UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS TECNOLOGICAS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

ONILDO SANTOS COELHO JUNIOR

Desenvolvimento de aplicação web para rastreamento de aeronaves

São Luís
2026

ONILDO SANTOS COELHO JUNIOR

Desenvolvimento de aplicação web para rastreamento de aeronaves

Monografia apresentado ao Curso de Engenharia de Computação, como parte dos requisitos necessários à obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Pedro Brandão

São Luís
2026

Coelho Junior, Onildo Santos.

Desenvolvimento de aplicação web para rastreamento de aeronaves. / Onildo Santos Coelho Junior. - São Luís - MA, 2026.
57f.

Trabalho de Conclusão de Curso (Curso de Engenharia de Computação Bacharelado) - Universidade Estadual do Maranhão, 2026.

Orientador: Prof. Me.Pedro Brandão Neto.

1. Desenvolvimento. 2. ReactJS. 3. Rastreamento. I. Título.

CDU:629.73:004

Elaborado por Luciana de Araújo - CRB 13/445

ONILDO SANTOS COELHO JUNIOR

DESENVOLVIMENTO DE APLICAÇÃO WEB PARA RASTREAMENTO DE AERONAVES

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia de Computação na Universidade Estadual do Maranhão – UEMA para obtenção do grau de Bacharel em Engenharia de Computação.

Aprovado em: / /

BANCA EXAMINADORA

gov.br
Documento assinado digitalmente
PEDRO BRANDAO NETO
Data: 20/03/2026 16:27:15-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Pedro Brandão Neto (Orientador)

Mestre em Engenharia Elétrica

Universidade Estadual do Maranhão

Prof. Dr. Reinaldo de Jesus da Silva

Doutor em Informática na Educação

Universidade Estadual do Maranhão

Prof. Dr. Luís Carlos Costa Fonseca

Doutor em Informática na Educação

Universidade Estadual do Maranhão

Agradecimentos

Agradeço a Deus pela força e pela sabedoria que me permitiram concluir mais esta etapa da minha vida.

Aos meus pais, pelo incentivo constante e por serem inspiração em minha caminhada.

Aos meus familiares, pelo apoio e suporte durante toda a trajetória acadêmica.

Aos meus amigos da UEMA, que estiveram presentes nos bons e maus momentos; em especial a Flávio Campos, Guilherme Oliveira e Lucas Vieira, pelos aprendizados, pelas risadas e pelas conquistas compartilhadas.

Aos meus professores, pelos ensinamentos e pela contribuição para minha formação; em especial ao meu orientador, Prof. Me. Pedro Brandão Neto, pela paciência, pelo direcionamento e pela orientação ao longo do desenvolvimento deste trabalho.

Resumo

Este trabalho apresenta o desenvolvimento de uma aplicação web destinada ao acompanhamento e rastreamento de aeronaves do Centro Tático Aéreo da Polícia Militar do Maranhão. A solução foi construída utilizando ReactJS e integra-se a uma API REST para exibir, em tempo real, informações de posição, trajetória e dados operacionais das aeronaves durante as missões. Além do rastreamento, a aplicação permite consultar o histórico de voos e realizar o gerenciamento de missões, aeronaves e usuários por meio de uma interface intuitiva e responsiva. Os testes realizados demonstraram que a ferramenta apresenta desempenho estável, atualização contínua das informações e boa adequação às necessidades operacionais do CTA, evidenciando o potencial do desenvolvimento web como apoio às atividades aéreas.

Palavras-chave: Desenvolvimento. ReactJS. Rastreamento.

Abstract

This work presents the development of a web application designed for the monitoring and tracking of aircraft operated by the Tactical Air Center of the Military Police of Maranhão. The solution was built using ReactJS and integrates with a REST API to display real-time information on aircraft position, trajectory and operational data during missions. In addition to tracking, the application provides access to flight history and enables the management of missions, aircraft and users through an intuitive and responsive interface. The tests conducted demonstrated that the tool delivers stable performance, continuous data updates and good adherence to the operational needs of the CTA, highlighting the potential of web development as support for aerial operations.

Keywords: Development. ReactJS. Tracking.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
ADS-B	Automatic Dependent Surveillance-Broadcast
API	Application Programming Interface
ASP.NET	Active Server Pages .NET
COVID-19	Coronavirus Diease - 19
CSS	Cascading Style Sheets
CTA	Centro Técnico de Aeronáutica
DOM	Document Object Model
GET	O método que solicita uma representação do recurso especificado.
GPS	Global Positioning System
HTML	HyperText Markup Language
HTML5	HyperText Markup Language version 5
HTTP	HyperText Transport Protocol
IETF	Internet Engineering Task Force
JS	JavaScript (Linguagem de programação interpretada.)
JSON	Notação de Objetos JavaScript (do inglês, JavaScript Object Notation)
MDN	Mozilla Developer Network
OSM	Open Street Map
PATCH	Método que atualiza representação do recurso especificado e atualiza dados
PHP	PHP: Hypertext Preprocessor
POST	Protocolo de Transferência de Estado (do inglês Protocol of State Transfer)
REST	Representação de estado de transferência do acrônimo em inglês Representational State Transfer
SPA	Single Page Application
UI	Interface com usuário do acrônimo em inglês User Interface
URL	Uniform Resource Locator
W3C	World Wide Web Consortioum, entidade que regulamenta os

	padrões de internet
WWW	World Wide Web
XML	EXtensible Markup Language

Sumário

1 Introdução

No contexto atual de transformação digital, as ferramentas de software desempenham um papel essencial no suporte às organizações públicas e privadas, otimizando processos, promovendo eficiência operacional e fornecendo recursos para tomada de decisão. A administração pública, por sua vez, tem adotado progressivamente soluções digitais para ampliar a transparência, a eficiência e a segurança de seus serviços (VIANA, 2021).

O Centro Tático Aéreo (CTA), grupamento integrante da Secretaria de Estado de Segurança Pública do Maranhão, é uma unidade especializada que realiza operações aéreas de segurança pública, salvamento e transporte aeromédico. Com bases operacionais distribuídas estrategicamente nas cidades de São Luís, Imperatriz e Presidente Dutra, o CTA emprega aeronaves para atender a ocorrências em todo o território maranhense, inclusive em regiões de difícil acesso. Suas atividades incluem patrulhamento aéreo, transporte de autoridades, apoio em situações de defesa civil, entre outras. Durante a pandemia da COVID-19, por exemplo, o CTA teve papel ativo no transporte de pacientes, materiais biológicos e equipes técnicas.

Apesar de sua importância estratégica e operacional, o CTA não dispõe de um sistema próprio que permita o registro preciso, em tempo real, da posição e do desempenho de suas aeronaves durante as missões. Atualmente, o controle de localização depende predominantemente da comunicação via rádio com as equipes em terra, limitando a rastreabilidade e a capacidade de planejamento das operações. Além disso, soluções comerciais de rastreamento e gerenciamento de voos, como sistemas baseados em ADS-B (Automatic Dependent Surveillance–Broadcast), apresentam custos elevados e não estão plenamente adaptadas à realidade orçamentária e operacional do grupamento.

Visando suprir essa lacuna, este trabalho propõe o desenvolvimento de uma aplicação web de gerenciamento e visualização de dados de voo, com ênfase em rastreamento, controle de missão e interface intuitiva, acessível via navegadores modernos. A aplicação, pensada para ser de baixo custo e com regras de negócio ajustadas às necessidades específicas do CTA, utiliza

tecnologias atuais de desenvolvimento front-end, priorizando responsividade, usabilidade e integração com fontes de dados provenientes de equipamentos embarcados nas aeronaves.

Assim, o projeto visa não apenas atender a uma demanda real do CTA, mas também demonstrar a aplicação prática dos conhecimentos adquiridos ao longo da graduação em Engenharia da Computação, com destaque para o desenvolvimento de interfaces web modernas e eficientes no contexto da segurança pública.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma aplicação web de baixo custo voltada ao Centro Tático Aéreo (CTA), com funcionalidades de gerenciamento de missões, histórico de voos e rastreamento de aeronaves, utilizando dados obtidos a partir de uma API. Visando maior controle, organização e eficiência das operações aéreas.

1.1.2 Objetivos Específicos

Projetar uma interface web responsiva, intuitiva e adequada às necessidades operacionais do CTA;

Permitir o cadastro e a visualização de missões com dados como datas, locais, pessoas envolvidas e tempo de duração;

Exibir o histórico de voos realizados pelas aeronaves, organizando as informações de forma acessível;

Implementar um sistema de rastreamento baseado em dados de coordenadas enviados por uma API externa;

Garantir controle de acesso por meio de autenticação para usuários credenciados;

Utilizar tecnologias modernas de desenvolvimento front-end para garantir desempenho e usabilidade.

1.2 Metodologia

O desenvolvimento da aplicação proposta foi realizado utilizando tecnologias modernas da área de desenvolvimento web, com foco exclusivo na camada de front-end. O framework escolhido foi o React, uma biblioteca JavaScript amplamente utilizada para a construção de interfaces de usuário dinâmicas e responsivas. Além disso, foi utilizada a biblioteca OpenStreetMap para exibição de mapas e dados geográficos, viabilizando a visualização em tempo real das posições das aeronaves.

A aplicação foi estruturada de forma modular, com componentes reutilizáveis, garantindo organização, escalabilidade e facilidade de manutenção. O sistema consome dados a partir de uma API REST, responsável por fornecer informações como voos, missões, localização e dados dos usuários. Para permitir a correta interpretação dos dados do lado do cliente, foram definidas regras de negócio que associam cada posição geográfica recebida a uma missão e a um operador identificado por meio do login no sistema.

O método de desenvolvimento adotado seguiu os princípios do desenvolvimento prototipado, que, conforme PRESSMAN (2018), consiste na criação de versões iniciais da aplicação com funcionalidades parciais, permitindo validação contínua, refinamento de requisitos e coleta de feedback ao longo do processo. Essa abordagem se mostrou eficaz para garantir o alinhamento entre as necessidades operacionais do Centro Tático Aéreo (CTA) e as funcionalidades implementadas no sistema.

Com base nas operações reais do CTA, foram definidos dois fluxos principais de uso da aplicação:

- **Fluxo planejado**: o usuário informa previamente os dados da missão, a identificação da aeronave e os participantes envolvidos. Em seguida, o sistema registra e organiza os dados de rastreamento conforme a missão ativa.
- **Fluxo emergencial**: utilizado em situações onde não é possível realizar o registro completo da missão com antecedência. Nesse fluxo, o operador inicia imediatamente o rastreamento vinculando apenas a aeronave em uso, com posterior complementação das informações.

Essa dualidade de uso proporciona flexibilidade à aplicação, tornando-a adequada tanto para contextos operacionais estruturados quanto para ocorrências urgentes, comuns nas atividades desempenhadas pelo CTA. A interface do sistema foi projetada com foco em usabilidade, acessibilidade e compatibilidade com diferentes dispositivos, permitindo que operadores possam acessá-la de qualquer local com conexão à internet e navegador web.

1.3 Estrutura do Trabalho

Este trabalho está organizado em quatro capítulos. O primeiro apresenta a introdução ao tema, contextualizando o problema enfrentado pelo Centro Tático Aéreo (CTA), a justificativa do projeto e os objetivos da aplicação desenvolvida.

O segundo capítulo trata da fundamentação teórica, abordando os principais conceitos e tecnologias utilizados, como React, consumo de APIs REST, uso do OpenStreetMap e princípios de usabilidade no desenvolvimento web.

No terceiro capítulo, são descritas as etapas de desenvolvimento da aplicação, incluindo a arquitetura adotada, a estrutura dos componentes, os fluxos de uso definidos e a integração com a API de dados.

Por fim, o quarto capítulo apresenta os resultados obtidos, discute os benefícios da aplicação para o CTA, aponta limitações encontradas e propõe possíveis melhorias para versões futuras do sistema.

2 Fundamentação Teórica

2.1 Desenvolvimento Web Front-End

De acordo com Laudon (2014) Sistemas web são conjuntos de componentes de software em que estão conectados de forma flexível, permitindo a troca de informações entre si através de linguagens e padrões de comunicação web. Esses serviços, conhecidos como Web services, possibilitam a comunicação entre diferentes sistemas, independentemente dos sistemas operacionais ou linguagens de programação em que foram desenvolvidos.

Os Web services têm um papel fundamental na integração de sistemas web, pois permitem que aplicativos e serviços se comuniquem de maneira padronizada e interoperável. Isso significa que diferentes sistemas podem se comunicar entre si de forma eficiente, sem a necessidade de adaptações complexas e demoradas.

Uma das grandes vantagens dos Web services é a sua capacidade de promover a integração entre sistemas de diferentes organizações. Eles podem ser utilizados para construir aplicativos baseados em padrões abertos que conectam sistemas de duas empresas diferentes. Além disso, os Web services também possibilitam a criação de aplicativos que interligam sistemas diversos dentro de uma única empresa, promovendo uma maior colaboração e eficiência entre os departamentos.

Os sistemas WEB são desenvolvidos utilizando os mais diversos tipos de tecnologias, incluindo linguagens de programação, banco de dados, protocolos de rede e frameworks. Dentre essas tecnologias existem 3 que são as principais no desenvolvimento de tais sistemas Hyper Text Markup Language (HTML), utilizado para estruturação do conteúdo das páginas web através de tags, Cascading Style Sheets (CSS), enquanto o HTML é usado para definir a estrutura do conteúdo o CSS é responsável pelo estilo do conteúdo definindo cores, posições e tamanhos e por último o JavaScript (JS).

O desenvolvimento web front-end é a área da engenharia de software voltada para a criação da interface visual das aplicações acessadas por meio de navegadores. Essa camada é responsável por garantir a interação

do usuário com os dados e serviços disponibilizados pela aplicação, sendo fundamental para a experiência de uso e a efetividade do sistema.

A Web, como conhecemos, é baseada na arquitetura cliente-servidor. O cliente — geralmente um navegador — realiza requisições a servidores utilizando o protocolo HTTP (*Hypertext Transfer Protocol*), que é o padrão para a comunicação entre sistemas distribuídos na Internet. O HTTP permite o envio e recebimento de dados entre cliente e servidor, geralmente utilizando os métodos GET, POST, PUT e DELETE, muito comuns em aplicações que consomem APIs REST.

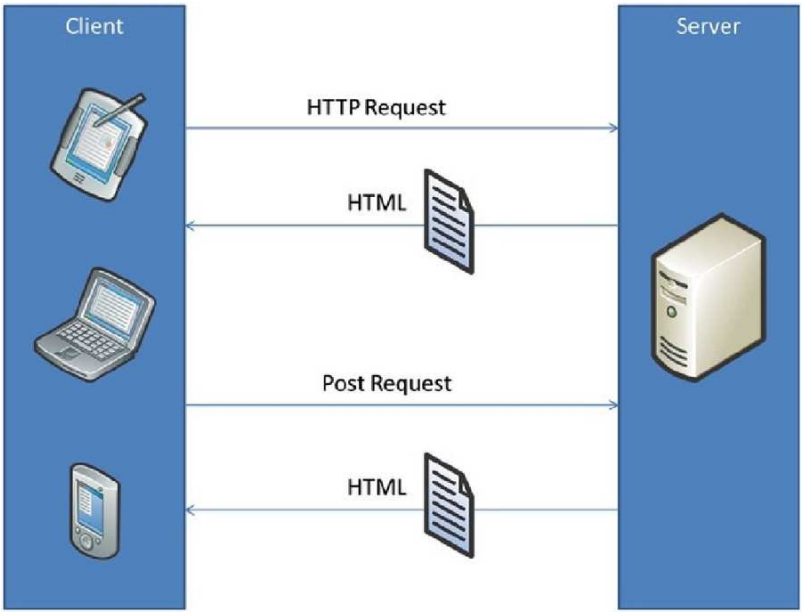
Com o avanço das tecnologias web, a comunicação assíncrona entre o front-end e o back-end se tornou essencial. Ferramentas como **Axios** (utilizada neste projeto) simplificam o consumo de dados por meio de requisições HTTP, promovendo maior dinamismo nas páginas web sem a necessidade de recarregamento completo.

2.2 Single Page Application

Segundo Flink e Flatow (2014) as aplicações Single Page Applications (SPAs) são uma abordagem moderna no desenvolvimento de aplicações web que proporcionam uma experiência mais interativa e fluida aos usuários. Ao contrário das aplicações web tradicionais, que envolvem o carregamento completo de páginas em cada interação do usuário, as SPAs consistem em apenas uma página principal que atua como um esqueleto para todas as outras páginas do aplicativo.

A principal característica das SPAs é o uso extensivo de tecnologias front-end, como JavaScript, HTML5 e CSS, para gerenciar as interações do usuário e atualizar o conteúdo da página dinamicamente. Isso permite que as ações do usuário sejam processadas rapidamente sem a necessidade de postbacks para o servidor ou recarregamento completo da página.

Figura 1 – Ciclo de vida de uma página web.



Fonte: FINK; FLATOW (2014)

Para permitir a navegação dentro da SPA, são utilizados mecanismos de roteamento e templating. O roteamento controla qual conteúdo será exibido com base na interação do usuário, enquanto o templating insere dinamicamente o conteúdo nas áreas designadas da página principal. Dessa maneira é passado a sensação de múltiplas páginas sem a necessidade de carregar novas páginas.

Outra vantagem das SPAs é a melhoria no desempenho. Como as SPAs transferem apenas os dados necessários para atualizar a página, ao invés de carregar a página completa, a quantidade de dados transferidos entre o cliente e o servidor é reduzida, resultando em um carregamento mais rápido e uma experiência mais eficiente para o usuário.

2.3 HTTP

Consoante a Internet Engineering Task Force (IETF) (1999) o Hypertext Transfer Protocol (HTTP) é um protocolo de nível de aplicação projetado para sistemas de informação distribuídos e colaborativos de hipermídia. É amplamente utilizado na comunicação entre clientes e servidores, permitindo o acesso e transferência de recursos, como páginas web, imagens, vídeos e

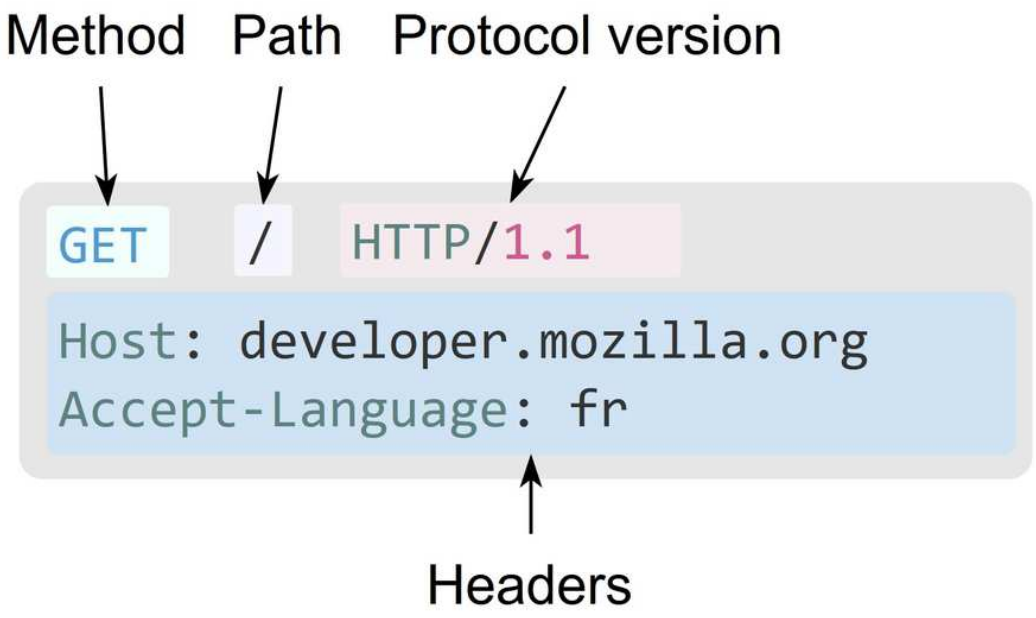
outros tipos de dados.

O HTTP é um protocolo genérico e sem estado, o que significa que cada requisição feita ao servidor é tratada de forma independente, sem conhecimento de requisições anteriores. Isso torna o protocolo leve e eficiente para transferência de informações.

Uma característica essencial do HTTP é a negociação e tipagem de representações de dados, permitindo que sistemas construídos independentemente possam interagir e trocar informações de maneira flexível. Isso significa que diferentes dispositivos e plataformas podem entender e processar os dados transferidos. Isso torna o protocolo adequado para uma ampla variedade de aplicações além de páginas web, como aplicações em dispositivos moveis e sistemas de gerenciamento de objetos distribuídos.

O HTTP é usado na internet para permitir a comunicação entre os clientes e os servidores. Ele define vários métodos de requisição que indicam a ação desejada a ser realizada em um recurso específico.

Figura 2 – Representação de uma requisição HTTP.



Autor: MOZILLA(2025)

De acordo com a MOZILLA (2025) esses métodos são chamados de verbos HTTP. Alguns dos principais métodos são GET, POST, PUT, DELETE e PATCH.

O método GET é usado para solicitar dados de um recurso no servidor. Por exemplo, quando você digita um endereço de site no navegador e pressiona Enter, ele faz uma requisição GET para obter a página web desse site.

Já o método POST é usado para enviar dados para serem processados por um recurso específico no servidor. Por exemplo, quando você preenche um formulário em um site e clica em “Enviar”, ele usa uma requisição POST para enviar os dados do formulário para o servidor.

O método PUT é usado para atualizar ou criar um recurso no servidor. Se você quiser modificar um recurso existente ou criar um novo, pode usar o método PUT.

Para excluir um recurso, utiliza-se o método DELETE. Ele é usado quando você quer remover completamente um item ou arquivo do servidor.

Por fim, o método PATCH é usado para fazer modificações parciais em um recurso. Ou seja, quando você quer fazer apenas algumas alterações em um item, pode usar o PATCH em vez de substituí-lo completamente com o PUT.

Tabela 1 – Tipos de códigos de status HTTP

Código	Status	Descrição
100	Continue	Essa resposta provisória indica que o cliente deve continuar a solicitação.
200	OK	A solicitação foi bem-sucedida.
301	Moved Permanently	A URL do recurso solicitado foi alterada permanentemente. A nova URL é fornecida na resposta.
404	Not Found	O servidor não pode encontrar o recurso solicitado.
500	Internal Server Error	O servidor encontrou uma situação com a qual não sabe lidar.

Fonte: Adaptado de MOZILLA (2025)

Os códigos de status HTTP são números de três dígitos que o servidor envia como resposta para a requisição feita. Eles fornecem informações sobre o resultado da solicitação e indicam o estado da resposta.

Existem cinco classes de códigos de status HTTP:

Códigos de status 1xx (Informational): Esses códigos indicam que a solicitação do cliente foi recebida, está sendo processada e o servidor está esperando mais informações antes de enviar uma resposta final.

Códigos de status 2xx (Successful): Esses códigos indicam que a solicitação do cliente foi bem-sucedida e o servidor conseguiu processá-la com sucesso. O código mais conhecido nessa classe é o 200 OK, que é retornado quando a requisição é bem-sucedida.

Códigos de status 3xx (Redirection): Esses códigos indicam que o cliente precisa realizar uma ação adicional para concluir a solicitação. Geralmente, ocorre quando o recurso foi movido para outro local ou quando é necessário seguir um redirecionamento para acessar o recurso.

Códigos de status 4xx (Client Error): Esses códigos indicam que houve um erro na solicitação feita pelo cliente. Por exemplo, o código 404 Not Found é retornado quando o servidor não encontra o recurso solicitado pelo cliente.

Códigos de status 5xx (Server Error): Esses códigos indicam que ocorreu um erro no servidor ao processar a requisição. Isso geralmente ocorre quando o servidor encontra um problema ao atender à solicitação do cliente.

Alguns exemplos comuns de códigos de status HTTP são:

200 OK: Indica que a solicitação foi bem-sucedida e o servidor retornou com sucesso os dados solicitados.
404 Not Found: Indica que o recurso solicitado não foi encontrado no servidor.
500 Internal Server Error: Indica que ocorreu um erro interno no servidor ao processar a requisição.

2.4 API

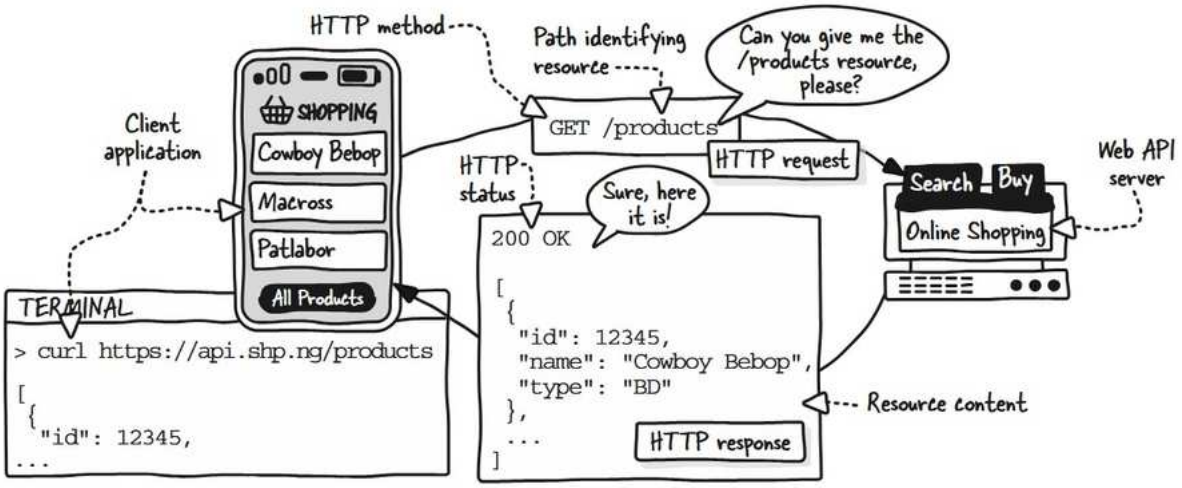
Uma API (Application Progamming Interface), de acordo com Geewax (2021), define a maneira pela qual sistemas de computador interagem e se comunicam entre si. Ela é um conjunto de regras, protocolos e ferramentas que permitem que diferentes componentes de software se conectem e troquem informações de forma padronizada e organizada. APIs são ampla-

mente usadas para permitir que diferentes sistemas, aplicativos e serviços trabalhem juntos de maneira eficiente e harmoniosa.

APIs são utilizadas em diversos contextos, disponibilizando maneiras rápidas de comunicação entre diferentes sistemas, as principais são:

- Bibliotecas e pacotes de linguagem: APIs podem ser encontradas em bibliotecas e pacotes de linguagem que fornecem funções e métodos específicos para realizar tarefas, como criptografia, manipulação de dados, interface com o sistema operacional, entre outros.
- Código próprio: Mesmo quando escrevemos nosso próprio código, criamos nossas próprias APIs definindo funções, métodos ou classes que encapsulam funcionalidades específicas. Essas APIs tornam nosso código modular e mais fácil de manter.
- Web APIs: O foco do trecho é nas Web APIs, são APIs projetadas para serem expostas na web e acessadas remotamente por diferentes pessoas e sistemas. Essas APIs permitem que aplicativos e serviços interajam uns com os outros por meio da internet. Por exemplo, serviços de mídia social, plataformas de pagamento online, serviços de geolocalização e muito mais são exemplos de Web APIs.

Figura 3 – Exemplo de requisição feita a uma API.



Fonte: LAURET (2025)

2.4.1 WEB APIs

As Web APIs são únicas em relação a outros tipos de APIs, pois são acessadas remotamente por meio de solicitações HTTP (ou outros protocolos) e retornam dados em formato estruturado, como JSON ou XML. Uma característica notável das Web APIs é que os desenvolvedores que as constroem têm o controle sobre as mudanças feitas na API. Quando uma mudança é feita em uma Web API, ela afeta diretamente os usuários que a consomem, pois não há cópias locais da API. Isso exige que os construtores da API tenham responsabilidade em manter a compatibilidade e comunicar as mudanças aos usuários.

2.5 Estrutura Básica de Uma Página Web

O desenvolvimento web moderno é sustentado por um conjunto de tecnologias base que operam em conjunto no lado do cliente (navegador). Essas tecnologias são HTML, CSS e JavaScript, que formam a tríade fundamental da construção de qualquer interface web.

Cada uma dessas linguagens cumpre um papel específico e complementar: o HTML define a estrutura do conteúdo, o CSS aplica a apresentação visual e o JavaScript introduz comportamento dinâmico. Quando integradas em um ambiente de desenvolvimento mais robusto, como frameworks modernos de front-end, essas tecnologias são responsáveis por tornar possíveis aplicações ricas, responsivas e interativas.

Esses três pilares são processados diretamente pelo navegador, sem a necessidade de compilação intermediária, o que contribui para o rápido desenvolvimento e fácil distribuição das aplicações web. A compreensão de seu funcionamento interno, limitações e boas práticas de uso é essencial para qualquer engenheiro da computação envolvido com desenvolvimento front-end.

2.5.1 HTML

De acordo com a W3C o HTML (HyperText Markup Language) é a linguagem fundamental de marcação que serve como base para a World Wide Web (WWW). Inicialmente, foi criada principalmente com o propósito de fornecer uma descrição semântica para documentos científicos, permitindo que pesquisadores e acadêmicos estruturassem e apresentassem seu conteúdo de maneira significativa. A linguagem foi pensando para incluir marcações e atributos que pudessem representar títulos, parágrafos, imagens, tabelas e outros elementos comuns em publicações acadêmicas.

Com o passar do tempo, o HTML mostrou ser altamente versátil e adaptável. Seu design geral permitiu que evoluísse e fosse modificado para acomodar uma ampla gama de tipos de documentos além de artigos científicos. Ao longo dos anos seguintes, desenvolvedores e designers começaram a utilizar o HTML para descrever diversos tipos de conteúdo, como artigos de notícias, postagens de blogs, descrições de produtos e muito mais. Além disso, a natureza flexível do HTML possibilitou que ele transcendesse o mundo de documentos estáticos e expandisse suas capacidades para incluir aplicações interativas.

Figura 4 – Exemplo de código HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Título da Página</title>
</head>
<body>

<h1>Isso é um cabeçalho</h1>
<p>Isso é um parágrafo.</p>

</body>
</html>
```

Fonte: Autor

A MDN (Mozilla Developer Network) define que a linguagem utiliza uma sintaxe composta de elementos (ou tags) que são cercados por ângulos (< >) e podem conter atributos para fornecer informações adicionais sobre o elemento.

Principais conceitos do HTML:

Elementos: São os blocos de construção do HTML e representam diferentes partes do conteúdo. Eles são identificados por tags, como <p> para parágrafos, <h1> para cabeçalhos, <a> para links e assim por diante.

Atributos: São usados para fornecer informações adicionais aos elementos e são especificados dentro das tags. Por exemplo, o atributo href é usado em um elemento <a> para definir o destino do link.

Estrutura: O HTML é organizado em uma estrutura de árvore, conhecida como Document Object Model (DOM). Os elementos são aninhados uns dentro dos outros para criar a hierarquia do conteúdo.

Tags vazias: Alguns elementos não têm conteúdo ou fechamento e são chamados de tags vazias. Eles são escritos com uma única tag, como
 para quebras de linha ou <img> para imagens.

Comentários: Os comentários em HTML são usados para adicionar notas ao código que não serão exibidas na página. Eles são escritos entre <!– e –>.

Versões do HTML: Existem várias versões do HTML, sendo a mais recente o HTML5. Cada versão introduz novos recursos e melhorias para melhorar a experiência de desenvolvimento e uso da web.

2.5.2 CSS

O Cascading Style Sheets (CSS) ou “Folhas de Estilo em Cascata”, em português, Segundo Meyer (2006), é uma linguagem de estilo utilizada para definir a apresentação e o layout de documentos HTML. Ela permite controlar a aparência dos elementos do documento, como texto, cores, fontes, espaçamento, posicionamento, imagens de fundo e muito mais. A principal finalidade do CSS é separar o conteúdo da estrutura e da apresentação, tornando o código HTML mais limpo e facilitando a manutenção e o design das páginas web.

A definição de “estilo em cascata” refere-se ao comportamento do CSS, onde várias regras de estilo podem ser aplicadas a um elemento. Uma prioridade é determinada pela especificidade das regras e pela ordem de aplicação. Isso permite que as propriedades sejam herdadas ou substituídas de forma hierárquica, criando assim um sistema de cascata.

Figura 5 – Exemplo de código CSS.

```css
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

Fonte: Autor

Em vez de definir estilos diretamente nos elementos HTML (usando atributos como “style”), o CSS é aplicado externamente a partir de um arquivo separado ou diretamente na tag <style> dentro do próprio documento HTML. Essa abordagem facilita a reutilização dos estilos em várias páginas e oferece maior controle sobre a aparência de todo o site.

Com o uso de CSS, é possível criar páginas web estilizadas, atraentes e responsivas, tornando a experiência do usuário mais agradável e consistente em diferentes dispositivos e navegadores. O CSS é uma das três principais tecnologias da web, juntamente com o HTML e o JavaScript, e desempenha um papel fundamental no design e na apresentação de conteúdo na internet.

2.5.3 JavaScript

Flanagan (2020) define JavaScript como a linguagem da Web, amplamente utilizada na maioria dos sites modernos. Todos os navegadores web em dispositivos como desktops, consoles de jogos, tablets e smartphones, incluem interpretadores de JavaScript, tornando-a a linguagem de programação mais difundida na história da computação.

Iniciando como uma linguagem de script, JavaScript evoluiu significativamente e se tornou uma linguagem de propósito geral, robusta e eficiente. A última versão da linguagem (ECMAScript 6) introduziu novos recursos que tornam o JavaScript adequado para o desenvolvimento de software em larga escala.

Figura 6 – Exemplo de código JavaScript.

```
const cars = ["BMW", "Volvo", "Saab", "Ford"];
let len = cars.length;

let i = 2;

let text = "";
for (; i < len; i++) {
  text += cars[i] + "<br>";
}

document.getElementById("demo").innerHTML = text;
```

Fonte: Autor

Como afirma o Haverbeke (2024), JavaScript é uma linguagem de programação interpretada, multiplataforma e orientada a objetos que adiciona interatividade e dinamismo às páginas da web. É uma das linguagens de programação mais populares do mundo, usada por milhões de desenvolvedores para criar uma ampla variedade de aplicativos da web, jogos e outros tipos de software.

JavaScript é uma linguagem poderosa e flexível que pode ser usada para criar uma ampla variedade de aplicativos da web, jogos e outros tipos de software. Se você está interessado em aprender JavaScript, existem muitos recursos disponíveis online e em bibliotecas. Alguns dos principais

conceitos de JavaScript incluem:

- Variáveis: Variáveis são usadas para armazenar dados. Elas podem ser usadas para armazenar valores simples, como números, strings e booleanos, ou objetos complexos, como arrays e objetos.
- Expressões: Expressões são usadas para calcular valores. Elas podem ser compostas por números, strings, booleanos, variáveis e funções.
- Comandos: Comandos são usados para executar ações. Eles podem ser usados para atribuir valores a variáveis, calcular expressões, imprimir texto e interagir com o usuário.
- Funções: Funções são blocos de código que podem ser reutilizados. Elas podem ser usadas para agrupar código relacionado, tornar o código mais organizado e facilitar a manutenção.
- Objetos: Objetos são estruturas de dados que podem ser usadas para armazenar dados e funções relacionados. Eles são semelhantes aos dicionários em linguagens de programação como Python e PHP.
- Eventos: Eventos são ações que ocorrem no navegador, como um usuário clicar em um botão ou mover o mouse. JavaScript pode ser usado para responder a eventos.
- Bibliotecas: Existem muitas bibliotecas JavaScript disponíveis, que fornecem funcionalidades adicionais ao JavaScript. Algumas bibliotecas populares incluem jQuery, React e Angular.

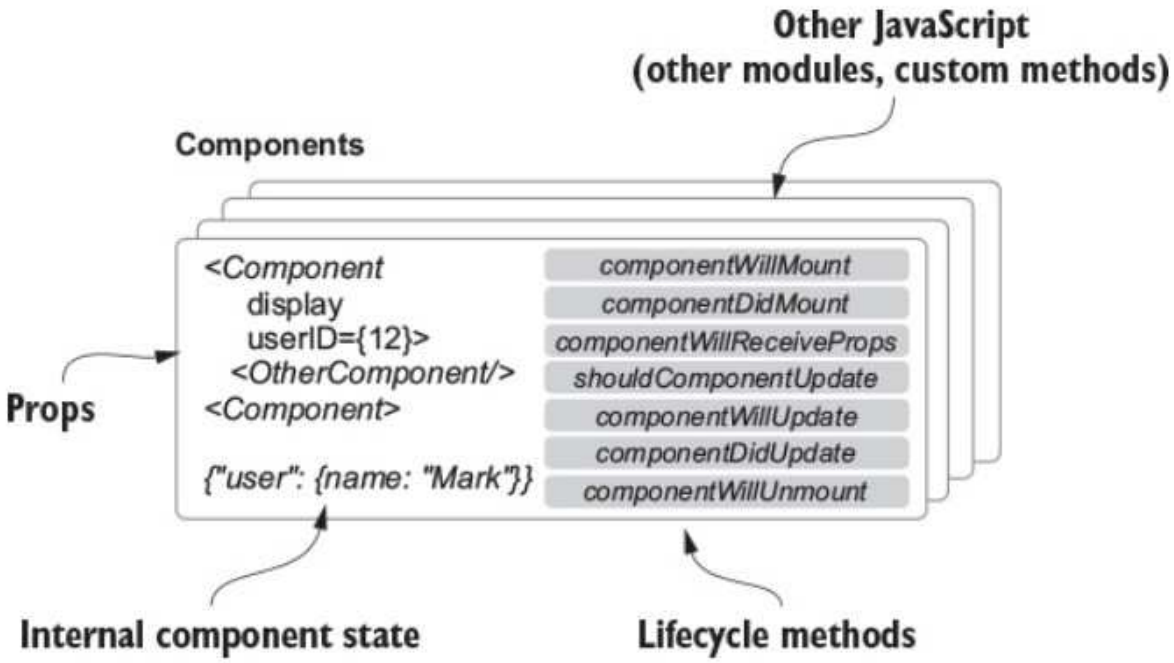
2.6 ReactJS

O React, é uma biblioteca de JavaScript para a criação de interfaces de usuário interativas e dinâmicas. Foi inicialmente criado por Jordan Walke, um engenheiro de software do Facebook. A história do React começa quando foi incorporado ao feed de notícias do Facebook em 2011 e posteriormente utilizado no Instagram após a aquisição desta pela Facebook em 2012.

Em 2013, o React foi disponibilizado como código aberto (open source) durante a JSConf, uma conferência dedicada ao JavaScript. Nesse momento, o React se juntou a outras bibliotecas populares, como jQuery, Angular, Dojo e Meteor, no segmento de bibliotecas de interface do usuário (UI).

Em janeiro de 2015, foram lançadas outras duas ferramentas relacionadas ao React. A primeira foi o React Native, uma biblioteca que permitia o desenvolvimento de aplicativos móveis usando o React. Isso possibilitou que os desenvolvedores reutilizassem seus conhecimentos e código entre projetos web e móveis. A segunda foi o ReactVR, uma ferramenta que levou o React para uma variedade mais ampla de destinos de renderização, permitindo a criação de experiências de realidade virtual com base na mesma abordagem de componentes do React.

Figura 7 – Visão geral de um component ReactJS.



Fonte: THOMAS (2018)

O React segue a abordagem baseada em componentes, onde as interfaces são divididas em pequenos elementos independentes chamados "componentes". Cada componente é responsável por uma parte específica da interface do usuário e pode ser reutilizado em diferentes partes da aplicação.

O React utiliza uma representação virtual do DOM (Virtual DOM) para otimizar o desempenho das atualizações na interface. Em vez de atualizar diretamente o DOM real toda vez que há uma mudança, o React compara a representação virtual com o DOM real, identificando as diferenças

e realizando atualizações apenas nos elementos que precisam ser alterados. Isso torna as atualizações da interface mais eficientes e rápidas.

O React é conhecido por sua eficiência, performance e escalabilidade, tornando-o uma escolha popular para o desenvolvimento de aplicações web e móveis modernas. Além disso, o React é frequentemente usado em conjunto com outras bibliotecas e frameworks, como Redux para gerenciamento de estado e React Router para gerenciamento de rotas, para criar aplicações mais complexas.

Em resumo, o ReactJS é uma poderosa biblioteca JavaScript que simplifica a criação de interfaces de usuário reativas e eficientes. Sua abordagem baseada em componentes torna o desenvolvimento de interfaces mais modular e gerenciável, permitindo que os desenvolvedores criem aplicações web e móveis com melhor desempenho e experiência do usuário. Com sua trajetória de sucesso e adoção pela comunidade, o React continua a ser uma das escolhas preferidas para desenvolvedores que desejam criar experiências interativas e modernas na web e em dispositivos móveis.

2.7 OpenStreetMap

A OpenStreetMap Foundation (2025), por meio do seu manual do usuário, explica que o OpenStreetMap (OSM) é um projeto colaborativo de mapeamento e cartografia que visa criar um mapa do mundo gratuito e de código aberto. Lançado em 2004, o OSM permite que qualquer pessoa contribua para a coleta de dados geoespaciais e informações geográficas, como estradas, pontos de interesse, edifícios, trilhas, rios e muito mais.

O projeto é baseado em uma comunidade global de voluntários, conhecidos como “mappers”, que usam dispositivos GPS, imagens de satélite e outras fontes para adicionar informações ao mapa. Esses dados são coletados e registrados em uma base de dados geográfica acessível publicamente.

Figura 8 – Estilo de mapa utilizado no OpenStreetMap

Fonte: Autor

Principais características e objetivos do OpenStreetMap:

Dados abertos: Todo o conteúdo do OSM é licenciado sob a Open Database License (ODbL), o que significa que os dados são disponibilizados de forma aberta e livre para uso por qualquer pessoa. Isso contrasta com os mapas proprietários e comerciais que geralmente têm restrições de licenciamento e acesso.

Atualização contínua: O OSM é dinâmico e está em constante atualização. A comunidade de mappers contribui continuamente para aprimorar e atualizar o mapa, refletindo mudanças na paisagem urbana, novos desenvolvimentos e outros eventos relevantes.

Diversidade geográfica: O OSM é uma plataforma global, o que significa que não se concentra apenas em áreas urbanas ou países desenvolvidos. Pessoas de todo o mundo contribuem com informações, permitindo que regiões menos mapeadas também sejam representadas com detalhes.

Flexibilidade e personalização: Como é um projeto de código aberto, o OSM pode ser personalizado e adaptado para atender a diversas necessidades. Empresas, organizações e desenvolvedores podem usar os dados e a

tecnologia do OSM para criar aplicativos específicos, mapas personalizados e serviços de localização.

Dados de qualidade: A comunidade do OSM possui sistemas de revisão e validação para garantir a qualidade dos dados adicionados. Isso inclui revisões automáticas, auditorias da comunidade e comparações com outras fontes de dados cartográficos.

Medeiros e Holanda (2017) destacam que o OSM se tornou uma fonte de dados geoespaciais amplamente utilizada em todo o mundo, sendo adotado em diversas aplicações, desde sistemas de navegação, mapas personalizados e soluções de localização, até pesquisas científicas. Sua natureza aberta e colaborativa é um de seus maiores trunfos, permitindo que a comunidade global contribua para a construção de um mapa detalhado, preciso e acessível para todos.

2.8 Sistemas Global de Posicionamento

O Sistema de Posicionamento Global, conhecido como GPS (Global Positioning System), é um sistema de navegação e posicionamento baseado em satélites que fornece informações de localização, velocidade e tempo em qualquer lugar da Terra ou em suas proximidades. O GPS foi concebido como um sistema de medição de distância entre posições conhecidas de satélites no espaço e posições desconhecidas na terra, mar, ar e espaço.

O funcionamento do GPS é baseado em uma rede de satélites que orbitam à Terra e transmitem sinais de rádio contendo informações de tempo e posição precisos. Um receptor GPS em terra, mar ou ar recebe os sinais dos satélites e, a partir desses sinais, determina sua posição com base nas distâncias e tempos de viagem calculados em relação aos satélites.

Além de fornecer informações de posição, o GPS também é usado para calcular a velocidade de veículos em movimento e para sincronizar o tempo com grande precisão. O sistema é utilizado em várias aplicações, desde navegação veicular e rastreamento de objetos até serviços de mapeamento e aplicativos de localização em dispositivos móveis.

O GPS é um sistema global, permitindo que pessoas em qualquer lugar

do mundo tenham acesso às informações de localização e tempo precisas. Sua capacidade de fornecer dados em tempo real e com alta precisão tornou-o uma ferramenta essencial em muitos campos, incluindo operações militares, transporte, aviação, exploração, cartografia, meteorologia, entre outros.

3 Desenvolvimento da Aplicação

3.1 Definição do Problema

O Centro Tático Aéreo (CTA), unidade especializada vinculada à Secretaria de Segurança Pública do Estado do Maranhão, desempenha papel essencial em missões de policiamento, resgate e transporte aeromédico em todo o território estadual (MARANHÃO, 2020). A natureza dessas operações exige ferramentas digitais e tecnológicas capazes de aprimorar o acompanhamento em tempo real das aeronaves, assegurando a precisão das informações e a eficiência das decisões operacionais.

Figura 9 – Aeronave aguardando para início de voo teste operado pelo Centro Tático Aéreo.

Fonte: Autor

Atualmente, o CTA não dispõe de um sistema integrado que permita o gerenciamento centralizado de aeronaves, missões e rastreamento em tempo real. As atividades de controle são baseadas em registros manuais e

comunicações por rádio, o que dificulta a rastreabilidade e o armazenamento histórico de dados. Soluções comerciais disponíveis — como plataformas baseadas em ADS-B — apresentam custos elevados e dependência tecnológica de terceiros, além de não atenderem plenamente às demandas do órgão.

Dessa forma, o problema consiste em desenvolver uma aplicação web de baixo custo e código aberto, capaz de consumir uma API REST existente para exibir a posição das aeronaves em um mapa interativo (OpenStreetMap), bem como gerenciar missões, históricos de voo e controle de usuários autenticados. O sistema deve operar de forma responsiva e intuitiva, permitindo acesso via navegador, priorizando confiabilidade e segurança da informação.

A proposta converge com os princípios da Estratégia Brasileira de Transformação Digital (E-Digital 2022–2026), que estabelece como diretriz o uso da tecnologia da informação para modernizar a administração pública e ampliar a eficiência e a transparência (BRASIL, 2022). Ademais, reflete a visão de modernização tecnológica destacada por Silva (2023), ao apontar que o avanço tecnológico é essencial para otimizar e aumentar a segurança das operações aéreas no CTA, demonstrando a importância de integrar sistemas digitais à aviação pública. Assim, o desenvolvimento deste sistema busca não apenas suprir uma lacuna operacional, mas também fortalecer a transformação digital no setor público maranhense.

3.2 Arquitetura da Aplicação

A aplicação proposta adota uma arquitetura cliente-servidor, na qual o front-end web, desenvolvido em ReactJS, consome uma API REST responsável por fornecer os dados relacionados às aeronaves, missões, usuários e histórico de voos. O sistema foi concebido para oferecer uma interface moderna, responsiva e de fácil uso, integrando mapas interativos, tabelas e formulários dinâmicos, de modo a atender às necessidades operacionais do Centro Tático Aéreo (CTA).

A camada de apresentação (front-end) é responsável por renderizar

os dados obtidos do servidor e gerenciar a interação com o usuário. A comunicação entre cliente e servidor ocorre por meio do protocolo HTTP, utilizando o formato JSON para troca de informações. O controle de acesso é realizado por autenticação de usuário, sendo o token armazenado em sessão e incluído automaticamente nos cabeçalhos das requisições.

O sistema web foi desenvolvido para atender aos requisitos operacionais do Centro Tático Aéreo, permitindo o acompanhamento das missões, aeronaves e usuários, bem como a visualização em tempo real da posição das aeronaves transmitida pelo servidor. Para isso, a aplicação segue três responsabilidades fundamentais:

1) Autenticação com a API – a aplicação realiza o processo de login e autenticação dos usuários, obtendo um token de sessão que é armazenado de forma segura. Esse token é incluído automaticamente nos cabeçalhos das requisições subsequentes, garantindo que apenas usuários autorizados possam acessar e manipular os dados disponíveis no sistema.
2) Carregamento e envio dinâmico de informações – o sistema é capaz de consultar e atualizar em tempo real os principais conjuntos de dados:
    - **Missões:** o front-end recupera da API as missões ativas ou planejadas e permite criar, editar e finalizar registros conforme o status operacional;
    - **Aeronaves:** os dados das aeronaves cadastradas, como modelo, prefixo e número de série, são carregados dinamicamente e atualizados conforme o uso;
    - **Usuários:** as informações de operadores e administradores são consultadas e mantidas pela aplicação, permitindo a inclusão e gerenciamento de perfis de acesso.

    Essa dinâmica de troca de informações assegura que qualquer alteração feita no servidor seja refletida instantaneamente na interface web, mantendo o sistema sincronizado com a base de dados central.
3) Apresentação de posições em tempo real – a aplicação exibe, em um mapa interativo baseado no OpenStreetMap, as posições das

aeronaves ou missões selecionadas, conforme os dados recebidos da API. O sistema atualiza automaticamente as coordenadas de latitude, longitude, velocidade e altitude, permitindo uma visualização contínua das operações aéreas.

Essa arquitetura garante modularidade e escalabilidade, permitindo futuras expansões sem comprometer o desempenho. Cada módulo do sistema — autenticação, missões, aeronaves e usuários — foi estruturado de forma independente, comunicando-se com o servidor por meio de endpoints específicos da API. A utilização de tecnologias abertas e amplamente consolidadas assegura a manutenção a longo prazo e a interoperabilidade com os demais sistemas que compõem o ecossistema de rastreamento do CTA.

3.3 Desenvolvimento da Aplicação

O desenvolvimento da aplicação seguiu um processo incremental, com o objetivo de garantir a integração entre os componentes da interface e os serviços disponibilizados pela API. Durante a implementação, buscou-se manter uma estrutura modular, em que cada funcionalidade do sistema fosse organizada em componentes independentes, favorecendo a reutilização e a manutenção do código.

A aplicação foi desenvolvida com base no framework ReactJS, utilizando o paradigma de componentização. Cada parte da interface — como menus, tabelas, formulários e o mapa de rastreamento — foi construída como um componente autônomo, capaz de ser renderizado dinamicamente conforme as requisições e interações do usuário. Essa abordagem possibilitou maior clareza na estrutura do projeto e facilitou a atualização de elementos sem a necessidade de recarregar toda a página, característica típica de aplicações Single Page Applications (SPA).

O fluxo de execução da aplicação inicia-se com o processo de autenticação, em que o usuário insere suas credenciais para obter um token de acesso. Após a validação com a API, o token é armazenado na sessão do navegador e passa a ser incluído automaticamente em todas as requisições subsequentes. Esse mecanismo garante segurança no acesso aos recursos

e evita a necessidade de autenticações repetidas durante o uso do sistema.

Uma vez autenticado, o usuário tem acesso às principais funcionalidades do sistema, estruturadas conforme os módulos descritos na arquitetura: Missões, Aeronaves e Usuários. O carregamento dos dados é feito de forma dinâmica por meio de chamadas à API, utilizando a biblioteca Axios. Essa integração permite que as informações apresentadas nas tabelas, listas e mapas sejam atualizadas em tempo real, refletindo qualquer alteração feita no servidor.

A camada de visualização geográfica utiliza a biblioteca React Leaflet, responsável pela renderização do mapa interativo com dados do OpenStreetMap. As posições das aeronaves são obtidas diretamente do servidor e exibidas como marcadores que indicam localização, altitude, velocidade e direção de voo. O sistema também permite ao usuário selecionar uma missão específica para acompanhar a trajetória associada, atualizando automaticamente o mapa com as informações da aeronave correspondente. Esse comportamento dinâmico é obtido por meio de consultas periódicas à API, que atualizam a interface sem a necessidade de intervenção manual.

A comunicação entre os componentes da aplicação é controlada por meio do gerenciamento de estado local, utilizando os *hooks* nativos do React, como useState e useEffect. Essa estratégia simplifica a lógica interna e permite que cada módulo manipule suas próprias informações sem comprometer o desempenho global da aplicação. Além disso, o uso do React Router garante uma navegação fluida entre as páginas, como o painel de missões, o cadastro de aeronaves, o histórico de voos e a lista de usuários.

Por fim, a interface foi projetada para oferecer usabilidade e responsividade, garantindo que o sistema possa ser acessado de diferentes dispositivos, incluindo notebooks, tablets e computadores de operação. Foram aplicados princípios de design simples e intuitivos, priorizando a clareza das informações e a agilidade de operação, essenciais em contextos de missão e tomada de decisão.

3.4 Interfaces da Aplicação

A interface da aplicação foi projetada com foco em clareza, organização e usabilidade, de modo a facilitar o acesso às informações e a execução das tarefas pelos operadores do Centro Tático Aéreo (CTA). As telas seguem um padrão visual unificado, com o uso de componentes responsivos e layout adaptável, garantindo boa experiência de uso em diferentes resoluções. A seguir, são apresentadas as principais telas que compõem o sistema, acompanhadas de suas funcionalidades.

3.4.1 Login

A primeira tela do sistema é responsável pela autenticação dos usuários.
Nela, o operador informa suas credenciais (e-mail e senha), validadas pela API. Após o login, é gerado um token de sessão, armazenado temporariamente para autorizar as próximas requisições. Essa camada de segurança garante que apenas usuários cadastrados possam acessar o sistema e interagir com as informações das aeronaves e missões.

Figura 10 – Tela de login da aplicação web

Fonte: Autor

3.4.2 Navegação Lateral

Após a autenticação, o usuário tem acesso à barra lateral de navegação, que se mantém fixa durante toda a utilização do sistema.
Esse componente contém atalhos para as seções principais: Rastreamento, Histórico de Voos, Missões, Usuários e Aeronaves, além do botão de Logout. O objetivo dessa estrutura é permitir navegação rápida e intuitiva entre as telas, característica essencial em aplicações de uso operacional.

Figura 11 – Barra lateral de navegação da aplicação.

Fonte: Autor

3.4.3 Rastreamento

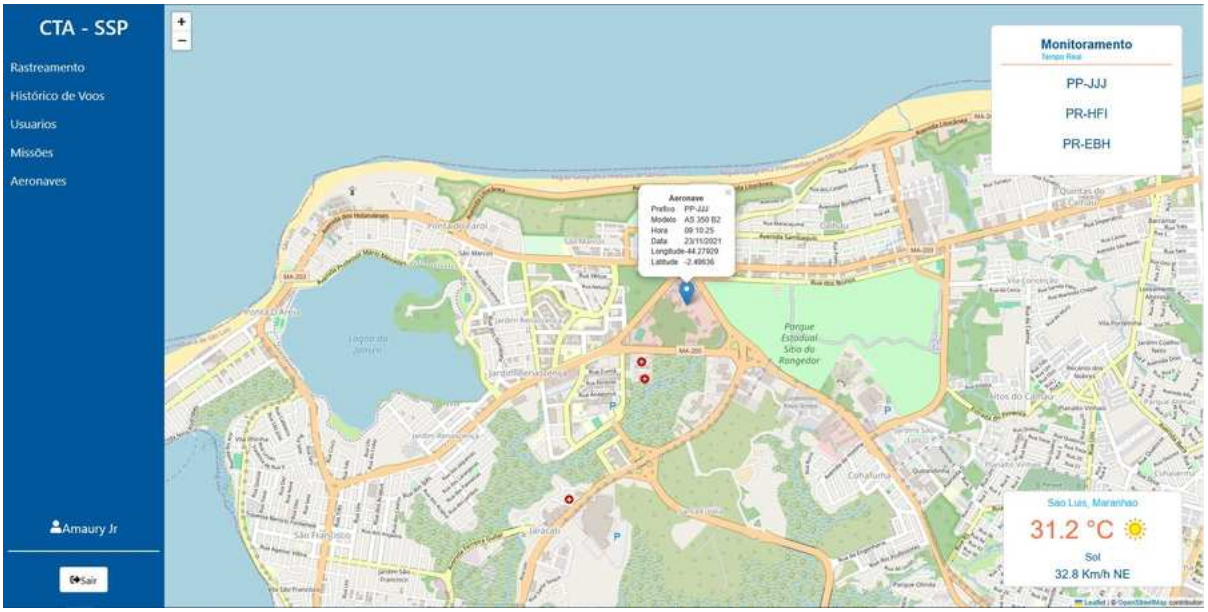
A tela de rastreamento representa o núcleo operacional do sistema. Por meio de um mapa interativo baseado no OpenStreetMap, são exibidas as posições em tempo real das aeronaves cadastradas. Cada marcador mostra dados como latitude, longitude, altitude, velocidade e horário da última atualização.

Foram implementados dois widgets funcionais:

- O primeiro exibe todas as aeronaves cadastradas, permitindo que o usuário selecione uma delas e o mapa seja automaticamente centralizado na posição atual.
- O segundo apresenta informações meteorológicas referentes à localização da aeronave, incluindo temperatura, condição climática, velocidade e direção do vento.

  Essa integração permite o acompanhamento simultâneo da frota e das condições ambientais durante as missões.

Figura 12 – Tela de Rastreamento de aeronaves.



Fonte: Autor

3.4.4 Histórico de Voos

A tela de histórico de voos tem como objetivo apresentar todas as missões já concluídas e seus respectivos trajetos registrados durante a operação. Assim como o módulo de rastreamento, essa tela utiliza o OpenStreetMap (OSM) integrado à biblioteca React Leaflet, permitindo a exibição de rotas completas percorridas pelas aeronaves em missões anteriores.

Cada voo é representado visualmente por uma linha de trajetória (polyline), desenhada no mapa a partir das coordenadas geográficas obtidas pela API. Essa linha conecta os pontos registrados pela aeronave durante o voo, permitindo visualizar com precisão todo o percurso realizado.
Ao interagir com a linha, o usuário pode acessar informações detalhadas de cada ponto registrado, incluindo:

- Altitude: indica a altura em relação ao nível do mar no momento do registro;
- Velocidade: mostra a velocidade instantânea da aeronave;
- Precisão: representa o grau de confiabilidade do ponto coletado, calculado conforme o sinal GPS.

Esses dados permitem uma análise técnica mais completa do desempenho das aeronaves, possibilitando a verificação de padrões de voo, variações de altitude e eventuais desvios de rota.

Além do mapa principal, a tela conta com um widget lateral que exibe a lista de voos armazenados. Cada item do widget contém informações resumidas da missão, como:

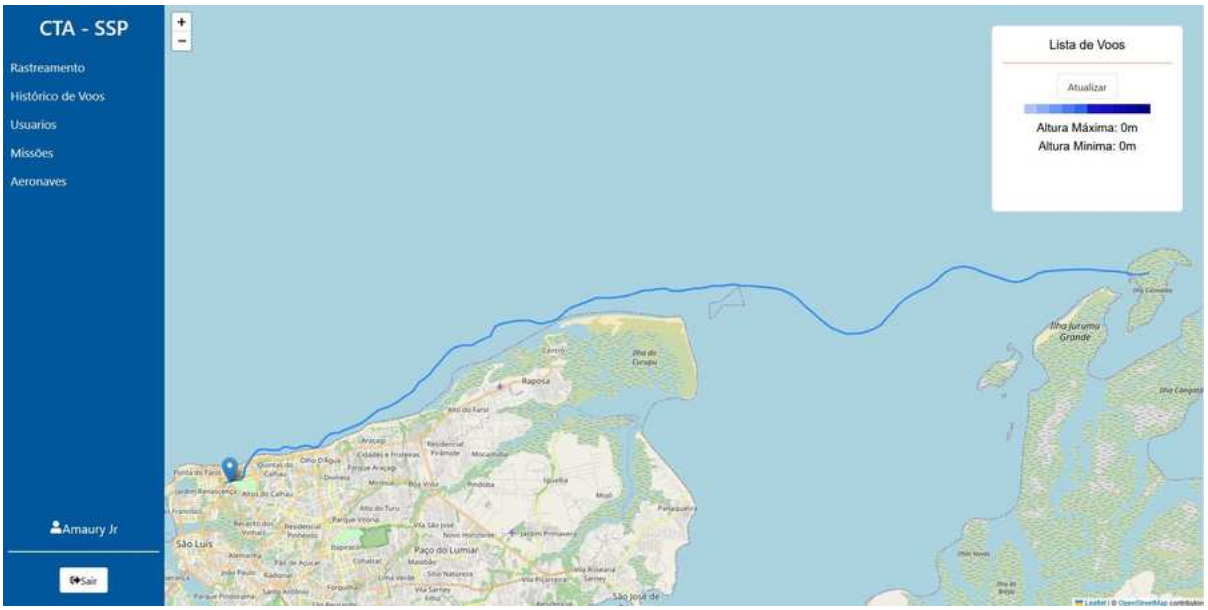
- Nome da aeronave utilizada;
- Status da missão (planejada, em andamento ou concluída);
- Data e hora programada do voo.

Ao selecionar um voo na lista, o mapa é automaticamente atualizado, centralizando a visualização no trajeto correspondente e destacando as informações daquele registro específico. Essa interação entre o mapa e o widget oferece uma navegação fluida e intuitiva, permitindo que o operador revise diferentes missões de forma rápida e organizada.

O módulo de histórico de voos, portanto, funciona não apenas como

um repositório de dados, mas como uma ferramenta analítica dentro da aplicação. Ele viabiliza o estudo e o acompanhamento dos padrões de voo, servindo de suporte para avaliações de desempenho, auditorias operacionais e relatórios técnicos do CTA.

Figura 13 – Tela de Histórico de Voos com mapa demonstrando um voo teste para a ilha de Carrapatal.



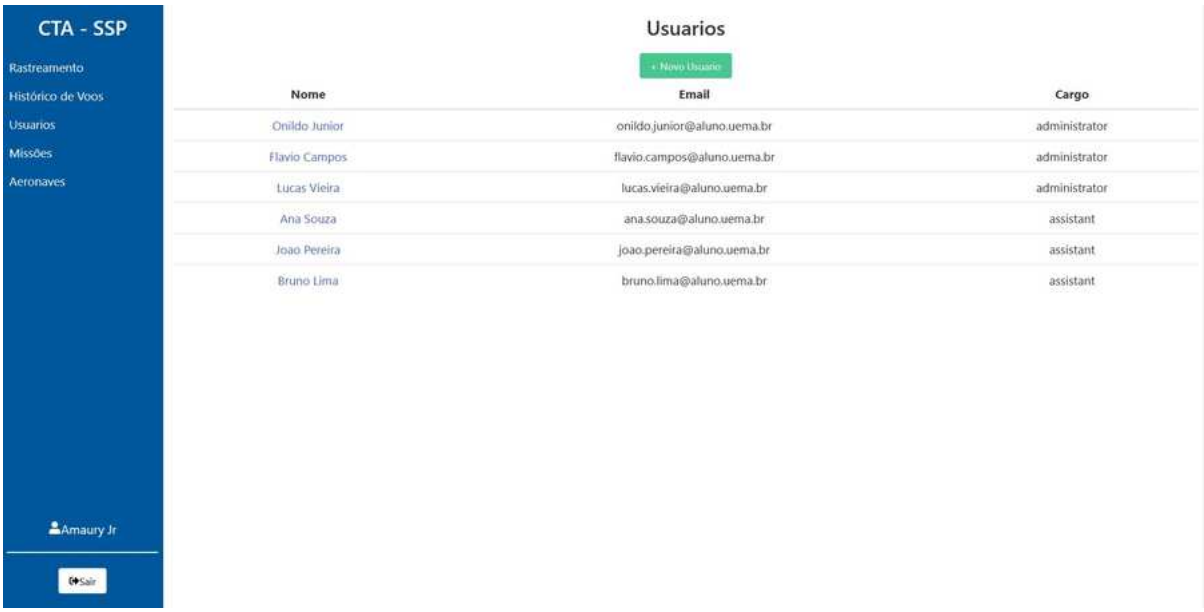
Fonte: Autor

3.4.5 Usuários

3.4.5.1 Lista de Usuários

Exibe todos os usuários cadastrados no sistema em formato de tabela. Cada linha contém nome, e-mail e cargo, com botões de ação para visualizar, editar ou excluir registros.
Essa tela centraliza o gerenciamento de contas e garante controle de acesso e rastreabilidade das ações executadas pelos operadores.

Figura 14 – Tela de listagem de Usuários.



Fonte: Autor

3.4.5.2 Detalhe do Usuário

Mostra os dados completos de um usuário específico, incluindo informações pessoais e permissões de acesso.
A partir desta tela, administradores podem alterar o cargo, redefinir senha ou atualizar informações cadastrais.

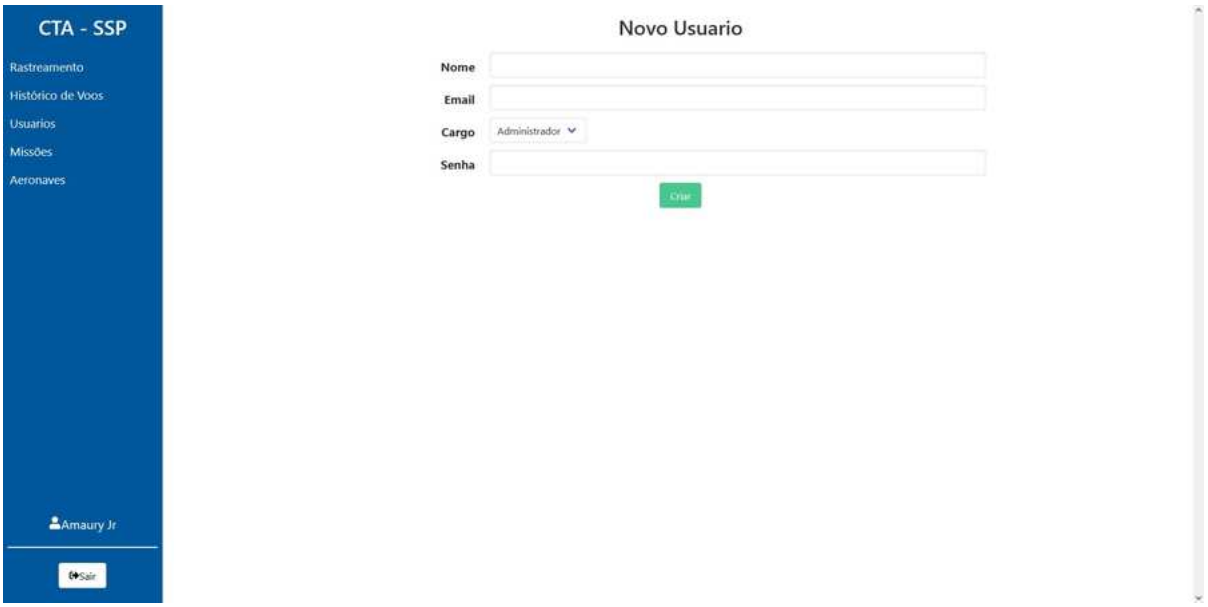
Figura 15 – Tela de Detalhe de Usuário.

Fonte: Autor

3.4.5.3 Novo Usuário

Permite o cadastro de novos usuários por meio de um formulário simples, com campos obrigatórios de nome, e-mail e cargo.
Essa funcionalidade é restrita a perfis administrativos, assegurando a integridade do controle de acesso.

Figura 16 – Tela de criação de Novo Usuário.



Fonte: Autor

3.4.6 Missões

3.4.6.1 Lista de Missões

Apresenta uma listagem de todas as missões cadastradas, exibidas em formato de cards.
Cada card contém informações como ID, local de origem, destino, data e hora programada, além do status atual (planejada, em andamento ou concluída). O usuário pode selecionar uma missão específica para visualizar detalhes ou editar suas informações.

Figura 17 – Tela de listagem de Missões.

Fonte: Autor

3.4.6.2 Detalhe da Missão

Mostra todas as informações detalhadas de uma missão específica, incluindo aeronave designada, tripulação, status, rota e horários.
A partir dessa tela, é possível iniciar ou encerrar o rastreamento da missão, além de acompanhar em tempo real as informações recebidas da API sobre o voo.

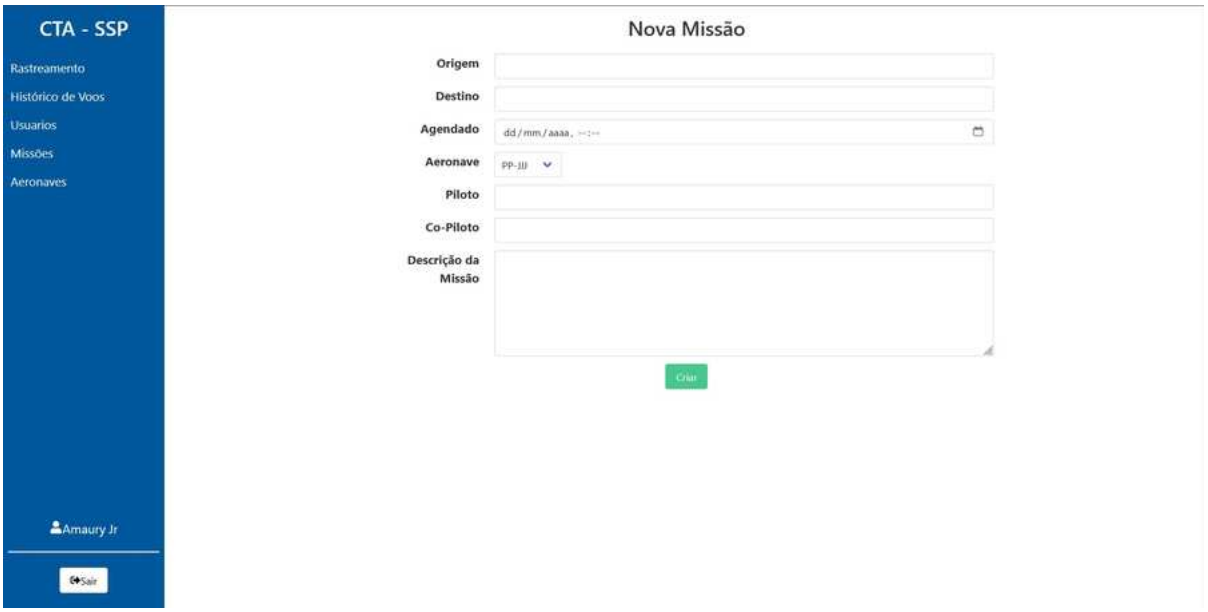
Figura 18 – Tela de Detalhe de Missão.

Fonte: Autor

3.4.6.3 Nova Missão

Permite o cadastro de novas missões, definindo origem, destino, data, horário e aeronave envolvida.
O formulário inclui campos de validação e comunicação direta com a API, garantindo que as novas entradas sejam salvas corretamente no banco de dados.

Figura 19 – Tela de criação de Nova Missão.



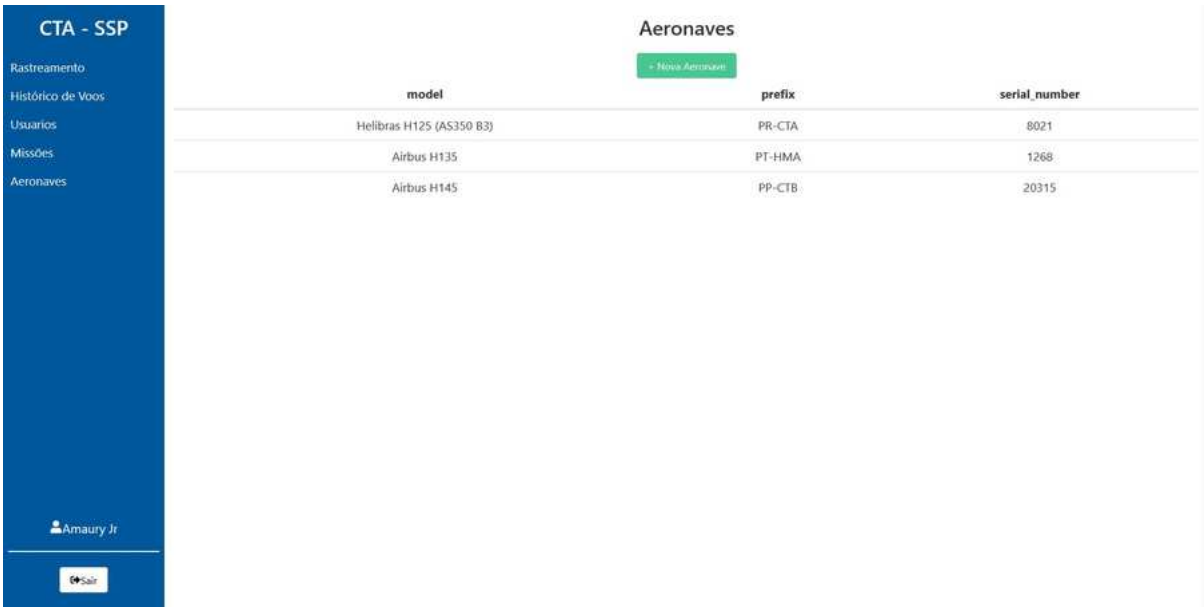
Fonte: Autor

3.4.7 Aeronaves

3.4.7.1 Lista de Aeronaves

Apresenta todas as aeronaves registradas no sistema, em uma tabela contendo modelo, prefixo e número serial.
A tela oferece ações de edição e exclusão, além de acesso à função de cadastro de novas aeronaves.
Essas informações são essenciais para vinculação nas missões e no rastreamento.

Figura 20 – Tela de listagem de Aeronaves.

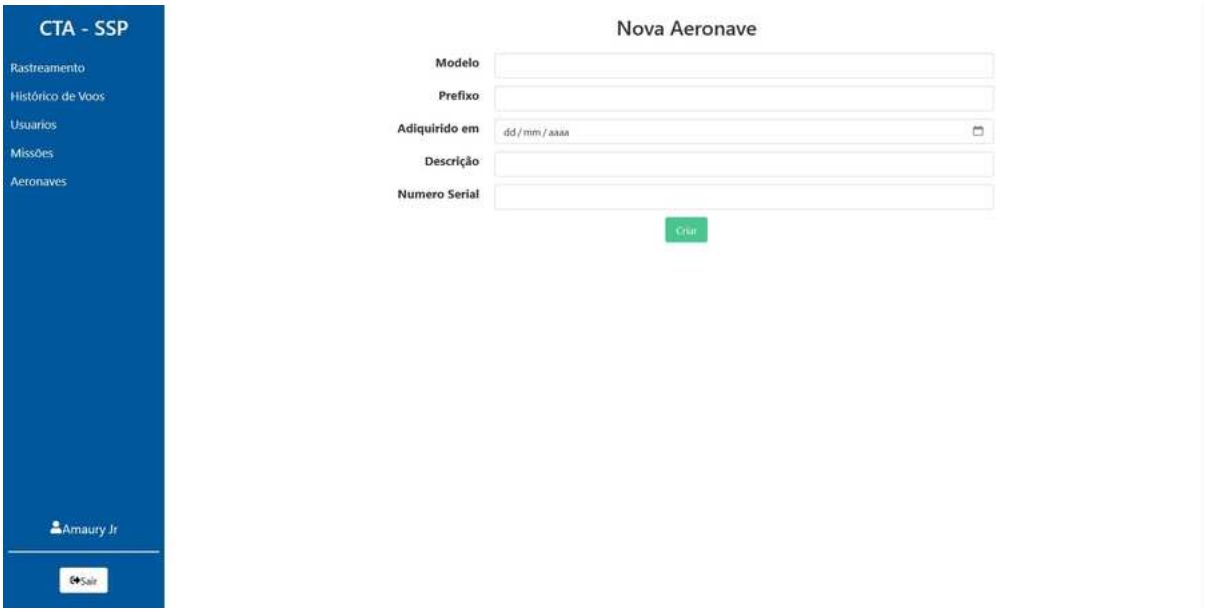


Fonte: Autor

3.4.7.2 Nova Aeronave

Permite o cadastro de novas aeronaves, informando os dados técnicos principais e o número de série.
A comunicação ocorre diretamente com a API, garantindo a persistência e atualização imediata na base de dados.

Figura 21 – Tela de cadastro de Nova Aeronave.



Fonte: Autor

Todas as telas da aplicação foram projetadas para manter consistência visual, legibilidade e praticidade, reduzindo o número de etapas necessárias para o operador realizar suas tarefas.

A interface se adapta a diferentes tamanhos de tela e mantém o foco na experiência do usuário, aspecto essencial em sistemas voltados a operações em tempo real.

4 Resultados e Discussão

A aplicação web desenvolvida encontra-se funcional e apta para utilização pelo Centro Tático Aéreo (CTA), apresentando desempenho estável, interface intuitiva e integração eficiente com a API de dados de aeronaves e missões. A estrutura do sistema permite carregamento dinâmico e atualização automática das informações no mapa, garantindo precisão e rapidez na visualização das posições.

A interface foi projetada com foco na simplicidade e usabilidade, adotando um design limpo e navegação lateral fixa, facilitando o acesso às funcionalidades e reduz a necessidade de treinamento prévio. Essa abordagem torna o sistema adequado ao ambiente operacional do CTA, permitindo a execução eficiente das atividades de rastreamento e consulta.

Durante os testes, a aplicação demonstrou bom desempenho na comunicação com a API, com tempos de resposta inferiores a dois segundos. A atualização em tempo real ocorreu de forma contínua, mesmo com múltiplas aeronaves monitoradas simultaneamente, sem sobrecarga perceptível no navegador, evidenciando a eficiência das tecnologias empregadas.

Na validação prática, foram realizados dois voos-teste com aeronaves do CTA, nos quais foi possível acompanhar, em tempo real, a trajetória das aeronaves, incluindo dados de posição, velocidade e altitude. A conexão com a API manteve-se estável durante toda a operação, confirmando a confiabilidade do sistema em cenários reais.

O módulo de histórico de voos apresentou resultados relevantes, permitindo a visualização completa das rotas realizadas, com dados detalhados de cada ponto do trajeto. Essa funcionalidade contribui para análises operacionais e planejamento de futuras missões.

De forma geral, o sistema atingiu os objetivos propostos, oferecendo uma solução web eficiente e de baixo custo para o monitoramento de aeronaves. Além disso, apresenta potencial para futuras expansões, com inclusão de novos recursos e integrações que podem ampliar sua aplicabilidade no contexto operacional.

5 Conclusão

O trabalho apresentou o desenvolvimento de uma aplicação web voltada ao acompanhamento e gerenciamento das operações aéreas do Centro Tático Aéreo (CTA). Considerando o contexto operacional da unidade, caracterizado pela necessidade de monitoramento contínuo das aeronaves e organização das missões, tornou-se evidente a demanda por uma ferramenta centralizada capaz de consolidar dados de rastreamento, cadastro e histórico em uma única plataforma.

A principal problemática identificada foi a ausência de um sistema web integrado que contemplasse rastreamento em tempo real, gerenciamento de missões, cadastro de aeronaves e usuários, além da disponibilização do histórico de voos. Para solucionar essa lacuna, o objetivo do trabalho consistiu em desenvolver uma solução baseada em tecnologias abertas, utilizando ReactJS e integração com uma API REST já existente, capaz de apresentar informações operacionais de forma clara, atualizada e acessível.

Os resultados obtidos demonstraram que o sistema atendeu aos objetivos propostos. A aplicação apresentou funcionamento estável, com atualização automática dos dados de telemetria, exibição geográfica por meio do OpenStreetMap e interface responsiva. Os testes realizados, incluindo dois voos acompanhados em tempo real, confirmaram a capacidade da solução de monitorar o deslocamento das aeronaves, apresentando informações de altitude, velocidade e precisão GPS durante toda a missão. O módulo de histórico de voos se mostrou eficiente na representação das rotas completas, permitindo análises posteriores de desempenho operacional.

A aplicação representa uma base sólida para evolução futura. Entre as possíveis extensões destacam-se o desenvolvimento de alertas automáticos, relatórios de desempenho, integração com novos sensores de telemetria, aprimoramento da gestão administrativa e adaptação da solução para dispositivos móveis. Tais melhorias podem ampliar o alcance da ferramenta e fortalecer sua contribuição para o processo de modernização tecnológica das operações aéreas do CTA.

Referências

BRASIL. Ministério da Ciência, Tecnologia e Inovações. **Estratégia Brasileira para a Transformação Digital (E-Digital).** Brasília, DF, 2022.

ELIAS, Elias Nasr Naim; FERNANDES, Vivian de Oliveira. **Qualidade dos Dados Geoespaciais do OpenStreetMap para os indicadores de Acurácia Posicional, Acurácia Temática e Completude**. Geografia (Londrina), Londrina, v. 30, n. 2, p. 255–275, 2021.

FINK, Gil; FLATOW, Ido. **Pro Single Page Application Development: using Backbone.js and ASP.NET**. 1. ed. Berkeley, CA: Apress, 2014.

FIELDING, Roy T. et al. **Hypertext Transfer Protocol – HTTP/1.1. RFC 2616**: Internet Engineering Task Force, 1999.

FLANAGAN, David. **JavaScript: o guia definitivo**. 6. ed. Porto Alegre: Bookman, 2013.

FRANKLIN, Gabriel L.; HOLANDA, Maristela T. de. **OpenStreetMap: uma análise acerca da evolução de dados geográficos colaborativos no Brasil**. In: CONFERÊNCIA IBÉRICA DE SISTEMAS E TECNOLOGIAS DE INFORMAÇÃO (CISTI), 12., 2017, Lisboa. Anais [. . .], 2017.

GEEWAX, J. J. **API Design Patterns**. Shelter Island, NY: Manning Publications, 2021.

HAVERBEKE, Marijn. **Eloquent JavaScript: a modern introduction to programming**. 3. ed. San Francisco: No Starch Press, 2018.

LAUDON, Kenneth C.; LAUDON, Jane P. **Sistemas de informação gerenciais.** 11. ed. São Paulo: Pearson, 2014.

LAURET, Arnaud. **The Design of Web APIs**. Shelter Island, NY: Manning Publications, 2019.

MARANHÃO. **Relatório de atividades e horas de voo realizadas pelo Centro Tático Aéreo do Maranhão – CTA/MA: período janeiro a novembro de 2020.** São Luís, MA, 2020.

MEYER, Eric A. **Cascading Style Sheets: the definitive guide.** 3. ed. Sebastopol: O'Reilly, 2006.

MOZILLA. **MDN Web Docs: Uma visão geral do HTTP**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em:

25 nov. 2025.

OPENSTREETMAP WIKI. **About OpenStreetMap**. Disponível em: <https://wiki.openstreetmap.org/wiki/About_OpenStreetMap>. Acesso em: 25 nov. 2025.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software: uma abordagem profissional.** 8. ed. Porto Alegre: AMGH, 2016.

SCOTT, Emmit A., Jr. **SPA Design and Architecture: understanding single page web applications**. Shelter Island, NY: Manning Publications, 2015.

THOMAS, Mark Tielens. **React in Action**. Shelter Island, NY: Manning Publications, 2018.

UNITED STATES. National Coordination Office for Space-Based Positioning, Navigation, and Timing. **GPS.gov: official U.S. Government information about the Global Positioning System (GPS) and related topics**. Washington, DC, [20–]. Disponível em: <https://www.gps.gov/>. Acesso em: 25 nov. 2025.

VIANA, Ana Cristina Aguilar. **Transformação digital na administração pública: do governo eletrônico ao governo digital.** Revista Eurolatinoamericana de Derecho Administrativo, Santa Fe, v. 8, n. 1, p. 115–136, jan./jun. 2021.

WORLD WIDE WEB CONSORTIUM. **HTML 4.01 Specification. W3C Recommendation, 24 Dec. 1999**: W3C, 1999.