento.

2.1 Diagrama de Entidade-Relacionamento

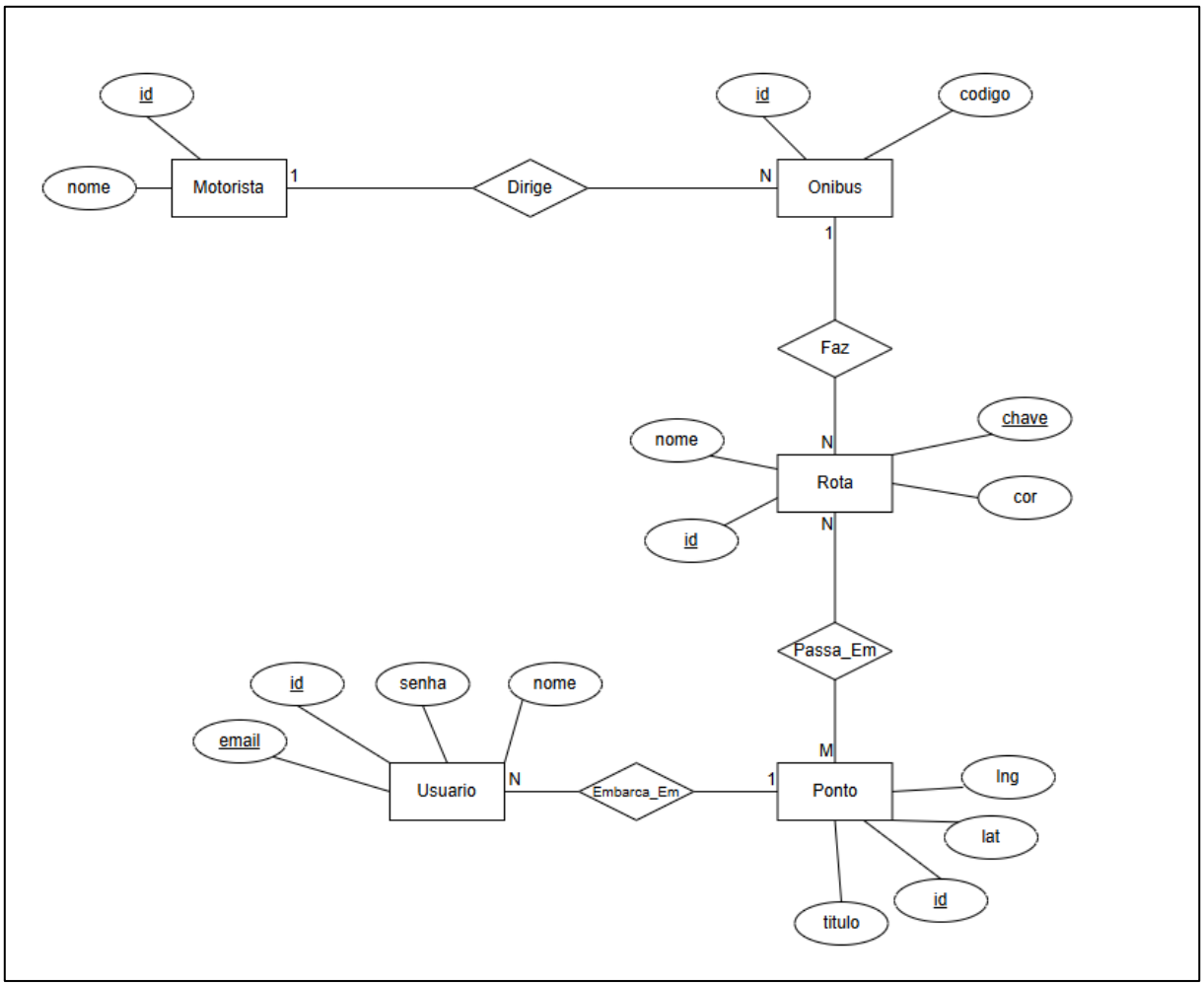

Figura 1: Diagrama de Entidade-Relacionamento do FindBus.
FONTE: Elaborado pelos autores (2025).

Iniciando pela entidade **Usuario**, esta representa os indivíduos que utilizarão a aplicação FindBus. Cada usuário é identificado por um id único. A entidade também armazena um nome (o nome completo do usuário), um e-mail (que deve ser único e

é usado para a autenticação no sistema) e uma senha (para validar o acesso). Esta entidade é crucial para a gestão de contas e personalização da experiência, conectando-se diretamente ao sistema de logística através do relacionamento **Embarca_Em** (ou Espera_Em). Esta ligação, de cardinalidade N:1 (Muitos-para-Um), permite que múltiplos (N) usuários se associem a um (1) Ponto específico, definindo-o como seu local de embarque principal.

Passando ao coração da funcionalidade, encontramos a entidade **Motorista**. Ela serve como um registro dos condutores, contendo um id (para identificação única) e o nome do profissional.

A entidade **Motorista** conecta-se à entidade **Onibus** por meio do relacionamento **Dirige**. A entidade **Onibus** simboliza um veículo específico da frota, identificado por um id (seu registro único) e um código (a designação visível do veículo, como "Ônibus #AZL-10"). Essa relação possui uma cardinalidade 1:N (Um para Muitos), o que estabelece que um motorista pode conduzir múltiplos ônibus, mas um ônibus em particular é operado por apenas um motorista.

Continuando o fluxo, a entidade **Onibus** possui uma ligação com a entidade **Rota** através do relacionamento **Faz**. A rota especifica um itinerário ou linha, sendo definida por um id (seu identificador numérico), um nome (ex: "Linha Verde - Setor Sul"), uma cor (para a representação visual no mapa) e uma chave (um código textual único, como "verde"). A cardinalidade aqui também é 1:N (Um para Muitos), indicando que um ônibus pode ser alocado para realizar diversas rotas, enquanto uma rota é executada por um único ônibus.

Chegamos à ligação mais importante do sistema, que une a entidade **Rota** à entidade **Ponto**. A entidade **Ponto** materializa uma parada física (uma localização geográfica específica), possuindo um id (seu identificador), um título (o nome da parada) e suas coordenadas geográficas lat (latitude) e lng (longitude).

A conexão entre rota e ponto é formalizada pelo relacionamento **Passa_Em**, do tipo N:M (Muitos para Muitos). Isto significa que uma rota consiste em variados pontos, e, inversamente, um ponto (como a ETEC) pode ser parte de várias rotas. O relacionamento **Passa_Em** é vital porque ele carrega um atributo próprio: a ordem. Este atributo é indispensável, pois determina a sequência exata das paradas (primeira, segunda, e assim por diante) em uma determinada rota, assegurando que o percurso seja apresentado de forma correta e sequencial.

2.2 Ferramentas utilizadas no desenvolvimento do projeto

Para a construção da interface do FindBus, foram utilizadas as seguintes tecnologias:

2.2.1 HTML 5

O HTML 5 é a versão mais moderna da linguagem de marcação usada para criar páginas da web. Ele é a base de qualquer site, trazendo melhorias na estrutura do código, compatibilidade com dispositivos móveis e suporte nativo a multimídia, o que torna a criação de sites interativos mais eficiente.

No projeto FindBus, o HTLM 5[2] foi empregado na estruturação semântica do conteúdo da página, definindo elementos como o cabeçalho, a área do mapa, os painéis de informação e os botões de interação.

2.2.2 CSS3 e Tailwind CSS

CSS3 é a terceira e mais recente versão das *Cascading Style Sheets* (Folha de Estilos em Cascata), utilizada para definir o design visual de um projeto web. Com recursos como transições, animações e sistemas de layout avançados como Flexbox e Grid, o CSS3 permite a criação de interfaces ricas e responsivas[3]. O Tailwind CSS é um framework que se baseia no CSS, mas adota uma abordagem utility-first, fornecendo um conjunto de classes predefinidas que podem ser aplicadas diretamente no HTML para construir designs personalizados de forma rápida, sem a necessidade de escrever CSS customizado[4].

No protótipo, o Tailwind CSS, um framework utility-first, foi utilizado para a estilização visual do website, permitindo a criação de um design moderno e responsivo

---

[2] Para saber mais sobre o HTML5, acesse o site https://www.task.com.br/blog/o-que-e-html5/. Acesso feito em 08 out. 2025.
[3] Para saber mais sobre o CSS3, acesse o site https://voitto.com.br/blog/artigo/o-que-e-css3. Acesso feito em 08 out. 2025.
[4] Para saber mais sobre o Tailwind CSS, acesse o site https://kinsta.com/pt/blog/tailwind-css/. Acesso feito em 08 out. 2025.

de forma ágil. O arquivo style.css foi usado para estilos personalizados, como o dimensionamento do container do mapa.

2.2.3 JavaScript

JavaScript[5] é uma linguagem de programação interpretada, o que significa que seu código é traduzido e executado diretamente no navegador do usuário por um motor de JavaScript. Juntamente com HTML e CSS, é uma das três principais tecnologias da World Wide Web, sendo a principal responsável por adicionar interatividade e comportamento dinâmico às páginas.

No FindBus, o JavaScript foi o núcleo da interatividade do protótipo. Ele foi utilizado para inicializar o mapa, processar os cliques do usuário nas rotas, chamar a API do Google Maps para desenhar o trajeto e atualizar dinamicamente os painéis de informação com os dados do ônibus e do motorista.

2.2.4 API do Google Maps

Uma API (Interface de Programação de Aplicações) de mapa, como a do Google Maps, é uma grande biblioteca de códigos que permite aos desenvolvedores integrar funcionalidades de mapas do Google em suas próprias aplicações ou sites. Com ela, é possível criar soluções que envolvem visualização de mapas, rastreamento, cálculo de rotas, entre outros[6].

Esta API foi a ferramenta central do protótipo do FindBus. Por meio dela, foi possível renderizar o mapa interativo da cidade de Matão e utilizar seus serviços de direções (DirectionsService) para calcular e desenhar as rotas de ônibus na tela, conectando os pontos de parada definidos no projeto.

---

[5] Para saber mais sobre o JavaScript, acesse o site https://aws.amazon.com/pt/what-is/javascript/. Acesso feito em 08 out. 2025.

[6] Para saber mais sobre a API do Google Maps, acesse o site https://www.geoambiente.com.br/blog/api-de-mapa-o-que-e-e-por-que-pode-ser-importante-para-o-seu-negocio/. Acesso feito em 08 out. 2025.

2.2.5 PHP

O PHP[7] (*Hypertext Preprocessor* – Pré-processador de Hipertexto) foi utilizado como a linguagem de programação do lado do servidor (backend) para o FindBus. Sendo uma linguagem de script de código aberto amplamente empregada para o desenvolvimento web, o PHP oferece uma sintaxe intuitiva e vasta compatibilidade com diversos bancos de dados, como o MySQL (PHP Group, 2023a). No FindBus, o PHP é responsável por toda a lógica de negócios que não pode ser executada no navegador do cliente. Suas funções principais incluem: a) processamento de formulários: recepcionar e validar os dados enviados pelos formulários de login, cadastro e atualização de perfil; b) gerenciamento de sessão: controlar a autenticação dos usuários, iniciando uma sessão (session_start()) após o login bem-sucedido e verificando-a em páginas restritas; c) segurança: criptografar as senhas dos usuários usando a função password_hash() antes de salvá-las no banco de dados e verificá-las com password_verify() durante o login; d) criação de API: Servir como uma API interna (ex: api/buscar_dados.php) que consulta o banco de dados, formata os dados das rotas em JSON e os entrega ao frontend (JavaScript).

2.2.6 MySQL

O MySQL[8] foi o Sistema de Gerenciamento de Banco de Dados (SGBD) relacional de código aberto escolhido para o projeto FindBus. Reconhecido por sua robustez, escalabilidade e desempenho, o MySQL é ideal para aplicações web que demandam armazenamento e recuperação eficientes de grandes volumes de dados (Oracle Corporation, 2023). Ele é responsável por armazenar de forma segura, estruturada e persistente todos os dados da aplicação, incluindo as informações de autenticação na tabela usuarios, bem como todas as entidades que compõem o sistema de rotas: motoristas, onibus, rotas, pontos e a tabela associativa rota_pontos.

---

[7] Para saber mais sobre o PHP, acesse o site https://www.php.net/manual/pt_BR/intro-whatis.php. Acesso feito em 23 out. 2025.
[8] Para saber mais sobre o MySQL, acesse o site https://www.mysql.com/. Acesso feito em 23 out. 2025.

3. DESENVOLVIMENTO

Apresentamos a seguir as funcionalidades do FindBus organizados em:

- Tela principal;
- Telas de login e de cadastro;
- Tela de perfil do usuário.

3.1 Tela Principal

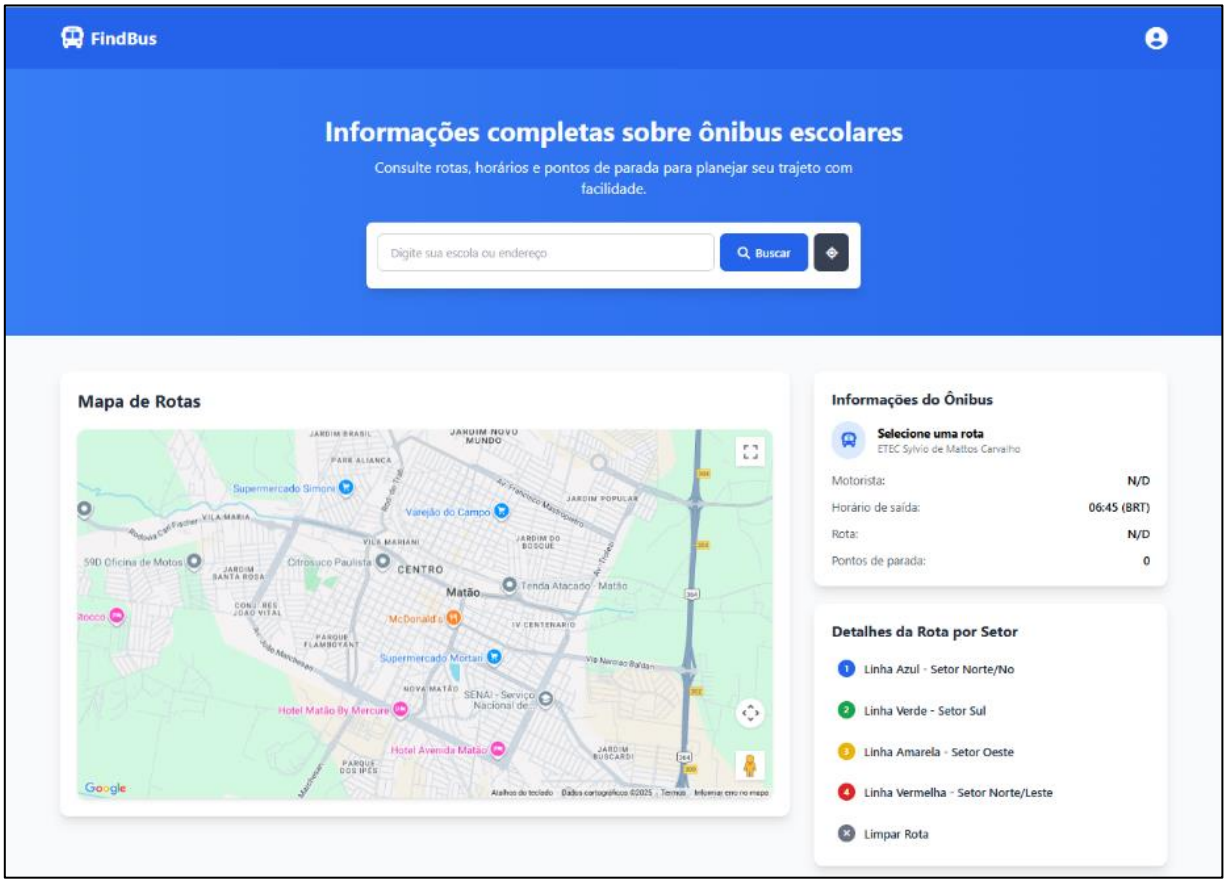

Figura 2: Tela inicial do projeto FindBus.
FONTE: Elaborado pelos autores (2025).

A tela principal, ilustrada na Figura 2, é o núcleo funcional do FindBus e é composta por diversos componentes interligados. O arquivo index.php é responsável por exibir o layout e gerenciar a autenticação do usuário. Utilizando PHP session_start() , o cabeçalho verifica se o $_SESSION['usuario_nome'] está definido, exibindo dinamicamente o menu do usuário logado ou o ícone de login padrão.

O script.js centraliza toda a interatividade. Ao carregar, a API do Google Maps executa a função initMap(), que por sua vez utiliza fetch para fazer uma requisição assíncrona ao api/buscar_dados.php. Esta API, escrita em PHP, realiza uma consulta otimizada ao banco de dados MySQL para buscar todas as rotas, motoristas e pontos, retornando-os como um objeto JSON.

Com os dados JSON carregados, o script.js ativa os elementos da interface:

1. **Mapa de Rotas:** O com id="map", estilizado com altura definida em css/style.css, é preenchido com o mapa do Google.
2. **Painel de Detalhes da Rota:** Os botões (ex: id="linha-azul") são configurados pela função setupRouteListeners. Um clique chama a função desenharRota(), que utiliza o DirectionsService do Google para calcular o trajeto e o DirectionsRenderer para desenhá-lo no mapa.
3. **Painel de Informações:** Os campos (ex: id="info-onibus-codigo") são preenchidos dinamicamente pela função desenharRota com os dados da rota selecionada.
4. **Barra de Busca:** A função setupSearch monitora o id="search-input" para autocompletar e o id="search-button" para executar a busca, que chama desenharRota() e centraliza o mapa no ponto encontrado.
5. **Histórico:** O script utiliza o localStorage do navegador para salvar e exibir as últimas buscas na div id="historicoBusca".

A seguir, apresentamos o código-fonte dessa tela (figura 2):

index.php

```
<?php
// session_start() é SEMPRE a primeira coisa em um script PHP que usa sessões.
// Ela carrega ou inicia a sessão do usuário, permitindo o acesso
// às variáveis $_SESSION (como $_SESSION['usuario_nome']).
session_start();
?>
<!DOCTYPE html>
```

```
<html lang="pt-BR">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>FindBus - Roteirização Otimizada</title>

    <script src="https://cdn.tailwindcss.com"></script>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

    <link rel="stylesheet" href="css/style.css">
</head>

<body class="bg-gray-50 min-h-screen">

    <header class="bg-blue-600 text-white shadow-lg">

        <div class="container mx-auto px-4 py-6">

            <div class="flex justify-between items-center">

                <div class="flex items-center space-x-2">
                    <i class="fas fa-bus text-3xl"></i>
                    <h1 class="text-2xl font-bold">FindBus</h1>
                </div>

                <div class="flex items-center space-x-4">

                <?php
                // Início do bloco PHP:
                // 'isset' verifica se a variável $_SESSION['usuario_nome'] EXISTE.
```

```
                // Esta variável só é criada no 'login.php' após um login bem-sucedido.
                // É assim que o site sabe que você está logado.
                if (isset($_SESSION['usuario_nome'])):
                ?>
                    <div class="hidden md:block relative">

                        <button id="user-menu-button" class="flex items-center space-x-2 text-white p-2 rounded-lg hover:bg-blue-700 transition-colors focus:outline-none">
                            <i class="fas fa-user-circle text-2xl"></i>

                            <span class="font-medium">Olá, <?php echo htmlspecialchars($_SESSION['usuario_nome']); ?>!</span>
                            <i class="fas fa-chevron-down text-xs"></i>
                        </button>

                        <div id="user-menu-dropdown" class="hidden absolute right-0 mt-2 w-48 bg-white rounded-xl shadow-lg py-1 z-50">
                            <a href="perfil.php" class="flex items-center w-full px-4 py-2 text-sm text-gray-700 hover:bg-gray-100">
                                <i class="fas fa-user-edit w-6 mr-2 text-gray-500"></i>
                                Meu Perfil
                            </a>
                            <a href="logout.php" class="flex items-center w-full px-4 py-2 text-sm text-red-600 hover:bg-red-50">
                                <i class="fas fa-sign-out-alt w-6 mr-2 text-red-500"></i>
                                Sair
                            </a>
                        </div>
                    </div>
                <?php
                // 'else' do PHP: Se 'isset($_SESSION['usuario_nome'])' for FALSO...
                else:
                ?>
```

```
                    <a href="login.html" title="Login / Cadastrar" class="hidden
md:block text-3xl hover:text-blue-200 transition-colors">
                        <i class="fas fa-user-circle"></i>
                    </a>
                <?php
                // 'endif;' termina o bloco 'if/else' do PHP
                endif;
                ?>

                <button class="md:hidden text-2xl" id="mobile-menu-button">
                    <i class="fas fa-bars"></i>
                </button>

                </div>
            </div>
        </div>

        <div class="md:hidden hidden bg-blue-700" id="mobile-menu">
            <div class="px-2 pt-2 pb-3 space-y-1">

                <?php if (isset($_SESSION['usuario_nome'])): ?>
                    <div class="px-2 py-2">
                        <p class="text-blue-200">Olá, <?php echo
htmlspecialchars($_SESSION['usuario_nome']); ?>!</p>
                    </div>
                    <a href="perfil.php" class="block px-3 py-2 rounded-md text-base
font-medium text-white hover:bg-blue-800">
                        Meu Perfil
                    </a>
                    <a href="logout.php" class="block px-3 py-2 rounded-md text-base
font-medium text-white bg-red-500 hover:bg-red-600">
                        Sair
                    </a>
                <?php else: ?>
```

```
                    <a href="login.html" class="block px-3 py-2 rounded-md text-base font-medium text-white hover:bg-blue-800">
                        Login / Cadastrar
                    </a>
                <?php endif; ?>
            </div>
        </div>
    </header>

    <section class="bg-gradient-to-r from-blue-500 to-blue-600 text-white py-16">
        <div class="container mx-auto px-4 text-center">
            <h2 class="text-4xl font-bold mb-4">Informações completas sobre ônibus escolares</h2>
            <p class="text-xl mb-8 max-w-2xl mx-auto">Consulte rotas, horários e pontos de parada para planejar seu trajeto com facilidade.</p>

            <div class="max-w-2xl mx-auto bg-white rounded-lg shadow-lg p-4">
                <div class="flex flex-col md:flex-row gap-2">

                    <div class="relative flex-grow">

                        <input type="text" placeholder="Digite sua escola ou endereço" class="w-full px-4 py-3 border border-gray-300 rounded-lg text-gray-800 focus:outline-none focus:ring-2 focus:ring-blue-500" id="search-input">

                        <div class="absolute z-10 w-full mt-1 bg-white border-gray-300 rounded-lg shadow-lg hidden search-dropdown" id="search-results"></div>
                    </div>

                    <button id="search-button" class="bg-blue-600 hover:bg-blue-700 text-white px-6 py-3 rounded-lg font-medium transition-colors">
                        <i class="fas fa-search mr-2"></i>Buscar
                    </button>
```

```
                    <button id="btnMinhaLocalizacao" class="bg-gray-700 hover:bg-gray-800 text-white px-4 py-3 rounded-lg font-medium transition-colors" title="Usar Minha Localização">
                        <i class="fas fa-location-crosshairs"></i>
                    </button>
                </div>

                <div id="historicoBusca" class="mt-2"></div>
            </div>
        </div>
    </section>

    <main class="container mx-auto px-4 py-12">
        <div class="grid grid-cols-1 lg:grid-cols-3 gap-8">

            <div class="lg:col-span-2">
                <div class="bg-white rounded-xl shadow-lg p-6">
                    <h3 class="text-2xl font-bold text-gray-800 mb-6">Mapa de Rotas</h3>

                    <div class="map-container" id="map"></div>
                </div>
            </div>

            <div class="space-y-6">

                <div class="bg-white rounded-xl shadow-lg p-6">
                    <h3 class="text-xl font-bold text-gray-800 mb-4">Informações do Ônibus</h3>
                    <div class="flex items-center mb-4">
                        <div class="bg-blue-100 p-3 rounded-full mr-4">
                            <i class="fas fa-bus text-blue-600 text-xl"></i>
                        </div>
```

```
                    <div>
                        <h4 class="font-bold" id="info-onibus-codigo">Selecione uma
rota</h4>
                        <p class="text-sm text-gray-500">ETEC Sylvio de Mattos
Carvalho</p>
                    </div>
                </div>
                <div class="space-y-3">
                    <div class="flex justify-between">
                        <span class="text-gray-600">Motorista:</span>
                        <span class="font-medium" id="info-rota-
motorista">N/D</span>
                    </div>
                    <div class="flex justify-between">
                        <span class="text-gray-600">Horário de saída:</span>
                        <span class="font-medium">06:45 (BRT)</span>
                    </div>
                    <div class="flex justify-between">
                        <span class="text-gray-600">Rota:</span>
                        <span class="font-medium" id="info-rota-nome">N/D</span>
                    </div>
                    <div class="flex justify-between">
                        <span class="text-gray-600">Pontos de parada:</span>
                        <span class="font-medium" id="info-rota-paradas">0</span>
                    </div>
                </div>
            </div>

            <div class="bg-white rounded-xl shadow-lg p-6">
                <h3 class="text-xl font-bold text-gray-800 mb-4">Detalhes da Rota
por Setor</h3>
                <div class="space-y-4">
```

```
                    <a id="linha-azul" class="flex items-center w-full text-left p-2 rounded-lg hover:bg-blue-50 transition cursor-pointer">
                        <div class="w-6 h-6 rounded-full bg-blue-600 flex items-center justify-center text-white text-xs font-bold mr-3">1</div>
                        <h4 class="font-medium text-gray-800">Linha Azul - Setor Norte/No</h4>
                    </a>

                    <a id="linha-verde" class="flex items-center w-full text-left p-2 rounded-lg hover:bg-green-50 transition cursor-pointer">
                        <div class="w-6 h-6 rounded-full bg-green-600 flex items-center justify-center text-white text-xs font-bold mr-3">2</div>
                        <h4 class="font-medium text-gray-800">Linha Verde - Setor Sul</h4>
                    </a>

                    <a id="linha-amarela" class="flex items-center w-full text-left p-2 rounded-lg hover:bg-yellow-50 transition cursor-pointer">
                        <div class="w-6 h-6 rounded-full bg-yellow-500 flex items-center justify-center text-white text-xs font-bold mr-3">3</div>
                        <h4 class="font-medium text-gray-800">Linha Amarela - Setor Oeste</h4>
                    </a>

                    <a id="linha-vermelha" class="flex items-center w-full text-left p-2 rounded-lg hover:bg-red-50 transition cursor-pointer">
                        <div class="w-6 h-6 rounded-full bg-red-600 flex items-center justify-center text-white text-xs font-bold mr-3">4</div>
                        <h4 class="font-medium text-gray-800">Linha Vermelha - Setor Norte/Leste</h4>
                    </a>

                    <a id="limpar-rota" class="flex items-center w-full text-left p-2 rounded-lg hover:bg-gray-100 transition cursor-pointer">
```

```
                        <div class="w-6 h-6 rounded-full bg-gray-500 flex items-center justify-center text-white text-xs font-bold mr-3"><i class="fas fa-times"></i></div>
                        <h4 class="font-medium text-gray-800">Limpar Rota</h4>
                    </a>
                </div>
            </div>
        </div>
    </main>

    <section class="bg-gray-100 py-16"></section>
    <footer class="bg-gray-800 text-white py-12"></footer>

    <script src="js/script.js"></script>

    <script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBfqDQMIY6dBQ35GU1rWb71nmZYqc3ERQw&callback=initMap">
    </script>

    <script>
        // Pega as referências dos elementos HTML pelos seus IDs
        const userMenuButton = document.getElementById('user-menu-button');
        const userMenuDropdown = document.getElementById('user-menu-dropdown');

        // Este 'if' é uma verificação de segurança.
        // Se o usuário não estiver logado, 'userMenuButton' não existe (é 'null').
        // Tentar adicionar um 'addEventListener' a 'null' causaria um erro no console.
        // Este 'if' garante que o código só rode se o usuário ESTIVER LOGADO.
        if (userMenuButton) {
```

```
            // Adiciona um "escutador" de evento de clique no botão do menu
            userMenuButton.addEventListener('click', () => {
                // 'classList.toggle' é uma função mágica:
                // Se a classe 'hidden' existe, ela a REMOVE.
                // Se a classe 'hidden' NÃO existe, ela a ADICIONA.
                // É isso que faz o menu aparecer e desaparecer.
                userMenuDropdown.classList.toggle('hidden');
            });

            // Opcional: Fecha o menu se o usuário clicar fora dele
            document.addEventListener('click', (event) => {
                // 'event.target' é o local exato onde o usuário clicou

                // Esta lógica verifica:
                // 1. Se o clique NÃO foi dentro do botão (!userMenuButton.contains(...))
                // E
                // 2. Se o clique NÃO foi dentro do dropdown (!userMenuDropdown.contains(...))
                if (!userMenuButton.contains(event.target) && !userMenuDropdown.contains(event.target)) {
                    // Se o clique foi "fora", força o menu a fechar adicionando 'hidden'
                    userMenuDropdown.classList.add('hidden');
                }
            });
        }
    </script>
</body>
</html>
```

script.js

```
// --- LÓGICA DO MENU MOBILE ---
// O código aqui é executado assim que o script.js é carregado.

// 1. Pega a referência do elemento HTML que tem o ID "mobile-menu-button" (o botão hambúrguer)
const mobileMenuButton = document.getElementById("mobile-menu-button");
// 2. Pega a referência do elemento HTML que tem o ID "mobile-menu" (o menu dropdown)
const mobileMenu = document.getElementById("mobile-menu");

// 3. Adiciona um "escutador" de evento de clique no botão hambúrguer
mobileMenuButton.addEventListener("click", () => {
  // 4. 'classList.toggle("hidden")' é uma função que:
  //    - Se a classe 'hidden' (do Tailwind) estiver no 'mobileMenu', ela é REMOVIDA (o menu aparece).
  //    - Se a classe 'hidden' NÃO estiver, ela é ADICIONADA (o menu desaparece).
  mobileMenu.classList.toggle("hidden");
});

// --- VARIÁVEIS GLOBAIS DO MAPA ---
// 'let' é usado porque essas variáveis começarão vazias (undefined) e
// serão preenchidas DEPOIS pela função 'initMap'.

let map; // Variável para guardar o objeto principal do Google Map
let directionsService; // Variável para o objeto que CALCULA as rotas (pede ao Google)
let directionsRenderer; // Variável para o objeto que DESENHA as rotas no mapa

// 'const' é usado para um valor que não vai mudar.
const mataoCoords = { lat: -21.6036, lng: -48.3658 }; // Coordenadas de Matão (centro do mapa)
```

```
// Esta é a variável que guardará os dados das rotas (ônibus, motorista, pontos)
// vindos da nossa API (buscar_dados.php). Começa como um objeto vazio.
let dadosDasRotas = {};

/**
 *
================================================================
=========
 * FUNÇÃO DE INICIALIZAÇÃO DO MAPA
 *
================================================================
=========
 * Esta é a função mais importante.
 * Ela é chamada AUTOMATICAMENTE pela API do Google Maps assim que ela termina de carregar.
 * (Isso acontece por causa do "...&callback=initMap" na tag <script> do index.php)
 * * 'async' na frente da função permite o uso de 'await' dentro dela.
 */
async function initMap() {

  // 1. CRIA O MAPA
  // Cria um novo objeto de Mapa do Google...
  map = new google.maps.Map(
    document.getElementById("map"), // ...e o anexa na <div id="map"> do HTML
    {
      center: mataoCoords, // Centraliza em Matão
      zoom: 14, // Define o zoom inicial
      mapTypeControl: false, // Esconde o controle de "Satélite/Mapa" (para limpar a interface)
    }
  );

  // 2. INICIALIZA OS SERVIÇOS DE ROTA
  // Cria o objeto que sabe como pedir rotas ao Google
```

```
  directionsService = new google.maps.DirectionsService();
  // Cria o objeto que sabe como desenhar rotas em um mapa
  directionsRenderer = new google.maps.DirectionsRenderer({
    suppressMarkers: true, // Esconde os marcadores 'A' (origem) e 'B' (destino) padrão
  });
  // Conecta o desenhador ('renderer') ao nosso mapa ('map')
  directionsRenderer.setMap(map);

  // 3. BUSCA OS DADOS DA NOSSA API (PHP/MYSQL)
  // 'try...catch' é um bloco de tratamento de erros. Se qualquer coisa dentro do 'try'
  // falhar (ex: API offline, erro de PHP), o código pula para o 'catch'.
  try {
    // 'await' pausa a execução do script AQUI até que o 'fetch' termine.
    // 'fetch' é a função do navegador para fazer requisições HTTP (como o AJAX).
    // Ele está "chamando" o arquivo 'api/buscar_dados.php' no servidor.
    const response = await fetch("api/buscar_dados.php");

    // 'response.ok' verifica se a resposta do servidor foi um sucesso (código 200).
    // Se for um erro (como 404 - Não Encontrado ou 500 - Erro Interno do Servidor),
    // 'response.ok' será 'false' e nós "jogamos" (throw) um erro.
    if (!response.ok) {
      throw new Error("Falha ao carregar dados da API. Status: " + response.status);
    }

    // 'response.json()' pega o texto da resposta (que é um JSON) e o converte
    // em um objeto JavaScript real. 'await' é usado porque isso também leva um tempo.
    // 'dadosDasRotas' agora conterá: { azul: {...}, verde: {...}, ... }
    dadosDasRotas = await response.json();

    // 4. CONFIGURA AS FUNÇÕES DEPENDENTES
    // SÓ DEPOIS ('await') que os dados (dadosDasRotas) chegarem,
    // nós finalmente configuramos as outras partes do site que DEPENDEM desses dados.
    setupRouteListeners(); // Configura os cliques nos botões (Linha Azul, Verde...)
```

```
    setupSearch(); // Configura a barra de busca
    setupMyLocationButton(); // Configura o botão de GPS

    // Funções de inicialização
    limparRotaDoMapa(); // Garante que o painel de info comece limpo
    mostrarHistorico(); // Carrega o histórico de buscas do usuário (do localStorage)

  } catch (error) {
    // Bloco 'catch': Executado se o 'try' falhar.

    // Mostra o erro detalhado no console do navegador (F12) para o desenvolvedor
    console.error("Erro ao carregar dados das rotas:", error);

    // Mostra um alerta simples para o usuário final
    alert("Não foi possível carregar os dados das rotas. Tente recarregar a página.");
  }
} // Fim da função initMap


/**
 *
================================================================
=========
 * CONFIGURAÇÃO DOS BOTÕES DE ROTA
 *
================================================================
=========
 * Esta função é chamada por 'initMap' DEPOIS que os dados chegam.
 * Ela adiciona os "escutadores" de clique nos botões das rotas.
 */
function setupRouteListeners() {

  // 1. Botão Linha Azul
  // Pega o elemento <a id="linha-azul">
```

```
  document.getElementById("linha-azul")
    // Adiciona um "escutador" de evento de clique
    .addEventListener("click",
      // '() => ...' é uma 'arrow function' (função anônima)
      // Quando o clique acontece, ela executa a função 'desenharRota'
      // e passa para ela os dados da rota azul (que buscamos da API)
      () => desenharRota(dadosDasRotas.azul)
    );

  // 2. Botão Linha Verde (processo idêntico)
  document.getElementById("linha-verde")
    .addEventListener("click", () => desenharRota(dadosDasRotas.verde));

  // 3. Botão Linha Amarela (processo idêntico)
  document.getElementById("linha-amarela")
    .addEventListener("click", () => desenharRota(dadosDasRotas.amarela));

  // 4. Botão Linha Vermelha (processo idêntico)
  document.getElementById("linha-vermelha")
    .addEventListener("click", () => desenharRota(dadosDasRotas.vermelha));

  // 5. Botão Limpar Rota
  document.getElementById("limpar-rota").addEventListener("click", () => {
    limparRotaDoMapa(); // Chama a função de limpeza
    map.panTo(mataoCoords); // Recentraliza o mapa em Matão
    map.setZoom(14); // Reseta o zoom
  });
} // Fim da função setupRouteListeners


/**
 *
==========================================================
=========
```

```
 * FUNÇÃO PRINCIPAL PARA DESENHAR A ROTA
 *
================================================================
=========
 * Esta é a função que faz a mágica de desenhar no mapa.
 * @param {object} dadosDaRota - O objeto da rota que deve ser desenhada
 * (ex: dadosDasRotas.azul, que contém nome, cor, pontos, etc.)
 */
function desenharRota(dadosDaRota) {
  // 1. Limpa qualquer rota que já esteja desenhada no mapa
  limparRotaDoMapa();

  // 2. Pega o array de pontos de dentro do objeto da rota
  const pontosDaRota = dadosDaRota.pontos;

  // 3. Verificação de segurança: Se a rota não tiver pelo menos 2 pontos (origem e
destino), não faz nada.
  if (pontosDaRota.length < 2) {
    console.warn("Rota com menos de 2 pontos. Não é possível desenhar.");
    return; // Para a execução da função
  }

  // 4. Define a Origem (o primeiro ponto do array, índice 0)
  const origem = pontosDaRota[0]; // Ex: {lat: -21.1, lng: -48.1, title: "Ponto A"}

  // 5. Define o Destino (o último ponto do array)
  const destino = pontosDaRota[pontosDaRota.length - 1];

  // 6. Define os Pontos de Parada (Waypoints)
  // 'slice(1, -1)' pega TODOS os pontos ENTRE o primeiro (1) e o último (-1).
  // 'map(...)' transforma esse array de pontos no formato que a API do Google exige.
  const waypoints = pontosDaRota.slice(1, -1).map((ponto) => ({
    location: { lat: ponto.lat, lng: ponto.lng }, // O local da parada
    stopover: true, // 'true' significa que é uma parada oficial (afeta o cálculo de tempo)
```

```
  }));

  // 7. Monta o objeto de requisição (o "pedido" para o Google)
  const request = {
    origin: origem,       // Local de partida
    destination: destino, // Local de chegada
    waypoints: waypoints, // Paradas no caminho
    travelMode: google.maps.TravelMode.DRIVING, // Define como "dirigindo"
  };

  // 8. ENVIA O PEDIDO PARA O GOOGLE
  // Chama o serviço 'directionsService' com o 'request' que montamos.
  // O segundo argumento é uma função 'callback' (será chamada quando o Google
responder).
  directionsService.route(request, (result, status) => {

    // 'status' diz se o pedido deu certo
    if (status == "OK") {
      // SUCESSO! O Google calculou a rota.

      // 1. Manda o desenhador ('renderer') exibir o 'result' (a rota calculada)
      directionsRenderer.setDirections(result);

      // 2. Configura a aparência da linha (cor, espessura)
      directionsRenderer.setOptions({
        polylineOptions: { // Opções da linha
          strokeColor: dadosDaRota.cor, // Pega a cor (ex: '#3b82f6') do nosso objeto de
dados
          strokeWeight: 6, // Espessura da linha
          strokeOpacity: 0.8, // Transparência da linha
        },
      });
    } else {
      // ERRO! O Google não conseguiu calcular a rota.
```

```
      console.error("Erro ao calcular rota do DirectionsService: " + status);
      window.alert("Não foi possível calcular a rota: " + status);
    }
  }); // Fim do callback do 'route'

  // 9. ATUALIZA O PAINEL DE INFORMAÇÕES (à direita do mapa)
  // Pega os elementos HTML pelo ID e atualiza o texto (textContent) deles
  document.getElementById("info-onibus-codigo").textContent =
dadosDaRota.codigoOnibus;
  document.getElementById("info-rota-nome").textContent = dadosDaRota.nome;
  document.getElementById("info-rota-paradas").textContent =
dadosDaRota.pontos.length; // Conta quantos pontos tem
  document.getElementById("info-rota-motorista").textContent =
dadosDaRota.motorista;
} // Fim da função desenharRota


/**
 *
================================================================
=========
 * FUNÇÃO DE LIMPEZA
 *
================================================================
=========
 * Limpa a rota desenhada do mapa e reseta o painel de informações.
 */
function limparRotaDoMapa() {
  // 1. Manda o desenhador apagar a rota (passando um resultado vazio)
  directionsRenderer.setDirections({ routes: [] });

  // 2. Reseta o texto do painel de informações para o padrão
  document.getElementById("info-onibus-codigo").textContent = "Selecione uma rota";
  document.getElementById("info-rota-nome").textContent = "N/D";
```

```
  document.getElementById("info-rota-paradas").textContent = "0";
  document.getElementById("info-rota-motorista").textContent = "N/D";
} // Fim da função limparRotaDoMapa


/**
 *
==========================================================================
=========
 * FUNÇÃO DO BOTÃO DE GEOLOCALIZAÇÃO (GPS)
 *
==========================================================================
=========
 */
function setupMyLocationButton() {
  // Pega a referência do botão
  const btnMinhaLocalizacao = document.getElementById("btnMinhaLocalizacao");

  // Adiciona o "escutador" de clique
  btnMinhaLocalizacao.addEventListener("click", () => {

    // 1. Verifica se o navegador (ex: Chrome) suporta a API de Geolocalização
    if (navigator.geolocation) {

      // 2. Pede ao navegador a posição atual do usuário
      // 'getCurrentPosition' recebe duas funções: uma para SUCESSO, outra para
ERRO.
      navigator.geolocation.getCurrentPosition(

        // --- Função de SUCESSO ---
        // 'posicao' é um objeto com os dados (latitude, longitude)
        (posicao) => {
          const lat = posicao.coords.latitude;
          const lng = posicao.coords.longitude;
```

```
        const localizacaoAtual = { lat, lng };

        // 3. Centraliza o mapa na localização do usuário
        map.setCenter(localizacaoAtual);
        map.setZoom(17); // Aumenta o zoom

        // 4. (Opcional) Cria um marcador azul pulsante onde o usuário está
        new google.maps.Marker({
          position: localizacaoAtual,
          map: map, // Anexa o marcador ao nosso mapa
          title: "Você está aqui!",
          // Ícone personalizado (um círculo azul em vez do pino vermelho)
          icon: {
            path: google.maps.SymbolPath.CIRCLE, // Forma
            scale: 8, // Tamanho
            fillColor: "#4285F4", // Cor (azul Google)
            fillOpacity: 1,
            strokeColor: "white",
            strokeWeight: 2,
          },
        });
      },

      // --- Função de ERRO ---
      // 'erro' contém o motivo (ex: usuário negou a permissão)
      (erro) => {
        console.error("Erro na geolocalização: ", erro);
        alert(
          "Não foi possível obter sua localização. Verifique as permissões do
navegador."
        );
      }
    );
  } else {
```

```
      // Se o navegador for muito antigo e não tiver 'navigator.geolocation'
      alert("Seu navegador não suporta geolocalização.");
    }
  });
} // Fim da função setupMyLocationButton


/**
 *
================================================================
=========
 * SEÇÃO DE LÓGICA DE BUSCA E HISTÓRICO
 *
================================================================
=========
 * Usa o 'localStorage' do navegador para salvar as buscas.
 */

// Define o número máximo de itens para salvar no histórico
const MAX_HISTORICO = 5;

/**
 * Salva um termo buscado no 'localStorage' do navegador
 * @param {string} termoBuscado - O texto que o usuário buscou (ex: "ETEC")
 */
function salvarBusca(termoBuscado) {
  // 1. Validação: Ignora buscas vazias ou só com espaços
  if (!termoBuscado || termoBuscado.trim() === "") return;

  // 2. Pega o histórico salvo.
  // 'localStorage' só salva TEXTO, então usamos 'JSON.parse' para
  // converter o texto de volta em um array.
  // Se não houver nada salvo ('null'), usa um array vazio '[]'.
  let historico = JSON.parse(localStorage.getItem("historico")) || [];
```

```
  // 3. Remove o termo se ele já existir (para movê-lo para o topo)
  // 'filter' cria um novo array mantendo apenas os itens que passam no teste.
  historico = historico.filter(
    (item) => item.toLowerCase() !== termoBuscado.toLowerCase()
  );

  // 4. Adiciona o novo termo no INÍCIO do array
  historico.unshift(termoBuscado);

  // 5. Limita o array ao tamanho máximo (ex: 5 itens)
  // 'slice(0, 5)' pega os itens do índice 0 até o 4.
  const historicoLimitado = historico.slice(0, MAX_HISTORICO);

  // 6. Salva o array atualizado de volta no localStorage.
  // 'JSON.stringify' converte o array JavaScript em uma string de texto JSON.
  localStorage.setItem("historico", JSON.stringify(historicoLimitado));

  // 7. Atualiza a exibição do histórico na tela
  mostrarHistorico();
}

/**
 * Remove um item específico do histórico (quando o usuário clica no 'x')
 * @param {string} termoParaRemover - O texto a ser removido
 */
function removerDoHistorico(termoParaRemover) {
  let historico = JSON.parse(localStorage.getItem("historico")) || [];

  // 1. Filtra o array, mantendo apenas os itens DIFERENTES do termo
  const novoHistorico = historico.filter((item) => item !== termoParaRemover);

  // 2. Salva o novo array (sem o item)
  localStorage.setItem("historico", JSON.stringify(novoHistorico));
```

```
  // 3. Atualiza a exibição (para o item sumir da tela)
  mostrarHistorico();
}

/**
 * Desenha os itens do histórico na tela (cria os botões dinamicamente)
 */
function mostrarHistorico() {
  // Pega o <div id="historicoBusca"> do HTML
  const divHistorico = document.getElementById("historicoBusca");
  if (!divHistorico) return; // Segurança: se o div não existir, para a função

  // 1. Limpa qualquer botão que já estava lá
  divHistorico.innerHTML = "";

  // 2. Carrega o histórico salvo
  const historico = JSON.parse(localStorage.getItem("historico")) || [];

  // 3. Só continua se o histórico não estiver vazio
  if (historico.length > 0) {

    // Cria o título "Últimas buscas:"
    const titulo = document.createElement("p");
    titulo.className = "text-sm text-gray-500 mt-3 mb-1"; // Estilos Tailwind
    titulo.textContent = "Últimas buscas:";
    divHistorico.appendChild(titulo); // Adiciona o título na div

    // 4. Cria um conjunto de botões para cada item no histórico
    // 'forEach' faz um loop por todos os itens do array
    historico.forEach((termo) => {
      // Cria o container (a "pílula" cinza)
      const itemContainer = document.createElement("div");
```

```
    itemContainer.className = "inline-flex items-center bg-blue-50 rounded-full mr-2 mb-2";

    // Botão 1: O texto (que refaz a busca)
    const termoButton = document.createElement("button");
    termoButton.className = "text-left text-sm text-blue-600 hover:bg-blue-100 rounded-l-full px-3 py-1 transition-colors";
    termoButton.textContent = termo; // Ex: "ETEC"

    // Define o que acontece ao clicar no texto
    termoButton.onclick = () => {
      // 1. Põe o texto "ETEC" de volta na barra de busca
      document.getElementById("search-input").value = termo;
      // 2. Simula um clique no botão "Buscar" (com a lupa)
      document.getElementById("search-button").click();
    };

    // Botão 2: O 'X' para remover
    const deleteButton = document.createElement("button");
    deleteButton.innerHTML = "×"; // Símbolo 'x'
    deleteButton.className = "text-lg font-bold text-blue-400 hover:text-blue-700 hover:bg-blue-100 rounded-r-full px-2 py-1 transition-colors";
    deleteButton.title = `Remover "${termo}" do histórico`;

    // Define o que acontece ao clicar no 'X'
    deleteButton.onclick = (event) => {
      // 'event.stopPropagation()' é MUITO IMPORTANTE:
      // Impede que o clique no 'x' também acione o clique no 'termoButton'
      // (que está no mesmo container).
      event.stopPropagation();
      removerDoHistorico(termo); // Chama a função de remoção
    };

    // 5. Adiciona os botões (texto e 'x') ao container-pílula
```

```
      itemContainer.appendChild(termoButton);
      itemContainer.appendChild(deleteButton);
      // 6. Adiciona a pílula inteira na div principal do histórico
      divHistorico.appendChild(itemContainer);
    });
  }
} // Fim da função mostrarHistorico


/**
 *
==========================================================================
=========
 * FUNÇÃO DE CONFIGURAÇÃO DA BUSCA (A MAIS COMPLEXA)
 *
==========================================================================
=========
 * Configura a barra de busca (autocompletar e botão)
 */
function setupSearch() {
  // 1. Pega as referências dos elementos HTML
  const searchInput = document.getElementById("search-input");
  const searchResults = document.getElementById("search-results"); // O dropdown
  const searchButton = document.getElementById("search-button");

  // 2. PREPARA OS DADOS PARA A BUSCA
  // 'dadosDasRotas' é um objeto agrupado por rota (ex: dadosDasRotas.azul.pontos).
  // Para a busca, é mais fácil ter um array simples com TODOS os pontos.
  const searchableData = [];

  // Itera sobre as chaves do objeto (ex: 'azul', 'verde', 'amarela', 'vermelha')
  for (const key in dadosDasRotas) {
    const rota = dadosDasRotas[key]; // Pega o objeto da rota (ex:
dadosDasRotas.azul)
```

```
    // Itera sobre o array 'pontos' de cada rota
    rota.pontos.forEach((ponto) => {
      // Validação: Ignora pontos que não têm nome (título)
      if (ponto.title && ponto.title.trim() !== "") {
        // Adiciona um novo objeto ao array 'searchableData'
        searchableData.push({
          nome: ponto.title,  // O nome do ponto (ex: "ETEC")
          rota: rota,         // A referência para o objeto da rota inteira (para desenhar)
          data: ponto,        // A referência para o objeto do ponto (com lat/lng)
        });
      }
    });
  }
  // No final, 'searchableData' é um array [ {ponto1}, {ponto2}, {ponto3}, ... ]

  /**
   * Função auxiliar interna que executa a ação de busca (desenha a rota e centraliza no ponto)
   * @param {object} item - Um item do array 'searchableData' (como o {ponto1} acima)
   */
  function executarAcao(item) {
    desenharRota(item.rota); // Desenha a rota associada ao ponto (ex: Linha Azul)
    map.panTo({ lat: item.data.lat, lng: item.data.lng }); // Centraliza o mapa no ponto
    map.setZoom(17); // Dá zoom
    searchInput.value = item.nome; // Põe o nome completo do ponto no input
    searchResults.innerHTML = ""; // Limpa os resultados do dropdown
    searchResults.classList.add("hidden"); // Esconde o dropdown
    salvarBusca(item.nome); // Salva o termo buscado no histórico
  }

  // 3. LÓGICA DA BUSCA 'AO VIVO' (AUTOCOMPLETAR)
  // Adiciona um "escutador" do evento 'input' (dispara a cada tecla digitada)
  searchInput.addEventListener("input", () => {
```

```
    const termo = searchInput.value.toLowerCase(); // Pega o valor e põe em minúsculas

    // Não busca com menos de 2 letras (para economizar processamento)
    if (termo.length < 2) {
      searchResults.innerHTML = "";
      searchResults.classList.add("hidden");
      return;
    }

    // Filtra o array 'searchableData'
    // Mantém apenas os itens cujo 'nome' (em minúsculas) INCLUI o 'termo' digitado
    const resultadosFiltrados = searchableData.filter((item) =>
      item.nome.toLowerCase().includes(termo)
    );

    searchResults.innerHTML = ""; // Limpa resultados anteriores
    searchResults.classList.remove("hidden"); // Mostra o dropdown

    if (resultadosFiltrados.length > 0) {
      // Se houver resultados, cria um link '<a>' para cada um
      resultadosFiltrados.forEach((item) => {
        const resultItem = document.createElement("a"); // Cria um <a>
        resultItem.href = "#"; // Link "morto" (será parado pelo JS)
        resultItem.className = "block px-4 py-2 text-gray-800 hover:bg-blue-100"; // Estilos
        // Insere o HTML do item (ícone, nome do ponto, nome da rota)
        resultItem.innerHTML = `<i class="fas fa-map-marker-alt mr-2 text-red-500"></i> ${item.nome} <span class="text-xs text-gray-500">(${item.rota.nome})</span>`;

        // Adiciona um "escutador" de clique no item do resultado
        resultItem.addEventListener("click", (e) => {
          e.preventDefault(); // Impede o link '#' de pular a página
          executarAcao(item); // Executa a ação (desenha e centraliza)
```

```
        });
        // Adiciona o item <a> ao dropdown
        searchResults.appendChild(resultItem);
      });
    } else {
      // Se não houver resultados, mostra uma mensagem
      searchResults.innerHTML =
        '<div class="px-4 py-2 text-gray-500">Nenhum endereço encontrado.</div>';
    }
  });

  // 4. LÓGICA DO CLIQUE NO BOTÃO 'BUSCAR' (o da lupa)
  searchButton.addEventListener("click", () => {
    const termo = searchInput.value.toLowerCase();
    if (termo.length === 0) return; // Ignora clique se o campo estiver vazio

    // Filtra os dados (igual ao autocompletar)
    const resultadosFiltrados = searchableData.filter((item) =>
      item.nome.toLowerCase().includes(termo)
    );

    if (resultadosFiltrados.length > 0) {
      // Pega apenas o PRIMEIRO resultado (índice 0) e executa a ação
      executarAcao(resultadosFiltrados[0]);
    } else {
      // Se não achar nada, mostra mensagem de erro no dropdown
      searchResults.innerHTML = `<div class="px-4 py-2 text-gray-500">Nenhum
resultado para "${searchInput.value}".</div>`;
      searchResults.classList.remove("hidden"); // Garante que o dropdown apareça
    }
  });

  // 5. LÓGICA PARA FECHAR O DROPDOWN
  // Adiciona um "escutador" de clique no DOCUMENTO INTEIRO
```

```
  document.addEventListener("click", (e) => {
    // Se o local clicado (e.target) NÃO for o input E NÃO for o dropdown...
    if (!searchInput.contains(e.target) && !searchResults.contains(e.target)) {
      searchResults.classList.add("hidden"); // ...esconde o dropdown
    }
  });
} // Fim da função setupSearch
```

conexao.php

```
<?php
// Este é um script PHP puro. Não há HTML aqui.

// --- DEFINIÇÃO DAS CREDENCIAIS DE ACESSO ---
// Estas variáveis guardam as "chaves" para o banco de dados.

// 1. O servidor (host)
// 'localhost' (ou '127.0.0.1') significa que o banco de dados
// está rodando na MESMA máquina que o servidor web (o XAMPP).
$servidor = "localhost";

// 2. O usuário do banco
// 'root' é o usuário administrador padrão do MySQL no XAMPP/WAMP.
$usuario_db = "root";

// 3. A senha do banco
// Por padrão, o 'root' do XAMPP não tem senha (uma string vazia).
$senha_db = "";

// 4. O nome do banco de dados
// O nome exato do banco que você criou no phpMyAdmin (db_findbus).
$banco = "db_findbus";

// --- TENTATIVA DE CONEXÃO ---

// 'new mysqli(...)' é a forma orientada a objetos de iniciar uma conexão
```

```
// com o banco de dados MySQL no PHP.
// A variável '$conexao' agora é um objeto que representa esta conexão.
$conexao = new mysqli($servidor, $usuario_db, $senha_db, $banco);

// --- VERIFICAÇÃO DE ERRO ---
// É VITAL checar se a conexão funcionou antes de prosseguir.

// Se o objeto '$conexao' tiver a propriedade 'connect_error',
// significa que a conexão falhou (ex: MySQL desligado, senha errada, banco não existe).
if ($conexao->connect_error) {

    // 'die()' é uma função do PHP que faz duas coisas:
    // 1. Imprime na tela a mensagem de erro.
    // 2. Interrompe IMEDIATAMENTE a execução de todo o script PHP.
    //
    // '. $conexao->connect_error' concatena (junta) a nossa mensagem
    // com a mensagem de erro oficial do MySQL, para sabermos o motivo exato.
    // (Foi isso que gerou o erro "Target machine actively refused it").
    die("Falha na conexão com o banco de dados: " . $conexao->connect_error);
}

// --- CONFIGURAÇÃO DE CARACTERES (BOA PRÁTICA) ---

// 'set_charset("utf8")' diz ao MySQL que os dados que estamos enviando
// e recebendo devem usar o padrão UTF-8.
// Isso evita que nomes com acentos (ex: "São José", "Informações")
// sejam salvos ou lidos do banco com caracteres estranhos (ex: "S?o Jos?").
$conexao->set_charset("utf8");

// --- FIM DO SCRIPT ---

// Se o script chegou até aqui sem dar 'die()', a conexão foi um sucesso.
// A variável '$conexao' agora está "viva" e pronta para ser usada
```

```
// por qualquer outro arquivo que inclua este (como o 'api/buscar_dados.php').
?>
```

api/buscar_dados.php

```
<?php
// Este script PHP só retorna dados (em formato JSON).

// 1. DEFINIR O TIPO DE RESPOSTA
// header('Content-Type: application/json');
// Esta linha é crucial. Ela avisa ao navegador (e ao 'fetch' no script.js)
// que a resposta deste arquivo NÃO é HTML, mas sim texto JSON.
// Isso faz o 'fetch' entender que deve usar a função '.json()'.
header('Content-Type: application/json');

// 2. INCLUIR A CONEXÃO
// 'require_once' é mais seguro que 'include'.
// Ele "puxa" todo o código do 'conexao.php' para cá.
// Se o 'conexao.php' não for encontrado, o script para com um erro fatal.
// O '../' significa "subir um nível" na estrutura de pastas
// (da pasta 'api' para a pasta raiz, onde 'conexao.php' está).
require_once '../conexao.php';

// 3. INICIALIZAR O ARRAY DE DADOS
// '$dados_finais' será o array "pai" em PHP que vamos montar.
// Ele começará vazio e será preenchido com os dados do banco.
// No final, será convertido em JSON.
$dados_finais = [];

// 4. A CONSULTA ÚNICA (OTIMIZADA)
// Esta é a consulta SQL que busca TUDO de uma vez.
// Usamos 'AS' para renomear colunas (ex: 'r.nome' vira 'rota_nome')
// e evitar conflitos de nomes iguais (como 'motoristas.nome' e 'rotas.nome').
$sql = "SELECT
            r.chave,
```

```
            r.nome AS rota_nome,
            r.cor,
            o.codigo AS codigoOnibus,
            m.nome AS motorista,
            p.lat,
            p.lng,
            p.titulo AS title,
            rp.ordem
        FROM rotas r
        LEFT JOIN onibus o ON r.onibus_id = o.id
        LEFT JOIN motoristas m ON o.motorista_id = m.id
        LEFT JOIN rota_pontos rp ON r.id = rp.rota_id
        LEFT JOIN pontos p ON rp.ponto_id = p.id
        ORDER BY r.chave, rp.ordem ASC"; // ORDENAR É ESSENCIAL!
        // Ordenamos por 'chave' (azul, verde...) e depois por 'ordem' (0, 1, 2...).
        // Isso garante que todas as linhas da "rota azul" venham juntas
        // E que seus pontos venham na sequência correta.

// 5. EXECUTAR A CONSULTA
// '$conexao->query($sql)' envia a string SQL para o MySQL através
// da conexão que abrimos no 'conexao.php'.
// '$resultado' será um objeto com os dados retornados (ou 'false' se der erro).
$resultado = $conexao->query($sql);

// 6. PROCESSAR OS RESULTADOS EM PHP
// Verifica se a consulta funcionou E se retornou mais de 0 linhas.
if ($resultado && $resultado->num_rows > 0) {

    // 'while' faz um loop que passa por CADA LINHA que o SQL retornou, uma por uma.
    // '$resultado->fetch_assoc()' pega a linha atual e a transforma em um
    // array associativo (ex: $linha['chave'] será 'azul').
    while ($linha = $resultado->fetch_assoc()) {
```

```
        // Pega a chave da rota atual (ex: 'azul')
        $chave_rota = $linha['chave'];

        // Verifica se nós JÁ começamos a montar esta rota no nosso array '$dados_finais'.
        // 'isset' verifica se a chave 'azul' já existe em '$dados_finais'.
        // Na primeira vez que vemos a rota 'azul', '!isset' será VERDADEIRO.
        if (!isset($dados_finais[$chave_rota])) {

            // Se for a primeira vez, cria a "cabeça" da rota.
            // Isso só acontece uma vez por rota (uma vez para 'azul', uma para 'verde'...).
            $dados_finais[$chave_rota] = [
                'nome' => $linha['rota_nome'],
                'cor' => $linha['cor'],
                'motorista' => $linha['motorista'],
                'codigoOnibus' => $linha['codigoOnibus'],
                'pontos' => [] // Cria um array vazio para os pontos desta rota
            ];
        }

        // 7. ADICIONA O PONTO ATUAL NA ROTA
        // Esta parte do código roda para CADA linha (incluindo a primeira).

        // 'LEFT JOIN' pode trazer linhas nulas se uma rota não tiver pontos.
        // Esta verificação garante que só tentamos adicionar um ponto se
        // os dados de 'lat' (latitude) existirem (não forem nulos).
        if ($linha['lat'] !== null) {

            // Adiciona (push) um novo objeto de ponto no array 'pontos'
            // da rota que estamos processando (ex: $dados_finais['azul']['pontos']).
            $dados_finais[$chave_rota]['pontos'][] = [
                'lat' => (float) $linha['lat'], // (float) converte a string do SQL para um número decimal
                'lng' => (float) $linha['lng'], // (float) idem
```

```
            'title' => $linha['title']
        ];
    }
  } // Fim do loop 'while'
} // Fim do 'if ($resultado...)'

// 8. FECHA A CONEXÃO
// É uma boa prática fechar a conexão assim que não precisamos mais dela.
// Isso libera recursos no servidor MySQL.
$conexao->close();

// 9. RETORNA O JSON
// 'json_encode()' é a função do PHP que converte um array/objeto PHP
// (como o '$dados_finais' que montamos) em uma string de texto no formato JSON.
// 'echo' imprime essa string na tela.
// O 'fetch' no 'script.js' receberá esta string.
echo json_encode($dados_finais);
exit(); // Encerra o script PHP.
?>
```

css/style.css

```
/*
Este arquivo CSS é carregado no <head> do 'index.php'.
Ele contém estilos que são difíceis ou impossíveis de
fazer apenas com as classes do Tailwind CSS.
*/

/*
Este seletor '.map-container' tem como alvo qualquer elemento HTML
que tenha a classe "map-container".
No seu 'index.php', é a <div id="map" class="map-container">.
*/
.map-container {
  /* DEFINE A ALTURA DO MAPA */
  /* Esta é a regra mais importante. Por padrão, um <div> tem
```

```
    altura 0, a menos que tenha conteúdo. O Google Maps precisa
    que seu container TENHA uma altura definida para aparecer.
    Aqui, definimos 500 pixels. */
  height: 500px;

  /* Garante que o container ocupe 100% da largura da coluna
    onde ele está (no seu caso, a coluna 'lg:col-span-2'). */
  width: 100%;

  /* Adiciona bordas arredondadas (para combinar com o design
    dos outros cards brancos). */
  border-radius: 12px;

  /* Adiciona uma sombra sutil (também para combinar com
    os cards feitos com Tailwind). */
  box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1),
    0 2px 4px -1px rgba(0, 0, 0, 0.06);
}

/*
Este seletor estiliza a <div id="search-results">
(o dropdown de busca)
*/
.search-dropdown {
  /* Define uma altura máxima. Se houver muitos resultados de busca,
    o dropdown não ficará gigante na tela. */
  max-height: 300px;

  /* Adiciona uma barra de rolagem vertical ('auto')
    APENAS se o conteúdo ultrapassar os 300px de altura. */
  overflow-y: auto;
}
```

3.2 Tela de login e de cadastro

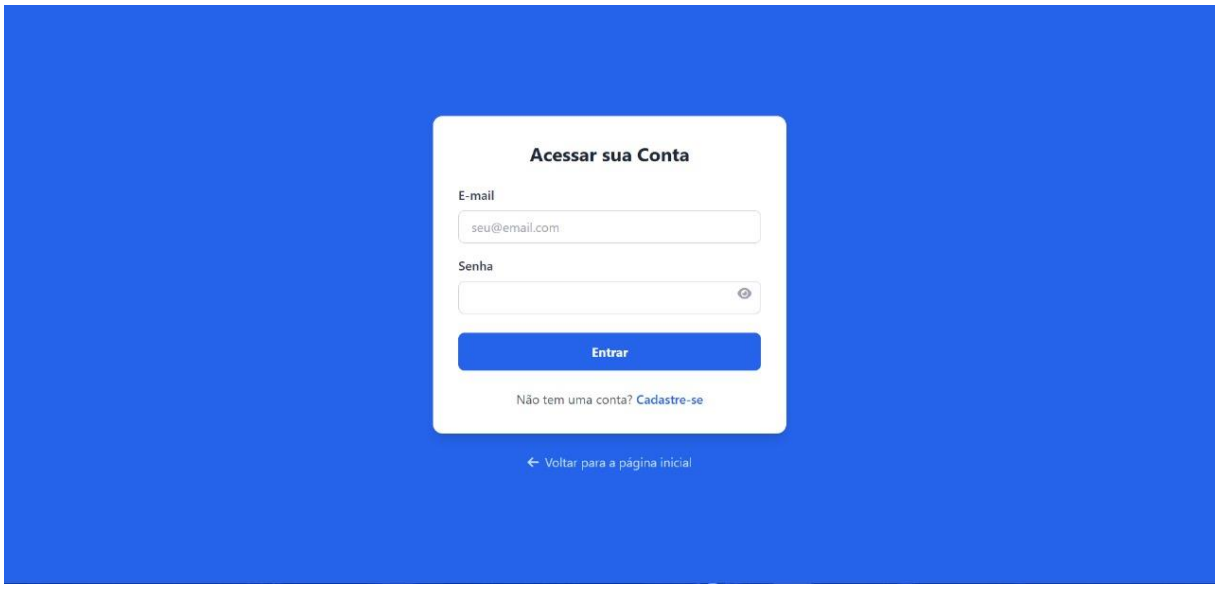

Figura 3: Tela de login do projeto FindBus.
FONTE: Elaborado pelos autores (2025).

login.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login / Cadastro - FindBus</title>

    <script src="https://cdn.tailwindcss.com"></script>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
    awesome/6.4.0/css/all.min.css">
</head>

<body class="bg-blue-600 min-h-screen flex flex-col items-center justify-center p-4">

    <div class="mb-8 text-center">
```

```
    <a href="index.php" class="flex items-center justify-center space-x-2 text-white hover:text-blue-200 transition-colors">
        <i class="fas fa-bus text-4xl"></i>
        <h1 class="text-3xl font-bold">FindBus</h1>
    </a>
</div>

<div class="max-w-md w-full bg-white rounded-xl shadow-lg p-8">

    <div id="form-login">
        <h2 class="text-2xl font-bold text-gray-800 mb-6 text-center">Acessar sua Conta</h2>

        <form action="login.php" method="POST">

            <div class="mb-4">
                <label for="email-login" class="block text-gray-700 font-medium mb-2">E-mail</label>
                <input type="email" id="email-login" name="email" placeholder="seu@email.com" required class="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">
            </div>

            <div class="mb-6 relative">
                <label for="senha-login" class="block text-gray-700 font-medium mb-2">Senha</label>
                <input type="password" id="senha-login" name="senha" placeholder="••••••••" required class="w-full px-4 py-2 pr-10 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">

                <i class="fas fa-eye absolute top-10 right-3 cursor-pointer text-gray-400" id="toggle-login"></i>
            </div>
```

```
            <button type="submit" class="w-full bg-blue-600 hover:bg-blue-700 text-white font-bold py-3 px-4 rounded-lg transition-colors">
                Entrar
            </button>
        </form>

        <p class="text-center text-gray-600 mt-6">
            Não tem uma conta?
            <a href="#" id="link-para-cadastro" class="text-blue-600 hover:underline font-medium">Cadastre-se</a>
        </p>
    </div>

    <div id="form-cadastro" class="hidden">
        <h2 class="text-2xl font-bold text-gray-800 mb-6 text-center">Crie sua Conta</h2>

        <form action="cadastro.php" method="POST">

            <div class="mb-4">
                <label for="nome-cadastro" class="block text-gray-700 font-medium mb-2">Nome Completo</label>
                <input type="text" id="nome-cadastro" name="nome" placeholder="Seu nome" required class="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">
            </div>

            <div class="mb-4">
                <label for="email-cadastro" class="block text-gray-700 font-medium mb-2">E-mail</label>
                <input type="email" id="email-cadastro" name="email" placeholder="seu@email.com" required class="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">
            </div>
```

```
            <div class="mb-6 relative">
                <label for="senha-cadastro" class="block text-gray-700 font-medium mb-2">Crie uma Senha</label>
                <input type="password" id="senha-cadastro" name="senha" placeholder="Mínimo 6 caracteres" required class="w-full px-4 py-2 pr-10 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">
                <i class="fas fa-eye absolute top-10 right-3 cursor-pointer text-gray-400" id="toggle-cadastro"></i>
            </div>

            <button type="submit" class="w-full bg-green-600 hover:bg-green-700 text-white font-bold py-3 px-4 rounded-lg transition-colors">
                Criar Conta
            </button>
        </form>

        <p class="text-center text-gray-600 mt-6">
            Já tem uma conta?
            <a href="#" id="link-para-login" class="text-blue-600 hover:underline font-medium">Faça Login</a>
        </p>
    </div>
</div>

<script>

    // --- LÓGICA DAS MENSAGENS DE STATUS (ERRO/SUCESSO) ---
    // Esta é uma "IIFE" (Immediately Invoked Function Expression).
    // É uma função que se auto-executa assim que a página carrega.
    (function() {
        // 1. Pega os parâmetros da URL (ex: ?status=erro_login)
        const urlParams = new URLSearchParams(window.location.search);
        const status = urlParams.get('status'); // Pega o valor de 'status'
```

```
        // Se não houver ?status=... na URL, a função para aqui.
        if (!status) {
            return;
        }

        // 2. Pega as referências dos formulários e seus títulos
        const formLogin = document.getElementById('form-login');
        const h2Login = formLogin.querySelector('h2');
        const formCadastro = document.getElementById('form-cadastro');
        const h2Cadastro = formCadastro.querySelector('h2');

        // 3. Cria o 'div' que mostrará a mensagem
        const mensagemDiv = document.createElement('div');
        // Estilos base do Tailwind para a mensagem
        mensagemDiv.className = 'p-4 rounded-lg mb-6 text-center text-sm font-
medium';

        // 4. Lógica para decidir qual mensagem mostrar

        if (status === 'cadastro_sucesso') {
            mensagemDiv.classList.add('bg-green-100', 'text-green-700');
            mensagemDiv.textContent = 'Cadastro realizado com sucesso! Faça o
login.';

            // Insere a mensagem DEPOIS do título (h2) do formulário de LOGIN
            if (h2Login) h2Login.after(mensagemDiv);

            // Garante que o formulário de LOGIN esteja visível
            formLogin.classList.remove('hidden');
            formCadastro.classList.add('hidden');

        } else if (status === 'erro_login') {
            mensagemDiv.classList.add('bg-red-100', 'text-red-700');
```

```
            mensagemDiv.textContent = 'E-mail ou senha inválidos. Tente novamente.';

            // Insere a mensagem no formulário de LOGIN
            if (h2Login) h2Login.after(mensagemDiv);

            // Garante que o formulário de LOGIN esteja visível
            formLogin.classList.remove('hidden');
            formCadastro.classList.add('hidden');

        } else if (status === 'erro_email_existente') {
            mensagemDiv.classList.add('bg-red-100', 'text-red-700');
            mensagemDiv.textContent = 'Este e-mail já está cadastrado. Tente outro ou faça login.';

            // Insere a mensagem no formulário de CADASTRO
            if (h2Cadastro) h2Cadastro.after(mensagemDiv);

            // Garante que o formulário de CADASTRO esteja visível
            formLogin.classList.add('hidden');
            formCadastro.classList.remove('hidden');
        }
    })(); // Os '()' no final fazem a função se auto-executar


    // --- LÓGICA PARA MOSTRAR/ESCONDER SENHA ---

    /**
     * Função reutilizável para configurar a funcionalidade de "mostrar senha".
     * @param {string} inputId - O ID do campo de senha (input)
     * @param {string} toggleId - O ID do ícone de olho (i)
     */
    function setupPasswordToggle(inputId, toggleId) {
        const passwordInput = document.getElementById(inputId);
        const toggleIcon = document.getElementById(toggleId);
```

```
        // Adiciona um "escutador" de clique no ícone de olho
        toggleIcon.addEventListener('click', function() {
            // 1. Verifica o tipo atual do input
            //    Se for 'password', muda para 'text'. Senão, muda para 'password'.
            const type = passwordInput.getAttribute('type') === 'password' ? 'text' : 'password';
            passwordInput.setAttribute('type', type);

            // 2. Alterna o ícone de olho (aberto/fechado) do Font Awesome
            this.classList.toggle('fa-eye');
            this.classList.toggle('fa-eye-slash');
        });
    }

    // --- ATIVAÇÃO DAS FUNÇÕES ---

    // Ativa a função de mostrar/esconder senha para o formulário de LOGIN
    setupPasswordToggle('senha-login', 'toggle-login');

    // Ativa a função de mostrar/esconder senha para o formulário de CADASTRO
    setupPasswordToggle('senha-cadastro', 'toggle-cadastro');


    // --- LÓGICA PARA ALTERNAR ENTRE FORMULÁRIOS ---

    // 1. Pega as referências dos links e dos 'divs' dos formulários
    const linkParaCadastro = document.getElementById('link-para-cadastro');
    const linkParaLogin = document.getElementById('link-para-login');
    const formLogin = document.getElementById('form-login');
    const formCadastro = document.getElementById('form-cadastro');

    // 2. Adiciona o "escutador" no link "Cadastre-se"
    if (linkParaCadastro) { // Verifica se o link existe
```

```
        linkParaCadastro.addEventListener('click', function(event) {
            event.preventDefault(); // Impede o link '#' de pular a página

            // Esconde o login e mostra o cadastro
            formLogin.classList.add('hidden');
            formCadastro.classList.remove('hidden');
        });
    }

    // 3. Adiciona o "escutador" no link "Faça Login"
    if (linkParaLogin) { // Verifica se o link existe
        linkParaLogin.addEventListener('click', function(event) {
            event.preventDefault(); // Impede o link '#' de pular a página

            // Esconde o cadastro e mostra o login
            formCadastro.classList.add('hidden');
            formLogin.classList.remove('hidden');
        });
    }
  </script>

</body>
</html>
```

login.php

```
<?php
// 1. INICIA A SESSÃO
// session_start() é OBRIGATÓRIO para que possamos
// criar ou acessar as variáveis $_SESSION.
session_start();

// 2. INCLUI A CONEXÃO
// Traz o código do 'conexao.php', que cria a variável $conexao
```

```
require_once 'conexao.php';

// 3. PEGA OS DADOS DO FORMULÁRIO (via POST)
// $_POST['email'] pega o valor do <input name="email">
$email = $_POST['email'];
// $_POST['senha'] pega o valor do <input name="senha">
$senha_pura = $_POST['senha']; // A senha que o usuário digitou

// 4. PREPARA A CONSULTA SQL (COM SEGURANÇA)
// Esta é a forma segura de fazer uma consulta, usando "Prepared Statements".
// O '?' é um placeholder (marcador de lugar).
// NUNCA coloque as variáveis ($email, $senha_pura) direto no SQL,
// pois isso causa vulnerabilidade a SQL Injection.
$stmt = $conexao->prepare("SELECT id, nome, senha FROM usuarios WHERE email
    = ?");

// 5. VINCULA OS PARÂMETROS
// 'bind_param' substitui os '?' pelos valores das variáveis de forma segura.
// "s" significa que a variável '$email' é uma String.
$stmt->bind_param("s", $email);

// 6. EXECUTA A CONSULTA
$stmt->execute();

// 7. PEGA O RESULTADO
// 'get_result' pega os dados que o banco de dados retornou.
$resultado = $stmt->get_result();

// 8. VERIFICA SE O USUÁRIO FOI ENCONTRADO
// '$resultado->num_rows' conta quantas linhas o banco retornou.
// Se for === 1, significa que encontramos EXATAMENTE um usuário com aquele e-
    mail.
if ($resultado->num_rows === 1) {
```

```
    // Usuário encontrado!
    // 'fetch_assoc' pega a linha do usuário e a transforma em um array associativo
    // $usuario agora será algo como: ['id' => 1, 'nome' => 'Felipe', 'senha' => '$2y...']
    $usuario = $resultado->fetch_assoc();

    // 9. VERIFICA A SENHA (A PARTE MAIS IMPORTANTE)
    // 'password_verify' é a função que compara a senha pura digitada ($senha_pura)
    // com o HASH criptografado que está salvo no banco ($usuario['senha']).
    // Ela NUNCA deve ser comparada com "==".
    if (password_verify($senha_pura, $usuario['senha'])) {

        // --- SUCESSO! Senha correta. ---

        // 10. SALVA OS DADOS NA SESSÃO
        // Agora o PHP "lembrará" desse usuário em todas as páginas.
        // O 'index.php' usará essas variáveis.
        $_SESSION['usuario_id'] = $usuario['id'];
        $_SESSION['usuario_nome'] = $usuario['nome'];

        // 11. REDIRECIONA PARA A PÁGINA PRINCIPAL
        // 'header()' envia um cabeçalho HTTP para o navegador,
        // mandando ele ir para 'index.php' (o mapa).
        header("Location: index.php");
        exit(); // 'exit()' é importante após um redirecionamento.
    }
}

// 12. FALHA NO LOGIN
// Se o script chegou até aqui, ou o usuário não foi encontrado (num_rows não era 1)
// ou a senha estava errada (password_verify falhou).
// Redireciona de volta para 'login.html' com uma mensagem de erro na URL.
header("Location: login.html?status=erro_login");
exit();
// (O '$stmt->close()' e '$conexao->close()' são tecnicamente bons,
```

```
// mas como o 'exit()' para o script, eles não são estritamente necessários aqui)
?>
```

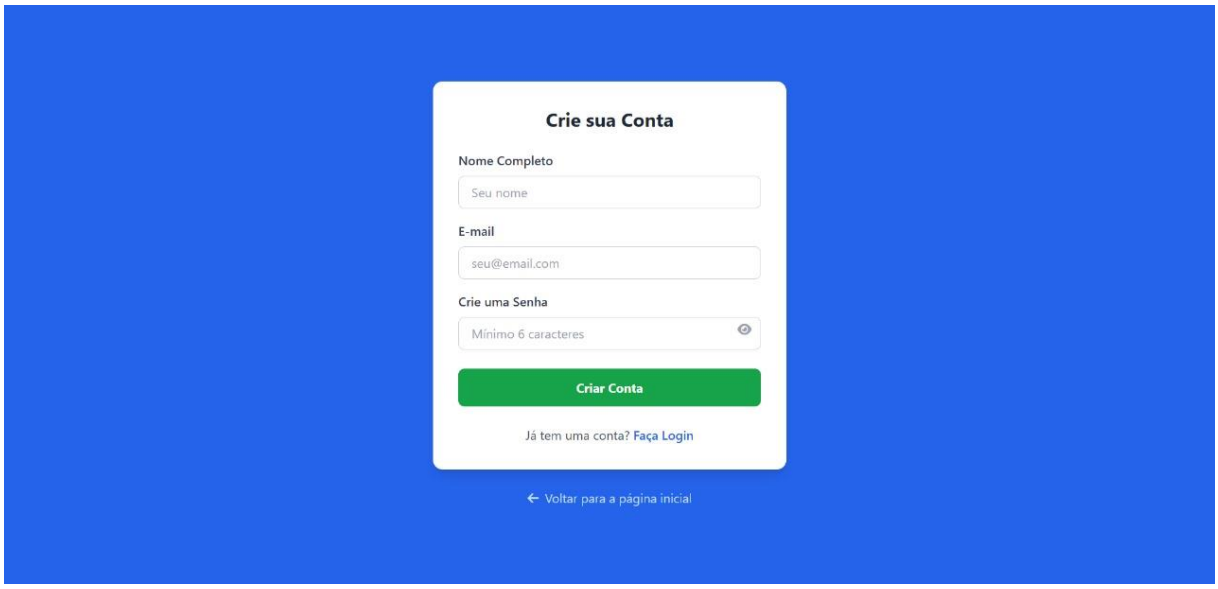

Figura 4: Tela de cadastro do projeto FindBus.
FONTE: Elaborado pelos autores (2025).

cadastro.php

```
<?php
// 1. INCLUI A CONEXÃO
// Traz o código do 'conexao.php', que cria a variável $conexao
require_once 'conexao.php';

// 2. PEGA OS DADOS DO FORMULÁRIO (via POST)
$nome = $_POST['nome'];
$email = $_POST['email'];
$senha_pura = $_POST['senha']; // A senha que o usuário digitou

// 3. VERIFICAÇÃO DE SEGURANÇA: O E-MAIL JÁ EXISTE?
// Esta é uma consulta crucial para evitar e-mails duplicados.
// O banco (UNIQUE INDEX) já nos protege, mas verificar antes
// permite enviar uma mensagem de erro amigável.
$stmt = $conexao->prepare("SELECT id FROM usuarios WHERE email = ?");
// "s" = String
$stmt->bind_param("s", $email);
```

```
$stmt->execute();
// 'store_result' é necessário para que possamos checar 'num_rows'
$stmt->store_result();

// 4. SE O E-MAIL JÁ EXISTIR, PARA O SCRIPT
if ($stmt->num_rows > 0) {
    // Se 'num_rows' for > 0, o e-mail já está em uso.
    // Redireciona de volta para 'login.html' com a mensagem de erro específica.
    header("Location: login.html?status=erro_email_existente");
    exit(); // Para o script aqui.
}

// 5. FECHA A PRIMEIRA CONSULTA
// É uma boa prática fechar o 'statement' anterior antes de abrir um novo.
$stmt->close();

// 6. CRIPTOGRAFA A SENHA (A PARTE MAIS IMPORTANTE)
// 'password_hash' é a função de segurança que transforma a senha
// (ex: "123456") em um hash longo e seguro (ex: "$2y$10...").
// NUNCA, JAMAIS, salve a '$senha_pura' no banco.
// 'PASSWORD_DEFAULT' usa o algoritmo mais forte disponível no PHP (atualmente,
    Bcrypt).
$senha_hash = password_hash($senha_pura, PASSWORD_DEFAULT);

// 7. PREPARA A NOVA CONSULTA (INSERIR USUÁRIO)
// Prepara o comando SQL 'INSERT' com placeholders (?)
$stmt = $conexao->prepare("INSERT INTO usuarios (nome, email, senha) VALUES
    (?, ?, ?)");
// "sss" = três parâmetros, todos do tipo String
$stmt->bind_param("sss", $nome, $email, $senha_hash); // Note que salvamos o
    HASH, não a senha pura

// 8. EXECUTA A INSERÇÃO E VERIFICA
if ($stmt->execute()) {
```

```
    // --- SUCESSO! Usuário cadastrado. ---

    // Redireciona de volta para 'login.html' com a mensagem de sucesso.
    // O 'login.html' (via JS) verá esse status e mostrará o formulário de LOGIN.
    header("Location: login.html?status=cadastro_sucesso");
    exit();
} else {
    // --- FALHA! ---
    // Se 'execute()' falhar (ex: erro inesperado no banco),
    // o script morre e mostra uma mensagem de erro técnica.
    die("Erro ao cadastrar o usuário: " . $stmt->error);
}

// 9. FECHA TUDO
$stmt->close();
$conexao->close();
?>
```

conexao.php

```
<?php
// Este é um script PHP puro. Não há HTML aqui.

// --- DEFINIÇÃO DAS CREDENCIAIS DE ACESSO ---
// Estas variáveis guardam as "chaves" para o banco de dados.

// 1. O servidor (host)
// 'localhost' (ou '127.0.0.1') significa que o banco de dados
// está rodando na MESMA máquina que o servidor web (o XAMPP).
$servidor = "localhost";

// 2. O usuário do banco
// 'root' é o usuário administrador padrão do MySQL no XAMPP/WAMP.
$usuario_db = "root";
```

```
// 3. A senha do banco
// Por padrão, o 'root' do XAMPP não tem senha (uma string vazia).
$senha_db = "";

// 4. O nome do banco de dados
// O nome exato do banco que você criou no phpMyAdmin (db_findbus).
$banco = "db_findbus";

// --- TENTATIVA DE CONEXÃO ---

// 'new mysqli(...)' é a forma orientada a objetos de iniciar uma conexão
// com o banco de dados MySQL no PHP.
// A variável '$conexao' agora é um objeto que representa esta conexão.
$conexao = new mysqli($servidor, $usuario_db, $senha_db, $banco);

// --- VERIFICAÇÃO DE ERRO ---
// É VITAL checar se a conexão funcionou antes de prosseguir.

// Se o objeto '$conexao' tiver a propriedade 'connect_error',
// significa que a conexão falhou (ex: MySQL desligado, senha errada, banco não
    existe).
if ($conexao->connect_error) {

    // 'die()' é uma função do PHP que faz duas coisas:
    // 1. Imprime na tela a mensagem de erro.
    // 2. Interrompe IMEDIATAMENTE a execução de todo o script PHP.
    //
    // '. $conexao->connect_error' concatena (junta) a nossa mensagem
    // com a mensagem de erro oficial do MySQL, para sabermos o motivo exato.
    // (Foi isso que gerou o erro "Target machine actively refused it").
    die("Falha na conexão com o banco de dados: " . $conexao->connect_error);
}
```

```
// --- CONFIGURAÇÃO DE CARACTERES (BOA PRÁTICA) ---

// 'set_charset("utf8")' diz ao MySQL que os dados que estamos enviando
// e recebendo devem usar o padrão UTF-8.
// Isso evita que nomes com acentos (ex: "São José", "Informações")
// sejam salvos ou lidos do banco com caracteres estranhos (ex: "S?o Jos?").
$conexao->set_charset("utf8");

// --- FIM DO SCRIPT ---

// Se o script chegou até aqui sem dar 'die()', a conexão foi um sucesso.
// A variável '$conexao' agora está "viva" e pronta para ser usada
// por qualquer outro arquivo que inclua este (como o 'login.php' e 'cadastro.php').
?>
```

3.4 Tela do perfil do usuário

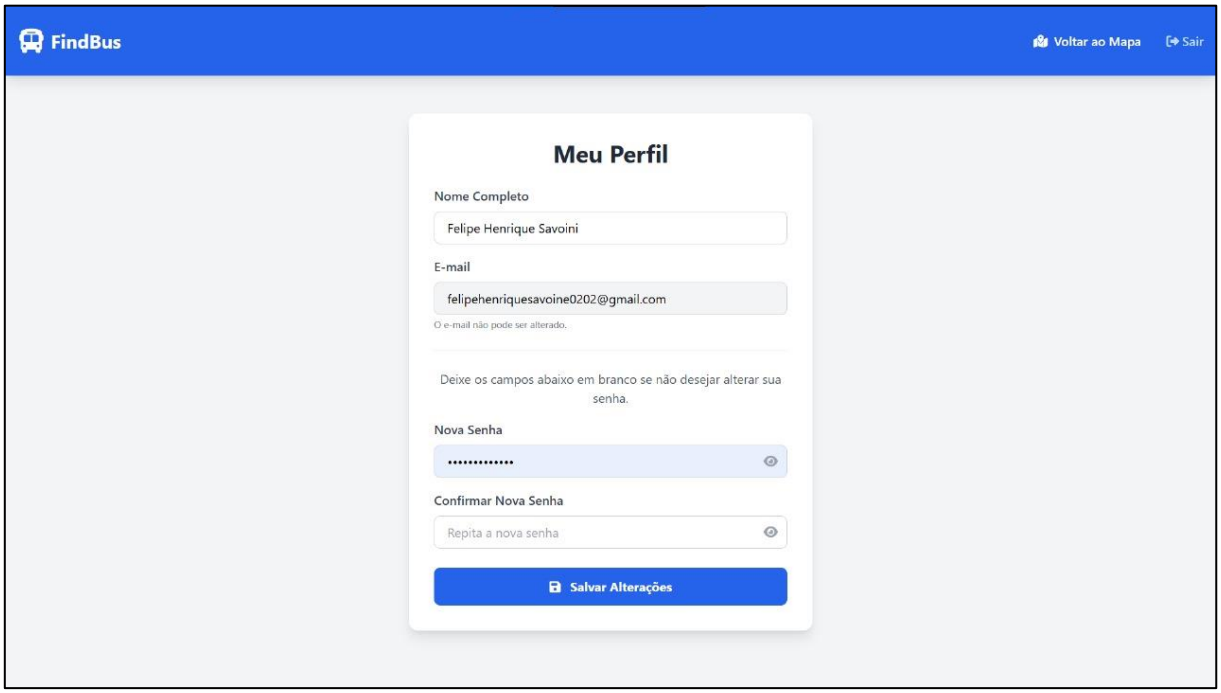

Figura 5: Tela do perfil do usuário do projeto FindBus.
FONTE: Elaborado pelos autores (2025).

A tela de perfil (perfil.php) é uma área restrita que demonstra o gerenciamento de sessão. O script inicia verificando a existência de $_SESSION['usuario_id'] ; se o usuário não estiver autenticado, ele é imediatamente redirecionado ao login.html.
Para usuários autenticados, a página busca os dados atuais do usuário no banco (nome e e-mail) e os utiliza para pré-preencher os campos do formulário . O campo de e-mail é definido como readonly, impedindo sua alteração.

O formulário permite a atualização do nome e, opcionalmente, da senha. O JavaScript local realiza uma validação *client-side*, verificando se as senhas digitadas coincidem e se possuem o comprimento mínimo antes de permitir o envio .

Ao submeter o formulário, os dados são enviados ao atualizar.php . Este script de backend verifica se o campo nova_senha foi preenchido. Caso positivo, ele atualiza o nome e a nova senha (criptografada com password_hash); caso contrário, atualiza apenas o nome. Após a atualização bem-sucedida, o script atualiza $_SESSION['usuario_nome'] (para que o cabeçalho do site reflita a mudança imediatamente) e redireciona de volta ao perfil.php com um status de sucesso. O botão "Sair" aponta para o logout.php , que destrói a sessão e redireciona o usuário ao index.php.

perfil.php

```
<?php
// 1. INICIAR A SESSÃO E VERIFICAR SE O USUÁRIO ESTÁ LOGADO
// session_start() é OBRIGATÓRIO para acessar as variáveis $_SESSION
session_start();
// Inclui o arquivo de conexão com o banco
require_once 'conexao.php';

// 2. VERIFICAÇÃO DE SEGURANÇA
// 'isset($_SESSION['usuario_id'])' verifica se o usuário está logado.
// O '!' inverte a lógica: "Se o usuário NÃO ESTIVER logado..."
if (!isset($_SESSION['usuario_id'])) {
    // ...redireciona-o para a página de login e para o script.
    header("Location: login.html");
    exit();
```

```
}

// 3. CARREGAR OS DADOS ATUAIS DO USUÁRIO
// Pega o ID do usuário que foi salvo na sessão durante o login
$usuario_id = $_SESSION['usuario_id'];
$usuario_atual = null; // Prepara uma variável para guardar os dados

// Prepara uma consulta SQL segura para buscar o nome e email do usuário
$stmt = $conexao->prepare("SELECT nome, email FROM usuarios WHERE id = ?");
// "i" = integer (inteiro). Substitui o '?' pelo $usuario_id
$stmt->bind_param("i", $usuario_id);
// Executa a consulta
$stmt->execute();
// Pega os resultados
$resultado = $stmt->get_result();

// Se o banco retornou exatamente 1 linha, armazena os dados
if ($resultado->num_rows === 1) {
    $usuario_atual = $resultado->fetch_assoc();
}
// Fecha a consulta (boa prática)
$stmt->close();

// 4. VERIFICAÇÃO DE INTEGRIDADE
// Se, por algum motivo, o $usuario_atual for nulo (ex: usuário deletado
// mas a sessão ainda existe), força o logout.
if (!$usuario_atual) {
    session_destroy(); // Destrói a sessão corrompida
    die("Erro crítico: usuário não encontrado. Por favor, faça login novamente.");
}

// 5. VERIFICAR MENSAGEM DE SUCESSO (da URL)
// Prepara uma variável vazia
$mensagem_sucesso = '';
```

```
// Verifica se a URL contém "?status=sucesso"
// (Isso acontece após o 'atualizar.php' redirecionar para cá)
if (isset($_GET['status']) && $_GET['status'] === 'sucesso') {
    $mensagem_sucesso = "Seus dados foram atualizados com sucesso!";
}

// O PHP ACABA AQUI. O HTML COMEÇA.
?>
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Meu Perfil - FindBus</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
</head>
<body class="bg-gray-100 min-h-screen">

    <header class="bg-blue-600 text-white shadow-lg">
        <div class="container mx-auto px-4 py-6">
            <div class="flex justify-between items-center">
                <a href="index.php" class="flex items-center space-x-2 hover:opacity-80 transition-opacity">
                    <i class="fas fa-bus text-3xl"></i>
                    <h1 class="text-2xl font-bold">FindBus</h1>
                </a>
                <div class="flex items-center space-x-6">
                    <a href="index.php" class="font-medium text-white p-2 rounded-lg hover:bg-blue-700 transition-colors">
                        <i class="fas fa-map-marked-alt mr-2"></i>Voltar ao Mapa
                    </a>
```

```
                <a href="logout.php" class="font-medium text-blue-200 hover:text-white
hover:underline transition-colors">
                    <i class="fas fa-sign-out-alt mr-1"></i>Sair
                </a>
            </div>
        </div>
    </header>

    <main class="container mx-auto px-4 py-12 flex justify-center">
        <div class="max-w-lg w-full bg-white rounded-xl shadow-lg p-8">
            <h2 class="text-3xl font-bold text-gray-800 mb-6 text-center">Meu Perfil</h2>

            <?php
            // Bloco PHP para exibir a mensagem de sucesso
            // Se a variável $mensagem_sucesso (definida lá em cima) NÃO estiver
vazia...
            if (!empty($mensagem_sucesso)):
            ?>
                <div class="bg-green-100 border-l-4 border-green-500 text-green-700 p-4
mb-6" role="alert">
                    <p class="font-bold">Sucesso!</p>
                    <p><?php echo $mensagem_sucesso; ?></p>
                </div>
            <?php
            // Fim do 'if'
            endif;
            ?>

            <form id="perfil-form" action="atualizar.php" method="POST">

                <div class="mb-4">
                    <label for="nome" class="block text-gray-700 font-medium mb-2">Nome
Completo</label>
```

```
            <input type="text" id="nome" name="nome" value="<?php echo
htmlspecialchars($usuario_atual['nome']); ?>" required class="w-full px-4 py-2 border
border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">
            </div>

            <div class="mb-4">
              <label for="email" class="block text-gray-700 font-medium mb-2">E-
mail</label>
              <input type="email" id="email" name="email" value="<?php echo
htmlspecialchars($usuario_atual['email']); ?>" readonly class="w-full px-4 py-2 border
border-gray-300 rounded-lg bg-gray-100 cursor-not-allowed">
              <p class="text-xs text-gray-500 mt-1">O e-mail não pode ser
alterado.</p>
            </div>

            <hr class="my-6">
            <p class="text-gray-600 text-center mb-4">Deixe os campos abaixo em
branco se não desejar alterar sua senha.</p>

            <div class="mb-4">
              <label for="nova_senha" class="block text-gray-700 font-medium mb-
2">Nova Senha</label>
              <div class="relative">
                <input type="password" id="nova_senha" name="nova_senha"
placeholder="Mínimo 6 caracteres" class="w-full px-4 py-2 pr-10 border border-gray-
300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">
                <i class="fas fa-eye absolute top-1/2 right-3 -translate-y-1/2 cursor-
pointer text-gray-400" id="toggle-nova-senha"></i>
              </div>
            </div>

            <div class="mb-6">
              <label for="confirmar_senha" class="block text-gray-700 font-medium
mb-2">Confirmar Nova Senha</label>
```

```
                <div class="relative">
                    <input type="password" id="confirmar_senha" name="confirmar_senha" placeholder="Repita a nova senha" class="w-full px-4 py-2 pr-10 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500">
                    <i class="fas fa-eye absolute top-1/2 right-3 -translate-y-1/2 cursor-pointer text-gray-400" id="toggle-confirmar-senha"></i>
                </div>
            </div>

            <div id="error-message" class="text-red-600 text-center font-medium mb-4"></div>

            <button type="submit" class="w-full bg-blue-600 hover:bg-blue-700 text-white font-bold py-3 px-4 rounded-lg transition-colors">
                <i class="fas fa-save mr-2"></i>
                Salvar Alterações
            </button>
        </form>
    </div>
</main>

<script>
    // --- LÓGICA DE VALIDAÇÃO DE SENHA (CLIENT-SIDE) ---
    // Pega as referências dos elementos do formulário
    const form = document.getElementById('perfil-form');
    const novaSenha = document.getElementById('nova_senha');
    const confirmarSenha = document.getElementById('confirmar_senha');
    const errorMessage = document.getElementById('error-message'); // A div de erro

    // Adiciona um "escutador" para o evento 'submit' (quando o usuário clica em "Salvar")
    form.addEventListener('submit', function(event) {
```

```
        // 1. Limpa qualquer mensagem de erro antiga
        errorMessage.textContent = '';

        // 2. Verifica se o usuário DIGITOU algo no campo "Nova Senha"
        if (novaSenha.value !== '') {

            // 3. Validação de Tamanho Mínimo
            if (novaSenha.value.length < 6) {
                // 'event.preventDefault()': ISSO É CRUCIAL.
                // Impede o formulário de ser enviado para o 'atualizar.php'.
                event.preventDefault();
                // Mostra a mensagem de erro
                errorMessage.textContent = 'A nova senha deve ter no mínimo 6 caracteres.';
                return; // Para a verificação
            }

            // 4. Validação de Correspondência
            if (novaSenha.value !== confirmarSenha.value) {
                // Impede o envio do formulário
                event.preventDefault();
                // Mostra a mensagem de erro
                errorMessage.textContent = 'As senhas não coincidem. Por favor, tente novamente.';
            }
        }
        // Se a senha estiver vazia, ou se as senhas baterem,
        // o 'event.preventDefault()' não é chamado e o formulário é enviado.
    });


    // --- LÓGICA PARA MOSTRAR/ESCONDER SENHA ---
    // Esta é a mesma função de 'login.html'
```

```
        /**
         * Função reutilizável para configurar a funcionalidade de "mostrar senha".
         * @param {string} inputId - O ID do campo de senha (input).
         * @param {string} toggleId - O ID do ícone de olho (i).
         */
        function setupPasswordToggle(inputId, toggleId) {
            const passwordInput = document.getElementById(inputId);
            const toggleIcon = document.getElementById(toggleId);

            // Garante que o ícone exista antes de adicionar o evento
            if (toggleIcon) {
                toggleIcon.addEventListener('click', function() {
                    // Alterna o tipo do input
                    const type = passwordInput.getAttribute('type') === 'password' ? 'text' :
'password';
                    passwordInput.setAttribute('type', type);

                    // Alterna o ícone (olho aberto/fechado)
                    this.classList.toggle('fa-eye');
                    this.classList.toggle('fa-eye-slash');
                });
            }
        }

        // Ativa a funcionalidade para os dois campos de senha da página
        setupPasswordToggle('nova_senha', 'toggle-nova-senha');
        setupPasswordToggle('confirmar_senha', 'toggle-confirmar-senha');
    </script>
</body>
</html>
```

atualizar.php

```
<?php
// 1. INICIA A SESSÃO
```

```
// OBRIGATÓRIO para acessar o $_SESSION['usuario_id'] e saber
// QUEM estamos atualizando.
session_start();
// Inclui o arquivo de conexão
require_once 'conexao.php';

// 2. VERIFICAÇÃO DE SEGURANÇA
// Se o usuário não estiver logado (ex: acessou este URL direto),
// ele é interrompido.
if (!isset($_SESSION['usuario_id'])) {
    die("Acesso negado. Por favor, faça login.");
}

// 3. PEGA OS DADOS (DA SESSÃO E DO FORMULÁRIO)
// Pega o ID da sessão (seguro, não vem do formulário)
$usuario_id = $_SESSION['usuario_id'];
// Pega os dados que vieram do formulário (método POST)
$novo_nome = $_POST['nome'];
$nova_senha_pura = $_POST['nova_senha'];

// 4. PREPARA A CONSULTA SQL (LÓGICA CONDICIONAL)
// Esta é a parte mais complexa:
// O usuário QUER mudar a senha ou só o nome?

// Verifica se o campo 'nova_senha' NÃO ESTÁ VAZIO
if (!empty($nova_senha_pura)) {

    // --- LÓGICA 1: ATUALIZAR NOME E SENHA ---

    // 1. Criptografa a nova senha (SEGURANÇA)
    $nova_senha_hash = password_hash($nova_senha_pura,
PASSWORD_DEFAULT);

    // 2. Prepara o SQL para atualizar AMBOS os campos (nome E senha)
```

```
    $stmt = $conexao->prepare("UPDATE usuarios SET nome = ?, senha = ? WHERE
id = ?");

    // 3. Vincula os parâmetros:
    // "ssi" = string (nome), string (senha_hash), integer (id)
    $stmt->bind_param("ssi", $novo_nome, $nova_senha_hash, $usuario_id);

} else {

    // --- LÓGICA 2: ATUALIZAR APENAS O NOME ---
    // (O usuário deixou o campo de senha em branco)

    // 1. Prepara o SQL para atualizar APENAS o nome
    $stmt = $conexao->prepare("UPDATE usuarios SET nome = ? WHERE id = ?");

    // 2. Vincula os parâmetros:
    // "si" = string (nome), integer (id)
    $stmt->bind_param("si", $novo_nome, $usuario_id);
}

// 5. EXECUTAR E VERIFICAR
// $stmt->execute() tenta rodar o SQL (seja o da Lógica 1 ou 2)
if ($stmt->execute()) {

    // --- SUCESSO! O banco foi atualizado. ---

    // 6. ATUALIZA A SESSÃO COM O NOVO NOME
    // ISSO É CRUCIAL!
    // Se não fizermos isso, o "Olá, [nome]" no 'index.php'
    // continuaria mostrando o nome antigo até o próximo login.
    $_SESSION['usuario_nome'] = $novo_nome;

    // 7. REDIRECIONAR DE VOLTA PARA O PERFIL
    // Redireciona o usuário de volta para 'perfil.php'
```

```
    // e adiciona "?status=sucesso" na URL. O 'perfil.php'
    // vai ler isso (como vimos no código dele) e mostrar a mensagem verde.
    header("Location: perfil.php?status=sucesso");
    exit();

} else {
    // --- FALHA! ---
    // Se o 'execute()' falhar, mostra um erro.
    die("Ocorreu um erro ao atualizar seu perfil: " . $stmt->error);
}

// 8. FECHA TUDO
$stmt->close();
$conexao->close();
?>
```

logout.php

```
<?php
// 1. INICIA A SESSÃO
// É impossível destruir uma sessão sem antes iniciá-la
// ou "se conectar" a ela.
session_start();

// 2. LIMPA A SESSÃO
// session_unset() remove todas as variáveis da sessão
// (ex: $_SESSION['usuario_id'] e $_SESSION['usuario_nome'] são apagadas).
session_unset();

// session_destroy() destrói o arquivo da sessão no servidor.
// Esta é a limpeza final e completa.
session_destroy();

// 3. REDIRECIONA O USUÁRIO
// Manda o usuário (agora deslogado) de volta para a página
// principal (o mapa).
```

```
header("Location: index.php");
exit(); // Encerra o script.
?>
```

conexao.php

```
<?php
// Este é um script PHP puro. Não há HTML aqui.

// --- DEFINIÇÃO DAS CREDENCIAIS DE ACESSO ---
// Estas variáveis guardam as "chaves" para o banco de dados.

// 1. O servidor (host)
// 'localhost' (ou '127.0.0.1') significa que o banco de dados
// está rodando na MESMA máquina que o servidor web (o XAMPP).
$servidor = "localhost";

// 2. O usuário do banco
// 'root' é o usuário administrador padrão do MySQL no XAMPP/WAMP.
$usuario_db = "root";

// 3. A senha do banco
// Por padrão, o 'root' do XAMPP não tem senha (uma string vazia).
$senha_db = "";

// 4. O nome do banco de dados
// O nome exato do banco que você criou no phpMyAdmin (db_findbus).
$banco = "db_findbus";

// --- TENTATIVA DE CONEXÃO ---

// 'new mysqli(...)' é a forma orientada a objetos de iniciar uma conexão
// com o banco de dados MySQL no PHP.
// A variável '$conexao' agora é um objeto que representa esta conexão.
$conexao = new mysqli($servidor, $usuario_db, $senha_db, $banco);
```

```
// --- VERIFICAÇÃO DE ERRO ---
// É VITAL checar se a conexão funcionou antes de prosseguir.

// Se o objeto '$conexao' tiver a propriedade 'connect_error',
// significa que a conexão falhou (ex: MySQL desligado, senha errada, banco não existe).
if ($conexao->connect_error) {

    // 'die()' é uma função do PHP que faz duas coisas:
    // 1. Imprime na tela a mensagem de erro.
    // 2. Interrompe IMEDIATAMENTE a execução de todo o script PHP.
    //
    // '. $conexao->connect_error' concatena (junta) a nossa mensagem
    // com a mensagem de erro oficial do MySQL, para sabermos o motivo exato.
    // (Foi isso que gerou o erro "Target machine actively refused it").
    die("Falha na conexão com o banco de dados: " . $conexao->connect_error);
}

// --- CONFIGURAÇÃO DE CARACTERES (BOA PRÁTICA) ---

// 'set_charset("utf8")' diz ao MySQL que os dados que estamos enviando
// e recebendo devem usar o padrão UTF-8.
// Isso evita que nomes com acentos (ex: "São José", "Informações")
// sejam salvos ou lidos do banco com caracteres estranhos (ex: "S?o Jos?").
$conexao->set_charset("utf8");

// --- FIM DO SCRIPT ---

// Se o script chegou até aqui sem dar 'die()', a conexão foi um sucesso.
// A variável '$conexao' agora está "viva" e pronta para ser usada
// por qualquer outro arquivo que inclua este (como o 'perfil.php' e 'atualizar.php').
?>
```

CONSIDERAÇÕES FINAIS

O presente trabalho foi motivado por uma dificuldade prática observada na comunidade discente da Etec Sylvio de Mattos Carvalho: a desinformação e a confusão dos alunos, especialmente dos ingressantes, em relação aos horários, pontos de parada e trajetos do transporte escolar. Diante deste cenário, o projeto FindBus foi proposto com o objetivo central de solucionar essa lacuna, desenvolvendo uma plataforma web que centralizasse essas informações de forma clara, confiável e acessível.

Ao final do desenvolvimento, considera-se que os objetivos específicos foram plenamente alcançados. A plataforma desenvolvida entrega um sistema funcional onde os alunos podem consultar os horários de cada rota, visualizar o traçado exato das paradas em um mapa interativo e localizar pontos de embarque específicos através de uma ferramenta de busca. A arquitetura escolhida, utilizando PHP e MySQL para o backend e JavaScript integrado à API do Google Maps para o frontend, provou-se robusta para a solução, permitindo também a implementação bem-sucedida de um sistema de autenticação e gerenciamento de perfil de usuário.

Por essa razão, o FindBus atinge sua proposta principal: oferecer uma ferramenta que confere agilidade, confiabilidade e autonomia aos estudantes no uso do transporte escolar. Ao substituir a desorganização informacional por uma interface digital e interativa, o projeto contribui diretamente para a inclusão digital e auxilia os alunos na tomada de decisões diárias sobre sua locomoção. Além disso, o desenvolvimento representou uma aplicação prática e concreta dos conhecimentos adquiridos no curso, transformando uma demanda real da comunidade escolar em uma solução tecnológica funcional.

Contudo, reconhecemos que o projeto, em seu estado atual, foca na disponibilização das rotas pré-definidas. Como sugestões para trabalhos futuros e aprimoramentos, vislumbra-se a implementação de um sistema de rastreamento GPS para a localização dos ônibus em tempo real. Adicionalmente, o desenvolvimento de um aplicativo móvel nativo (para Android e iOS) poderia ampliar significativamente o acesso e a conveniência. Por fim, sugere-se a criação de um painel administrativo dedicado, permitindo que a própria equipe escolar possa gerenciar e atualizar horários e rotas dinamicamente, sem a necessidade de intervenção técnica direta no banco de dados

REFERÊNCIAS

AMAZON WEB SERVICES. **O que é JavaScript?** [S.l.]: AWS, [S.d.]. Disponível em: https://aws.amazon.com/pt/what-is/javascript/. Acesso em: 8 out. 2025.

BRASIL. Lei nº 10.880, de 9 de junho de 2004. Institui o Programa Nacional de Apoio ao Transporte do Escolar - PNATE e o Programa de Apoio aos Sistemas de Ensino para Atendimento à Educação de Jovens e Adultos, dispõe sobre o repasse de recursos financeiros do Programa Brasil Alfabetizado, altera o art. 4º da Lei nº 9.424, de 24 de dezembro de 1996, e dá outras providências. **Diário Oficial da União.** Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/lei/l10.880.htm. Acesso feito em 29 abr. 2025.

BRASIL. Lei nº 9.394, de 20 de dezembro de 1996. Estabelece as diretrizes e bases da educação nacional. **Diário Oficial da União**. Disponível em: https://www.planalto.gov.br/ccivil_03/leis/l9394.htm. Acesso feito em 29 abr. 2025.

GEOAMBIENTE. **API de mapa: o que é e por que pode ser importante para o seu negócio?** [S.l.]: Geoambiente, [S.d.]. Disponível em: https://www.geoambiente.com.br/blog/api-de-mapa-o-que-e-e-por-que-pode-ser-importante-para-o-seu-negocio/. Acesso em: 8 out. 2025.

GRUPO VOITTO. **O que é CSS3? As 7 principais novidades!** [S.l.]: Grupo Voitto, [S.d.]. Disponível em: https://voitto.com.br/blog/artigo/o-que-e-css3. Acesso em: 8 out. 2025.

KINSTA. **O que é Tailwind CSS? Um guia para iniciantes sobre o framework CSS.** [S.l.]: Kinsta, [S.d.]. Disponível em: https://kinsta.com/pt/blog/tailwind-css/. Acesso em: 8 out. 2025.

ORACLE CORPORATION. **MySQL.** [S.l.]: Oracle, 2023. Disponível em: https://www.mysql.com/. Acesso feito em 23 out. 2025.

TASK. **O que é HTML5 e quais suas vantagens?** [S.l.]: Task, [S.d.]. Disponível em: https://www.task.com.br/blog/o-que-e-html5/. Acesso em: 8 out. 2025.

THE PHP GROUP. **O que é o PHP?** [S.l.]: The PHP Group, [S.d.]. Disponível em: https://www.php.net/manual/pt_BR/intro-whatis.php. Acesso em: 23 out. 2025.

APÊNDICE A: Diário de Bordo

Data: 24/03/2025
Assunto: Planejamento Inicial da Plataforma FindBus
Descrição das atividades desenvolvidas nesse dia: Avançamos no planejamento do FindBus, focado em entender nosso público-alvo (alunos da Etec com dúvidas sobre transporte). Definimos os objetivos principais (mapa interativo com rotas e horários) e organizamos um cronograma inicial. Começamos a preparar um slide de apresentação e um resumo do projeto.
Ferramentas utilizadas:
Microsoft Word
Microsoft PowerPoint

Data: 23/03/2025
Assunto: Planejamento Inicial da Plataforma FindBus
Descrição das Atividades Realizadas: Avançamos no planejamento do FindBus, focado em entender nosso público-alvo (alunos da Etec com dúvidas sobre transporte). Definimos os objetivos principais (mapa interativo com rotas e horários) e organizamos um cronograma inicial. Começamos a preparar um slide de apresentação e um resumo do projeto.
Fontes/Ferramentas Utilizadas:
Microsoft Word
Microsoft PowerPoint

Data: 28/03/2025
Assunto: Avanços no Protótipo Visual e Planejamento do Projeto
Descrição das Atividades Realizadas: Iniciamos a criação do protótipo (mockup) do nosso site, focando na parte visual e estrutura do design da index.php (mapa) e login.html (cadastro/login). Além disso, continuamos ajustando o slide de apresentação e o cronograma.
Fontes/Ferramentas Utilizadas:
Canva
Microsoft PowerPoint

Data: 04/04/2025
Assunto: Ajustes no protótipo e Identidade Visual
Descrição das Atividades Realizadas: No dia de hoje, continuamos com a criação das telas do protótipo (mockup) no Canva, focando na Tela Principal e na de Cadastro. Também revisamos e alteramos o logotipo do FindBus, melhorando a aparência e estrutura do nosso projeto.
Fontes/Ferramentas Utilizadas:
Canva

Data: 11/04/2025
Assunto: Criação do DER, reajustes no protótipo e Formulário.
Descrição das Atividades Realizadas: No dia de hoje, começamos a estruturar o DER do nosso projeto (definindo Usuario, Rota, Ponto, Motorista etc.) para que fosse possível iniciarmos o desenvolvimento do banco de dados. Também arrumamos alguns detalhes nas telas no protótipo. Modificamos o Formulário (questionário) para os alunos, para uma melhor compreensão.
Fontes/Ferramentas Utilizadas:
Lucidchart, Canva
Microsoft Word

Data: 25/04/2025
Assunto: Desenvolvimento do site e finalização do DER.
Descrição das Atividades Realizadas: Hoje finalizamos a estrutura do DER. Demos início ao desenvolvimento do site, criando a Tela Principal (index.php) e aplicando a estilização inicial com Tailwind CSS. Criamos o banco db_findbus no MySQL.
Fontes/Ferramentas Utilizadas:
Visual Studio Code
MySQL Workbench

Data: 02/05/2025
Assunto: Ajustes no slide de apresentação e desenvolvimento do site.
Descrição das Atividades Realizadas: Neste dia, continuamos com o desenvolvimento do site, ajustando as telas de login.html e perfil.php. Com a ajuda do nosso orientador, mudamos algumas informações no slide de apresentação para melhor compreensão.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (HTML, PHP)
Microsoft PowerPoint

Data: 09/05/2025
Assunto: Ajustes no slide de apresentação.
Descrição das Atividades Realizadas: No dia de hoje, com a ajuda do nosso orientador, fizemos alguns ajustes no slide de apresentação para a prévia.
Fontes/Ferramentas Utilizadas:
Microsoft PowerPoint

Data: 16/05/2025
Assunto: Reajustes no slide de apresentação
Descrição das Atividades Realizadas: Hoje fizemos os últimos reajustes no slide de apresentação do TCC do primeiro bimestre.
Fontes/Ferramentas Utilizadas:
Microsoft PowerPoint

Data: 23/05/2025
Assunto: Apresentação da Prévia do TCC (1º Bimestre)
Descrição das Atividades Realizadas: No dia de hoje, apresentamos nosso projeto para a banca de validação e recebemos feedbacks para melhorias do projeto.

Data: 30/05/2025
Assunto: Correção Slides e Escopo
Descrição das Atividades Realizadas: No dia de hoje, nós corrigimos os slides e melhoramos o escopo do nosso projeto (incluindo a nova relação Embarca_Em), com base no feedback recebido pela banca.
Fontes/Ferramentas Utilizadas:
Microsoft PowerPoint
Lucidchart

Data: 06/06/2025
Assunto: Desenvolvimento do site (Frontend)
Descrição das Atividades Realizadas: Neste dia começamos o desenvolvimento da parte visual do nosso site, avançamos com as telas de cadastro (login.html) e a página inicial (index.php), estruturando o HTML e o CSS (Tailwind).
Fontes/Ferramentas Utilizadas:
Visual Studio Code (HTML, CSS)
Tailwind CSS

Data: 13/06/2025
Assunto: Relatório técnico e desenvolvimento (Backend).
Descrição das Atividades Realizadas: No dia de hoje, demos continuidade com as telas de backend (cadastro.php e login.php). Começamos também a editar o relatório técnico com a parte da introdução.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (PHP)
MySQL Workbench
Microsoft Word

Data: 20/06/2025
Assunto: Relatório técnico e desenvolvimento.
Descrição das Atividades Realizadas: Finalizamos o desenvolvimento das páginas de cadastro e tela principal. Seguimos com a edição do relatório técnico com dados da introdução e metodologia.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (PHP, HTML)
Microsoft Word

Data: 27/06/2025
Assunto: Mudança no design, desenvolvimento e Relatório técnico.
Descrição das Atividades Realizadas: Fizemos algumas alterações no design (Tailwind). Criamos também a funcionalidade de "Histórico de Busca" (no script.js com localStorage). Além disso, continuamos com o incremento de dados no relatório técnico na parte de metodologia.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (JavaScript, CSS)
Microsoft Word

Data: 04/07/2025
Assunto: Relatório técnico e desenvolvimento.
Descrição das Atividades Realizadas: Demos sequência no desenvolvimento da tela principal (index.php), focando na parte visual da exibição das rotas no mapa. Também continuamos com a inserção de informações no relatório técnico, na parte da introdução.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (HTML, CSS)
Microsoft Word

Data: 11/07/2025
Assunto: Relatório técnico e críticas.
Descrição das Atividades Realizadas: No dia de hoje nosso professor orientador nos deu algumas críticas construtivas sobre o relatório técnico, da mesma forma, alteramos parte do texto de introdução e metodologia.
Fontes/Ferramentas Utilizadas:
Microsoft Word

Data: 18/07/2025
Assunto: Desenvolvimento e relatório técnico.
Descrição das Atividades Realizadas: Hoje demos início ao desenvolvimento da tela de perfil (perfil.php), apenas a parte visual. Também prosseguimos com o incremento de dados no relatório técnico, finalizando a parte de metodologia e começando a usar as ferramentas utilizadas.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (HTML, CSS)
Microsoft Word

Data: 25/07/2025
Assunto: Desenvolvimento e relatório técnico.
Descrição das Atividades Realizadas: Com a ajuda do professor orientador, ajustamos algumas partes do texto de metodologia. Terminamos o desenvolvimento da tela de perfil e começamos a integração do script.js com a API do Google Maps.
Fontes/Ferramentas Utilizadas: Visual Studio Code (HTML, JS)
Google Maps API
Microsoft Word

Data: 01/08/2025
Assunto: Desenvolvimento e relatório técnico.
Descrição das Atividades Realizadas: No dia de hoje, consultamos um profissional (orientador) para nos ajudar com o DER do nosso projeto (relação Embarca_Em) . Terminamos a integração básica do Google Maps e começamos a desenvolver a API api/buscar_dados.php.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (PHP, JS)
MySQL Workbench
Lucidchart

Data: 08/08/2025
Assunto: Relatório técnico e desenvolvimento.
Descrição das Atividades Realizadas: Prosseguimos com o desenvolvimento da API api/buscar_dados.php (lógica dos JOINs). Fizemos também alguns últimos ajustes no relatório técnico na parte do DER e ferramentas.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (PHP)
MySQL Workbench
Microsoft Word

Data: 15/08/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: No dia de hoje fizemos apenas o término das funções desenharRota() e setupRouteListeners() no script.js.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (JavaScript)

Data: 22/08/2025
Assunto: Tabulação do formulário.
Descrição das Atividades Realizadas: No dia de hoje, tabulamos os dados do formulário (questionário) que aplicamos aos alunos da Etec, para validar a necessidade do projeto.
Fontes/Ferramentas Utilizadas:
Microsoft Word

Data: 29/08/2025
Assunto: Desenvolvimento e relatório técnico.
Descrição das Atividades Realizadas: Hoje prosseguimos com os dados do relatório com a parte dos prints das telas prontas (index.php e login.html). Começamos a popular o banco de dados (inserindo as rotas, motoristas).
Fontes/Ferramentas Utilizadas:
Visual Studio Code
MySQL Workbench
Microsoft Word

Data: 05/09/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Hoje finalizamos a parte visual e funcional do script.js, incluindo a função de busca setupSearch e o historicoBusca.
Fontes/Ferramentas Utilizadas:
Visual Studio Code (JavaScript, CSS)

Data: 12/09/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Hoje entramos na reta final, finalizamos a estrutura final para popular o banco de dados db_findbus com todos os pontos (latitude/longitude) e rota_pontos.
Fontes/Ferramentas Utilizadas:
MySQL Workbench
Google Maps

Data: 24/09/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Neste dia, nós cadastramos os primeiros dados (rotas e pontos). Finalizamos as últimas telas (backend atualizar.php e frontend perfil.php).
Fontes/Ferramentas Utilizadas:
Visual Studio Code (PHP)
MySQL Workbench

Data: 03/10/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Neste dia, nós cadastramos mais dados (rotas, motoristas) e finalizamos a correção de bugs na programação (ex: login.php).
Fontes/Ferramentas Utilizadas:
MySQL Workbench
Visual Studio Code (PHP)

Data: 08/10/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Seguimos cadastrando mais dados (associação rota_pontos).
Fontes/Ferramentas Utilizadas:
MySQL
Workbench

Data: 09/10/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Nós seguimos cadastrando mais dados e mostramos o site e os documentos para o nosso professor orientador.
Fontes/Ferramentas Utilizadas:
Microsoft Word
Visual Studio Code
MySQL Workbench

Data: 10/10/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Seguimos cadastrando mais dados e revisamos os documentos do TCC, mudando os pontos pontuados pelo professor.
Fontes/Ferramentas Utilizadas:
Microsoft Word
Visual Studio Code
MySQL Workbench

Data: 17/10/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Começamos as correções do site (HTML/PHP), com base no feedback (ex: htmlspecialchars no perfil.php).
Fontes/Ferramentas Utilizadas:
Microsoft Word
Visual Studio Code

Data: 22/10/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Concluímos o código e começamos o processo de inserção dos códigos-fonte comentados no relatório. Mostramos o código pronto para o orientador.
Fontes/Ferramentas Utilizadas:
Visual Studio Code
Microsoft Word

Data: 30/10/2025
Assunto: Desenvolvimento.
Descrição das Atividades Realizadas: Mostramos o TCC finalizado (Relatório Técnico) para o professor orientador. Ajustamos o Sumário, Referências e "Palavras-chave".
Fontes/Ferramentas Utilizadas:
Microsoft Word
Visual Studio Code