MCBusão: Uma plataforma PWA para acesso a informações sobre rotas de ônibus na cidade de Monte Carmelo

Emmanuel de França Antunes Reis

Universidade Federal de Uberlândia
Faculdade de Computação
Bacharelado em Sistemas de Informação

Monte Carmelo - MG
2023

Emmanuel de França Antunes Reis

MCBusão: Uma plataforma PWA para acesso a informações sobre rotas de ônibus na cidade de Monte Carmelo

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Prof. Dr. Rafael Dias Araújo

Monte Carmelo - MG
2023

Dedico este trabalho a todos os estudantes da UFU principalmente do campus de Monte Carmelo, e espero que por meio deste eles possam desfrutar de um uso mais prático dos ônibus.

Agradecimentos

Agradeço primeiramente a Deus, pois Ele que nos dá a possibilidade de fazermos o que fazemos.

Agradeço também meus pais Cybele Vanuza e João Edson, que com certeza sem os esforços deles eu não estaria aqui.

Agradeço a minha esposa Débora Araújo que durante um período em que eu estava desanimado nunca desistiu de mim e sempre me motivou para continuar com este trabalho.

Agradeço meu professor orientador Rafael Dias por também não desistir, sempre me apoiando e me incentivando a continuar.

"A vida não é sobre quão duro você é capaz de bater, mas sobre quão duro você é capaz de apanhar e continuar indo em frente."

(Rocky Balboa)

Resumo

Nos últimos anos, a evolução tecnológica tem sido rápida, transformando os celulares em instrumentos multifuncionais essenciais para comunicação e informação. Em cidades como Monte Carmelo, que recebem muitos novos habitantes, especialmente jovens, a falta de serviços de apoio e conhecimento sobre a cidade pode dificultar a adaptação, evidenciando a necessidade de soluções tecnológicas para facilitar a integração desses indivíduos. Assim, este trabalho discorre em cima da construção de um sistema *WEB* utilizando PWA, a fim de disponibilizar aos alunos da Universidade Federal de Uberlândia do Campus de Monte Carmelo, todas as informações pertinentes aos ônibus que circulam na cidade, como localização dos pontos, horários, avisos, etc. Será abordado toda a parte teórica e prática de como este sistema foi desenvolvido, as tecnologias utilizadas, tanto de front-end como, HTML, CSS e Java, quanto back-end com NodeJS, MongoDB e Redis além da infraestrutura em cloud. Também é apresentado detalhadamente toda a parte de engenharia de software, desde o levantamento dos requisitos, modelagem de dados e casos de uso até a implantação e instalação da aplicação, por fim é apresentado os resultados obtidos após a implantação da primeira versão da aplicação.

Palavras-chave: ITS, Ônibus, PWA, Cloud, Engenharia de Software.

Lista de ilustrações

Lista de siglas

APTS Sistema Avançado de Transporte Público

ATIS Sistemas Avançados de Informação ao Viajante

ATMS Sistemas Avançados de Gerenciamento de Tráfego

AVCS Sistemas Avançados de Controle Veicular

CLI Interface de Linha de Comando

CSS Folhas de Estilo em Cascata

CVO Operação de Veículos Comerciais

ETC Coleta Eletrônica de Pedágio

HTML Linguagem de Marcação de HiperTexto

HTTPS Protocolo de Transferência de Hipertexto Seguro

IaaS Infraestutura como Serviço

IFPI Instituto Federal de Educação, Ciência e Tecnologia do Piauí

ITS Sistema de transporte Inteligente

INPI Instituto Nacional da Propriedade Industrial

PaaS Plataforma como Serviço

PDF Formato de Documento Portátil

PWA Aplicação Web Progressiva

SIU Sistema de Informação ao Usuário

SaaS Software como Serviço

SSL Camada de Soquete Seguro

UFF Universidade Federal Fluminense

UFU Universidade Federal de Uberlândia

URL Localizador Uniforme de Recursos

W3C World Wide Web Consortium

Sumário

Capítulo 1

Introdução

A tecnologia tem se mostrado em uma evolução cada vez mais rápida durante os últimos anos. Um aparelho celular que originalmente foi construído para realizar o contato entre duas pessoas em lugares distintos do mundo através da fala, hoje se tornou o principal instrumento utilizado para recepção de informações de qualquer que seja a fonte, comunicação em tempo real, sendo ela através de vídeo, fala ou texto entre duas ou mais pessoas (JUNIOR; LIMA; CUNHA, 2014).

Entretanto, muitas vezes, esse potencial que a tecnologia oferece não é utilizado, por exemplo, nas cidades interioranas, é possível encontrar com facilidade estabelecimentos que ainda utilizam recursos em papel, como comanda, nota de serviço, dentre outros.

A evolução da tecnologia tem também como objetivo auxiliar e facilitar tarefas para o usuário, um sistema de pedidos para um restaurante, traz ao garçom uma maior facilidade ao realizar um pedido com apenas alguns cliques, ao invés de ser necessário escrever tudo que lhe é pedido. Carros autônomos já estão circulando as ruas possibilitando a execução de outras tarefas mais importantes durante o período de trânsito, robôs aspiradores que também passam pano, auxiliam na faxina diária, entre outros facilitadores que nos proporcionam mais tempo para resolver outros problemas que não comuns no dia-a-dia.

Ao que se refere este trabalho, será disponibilizado aos habitantes da cidade de Monte Carmelo, em especial os estudantes da UFU nesta cidade, o que ainda não é oferecido a eles de uma forma descomplicada, a facilidade de encontrar em um único lugar todas as informações necessárias em relação aos ônibus que circulam na cidade.

1.1 Motivação

A cada semestre Monte Carmelo por conta da UFU recebe centenas de novos habitantes. Em sua maioria jovens adolescentes saindo de casa pela primeira vez, enfrentando novos desafios e iniciando a jornada à vida adulta, vida de responsabilidades. Uma das dificuldades para a adaptação em uma nova cidade é a falta de serviços de apoio e o conhecimento sobre ela, no estudo realizado por (CANHA, 2009), mostra que um dos principais

auxílios que se pode ter para uma melhor adaptação, é alguém mostrar a cidade para a pessoa que está se mudando.

Contudo nem sempre é possível ter alguém para realizar esta ação, assim ficando a cargo do aluno buscar informações para apoiá-lo nessa jornada. Com papel principal o transporte coletivo pode contribuir, porém atualmente não se tem muitas informações disponíveis que facilitam a utilização deste na cidade, isto acontece por ser uma cidade pequena em um país em desenvolvimento onde os processos de automatização e informatização ainda estão em crescimento.

A linha Intercampi, ofertada pela própria UFU, tem os seus horários informados em um aplicativo institucional disponível aos estudantes, enquanto há uma linha de ônibus municipal, que é ofertada pela prefeitura da cidade, cujos horários das rotas são disponibilizadas por meio de um arquivo PDF que circula de tempos em tempos por e-mail e ou mensagens de aplicativos de conversa, sendo que em nenhum desses locais oferta precisamente a localização dos pontos em uma mapa.

Visto a necessidade aliada à evolução e importância da tecnologia, identificou-se a oportunidade de aplicar o desenvolvimento da solução que discorrerá neste trabalho.

1.2 Hipótese

O sistema desenvolvido neste trabalho é uma Aplicação Web Progressiva (PWA) que deverá possibilitar o acesso as informações dos ônibus que circulam na cidade de Monte Carmelo para os estudantes da Universidade Federal de Uberlândia com clareza e facilidade, obtendo informações de uso diário e possibilitando a propagação via notificação de alertas relacionados aos ônibus.

Sendo assim a hipótese se baseia na usabilidade do sistema e a praticidade do acesso a informação, esperando acessos relevantes diários, principalmente no início dos semestres. Não só a quantidade de acessos será relevante quanto também as sugestões e opiniões dos usuários, que serão indagados a partir de um formulário na própria interface da aplicação em relação as hipóteses levantadas, usabilidade e praticidade.

1.3 Objetivos

O objetivo geral deste trabalho é criar um sistema de informação que permita que a comunidade acadêmica da Universidade Federal de Uberlândia (UFU) no campus Monte Carmelo e os habitantes possam acessar informações sobre rotas, horários e eventuais problemas das linhas de ônibus que atendem a cidade.

Para isso, os seguintes objetivos específicos foram traçados:

a) Projetar e desenvolver um sistema permitindo a população de Monte Carmelo acessar de forma rápida as informações do ônibus, como, ponto (com localização),

horário, sentido, e poder relatar incidentes, centralizando a informação em um único lugar;

b) Ofertar aos usuários a localização dos ônibus em tempo real;

c) Disponibilizar uma versão, como prova de conceito, para uso público;

d) Auxiliar novos moradores e estudantes da UFU do campus de Monte Carmelo a adaptação, tendo fácil acesso a informação;

e) Contribuir com a evolução tecnológica na cidade de Monte Carmelo.

1.4 Organização da Monografia

Este trabalho está divido em cinco capítulos, sendo o primeiro uma introdução e apresentação das hipóteses, motivações e objetivos. Nos segundo capítulo será apresentado a parte teórica do trabalho e alguns trabalhos relacionados. No terceiro capítulo veremos a proposta do trabalho colocando em prática o que foi discutido no capítulo dois, e no quarto capítulo temos principalmente os resultados coletados dessa proposta. Por fim, no quinto capítulo, serão apresentadas as considerações finais sobre este trabalho, bem como o que pode ser acrescentado e continuado no futuro.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta o contexto em que a aplicação proposta está inserida, assim como, as tecnologias e ferramentas utilizadas para o desenvolvimento. Além disso, também são apresentados trabalhos relacionados encontrados na literatura.

2.1 Sistema de Informação ao Usuário

A informatização do transporte coletivo tem sido cada vez mais necessária e vem ganhando visibilidade, existem vários projetos ao redor do mundo, (SILVA, 2000) apresenta sistemas já implantados em diversos países e nos introduz ao que abrange o Sistema de transporte Inteligente (ITS), que promovem a segurança e eficiência nas operações do transporte.

O ITS está subdividido em seis categorias

- Sistemas Avançados de Gerenciamento de Tráfego (ATMS) que emprega tecnologias avançadas para o controle do tráfego;
- Sistemas Avançados de Informação ao Viajante (ATIS) utiliza de tecnologia para atualizar o usuário com informações relevantes ao trânsito e meio ambiente;
- Sistemas Avançados de Controle Veicular (AVCS) que auxiliam os condutores a manterem a segurança no trânsito através de monitoramento da dirigibilidade e possível tomada de ação evitando acidente;
- Operação de Veículos Comerciais (CVO) operam para melhoria no gerenciamento em geral como por exemplo rotas, dos serviços de carga;
- Coleta Eletrônica de Pedágio (ETC) buscam a eficiência na cobrança de pedágios;
- por fim o que será abordado neste trabalho o Sistema Avançado de Transporte Público (APTS), este propõe a melhoria dos sistemas de transporte público beneficiando principalmente os usuários, dividido em cinco categorias Sistemas de

gerenciamento de frota, SIU, Sistemas de pagamento eletrônico, Gerenciamento da demanda de transporte e Veículos inteligentes de transporte coletivo.

O Sistema de Informação ao Usuário (SIU) uma das categorias do APTS, traz em sua essência o poder ao usuário de extrair todo o potencial que o transporte coletivo oferece, obter a informação de forma rápida e eficaz, sensibiliza o usuário a confiar no produto que está sendo oferecido, reforça a segurança e qualidade instigando a utilização (NASCIMENTO; LIMA, 2017).

2.2 Tecnologias Utilizadas

Serão apresentadas as tecnologias utilizadas no front-end, back-end e também na infraestrutura.

2.2.1 Front-End

Foi utilizado as tecnologias básica para o desenvolvimento de uma aplicação *WEB*.

2.2.1.1 HTML

A Linguagem de Marcação de HiperTexto (HTML) é uma linguagem de marcação e o padrão oficial da internet, adotado pela W3C, utilizado para estruturação de texto. A principal funcionalidade é a criação de páginas *WEB*, criado por Tim Berners-Lee segue um sistema de hipertexto, um texto fazendo referências de outro texto podendo ser acessado instantaneamente. Devido a limitação de não poder ter funcionalidades dinâmicas o HTML não é considerado uma linguagem de programação (LONGEN, 2023).

A Figura 1 apresenta arquivo básico de um HTML, contendo as principais tags, `<!DOCTYPE>`, `<html>`, `<head>` e `<body>` (KELLY, 2020).

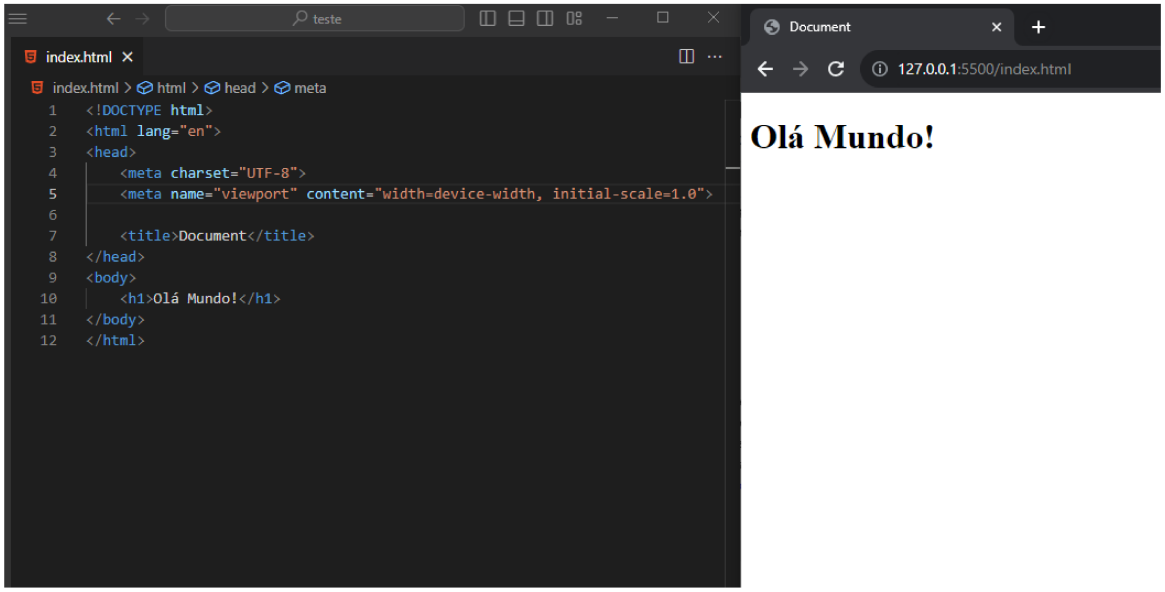

Figura 1 – Arquivo HTML básico

- A tag `!DOCTYPE` informa ao navegador qual a versão do HTML está sendo usada.
- A tag `html` é a principal do documento, encapsula todas as outras tags e define a linguagem do documento que por padrão vem "en"(Inglês).
- A tag `head` define o cabeçalho do documento, este elemento não é apresentado no navegar para o usuário, dentro dele é adicionado metadados e informações importantes sobre o documento, como por exemplo a tag `<title>` que é o título, este é apresentado na aba do navegador.
- A tag `body` contém tudo que será apresentado para o usuário no navegador, no exemplo temos uma outra tag `<h1>` que é uma tag de texto com a seguinte informação "Olá Mundo!".

2.2.1.2 CSS

O Folhas de Estilo em Cascata (CSS) é uma linguagem de estilo, ou seja, ela descreve como será apresentado visualmente um elemento (CONTRIBUTORS, 2022), decorando os elementos da linguagem de marcação, como o HTML.

Usando o mesmo exemplo do arquivo básico de HTML, é possível adicionar a tag `<style>` a tag `<head>` e definir um estilo para as informações que serão apresentadas, outra possibilidade também é criar um arquivo com a extensão .css e adicionar através da tag `<link>` no `head`. Conforme apresentado na Figura 2 foi adicionado através da classe "cor-azul", a cor azul ao texto que antes estava padrão, na cor preta.

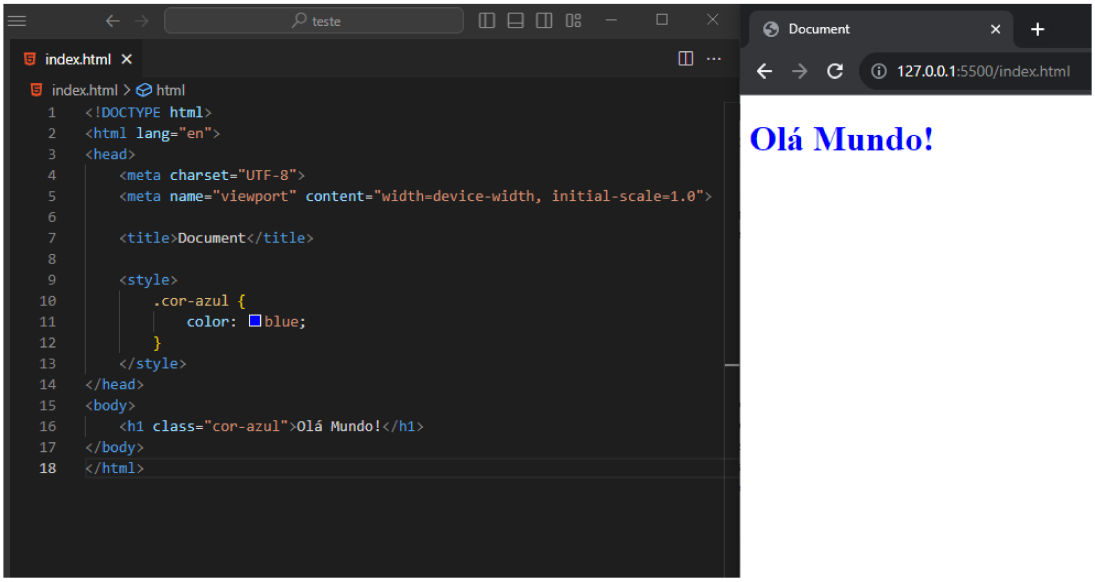

Figura 2 – Texto com estilo, usando css

2.2.1.3 JavaScript

O JavaScript é uma das principais linguagens de programação utilizadas hoje em dia, inicialmente criada para criar páginas interativas na *WEB* que somente com HTML não é possível operando apenas do lado cliente, hoje ela é utilizada também no lado servido, devido ao seu amadurecimento (AWS, 2023a).

A partir da tag `<script>` ou criando um arquivo com a extensão .js e importando com a tag `<link>` adiciona-se dinamicidade à página, no exemplo na Figura 3 mostra o estado inicial, onde a frase "Olá Mundo!"está com a cor azul definida pela classe de estilo ".cor-azul", ao lado o código javascript que atribui ao elemento texto, tag `<h1>`, um evento de *click*, que ao ser executada esta ação a classe de estilo "cor-azul"alterna, ou seja, se ela estiver presente no elemento, é removida, ou, se ela não estiver presente no elemento, é adicionada, assim na Figura 4 após o *click* no elemento o resultado é a cor padrão preta, pois foi removida a classe de estilo "cor-azul".

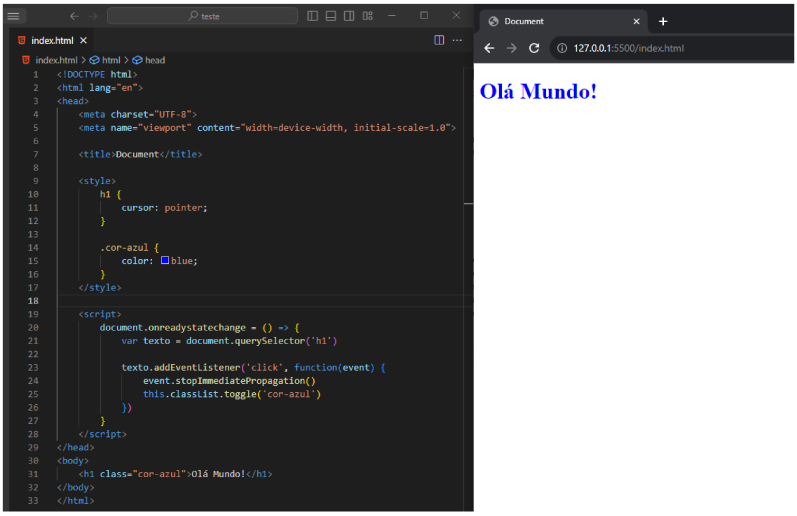

Figura 3 – Adicionando dinamicidade à página

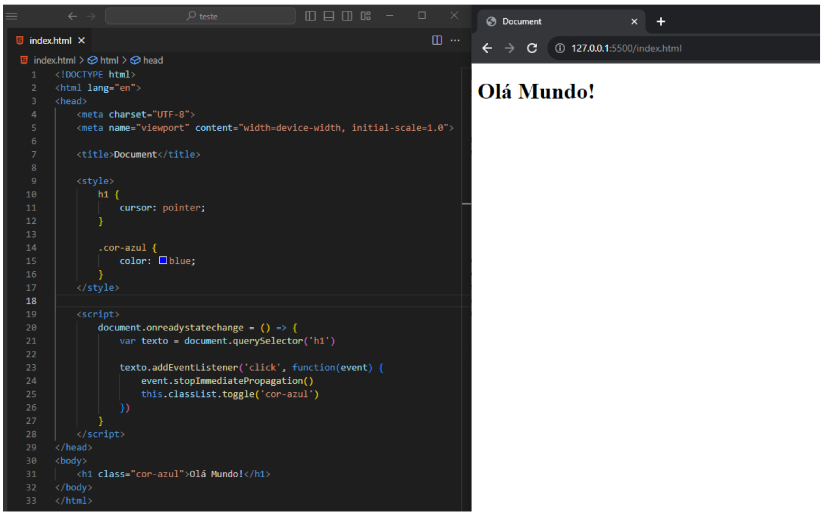

Figura 4 – Adicionando dinamicidade à página ao clicar

2.2.2 Back-End

O servidor foi desenvolvido em NodeJS e para a camada de dados foi utilizado o MongoDB <https://www.mongodb.com/> e o Redis.

2.2.2.1 NodeJS

O NodeJS é um ambiente que permite a execução do JavaScript sem a dependência de um navegador, com ele é possível executarmos código JavaScript no servidor. Para o projeto a utilização dele traz uma vantagem que nos permite ter apenas uma linguagem no ecossistema da nossa aplicação, tanto do lado do cliente (navegador/front-end), quanto do lado do servidor(back-end) (MELO, 2021).

2.2.2.2 Redis

Redis é um armazenador de estrutura de dados em memória de chave-valor open-source com diversas funcionalidades como banco de dados, cache, pubsub, etc (AWS, 2023b). O principal objetivo é reduzir a latência das aplicações com o consumo dos dados, principalmente os mais acessados e menos voláteis, salvando-os em memória com acesso rápido.

Um exemplo simples de utilização do redis como cache, na Figura 5 através da Interface de Linha de Comando (CLI) acessando em localmente o redis pela porta padrão 6379, foram criadas duas chaves com o comando *SET* com seus respectivos valores e recuperadas utilizando o comando *GET*, há ainda uma exemplificação do retorno quando a chave não existe e a forma de recuperar todas as chaves criadas pelo comando *KEYS*.

```
# redis-cli
127.0.0.1:6379> SET CHAVE_1 valor_1
OK
127.0.0.1:6379> SET CHAVE_2 valor_2
OK
127.0.0.1:6379> GET CHAVE_1
"valor_1"
127.0.0.1:6379> GET CHAVE_3
(nil)
127.0.0.1:6379> GET CHAVE_2
"valor_2"
127.0.0.1:6379> KEYS CHAVE
(empty array)
127.0.0.1:6379> KEYS CHAVE*
1) "CHAVE_2"
2) "CHAVE_1"
127.0.0.1:6379>
```

Figura 5 – Redis CLI

2.2.2.3 MongoDB

MongoDB é um banco de dados não relacional open-source de alta performance, é orientado a documentos JSON que compõe coleções, e esses documentos são armazenando-os em modo binário (BSON)(JUNIOR, 2022).

Para fins de comparação com um banco de dados relacional como MySQL, Postgres, etc, coleções se assemelham com tabelas e os documentos com os registros presentes em cada tabela. O MongoDB fornece para o desenvolvimento maior dinamicidade, visto que os documentos não precisam seguir um esquema, diferente de um banco relacional em que a tabela possui um esquema e uma alteração afeta todos os registros contidos nela.

A Figura 6 é um exemplo de um documento que foi criado na coleção "bus", esta coleção não possui um esquema definido e este documento pode ser facilmente alterado atomicamente.

Figura 6 – Exemplo de documento armazenado no MongoDB

2.2.3 Aplicação *WEB* Progressiva (PWA)

Uma PWA é um pouco de dois tipos de aplicação, a aplicação *WEB* e a aplicação nativa para um celular. Isso significa que utilizando as tecnologias citadas anteriormente HTML, CSS e JavaScript, uma aplicação padrão *WEB* fica apta a utilizar alguns recursos nativos, que normalmente estariam disponíveis apenas para aplicações desenvolvidas exclusivamente para a plataforma Android ou IOS, como, permitir o usuário adicionar a página *WEB* na tela principal do celular deixando mais fácil a localização da aplicação, evitando a necessidade de abrir o navegador para pesquisar o site ou ter que digitar o Localizador Uniforme de Recursos (URL), receber notificações como qualquer outro aplicativo, utilizar a aplicação *offline* etc. Podemos considerar o custo para o desenvolvimento da aplicação uma outra vantagem para a PWA, com poucos passos de configurações a mais é possível tornar sua aplicação *WEB* em PWA além de apenas desenvolver uma única vez para três plataforma distintas, *WEB*, Android e IOS, não necessitando o desenvolvimento e implantação diferente para cada plataforma (DIGITAIS, 2022).

2.2.4 Infraestrutura

A aplicação foi disponibilizada na internet através de uma plataforma de hospedagem que simplifica a publicação de aplicativos e sites, o Fly.io <https://fly.io/> utiliza das principais *cloud's* públicas e possui uma solução gratuita para os desenvolvedores publicarem suas aplicações.

Cloud Computing ou computação em nuvem são recursos de infraestrutura com acesso remoto, pela internet, na maioria dos casos espalhados por todo o mundo (VALINOR, 2022). Existem alguns tipos de *cloud*:

- *Cloud* privada ou *onpremises* em que a organização é responsável pela manutenção do software e infraestrutura, garantindo ao não compartilhamento dos seus recursos.
- *Cloud* pública em que uma empresa disponibiliza os recursos de infraestrutura para outras organizações, tendo como responsabilidade a manutenção desses recursos ape-

nas, e a organização apenas com a manutenção do software, assim temos um modelo compartilhado de responsabilidades. As principais *cloud's* públicas são AWS, Google Cloud, Microsoft Azure, Alibaba, etc.

- ❑ *Cloud* hibrida onde a organização possui uma solução de implantação de uma *cloud* pública e uma privada, podendo essas estarem ou não conectadas.

As *cloud's* públicas, oferecem diversos níveis de serviço gerenciados ou não como Infraestutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS), a Figura 7 apresenta os serviços e a responsabilidade, em verde a responsabilidade de gerenciamento e manutenção de quem está contratando e em azul da provedora de *cloud*, por exemplo, em uma contartação de um serviço do tipo PaaS na *cloud* Amazon Web Services, o contratante deve zelar pela aplicação e pelos dados, enquanto a AWS é responsável pelo resto.

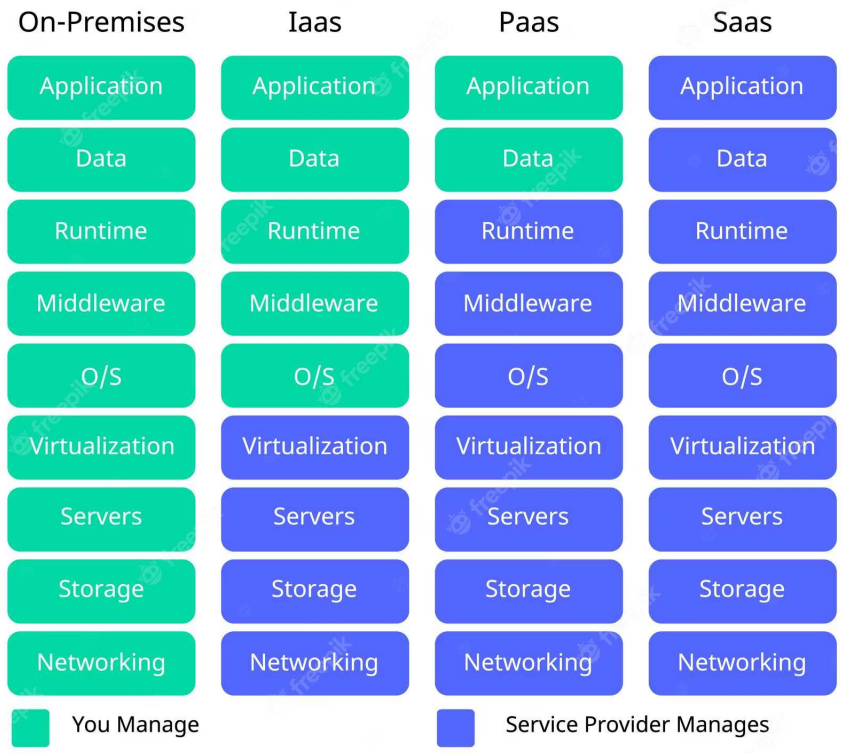

Figura 7 – Níveis de serviços e gerenciamento de cloud

2.3 Trabalhos Relacionados

O aplicativo Topin desenvolvido por estudantes da Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI) (FERREIRA; SILVA; NUNES, 2018) possui a mesma motivação, oferecer uma melhor experiência ao usuário do transporte público da cidade,

no caso Picos-PI, fornecendo as informações necessárias, rotas, pontos, horários. O aplicativo foi disponibilizado apenas para a plataforma Android e tem como diferencial a disponibilização em tempo real da localização dos ônibus, algo que foi previsto, porém não implementado na primeira versão do nosso sistema.

Na cidade do Rio de Janeiro-RJ foi implementado também como prova de conceito o aplicativo TrackBus (VAZ; AMARAL, 2017) através do trabalho de conclusão de curso da Universidade Federal Fluminense (UFF), com objetivo similar, porém contendo algumas funcionalidades a mais devido a diferença do público a ser atingido, como turistas, este aplicativo além de apresentar ao usuário as informações de rota, pontos, horários e localização dos ônibus, também possui os pontos turísticos da cidade e alerta os usuários quando o ônibus está próximo. Apesar de ter a sua construção em um framework híbrido podendo ser disponibilizado nas plataformas Android e IOS, a sua implantação foi apenas para Android devido ao custo elevado para disponibilização para IOS.

Tendo como proposta a colaboração dos usuários o aplicativo Crowdbus, sistema genérico que não foi implantado em lugar específico, oferece aos usuários todas as informações como localização, rotas, horários, problemas, tudo alimentado pela utilização dos próprios usuários, ou seja, não há uma base de dados previamente populada, a partir da colaboração dos usuário o aplicativo vai disponibilizando essas informação, como por exemplo a localização do ônibus é acessada através do dispositivo de uma pessoa que está utilizando este transporte (JUNIOR; LIMA; CUNHA, 2014). Parecido com essa ideia de colaboração, temos no sistema a possibilidade do usuário informar caso ocorra algum problema afetando a utilização de algum ônibus.

Capítulo 3

Proposta

Este capítulo discorre sobre os detalhes de projeto e implementação da aplicação proposta. Primeiramente, são apresentados os requisitos, uma visão geral dos casos de uso e o modelo de dados. Na sequência, é apresentada uma visão geral da arquitetura proposta e detalhes específicos de implementação, como o cálculo de distância entre dois pontos e os registros de interação dos usuários com o sistema.

3.1 Atores

O sistema tem os seguintes atores:

a) Administrador: usuário com acessos e permissões, tendo a possibilidade de cadastro, alteração, remoção e leitura de ônibus, pontos, rotas, além de aprovação ou rejeição de alertas.

b) Usuário: usuário comum, capaz de visualizar a lista de todas as rotas cadastradas no sistema, alertas aprovados e cadastrar alertas.

3.2 Requisitos Funcionais

Requisitos funcionais descrevem as funcionalidades do sistema, são eles:

a) Autenticação (RF1): o sistema deve fornecer uma tela de login, onde o usuário deve prover suas informações de acesso, usuário e senha, para usufruir das funcionalidades da área logada.

b) Controle de sessão (RN2): o sistema deve ser capaz de controlar a seção do usuário, mantendo em cache um token de acesso, capaz de logar automaticamente quando não expirado, evitando a solicitação da senha ao usuário.

c) Cadastro de ponto de ônibus (RF3): o sistema deve disponibilizar na área logada uma tela para que o usuário possa realizar o cadastro de ponto de ônibus, assim como alteração exclusão e leitura.

d) Cadastro de rota (RF4): o sistema deve disponibilizar na área logada uma tela para que o usuário possa realizar o cadastro de rota, assim como alteração exclusão e leitura.

e) Cadastro de alerta (RF5): o sistema deve disponibilizar na área não logada uma tela para que qualquer usuário possa realizar o cadastro de alertas.

f) Cadastro de uso (RF6): o sistema deve ser capaz de armazenar quantidade de acessos por usuário, e uso dos ônibus a partir da resposta por fomulário.

g) Cadastro de ônibus (RF7): o sistema deve disponibilizar na área logada uma tela para que o usuário possa realizar o cadastro de ônibus, assim como alteração exclusão e leitura.

h) Listagem de alertas (RF8): o sistema deve disponibilizar na área logada uma tela listando os alertas cadastrados pendentes de ação (aprovação ou rejeição), assim como a possibilidade de aprovação ou rejeição desses alertas pelo usuário logado.

i) Listagem de rotas (RF9): o sistema deve disponibilizar na área não logada uma tela listando todas as rotas cadastradas.

j) Envio de notificação (RF10): o sistema deve ser capaz de enviar notificação para os usuários quando um alerta for aprovado.

k) Solicitação de feedback (RF11): o sistema deve ser capaz de solicitar aos usuários um retorno em relação a satisfação do uso da aplicação através de um formulário.

3.3 Requisitos Não Funcionais

Requisitos não funcionais descrevem atributos de qualidade do sistema, são eles:

a) Segurança (RNF1): o sistema deve ter uma área segura/protegida, onde somente usuários com acesso e permissões tenham a possibilidade de operá-lo.

b) Compatibilidade com navegadores (RNF2): o sistema deverá rodar em qualquer navegador, sem diferença de usabilidade.

c) Compatibilidade com dispositivos móveis (RNF3): o sistema deve ser capaz de operar em qualquer dispositivo móvel, de qualquer sistema operacional.

d) Usabilidade (RNF4): os usuários devem ser capaz de utilizar o sistema mesmo sem um treinamento, deve-se ter facilidade no uso.

3.4 Diagrama de Casos de Uso

Partindo dos requisitos descritos na seção anterior, foi possível fazer a modelagem das funcionalidades do sistema. A Figura 8 apresenta uma visão geral das funcionalidades e atores por meio do Diagrama de Casos de Uso do sistema proposto.

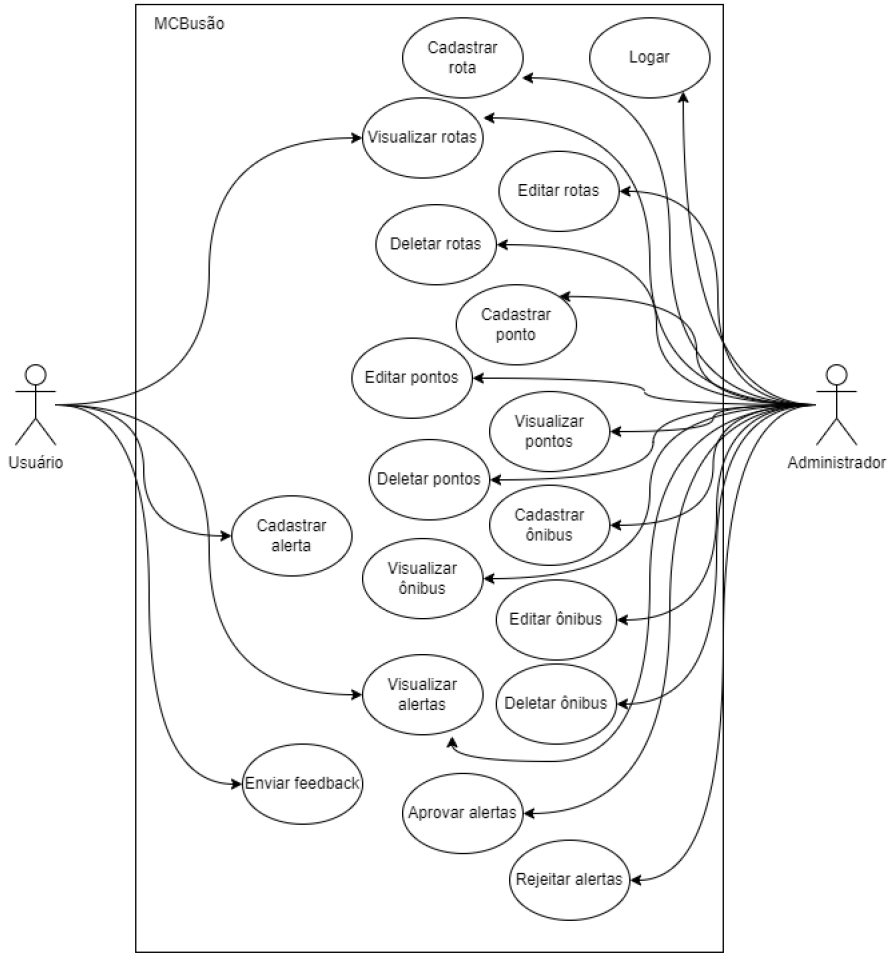

Figura 8 – Casos de uso

Podemos perceber que, dentre todas as funcionalidades, o usuário tem acesso a apenas quatro, sendo elas, visualizar rotas, visualizar alertas, cadastrar alertas e enviar *feedback*, as demais em sua grande maioria são protegidas, tendo o acesso apenas pela área logada, tendo como ator o perfil de administrador.

3.5 Modelo de Dados

Foi utilizado o Redis uma ferramenta de cache que armazenada dados em esquema de chave-valor, com a proposta de reduzir o consumo de serviços mais pesados e de uso frequente, como por exemplo a listagem dos pontos feita na página principal. Guardando esta informação pouco volátil em cache o consumo fica mais rápido pis já está na memória, diminuindo a latência da aplicação.

O banco de dados é composto por coleções de documentos. Esta foi a solução proposta devido a facilidade de alterações constantes nos esquemas das coleções, contribuindo com a dinamicidade do desenvolvimento, e também pelo nível gratuito que o MongoDB fornece em seu site.

As seguintes coleções foram criadas:

- bus: coleção que armazena documentos relacionados aos ônibus tendo como primeira versão a seguinte estrutura.
  a) _id: código de identificação gerado pelo próprio banco de dados.
  b) id: código de identificação gerado pelo sistema.
  c) bus: nome do ônibus.
  d) color: cor utilizada para identificação do ônibus no sistema.
  e) active: campo que determina se o ônibus está ativo ou não no sistema.

- routes: coleção que armazena documentos relacionados as rotas tendo como primeira versão a seguinte estrutura.
  a) _id: código de identificação gerado pelo próprio banco de dados.
  b) id: código de identificação gerado pelo sistema.
  c) bus: código do ônibus responsável pela rota.
  d) way: sentido da rota.
  e) spots: campo com nenhuma ou várias coleções compostas da seguinte forma
     – spot: código do ponto.
     – time: horário programado para o ônibus passar nesse ponto nessa rota.
  f) active: campo que determina se a rota está ativa ou não no sistema.

- spots: coleção que armazena documentos relacionados aos pontos tendo como primeira versão a seguinte estrutura.
  a) _id: código de identificação gerado pelo próprio banco de dados.
  b) id: código de identificação gerado pelo sistema.
  c) spot: nome do ponto.
  d) way: sentido do ponto.
  e) latitude: latitude do ponto.
  f) longitude: longitude do ponto.
  g) active: campo que determina se o ponto está ativo ou não no sistema.

- subscriptions: coleção que armazena documentos relacionados as subscrições para o recebimento de notificações tendo como primeira versão a seguinte estrutura.
  a) _id: código de identificação gerado pelo próprio banco de dados.
  b) fingerprint: código de identificação gerado pelo dispositivo utilizado pelo usuário.
  c) subscription: coleção com informações para envio de notificação seguindo o esquema
     – endpoint: url para o envio da notificação.

  - expirationTime: tempo de expiração da subscrição.
  - keys: chaves de criptografia e autenticação.

- ❏ users: coleção que armazena documentos relacionados aos usuários do sistemas tendo como primeira versão a seguinte estrutura.
  a) _id: código de identificação gerado pelo próprio banco de dados.
  b) id: código de identificação gerado pelo sistema.
  c) username: nome do ônibus.
  d) password: cor utilizada para identificação do ônibus no sistema.
  e) role: função do usuário no sistema.
  f) active: campo que determina se o usuário está ativo ou não no sistema.

- ❏ uses: coleção que armazena documentos relacionados aos usos dos sitema tendo como primeira versão a seguinte estrutura.
  a) _id: código de identificação gerado pelo próprio banco de dados.
  b) id: código de identificação gerado pelo sistema.
  c) date: data do uso.
  d) users: campo com uma ou várias coleções compostas da seguinte forma
     - fingerprint: código de identificação gerado pelo dispositivo utilizado pelo usuário.
     - qtd: quantidade de acesso no dia desse código de identificação.

- ❏ warnings: coleção que armazena documentos relacionados aos pontos tendo como primeira versão a seguinte estrutura.
  a) _id: código de identificação gerado pelo próprio banco de dados.
  b) id: código de identificação gerado pelo sistema.
  c) start: data e hora do início do alerta.
  d) bus: código do ônibus referente ao alerta.
  e) title: título do alerta.
  f) message: mensagem do alerta.
  g) end: data e hora do fim do alerta.
  h) fingerprint: código de identificação gerado pelo dispositivo utilizado pelo usuário que criou o alerta.
  i) pending: informa se o alerta está pendente de aprovação ou rejeição.
  j) approved: informa se o alerta foi aprovado.

- ❏ passangers: coleção que armazena documentos relacionados a utilização dos ônibus tendo como primeira versão a seguinte estrutura.

a) _id: código de identificação gerado pelo próprio banco de dados.

b) id: código de identificação gerado pelo sistema.

c) routeId: código da rota.

d) spotId: código do ponto.

e) use: informa se o usuário utilizou ou não(embarque/desembarque) o ônibus na rota e ponto informado.

f) fingerprint: código de identificação gerado pelo dispositivo utilizado pelo usuário.

❏ feedback: coleção que armazena documentos contendo os feedbacks dos usuários.

a) _id: código de identificação gerado pelo próprio banco de dados.

b) id: código de identificação gerado pelo sistema.

c) rating: número de 0 a 5 relacionado ao nível de satisfação do usuário em relação a aplicação, em que quanto maior melhor.

d) message: sugestões ou críticas feitas pelo usuário, pode ser preenchida ou não.

e) fingerprint: código de identificação gerado pelo dispositivo utilizado pelo usuário que deu o feedback.

3.6 Visão Geral da Arquitetura

O sistema proposto foi projetado e desenvolvido utilizando três camadas, dividindo responsabilidades de aplicação, de serviço e de dados, como mostra a Figura 9.

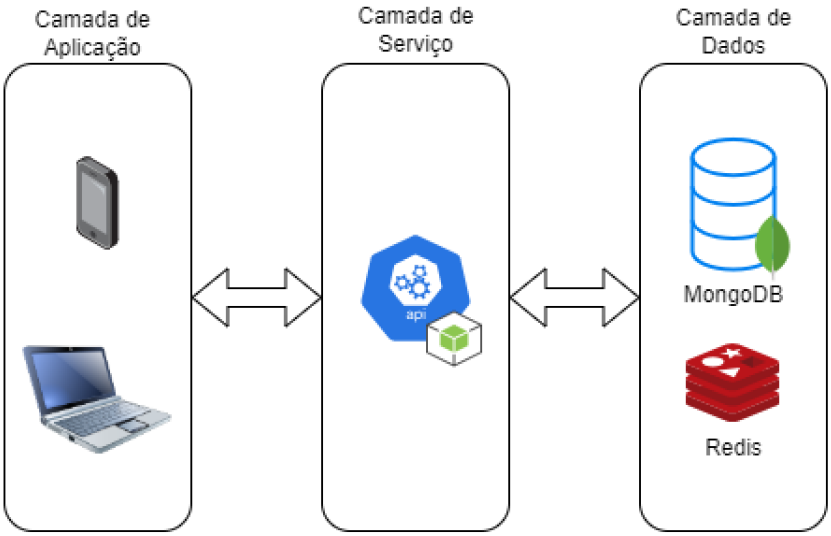

Figura 9 – Arquitetura do sistema

A camada de dados é a responsável por persistir todos os dados pertinentes ao sistema. Foi utilizado o MongoDB, um banco de dados não relacional baseado em documentos e

redis para cache. Já a camada de serviço tem o papel de intermediação entre a camada de dados e a camada de aplicação, provendo o que é necessário para a aplicação através dos dados armazenados na camada de dados. Na camada de aplicação, a interface *WEB* é disponibilizada para acesso dos usuários e administradores, onde é disponibilizada todas as informações de rotas, ônibus, pontos, localização, avisos. Essa camada foi desenvolvida para *WEB* e utilizado o recurso de PWA possibilitando a flexibilidade de ser acessado tanto por navegadores em notebooks e *smartphones*, como um aplicativo nativo nos *smartphones*, facilitando o uso.

Para utilizar a aplicação como uma PWA, é necessário algumas etapas, primeiramente adicionando dois arquivos de configuração "manifest.json" e "service-worker.js" no projeto, como mostra a Figura 10.

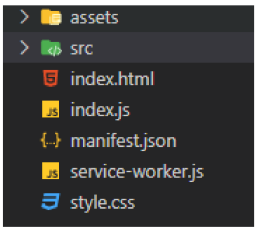

Figura 10 – Localização dos arquivos para PWA

O arquivo "manifest.json" contém informações relacionadas à aplicação, mostradas pela Figura 11. Esse arquivo é o responsável por transformar a aplicação *WEB* em uma aplicação instalável, seu conteúdo compõe de informações como, nome do aplicativo, cor principal, cor de fundo, ícone, tela de início e orientação do display.

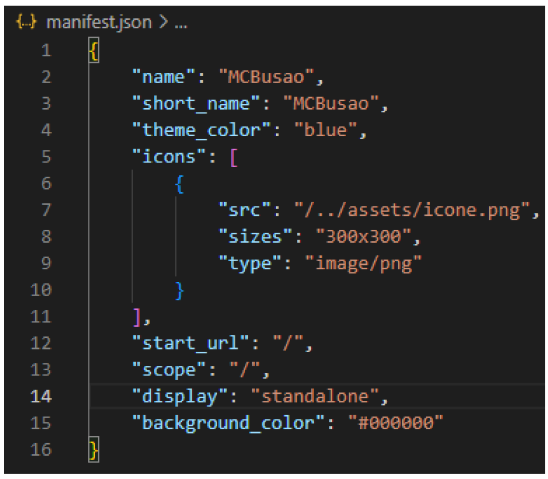

Figura 11 – Arquivo de metadados sobre a aplicação.

O arquivo "service-worker.js" é adicionado para instalação de um serviço executado em *background*, possibilitando adicionar recursos como cache, notificações etc. A Figura 12 mostra a função executada para instalar o serviço.

```
service-worker.js > ...
1   oninstall = () => {
2       console.log('used to register the service worker')
3
4       self.skipWaiting()
5   }
```

Figura 12 – Código de instalação de um *service worker*.

Por fim, no arquivo JavaScript principal da aplicação, é necessário registrar o serviço criado para que ele seja instalado na aplicação. Antes disso, é preciso verificar se o navegador que o usuário está utilizando tem compatibilidade com o recurso de *service worker*, como mostra a Figura 13. Além disso, também é necessário que a aplicação esteja em um ambiente seguro com certificado Camada de Soquete Seguro (SSL), seguindo os padrões para utilização do protocolo Protocolo de Transferência de Hipertexto Seguro (HTTPS).

```
index.js > ...
14      if ('serviceWorker' in navigator) {
15          navigator.serviceWorker.register('/service-worker.js', { scope: '/' })
16      }
```

Figura 13 – Código para registro de um *service worker* na aplicação.

Dada a compatibilidade e a instalação com êxito do *service worker*, o usuário já estará apto a utilizar os recursos que a PWA oferece, como a instalação da aplicação no dispositivo, indicada passo-a-passo pelas Figura 14, Figura 15 e Figura 16.

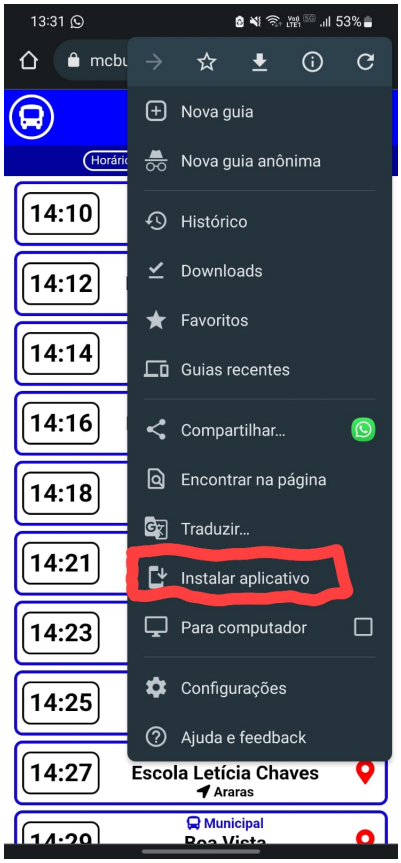

Figura 14 – Captura de tela que mostra a possibilidade de instalação da PWA no *smartphone*.

Figura 15 – Captura de tela que mostra a caixa de mensagem para confirmação da instalação do aplicativo.

Figura 16 – Captura de tela que mostra o ícone do aplicativo instalado.

3.7 Rotas e Proximidade de Pontos

Para poder mensurar a utilização dos ônibus, foi adicionada uma funcionalidade ao sistema para perguntar ao usuário se ele utilizou ou irá utilizar o ônibus, desembarcando ou embarcando em um ponto de ônibus de uma rota. Para isso, ao abrir a aplicação, é solicitada a permissão de acesso a sua geolocalização por meio de um recurso do navegador *WEB*, como mostra a Figura 17. Caso a permissão seja concedida, é calculado se a localização está próximo a algum ponto e horário de alguma rota. Em que caso positivo, o usuário recebe um alerta solicitando a confirmação de embarque ou desembarque.

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(({ coords: { latitude, longitude }}) =>
        this.#sendLocation({ latitude, longitude }),
    )
}
```

Figura 17 – Código para solicitar e recuperar informação de localização

Para solicitação dessa permissão, faz-se necessário acessar o componente de geolocalização já disponibilizado nos navegadores. Caso o navegador tenha compatibilidade com esse recurso, abrirá uma janela de confirmação, mostrada na Figura 18. No caso de permissão concedida, a informação de latitude e longitude é enviada para a camada de serviços, que realiza o cálculo de proximidade.

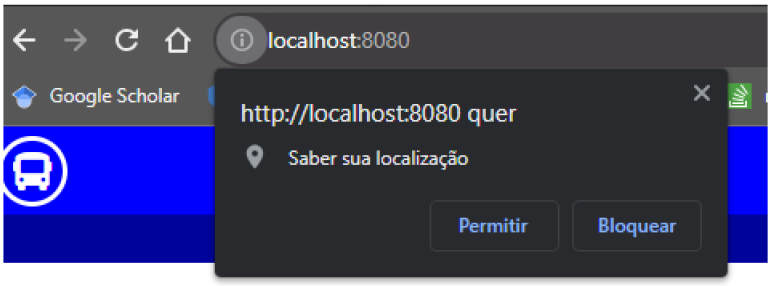

Figura 18 – Navegador solicitando acesso a localização

Foi utilizada para o cálculo a fórmula de Haversine Figura 19 (SINNOTT, 1984), uma das principais utilizadas na navegação para calcular a distância entre dois pontos em uma esfera a partir da latitude e longitude, devido ao fato da Terra não ser uma esfera perfeita, esta fórmula quando aplicada na Terra tem como resultado uma aproximação (CHAMBERLAIN, 1997).

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Figura 19 – Fórmula de Haversine

O trecho de código apresentado na Figura 20 possui as seguintes instruções:

1. Para cada ponto recuperado do banco, calcula-se primeiramente a lei dos Haversines, que permite ser calculado diretamente com os radianos da latitude e longitude dos dois pontos;

2. Com o valor resultante, a função arco seno é aplicada para obtenção da distância em quilômetro;

3. Se a distância for menor do que 100 metros, é apresentado para o usuário o alerta para ser respondido se foi ou irá utilizar o ônibus (embarque/desembarque).

```
for(const spot of spots) {
    const hav = Math.sin((rad(spot.latitude) - rad(latitude)) / 2) ** 2  +
                    Math.cos(rad(spot.latitude)) *
                    Math.cos(rad(latitude)) *
                    (Math.sin((rad(spot.longitude) - rad(longitude)) / 2) ** 2)
    const distance = radiusEarth * 2 * Math.atan2(Math.sqrt(hav), Math.sqrt(1 - hav))

    if(distance.toFixed(2) < 0.1) {
        return res.status(200).json({ routeId: spot.routeId, spotId: spot.id, busName
    }
}
```

Figura 20 – Trecho de código para cálculo de proximidade entre dois pontos em uma esfera.

3.8 Logs de Interações

Além do sistema solicitar a interação do usuário para armazenar os dados de uso dos ônibus explicado na seção anterior, há também o armazenamento da quantidade de uso do sistema por usuário. Ou seja, sempre que o usuário acessa a aplicação em determinado dia é computado mais um acesso. Ao carregar a lista dos pontos a serem apresentados na camada de aplicação, a camada de serviço captura o *fingerprint*, recebido pela camada de aplicação e envia para a camada de dados. Assim, é possível obter quantos acessos teve por dia e quantos usuário distintos utilizaram o sistema. O Diagrama de Sequência apresentado na Figura 21 mostra esse fluxo de coleta de log de uso dos usuários.

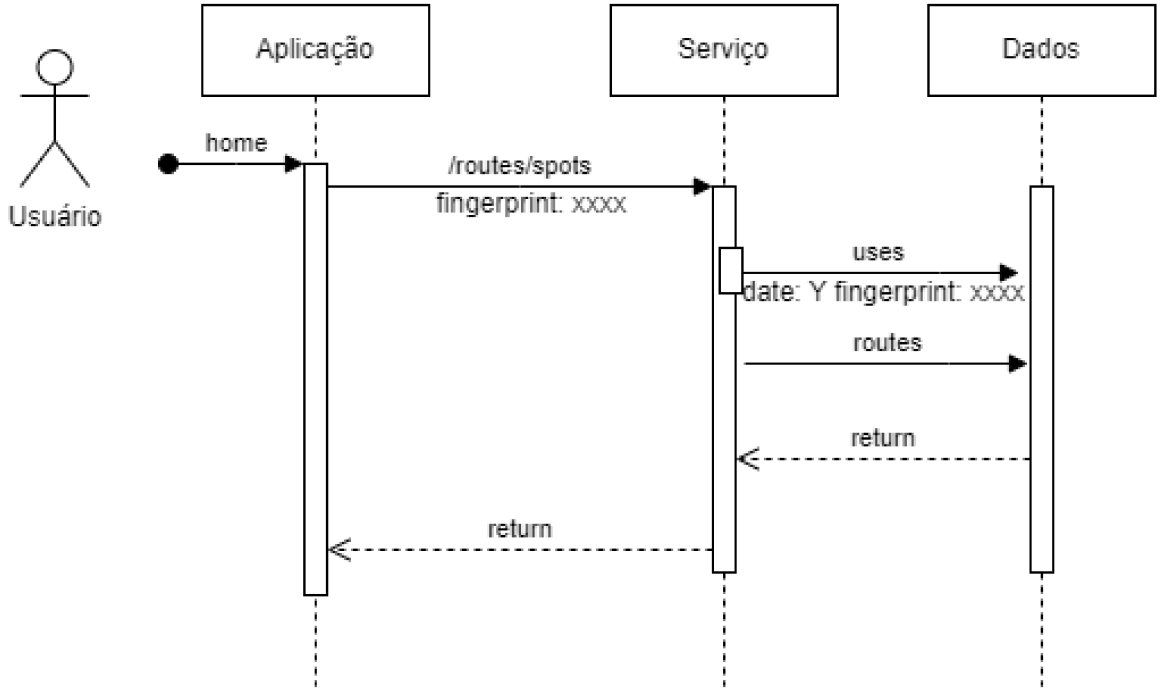

Figura 21 – Diagrama de sequência: Log de uso.

Para que a coleta dos dados tenha maior precisão e desprenda da necessidade do usuário criar um login para identificá-lo, foi utilizado uma biblioteca *Javascript* chamada FingerprintJS <https://fingerprint.com/> que utiliza os dados do navegador para criação de um *fingerprint*(espécie de um identificador), possibilitando uma maior acurácia em relação a informação de uso. Como foi utilizado a versão gratuita da biblioteca que possui uma acurácia menor na geração do *fingerprint*, ao ser gerado o primeiro, este é salvo no armazenamento local do navegador e assim sendo reutilizado sem a necessidade de ser gerado novamente pela biblioteca.

Capítulo 4

Resultados

Este capítulo apresenta os resultados obtidos com o trabalho proposto. Uma prova de conceito foi implementada e disponibilizada para a comunidade da UFU no campus de Monte Carmelo no final do semestre letivo de 2022/2 e início de 2023/1. Foram coletados dados de utilização do sistema pelos usuários e os resultados preliminares são apresentados.

4.1 Prova de Conceito

Como mencionado, foi implementada uma prova de conceito utilizando tecnologias *WEB* para disponibilização de uma versão funcional do sistema para uso e obtenção de *feedback*. A prova de conceito foi dividida em uma área de acesso público e uma área de acesso restrito, apresentadas nas próximas subseções.

4.1.1 Área de acesso público

A área de acesso público possui três telas disponíveis para os usuários. A tela principal, mostrada na Figura 22, é possível visualizar todos os horários das linhas com indicação do nome da linha, sentido e localização dos pontos e caso o ônibus tenha algum alerta ativo Figura 23. É possível aplicar filtro para visualização específica de cada linha e do sentido desejado. Ainda nessa tela, é possível acessar a tela de avisos.

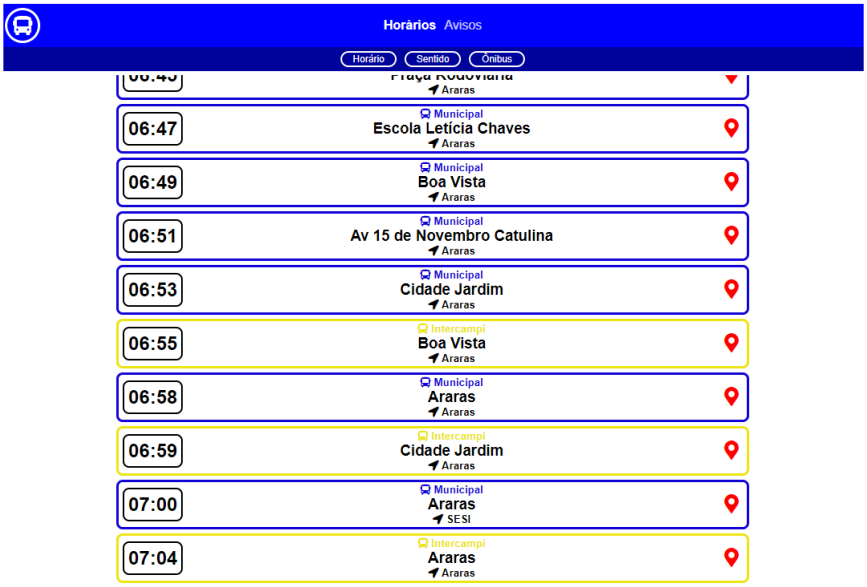

Figura 22 – Lista de horários

Figura 23 – Lista de horários ônibus com alerta ativo

A tela de avisos (ou alertas), mostrada na Figura 24, mostra todos os alertas ativos (ainda vigentes) e permite que os usuários criem alertas específicos para cada linha de ônibus Figura 25. O alerta contém um título (indicando o assunto do alerta), uma mensagem explicativa e uma previsão de retorno.

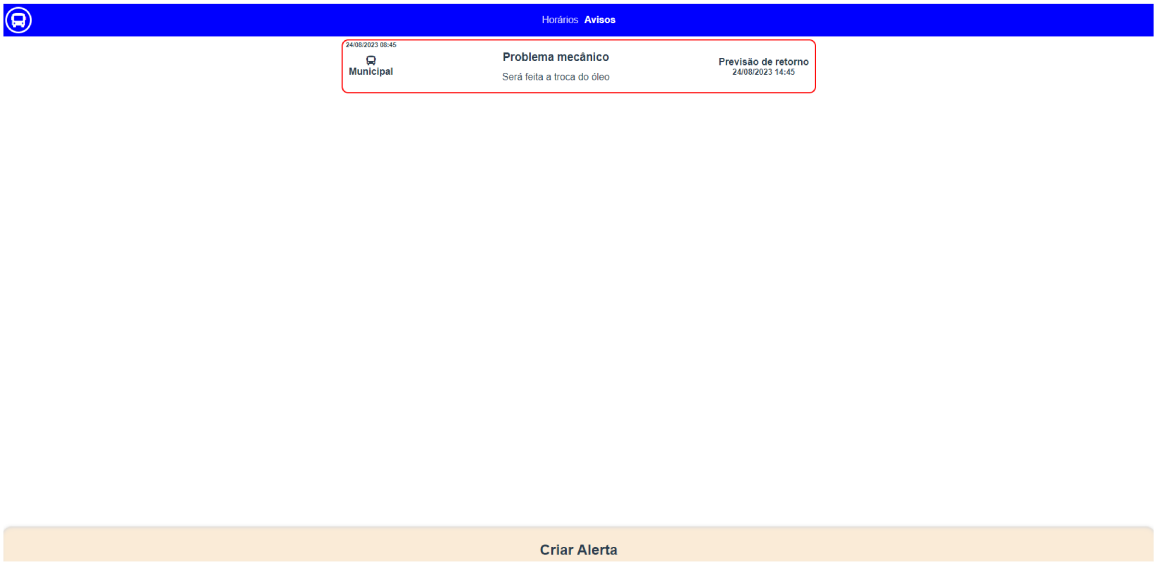

Figura 24 – Lista de alertas

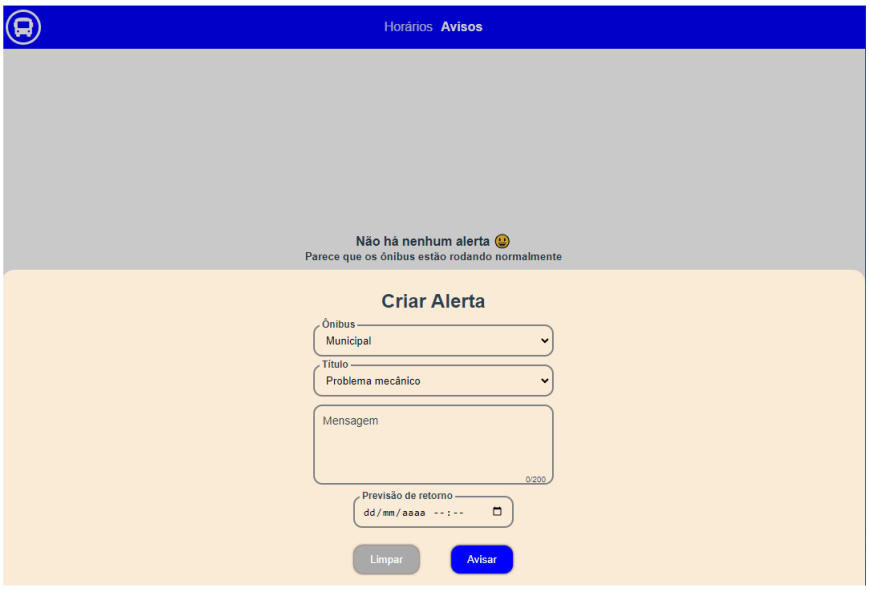

Figura 25 – Cadastro de alertas

Por fim, há também um formulário que aparece na tela principal para o usuário quando ele utiliza o aplicativo por mais de um dia. Esse formulário solicita uma avaliação do aplicativo, com estrelas (1 a 5) e opcionalmente um campo de texto para indicação de sugestões ou críticas, como mostra a Figura 26.

Figura 26 – Formulário de *feedback*

4.1.2 Área de acesso restrito

A área de acesso restrito foi criada para que usuários com o perfil de administrador possam executar algumas operações, está área não está disponível para acesso público, tem um difícil acesso para ser utilizada apenas pelos administradores. Primeiramente, o usuário administrador precisa informar suas credenciais (usuário e senha) para fazer login, como mostra a Figura 27.

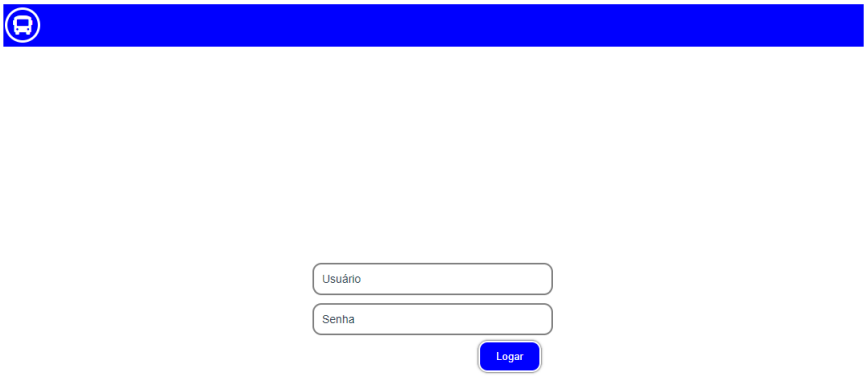

Figura 27 – Captura da tela de login

Na área logada, o usuário administrador terá, como tela principal, a visualização dos alertas criados pelos usuários que ainda não foram autorizados. É necessária uma aprovação prévia para que o alerta criado seja disponibilizado a todos os usuários. A

Figura 29 mostra exemplos de alertas pendentes de aprovação. Note que é possível aprovar ou rejeitá-los.

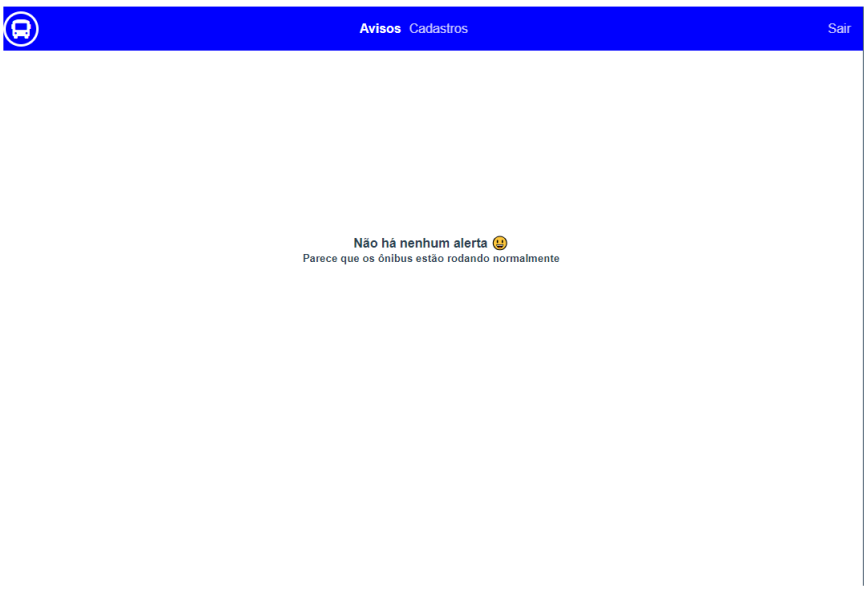

Figura 28 – Lista de alertas para aprovação sem alertas pendentes

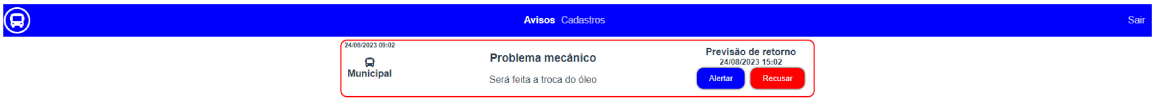

Figura 29 – Lista de alertas para aprovação com alertas pendentes

O usuário administrador também tem acesso às telas de cadastro que permitem as operações básicas de criação, alteração e exclusão de linhas de ônibus (Figura 30).

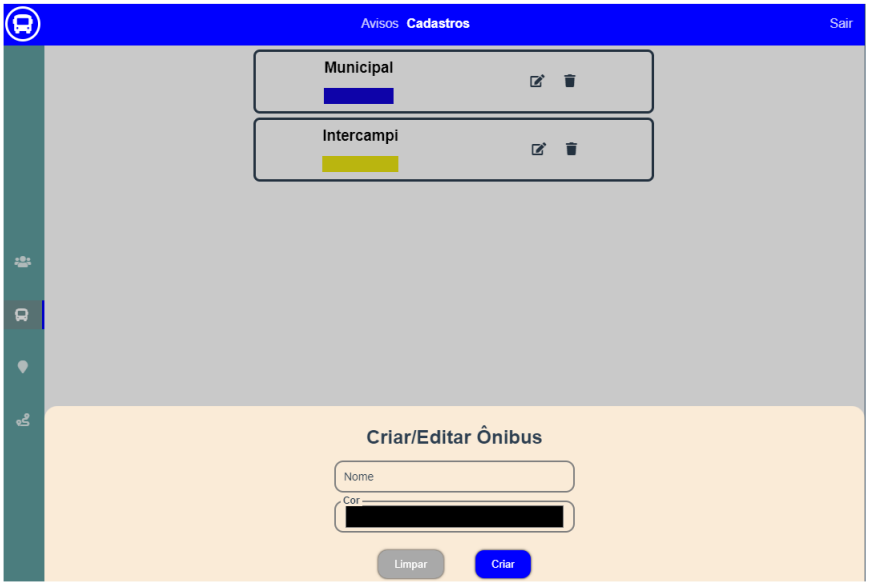

Figura 30 – Cadastro de ônibus

Para cada linha de ônibus, o usuário administrador também pode criar, alterar e excluir rotas (que indicam os sentidos atendidos por cada linha) e seus pontos de parada, mostrados pela Figura 32 e pela Figura 31, respectivamente. Note que para o cadastro de ponto de parada é preciso informar a latitude e longitude do ponto. Para isso, é necessário utilizar um aplicativo externo, como o Google Maps, para recuperar essas informações.

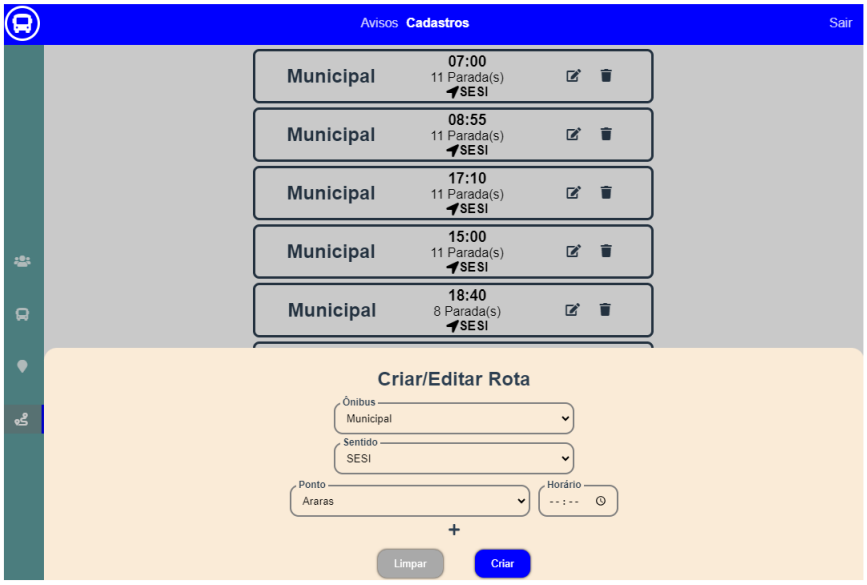

Figura 31 – Cadastro de rota de ônibus

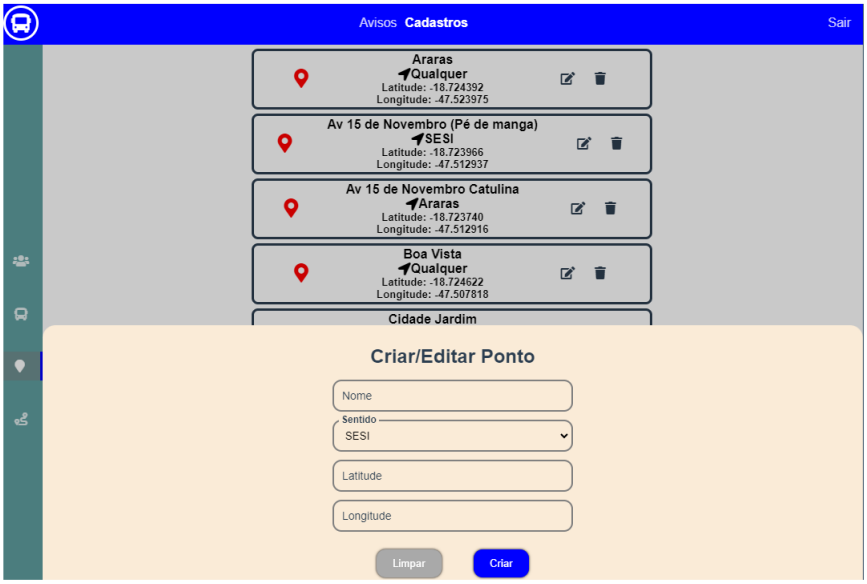

Figura 32 – Cadastro de ponto de parada de uma rota

4.2 Implantação

A implantação de todo o sistema foi executada em duas partes, para a camada de dados foi utilizado a solução gratuita do MongoDB disponibilizado através do site <https://www.mongodb.com/>, e o Redis e as camadas de serviço e aplicação utilizando a solução gratuita também do site <https://fly.io/>.

Após a disponibilização do sistema no site <https://mcbusao.fly.dev>, foi realizado todos os cadastros para proporcionar as informações aos usuários, como, os dois ônibus principais que circulam na cidade para os alunos da UFU e todos os pontos e rotas de ambos.

Realizado o cadastro foi enviado um link para os alunos via WhatsApp em dois grupos, A.A.A. MOCA II, grupo relacionado a atlética da UFU do campus de Monte Carmelo e também para o grupo S.I que contém os alunos de Sistemas de Informação, este mesmo link também foi divulgado no Instagram @spotted.ufumc que possui 731 seguidores até o momento dentre estes, alunos e ex-alunos, professores, moradores da cidade de Monte Carmelo e região.

4.3 Resultados preliminares

Em 78 dias de operação, o sistema obteve uma quantidade de acesso total de 532, sendo que, desses 78 dias, apenas 42 dias teve, ao menos, um acesso. Isso ocorreu devido ao fato de ter havido um recesso acadêmico (julho/2023) logo após a implantação inicial do sistema. Desses 42 dias com acessos registrados, dez dias antecederam o recesso, três dias foram durante o recesso e o restante aconteceu após o recesso. A Figura 33 mostra um

gráfico com o quantitativo de usuários por dia. Observa-se dois picos de usuários ativos, um na data da disponibilização do sistema online que aconteceu no dia 12/06/2023, e outro no dia 18/08/2023. A média de usuários ativos por dia é de 5,57.

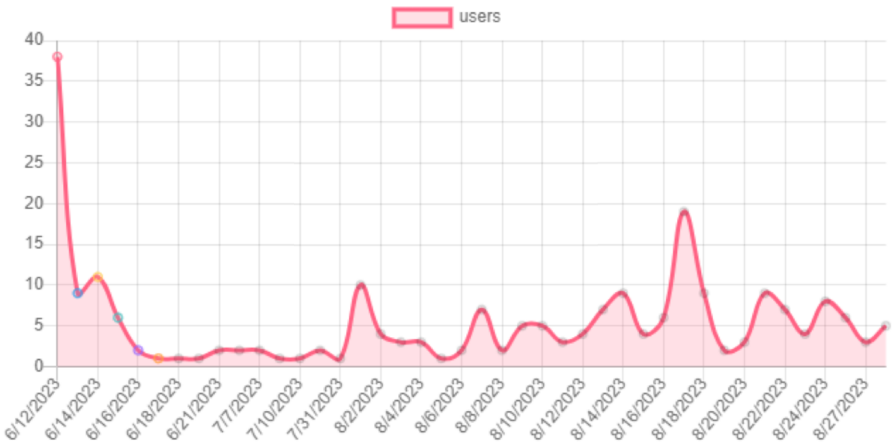

Figura 33 – Gráfico do quantitativo de usuários do sistema por dia.

O sistema obteve uma média de 2,36 acessos por dia para cada usuário. No total, constatou-se uma média de 12,66 acessos diários, sendo 91 usuários distintos ao todo. A Figura 34 mostra um gráfico com o quantitativo de uso por dia. Observa-se dois picos de acesso, um aconteceu no dia 12/06/2023, data em que o aplicativo foi disponibilizado, e outro no dia 02/08/2023, data em que foi reforçada a divulgação no início do semestre letivo.

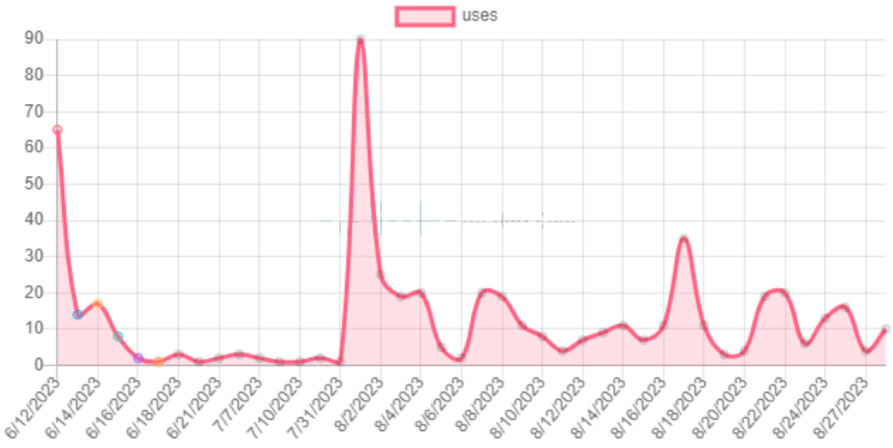

Figura 34 – Gráfico do quantitativo de uso por dia.

Como mencionado anteriormente, também foi solicitado, via sistema, uma breve avaliação dos usuários por meio de uma classificação em estrelas (1 a 5) e um campo aberto para críticas e sugestões. No total, foram recebidas 12 avaliações, resultando em uma média de 4,64 estrelas. 67% das avaliações foram classificadas com 5 estrelas.

Também foram recebidos *feedback* de forma textual, como mostrado nas Figura 35, Figura 36 e Figura 37. Nota-se, na Figura 36 e na Figura 37, que os usuários não marcaram a classificação em estrelas, sendo que um deles indicou esse detalhe no texto do comentário.

Figura 35 – Captura de tela com um feedback recebido com classificação em estrelas e um comentário.

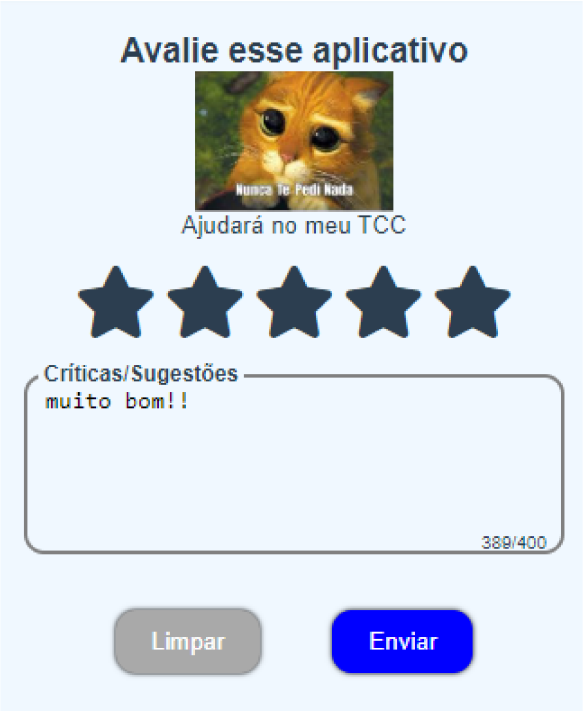

Figura 36 – Captura de tela com um *feedback* recebido sem classificação em estrelas, mas com um comentário.

Figura 37 – Captura de tela com um *feedback* recebido com um comentário indicando que esqueceu de fazer a classificação em estrelas.

No geral, os comentários enviados elogiam a proposta do sistema e indicam possíveis melhorias, como a exibição da localização dos ônibus em tempo real, previsão do horário de chegada no ponto e notificação aos usuários quando os ônibus estiverem se aproximando dos pontos de parada.

Capítulo 5

Considerações Finais

Em uma sociedade conectada digitalmente, o uso de sistemas de informação tem se tornado aliado no auxílio e facilitação da execução de tarefas do cotidiano. Especificamente na cidade de Monte Carmelo, Minas Gerais, ainda há um grande potencial do uso de tecnologia, mesmo que em ambientes mais comuns e rotineiros, como na identificação de horários e rotas de ônibus.

Tendo em vista a movimentação de pessoas causada semestralmente pela UFU na cidade, este trabalho objetivou construir uma solução rápida e prática para que estudantes da instituição no campus de Monte Carmelo pudessem encontrar informações de forma centralizada sobre os ônibus que circulam na cidade em direção ao campus.

Estudos como (NASCIMENTO; LIMA, 2017) apresenta a importância dos ITS, nos informando que quanto mais informações e confiança do usuário em relação a estas, melhor a satisfação dele e a priorização para utilização dos transportes públicos que vem sendo perdida ao não se sentir respeitado devido a falta de cuidado na transparência da informação.

Dessa forma, a proposta do trabalho foi a implementação de um sistema para facilitar o dia a dia dos estudantes, centralizando as informações em um único lugar diminui o trabalho que o usuário tem de encontrar o seu objetivo, localizando facilmente pontos, horários, problemas dos ônibus etc.

Uma prova de conceito da proposta foi desenvolvida, implantada e disponibilizada aos estudantes da UFU no campus de Monte Carmelo. Após a implantação, foi possível notar através dos *feedback* colhidos pelo formulário disponibilizado na própria aplicação, e mensagens recebidas, que mesmo a aplicação estando em sua primeira versão e não contendo todas as funcionalizadas que inicialmente foram pensadas para auxiliar ainda mais os usuários, foi bem aceita por aqueles que a utilizaram.

5.1 Limitações

Devido a entrega do sistema não incluir todas as funcionalidades previamente pensadas, o uso da aplicação ainda é limitado, apenas a visualização dos horários sem a localização em tempo real dos ônibus e disponível apenas para aqueles que tiveram acesso aos locais onde foram divulgados, não foi realizada até o momento uma divulgação a nível da faculdade.

Ainda também não foi feito uma pesquisa de usabilidade da aplicação, apesar de ter sido coletado os *feedbacks* através do formulário apresentado anteriormente, o quesito usabilidade não foi verificado.

5.2 Trabalhos Futuros

A aplicação proposta neste trabalho é apenas uma versão piloto para a informatização da informação existente sobre as rotas e horários dos ônibus. Diversos pontos de evolução podem ser apontados, como novas funcionalidades e atendimento aos outros campi da UFU. Algumas sugestões de melhorias também foram apontadas nos *feedback* dos usuários, como a localização do ônibus em tempo real. Assim, algumas possibilidades de trabalhos futuros foram identificadas:

a) Localização dos ônibus, garantindo confiabilidade na informação.

b) Previsão do horário de chegada no ponto.

c) Envio de notificação para o administrador quando um usuário criar uma alerta de um ônibus.

d) Notificar o usuário quando o ônibus estiver próximo ao seu ponto de partida.

e) Escalabilidade da aplicação, outros campus da UFU, outras universidades.

Por fim, pretende-se realizar um pedido de registro de software junto ao Instituto Nacional da Propriedade Industrial (INPI) e realizar divulgações sobre a proposta para gestores da UFU.

Referências

AWS. **O que é o JavaScript? – Explicação sobre o JavaScript (JS) – AWS — aws.amazon.com**. 2023. <https://aws.amazon.com/pt/what-is/javascript/>. [Accessed 08-Jul-2023]. Citado na página 18.

_____. **O que é o Redis? – Amazon Web Services (AWS) — aws.amazon.com**. 2023. <https://aws.amazon.com/pt/elasticache/what-is-redis/>. [Accessed 12-Jul-2023]. Citado na página 20.

CANHA, J. I. d. E. **Adaptação, saudades de casa e sintomatologia depressiva nos estudantes deslocados**. Tese (Doutorado), 2009. Citado na página 12.

CHAMBERLAIN, R. G. **Q5.1: What is the best way to calculate the great circle distance (which deliberately ignores elevation differences) between 2 points?** Online: Geographic Information Systems FAQ, 1997. <http://www.faqs.org/faqs/geography/infosystems-faq/>. [Acessado 02-08-2023]. Citado na página 34.

CONTRIBUTORS, M. **CSS | MDN — developer.mozilla.org**. 2022. <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. [Accessed 08-Jul-2023]. Citado na página 17.

DIGITAIS, R. **Saiba o que é um Progressive Web App (PWA) e quais são as vantagens desse tipo de aplicação — resultadosdigitais.com.br**. 2022. <https://resultadosdigitais.com.br/marketing/pwa/>. [Accessed 24-08-2023]. Citado na página 21.

FERREIRA, D.; SILVA, J.; NUNES, F. Topin: Aplicativo para melhoria da experiência dos usuários de transporte público de picos. In: **Anais da IV Escola Regional de Informática do Piauí**. Porto Alegre, RS, Brasil: SBC, 2018. p. 137–142. Disponível em: <https://sol.sbc.org.br/index.php/eripi/article/view/5157>. Citado na página 22.

JUNIOR, L. F. D. **Tutorial MongoDB para iniciantes em NoSQL - iMasters - We are Developers — imasters.com.br**. 2022. <https://imasters.com.br/mongodb/tutorial-mongodb-para-iniciantes-em-nosql>. [Accessed 12-Jul-2023]. Citado na página 20.

JUNIOR, S. R. de S.; LIMA, R. dos S.; CUNHA, R. A. H. da. Crowdbus: Aplicativo crowdsourcing para informação, localização, avaliação e fiscalização de frotas de ônibus.

SEGeT: Simpósio de Excelência em Gestão e Tecnologia, XI, v. 49, 2014. Citado 2 vezes nas páginas 12 e 23.

KELLY, G. **Comandos e tags HTML: confira os principais e mais usados! — blog.betrybe.com**. 2020. <https://blog.betrybe.com/desenvolvimento-web/comandos-e-tags-html/>. [Accessed 13-08-2023]. Citado na página 16.

LONGEN, A. **O que é HTML? Guia Completo com Lista de Comandos Básicos HTML — hostinger.com.br**. 2023. <https://www.hostinger.com.br/tutoriais/o-que-e-html-conceitos-basicos>. [Accessed 08-Jul-2023]. Citado na página 16.

MELO, D. **O que é Node.js? [Guia para iniciantes] – Tecnoblog — tecnoblog.net**. 2021. <https://tecnoblog.net/responde/o-que-e-node-js-guia-para-iniciantes/>. [Accessed 12-Jul-2023]. Citado na página 19.

NASCIMENTO, C. A. do; LIMA, J. P. O papel dos sistemas de informação como ferramenta para buscar melhorias na gestão dos serviços prestados ao usuário do transporte coletivo. 2017. Citado 2 vezes nas páginas 16 e 47.

SILVA, D. M. d. Sistemas inteligentes no transporte público coletivo por ônibus. 2000. Citado na página 15.

SINNOTT, R. W. Virtues of the haversine. **Sky and Telescope**, v. 68, n. 2, p. 158–159, 1984. Citado na página 34.

VALINOR, R. **Cloud Computing: tudo que você precisa saber sobre — remessaonline.com.br**. 2022. <https://www.remessaonline.com.br/blog/cloud-computing/>. [Accessed 13-08-2023]. Citado na página 21.

VAZ, M. C. d. R.; AMARAL, R. P. Trackbus: um aplicativo de apoio aos usuários de ônibus na cidade do rio de janeiro. 2017. Citado na página 23.