Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Ferramenta Web para Rastreamento e Monitoramento de Veículos de Transporte Escolar em Ambientes Reais

Lucas Teotônio Lima

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Rafael Frederico Alexandre

Abril, 2021
João Monlevade–MG

Lucas Teotônio Lima

Ferramenta Web para Rastreamento e Monitoramento de Veículos de Transporte Escolar em Ambientes Reais

Orientador: Rafael Frederico Alexandre

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Abril de 2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

L732f Lima, Lucas Teotonio .
Ferramenta web para rastreamento e monitoramento de veículos de transporte escolar em ambientes reais. [manuscrito] / Lucas Teotonio Lima. Lucas Teotônio Lima. - 2021.
63 f.: il.: color., tab..

Orientador: Prof. Dr. Rafael Frederico Alexandre.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Aplicações Web. 2. Aplicativos móveis. 3. Ônibus escolares - Medidas de segurança. 4. Sistemas de recuperação da informação - Transportes. 5. Transporte escolar - Ônibus . I. Lima, Lucas Teotônio. II. Alexandre, Rafael Frederico. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004.9

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Lucas Teotônio Lima

Ferramenta Web para Rastreamento e Monitoramento de Veículos de Transporte Escolar em Ambientes Reais

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 27 de abril de 2021

Membros da banca

Doutor - Rafael Frederico Alexandre - Orientador (Universidade Federal de Ouro Preto)
Mestre - Bruno Cerqueira Hott - (Universidade Federal de Ouro Preto)
Doutor - Darlan Nunes de Brito - (Universidade Federal de Ouro Preto)
Doutor - Marlon Paolo Lima - (Universidade Federal de Ouro Preto)

Rafael Frederico Alexandre, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 05/08/2021

Documento assinado eletronicamente por **Rafael Frederico Alexandre**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 05/08/2021, às 21:34, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0203269** e o código CRC **0E63226E**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Dedico este trabalho primeiramente a Deus, à minha mãe, irmãos e à toda minha família que, com muito carinho e apoio, não mediram esforços para que eu concluísse com êxito esta importante etapa da minha vida.

Agradecimentos

Agradeço, primeiramente a Deus, que me deu energia para me esforçar e concluir esse trabalho.

Agradeço à minha mãe que sempre se preocupou comigo, e sempre me incentivou a continuar na universidade durante todos esses anos.

Agradeço aos meus colegas que me apoiaram e me ajudaram a desenvolver algumas partes do projeto, me dando todo o suporte possível, sempre que eu precisei.

Agradeço ao Professor Rafael Frederico Alexandre, pela paciência e confiança no meu trabalho durante o projeto.

Enfim, agradeço à todas as pessoas que fizeram parte dessa etapa decisiva da minha vida, e que ainda tem muitos obstáculos para serem vencidos.

"Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar aonde a maioria não chega, faça o que a maioria não faz."

— Bill Gates.

Resumo

O transporte escolar é muito utilizado por crianças no Brasil, porém, na maioria das vezes, não se tem controle sobre o que acontece durante o percurso dos alunos entre casa e escola, e vice-versa. Com isso, alguns problemas têm a iminência de surgir, como, por exemplo: as crianças podem faltar às aulas sem o consentimento dos pais; pegar o transporte errado; ou se envolverem em acidentes, seja por descuidos pessoais, ou por condutas inapropriadas do condutor do veículo. Buscando melhorar a segurança dos alunos, além de oferecer maior tranquilidade para os pais, foi desenvolvido uma aplicação *web* chamada SRM, que propõe o monitoramento e o rastreamento de veículos de transporte escolar. Neste trabalho, foi utilizado um serviço de software embarcado que ficará nos veículos, que por meio do uso de redes móveis (3g/ 4g) ou redes 802.11 - popularmente conhecidas como redes *Wi-Fi*, do inglês *Wireless Fidelity*, enviará dados para o servidor através de uma API Restful. A aplicação web fornecerá informações em tempo real sobre a localização do veículo através de um *Websocket*. O desenvolvimento deste trabalho contou com várias tecnologias utilizadas para criação de aplicações web modernas na década de 2020.

Palavras-chaves: monitoramento de transporte escolar. veículo. ônibus. tempo real. transporte. websocket.

Abstract

School bus is widely used by children in Brazil, however, in most cases, there is no control over what happens during the students' journey. As a result, some problems may arise, for example: children may miss classes without parental consent; take the wrong transport; or get involved in accidents, whether due to personal carelessness or the wrong conduct of the vehicle driver. In order to improve the safety negligence, in addition to offering greater tranquility for parents, a web application was developed, which aims to monitor and track school transport vehicles. In this work, an embedded software service that will be used in vehicles is used and through the use of mobile networks (3g / 4g) or 802.11 networks, it will send data to the server through a Restful API. The application provides real-time information about the vehicle's location through a Websocket. The development of this work has several technologies used to create modern web applications today.

Key-words: school transport monitoring. vehicle. bus. real time. transport. websocket.

Lista de ilustrações

Figura 1 – Diagrama que descreve a arquitetura e os módulos do projeto, assim como os relacionamentos entre si. Este trabalho em específico está relacionado com a criação da *API RESTful*, o *WebSocket* e a página *WEB*. 16
Figura 2 – Organograma de tecnologias 27
Figura 3 – Comparação dos *Frameworks Front-End* mais utilizados atualmente. . 29
Figura 4 – Modelagem da aplicação. 37
Figura 5 – Arquitetura da *API Restful*. 38
Figura 6 – Arquitetura do *Websocket*. 39
Figura 7 – Demonstração da arquitetura reduzida do Sistema. 40
Figura 8 – Entrando no sistema embarcado para acessar o menu de Registrar rota. 43
Figura 9 – Cadastrando rota e sendo direcionado para tela de iniciar turno. 44
Figura 10 – Tela de login do sistema web. 45
Figura 11 – Tela de inicial do sistema web. 45
Figura 12 – Tela de monitoramento do sistema *web*. 46
Figura 13 – Tela de monitoramento do sistema *mobile*. 47
Figura 14 – Sistema *web*, tela que mostra que o monitoramento do veículo foi iniciado. 48
Figura 15 – Sistema *web*, tela com pop-up de informações do veículo monitorado. . 49
Figura 16 – Sistema *web*, tela com veículo fora de rota. 49
Figura 17 – Sistema *web*, tela de demonstração de movimento na rota. 50
Figura 18 – Embarcando um aluno no veículo. 51
Figura 19 – Modal do sistema *web* de alunos embarcados e não embarcados no veículo. 52
Figura 20 – Tela do sistema *web* com a *sidebar* oculta. 53
Figura 21 – Tela do sistema *web* em modo *full screen*. 53
Figura 22 – Tela do dispositivo embarcado para finalizar um turno. 54

Lista de abreviaturas e siglas

GPS Global Positioning System

WIFI Wireless Fidelity

PRV Problema de Roteamento de Veículos

SMS Short Message Service

IoT Internet of Things

RFID Radio Frequency Identification

API Application Programming Interface

ORM Object Relational Mapper

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

CSS Cascading Style Sheets

XML Extensible Markup Language

DAO Data Access Object

DOM Document Object Model

REST Representational State Transfer

TCP Transmission Control Protocol

DOM Document Object Model

DI Dependency Injection

IDE Integrated Development Environment

HTTP Hypertext Transfer Protocol

PO Product Owner

SM Scrum Master

BSS Bus Stop Selection

SBR School Buss Routing

SBRP School Buss Routing Project

DVR Digital Vídeo Recorder

RFID Radio Frequency Identification

CMS Content Management System

JSX JavaScript XML

DTO Data Transfer Object

Sumário

1 INTRODUÇÃO 14
1.1 Objetivos 17
1.2 Metodologia 18
1.3 Organização do trabalho 19

2 REVISÃO BIBLIOGRÁFICA 20
2.1 Trabalhos Correlatos 20
2.2 Tecnologias Utilizadas 26
2.2.1 API RESTful 27
2.2.2 Websocket 28
2.2.3 TypeScript 28
2.2.4 React 28
2.2.5 *Material UI* 29
2.2.6 Leaflet OpenStreetMap 30
2.2.7 Java 30
2.2.8 Spring 30
2.2.9 NodeJS 32
2.2.10 Socket.IO 32
2.2.11 PostgresSQL 33

3 DESENVOLVIMENTO 34
3.1 Back-End 35
3.1.1 Arquitetura da API Restful 38
3.1.2 Arquitetura do Websocket 38
3.2 Front-End 40

4 RESULTADOS 41
4.1 Metodologia para testes e validação 41
4.2 Resultados obtidos 42
4.2.1 Planejamento da viagem 42
4.2.2 Iniciar Turno 46
4.2.2.1 Embarque e desembarque de alunos 50
4.2.2.2 Funcionalidades Extras 52
4.2.3 Finalizar Turno 54
4.3 Conclusão sobre os resultados 54

5 CONCLUSÃO 56
5.1 Trabalhos futuros 56

REFERÊNCIAS 58

1 Introdução

Para que a ida para a escola das crianças seja mais segura, é preciso um roteamento efetivo. O Problema de Roteamento de Veículos (PRV) pode oferecer soluções para resolver o problema de roteamento de transporte escolar, pois, o PRV trata-se de um problema de otimização combinatória, cujo principal objetivo é encontrar o melhor caminho possível, levando em consideração algumas restrições laterais, para atender uma série de clientes (GOLDEN; RAGHAVAN; WASIL, 2008).

Segundo GOLDEN; RAGHAVAN; WASIL existe uma categoria de problemas de roteamento de veículos denominada VRPs "ricos", que são próximos dos problemas abordados no mundo real. As principais características destes VRPs compreendem o grande número de restrições não presentes no problema original, como vários depósitos, viagens a serem realizadas pelos veículos, vários tipos de veículos ou outras questões operacionais, como restrições de carregamento (LEMOS, 2019). Ainda segundo LEMOS o problema de roteamento de veículos de transporte escolar, se encaixa no grupo VRPs "ricos".

A tecnologia pode ser uma grande aliada na segurança e controle dos veículos de transporte escolar, além de reduzir custos para operadores e usuários. Ela pode ajudar a resolver o Problema de Roteamento de Veículos (PRV), onde cada veículo envolvido no trasporte escolar tem uma rota predefinida, considerando a necessidade de visitar um conjunto de pontos de ônibus para o embarque ou desembarque de alunos até o seu destino final, que pode se a casa ou a escola do aluno (STEINER et al., 2000). Logo, a problemática que está sendo abordada neste trabalho se trata de um PRV, que se configura num problema de roteamento de veículos de transporte escolar.

Na Figura 1 é possível observar a arquitetura atual do School Buss Routing Project (SBRP)[1], inicialmente proposta por LEMOS. Entre os módulos disponíveis é possível citar: I) O Bus Stop Selection (BSS), abordado no trabalho de AMARAL , é responsável pela identificação de pontos de ônibus candidatos e, logo após, realiza a alocação dos alunos de forma que o número de pontos de ônibus seja minimizado e, além disso, atenda a restrições de limite máximo de caminhada dos alunos. II) O School Buss Routing (SBR), que tem como objetivo definir as rotas dos veículos responsáveis por transportar os alunos dos pontos de ônibus até as suas respectivas escolas. O School Buss Routing (SBR), foi abordado em outros trabalhos como o de LEMOS, que desenvolveu uma solução web para gerir os dados relacionados ao School Buss Routing Project (SBRP), e o trabalho de MESSIAS, que desenvolveu uma ferramenta *Web* para Visualização de

[1] Projeto que tem como objetivo desenvolver soluções de software voltadas para tratar o problema de transporte escolar. Maiores informações podem ser consultadas através do domínio <www.sbrproject.org>

Rotas Aplicadas ao Problema de Roteamento de Veículos (PRV). III) O SBR *Mobile* que é o módulo responsável por coletar as informações dos veículos em tempo real. Esse módulo é executado em dispositivos móveis que são embarcados aos veículos e fornecem, além da localização do veículo em tempo real, informações dos alunos. Nota-se ainda que na arquitetura proposta existe um *Websocket*, que foi desenvolvido para apresentar informações em tempo real para o usuário na interface web, enquanto o veículo estiver em movimento. Por fim, a *API Restful* é utilizada para inserir e atualizar dados que são gerados através do dispositivo móvel e por todos os módulos que compõem o projeto, para possibilitar que o sistema de rastreamento e monitoramento de transporte escolar funcione corretamente.

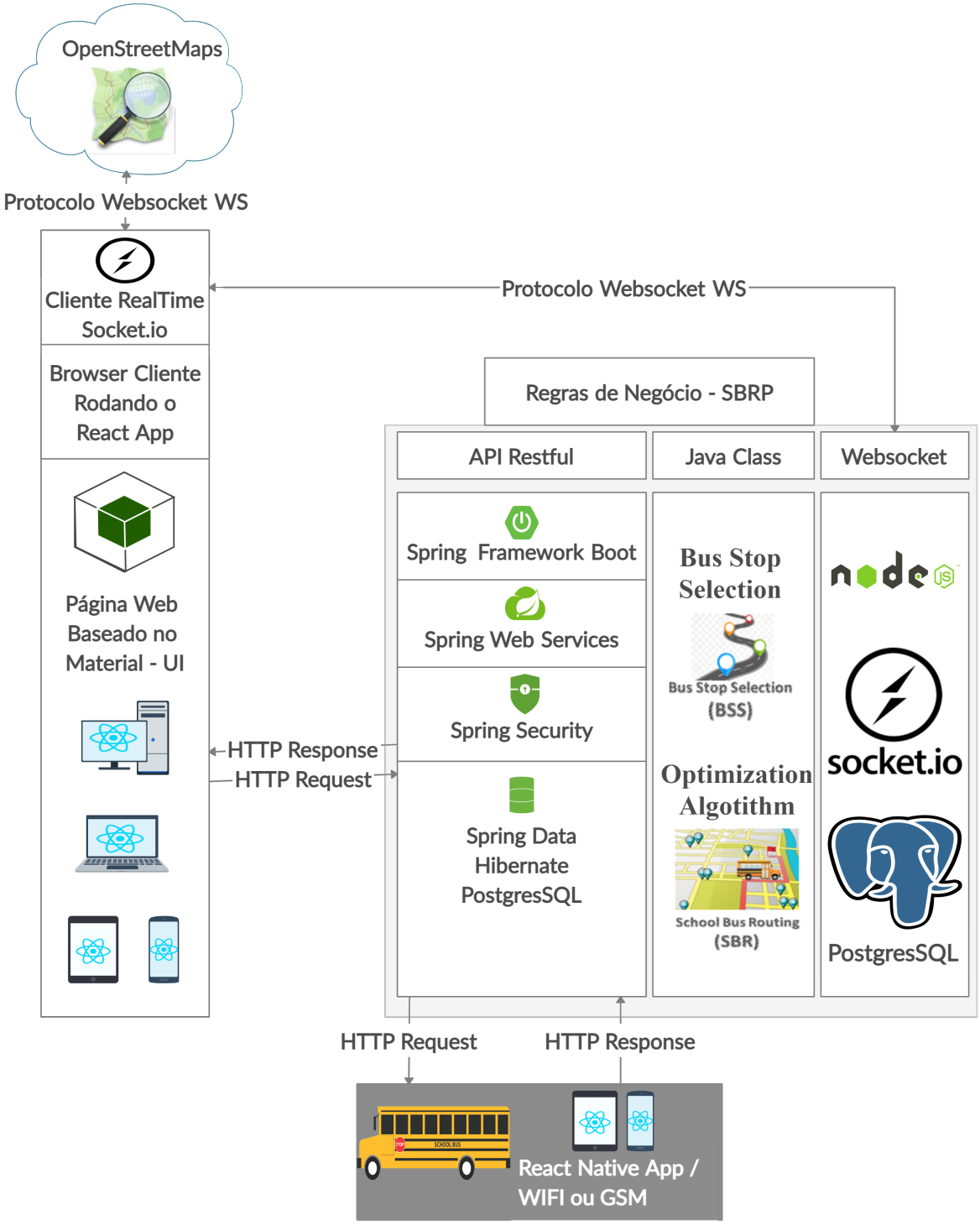


Figura 1 – Diagrama que descreve a arquitetura e os módulos do projeto, assim como os relacionamentos entre si. Este trabalho em específico está relacionado com a criação da *API RESTful*, o *WebSocket* e a página *WEB*.

Adaptado de: (LEMOS, 2019).

Este trabalho visa desenvolver a parte de rastreamento e monitoramento de veículos de transporte escolar do projeto SBRP. Ou seja, será desenvolvido a *API Restful* com

Spring, o sistema web onde será possível monitorar os veículos, e o *websocket*, que irá fornecer a possibilidade de o sistema web poder funcionar em tempo real. O dispositivo embarcado do veículo será adaptado neste trabalho segundo as novas rotas que serão criadas na *API Restful*. O dispositivo embarcado faz parte do projeto SBRP; não faz parte deste trabalho em questão. O mesmo será apenas adaptado a este projeto, mas é de extrema importância para que o sistema de rastreamento e monitoramento de veículos funcione, pois, é ele que enviará as informações de localização do veículo para o sistema web através da *API Restful*.

Segundo JISHA; JYOTHINDRANATH; KUMARY, em diversos países, como, por exemplo, a Índia, estão sendo desenvolvidas pesquisas e *softwares* com o objetivo de melhorar a segurança dos estudantes no transporte escolar e oferecer meios para que as escolas e os pais possam acompanhar os passos das crianças. Dessa forma, é possível ter acesso a pontos de entrada e saída das crianças do veículo; monitorar o transporte em tempo real; checar o horário de ida e volta do veículo; notificar conforme as ações que são executadas durante todo o percurso; e sinalizar rota incorreta.

1.1 Objetivos

O presente trabalho possui como objetivo geral desenvolver um módulo *WEB* para rastreamento e monitoramento de veículos de transporte escolar em ambientes reais, chamado SRM, que será incorporado à arquitetura atual do School Buss Routing Project (SBRP). Para isso, serão utilizadas as principais tecnologias *web* disponíveis na década de 2020.

Todos os dados utilizados pela aplicação desenvolvida serão armazenados em um banco de dados relacional. Para obter as informações de localização do veículo, serão utilizados dispositivos embarcados nos veículos, que enviarão as informações para o banco de dados. Entre os dados fornecidos pelo dispositivo embarcado, destacam-se as informações de geolocalização em tempo real, que serão enviadas por meio do uso de redes móveis (3G/ 4G) ou redes 802.11 - popularmente conhecidas como WIFI, sendo as principais testadas. Demais informações como a identificação dos alunos que estão embarcando e desembarcando do veículo, também serão enviadas através do dispositivo embarcado. É preciso ressaltar que a *interface* do dispositivo embarcado não faz parte do escopo desse trabalho, mas o *backend*, assim como os *endpoints*, foram desenvolvidos neste trabalho, e são utilizados por esse módulo.

Após o armazenamento das informações no banco de dados relacional, as mesmas são consumidas e apresentados em uma *interface web* em tempo real. Cabe ressaltar que esse dispositivo continuará enviando informações sobre a rota percorrida pelo veículo no decorrer da sua operação.

1.2 Metodologia

Os passos para a execução deste trabalho são definidos a seguir:

1. Revisar a literatura: pesquisa por trabalhos correlatos, por meio de livros e artigos científicos, buscando diferenciais que possam ser comparados ou acrescentados ao trabalho proposto.

2. Levantamento de requisitos: definição dos requisitos mínimos para o funcionamento do software em questão.

3. Histórias de usuário: utilização de uma linguagem cotidiana para definir histórias de usuário, com o intuito de narrar o que o usuário está fazendo, ou o que ele necessita fazer no sistema.

4. Definir o modelo: baseado nos requisitos que foram levantados e nas histórias de usuário, realizar a modelagem do banco de dados, buscando reformular o modelo anterior, que foi utilizado em outros projetos, visando adapta-lo para o projeto em questão.

5. Definir tecnologias: pesquisa através de sites, artigos, livros, revistas e blogs de tecnologia e programação, buscando identificar as tecnologias modernas mais utilizadas para desenvolvimento *web* na década de 2020, e definir todas as tecnologias a serem utilizadas no desenvolvimento do trabalho em questão.

6. Prototipação: criação de representações e protótipos de softwares utilizando papel e lápis, ou a ferramenta *pencil*[2], utilizando como base as histórias de usuários e os requisitos levantados. Formulação de ideias de como as telas serão apresentadas para o usuário final.

7. Implementar *API Restful*: criação de um nova *API Restful*, buscando utilizar conceitos de padrão de projetos, utilizando as principais tecnologias que foram pesquisadas anteriormente.

8. Comunicação em tempo real: desenvolvimento de um *WebSocket* para que seja possível a comunicação em tempo real entre a interface *web* e o veículo a ser monitorado.

9. Implementar o *software*: desenvolver o software *web* com base nas histórias de usuários e nos protótipos que foram desenvolvidos. Aplicação da integração com a *API Restful* e o *WebSocket*.

10. Teste de *software*: planejamento, desenvolvimento e realização de testes de *software*, buscando validar os requisitos funcionais do sistema.

[2] <https://pencil.evolus.vn/>

11. Fase final: agir em cima dos resultados obtidos procurando propor, e/ ou realizar melhorias até o final do projeto.

1.3 Organização do trabalho

O trabalho está estruturado em cinco capítulos. No capítulo primeiro, apresenta-se a introdução ao tema, contendo a problemática, os objetivos e os procedimentos metodológicos da pesquisa. O segundo capítulo apresenta a revisão bibliográfica acerca do tema, detalhando alguns sistemas correlatos e outras ferramentas utilizadas no trabalho. No terceiro capítulo apresenta-se a modelagem do sistema. Neste momento, buscou-se explicar como as ferramentas apresentadas no segundo capítulo foram utilizadas no trabalho, bem como descreve-se sobre a metodologia utilizada para desenvolver o projeto e realizar os encontros com o orientador. No quarto capítulo são apresentados os resultados obtidos no desenvolvimento do trabalho, utilizando o fluxo de funcionamento do sistema. Por fim, no quinto e último capítulo realiza-se a conclusão do trabalho desenvolvido e apresenta-se algumas sugestões para pesquisas futuras, que poderão se utilizar deste estudo como base.

2 Revisão bibliográfica

Este capítulo apresenta uma revisão da literatura, destacando os trabalhos correlatos encontrados na bibliografia. Nesta seção são apresentadas todas as ferramentas utilizadas durante o desenvolvimento do sistema.

2.1 Trabalhos Correlatos

No estudo intitulado "O ônibus inteligente: um sistema de rastreamento para ônibus escolares", realizado por GHAREEB et al., os autores destacaram que o trânsito no Líbano tem muito congestionamento, e os pais ficam em uma missão exaustiva de esperar os filhos sem saber a que horas realmente eles chegarão. De acordo com GHAREEB et al., o objetivo principal da aplicação proposta no trabalho, é detectar a localização atual do ônibus usando o Global Positioning System (GPS) para notificar automaticamente os pais alguns minutos antes da chegada dos filhos às suas casas. Além disso, ele dispara um alarme para o motorista desembarcar a criança no endereço previamente fornecido pelos pais. Por sua vez, os pais podem usar o aplicativo para detectar a localização atual do ônibus, receber a notificação do horário em que a criança irá chegar, e definir qualquer um dos seus filhos como ausente, se for o caso.

O sistema criado por GHAREEB et al., tem uma parte *web*, onde os pais podem definir seus filhos como ausentes e ver o histórico de ausência. Já a escola pode adicionar ônibus, crianças e parentes das crianças ao sistema. A parte *mobile* é quase toda gerenciada pelo motorista, onde é possível ver o mapa com as casas dos alunos, e outra funcionalidades automáticas, tais como envio automático de mensagens, atualização da localização e verificação da distância entre as casas dos alunos. Pode-se perceber que o sistema completo tem três domínios: a escola; o ônibus; e os pais, sendo que cada um desses domínios tem funcionalidades disponíveis no sistema, a saber:

- Escola: pode adicionar ou atualizar informações sobre os estudantes; designa o aluno para o ônibus correspondente de acordo com seu endereço; verifica ausências de cada aluno no registro.
- Pais: podem definir seu filho como ausente; podem visualizar a localização do ônibus escolar; podem ver o histórico de ausência do filho.
- Ônibus: atualiza a localização a cada 5 minutos; pode ver a casa de cada uma das crianças no mapa - casas vermelhas significa que a criança está ausente, casas verdes significa que a criança está presente; envia uma mensagem automática para os pais

quando a distância entre o ônibus e a casa do aluno for menor ou igual a 3 km, avisando que o filho vai chegar em casa em 5 minutos.

Pode-se perceber que o sistema proposto por GHAREEB et al. é focado na segurança das crianças. O transporte escolar é totalmente gerido pela escola, e busca informar os pais sobre as ações de seus filhos, possibilitando saber se o filho faltou a aula, quando ele vai chegar, ou quando ele pode sair de casa para pegar o ônibus. O sistema proposto pelos autores também ajuda a reduzir custos, já que os pais podem informar a ausência do filho, e isso pode evitar que o motorista pegue uma rota mais longa, diminuindo custos (GHAREEB et al., 2017).

Uma das principais características que envolvem um Sistema de Monitoramento Escolar é o fato de ele possibilitar o Rastreamento e o Monitoramento em tempo real. Neste contexto, GHAREEB et al. desenvolveram uma aplicação que possibilita que os pais das crianças acompanhem a localização de seus filhos no ônibus em tempo real. Entretanto, a taxa de atualização da localização do veículo, que é de 5 minutos, é muito alta e pode influenciar até mesmo no sistema de notificação que é utilizado para enviar mensagens para os pais, fazendo com que tais informações cheguem atrasadas. Além disso, existe um risco iminente de oferecer a localização errada do veículo.

O trabalho intitulado "Um sistema de rastreamento inteligente e seguro para monitoramento de ônibus escolar", proposto por AHMED et al., destaca a falta de segurança nos transportes públicos escolares, motivo pelo qual os pais acabam utilizando o transporte particular para levar os filhos para escola, o que acaba acarretando problemas relacionados ao tráfego em lugares densamente povoados como *Dhaka* e *Delhi*. AHMED et al. destacam que países como EUA, Canadá e Austrália, sugerem que o transporte público deve possuir um alto nível de segurança. Pode-se citar como exemplo o *Australian College of Road Safety* [1], que destaca que a viagem de ônibus é a forma mais segura entre os transportes rodoviários na Austrália, podendo ser até 14 vezes mais seguro do que veículos particulares (AHMED et al., 2019).

Segundo AHMED et al., Internet of Things (IoT) é um conceito que se refere à interconexão de objetos ao redor, sem a intervenção de pessoas. Os objetos trocam informações entre si e fornecem serviços avançados e inteligentes para os usuários (XIA et al., 2012). Utilizando conceitos de IoT, AHMED et al. propuseram um sistema inteligente de rastreamento de ônibus escolar, cujo principal objetivo era auxiliar os pais e responsáveis no gerenciamento de transporte escolar. Os pais receberam notificações no momento de chegada do ônibus, e também na entrada e saída de alunos do ônibus, devido a um dispositivo de autenticação que consegue identificar cada aluno individualmente. No

[1] A associação de membros de pico para profissionais de segurança no trânsito, defensores e membros do público que estão focados em salvar vidas e ferimentos graves em nossas estradas <https://acrs.org.au/>.

transporte, as janelas contam com um laser que durante o tempo de viagem pode ser acionado, e evitará que os alunos fiquem com partes do corpo para fora do veículo, como mãos ou cabeça. O autores consideram que tais funcionalidades ajudaram a garantir a segurança e a confiabilidade dos ônibus escolares.

No sistema proposto por AHMED et al., o aluno passa o cartão com leitor *Radio Frequency Identification (RFID)* quando entra no ônibus, o leitor RFID transmite a identificação do aluno para um *Digital Vídeo Recorder (DVR)* móvel. O DVR usa a rede 2G/3G/WI-FI para enviar a identificação do aluno para um servidor *Content Management System (CMS)*. Em seguida, o servidor CMS envia um SMS para o celular dos pais informando que a criança entrou no ônibus. Durante a viagem do ônibus, o DVR faz gravações de áudio e vídeo nas paradas do veículo, e tais informações também são enviadas para o CMS. Desta forma, é possível que os pais verifiquem visualmente o ônibus. Além disso, o GPS instalado no veículo é utilizado para rastrear a localização dos ônibus a partir de uma interface disponível na *web* (AHMED et al., 2019). Nesta arquitetura, o sistema possui três domínios, a saber:

- Admin: este domínio é responsável por adicionar e atualizar informações do aluno e do ônibus no sistema. Qualquer alteração realizada por esse domínio pode ser refletida em tempo real, devido ao seu nível de acesso (AHMED et al., 2019).
- Serviço: este domínio é responsável pelas tecnologias implementadas nos ônibus escolares. O serviço é prestado através de um leitor RFID e impressão digital, que envia mensagens para os pais a cada entrada ou saída do aluno do transporte. Conta também com um laser para evitar acidentes nas janelas e GPS para oferecer a possibilidade de rastrear o veículo (AHMED et al., 2019).
- Usuário: este domínio conta com um aplicativo *Android* que enviará notificações para os pais sobre o status atual de seus filhos. O aplicativo também informa em tempo real, a localização do ônibus através de um mapa, bem como outras informações como o número de contato do motorista (AHMED et al., 2019).

O sistema desenvolvido no trabalho proposto por AHMED et al. é para a cidade de *Dhaka*, capital de Bangladesh, e tenta garantir mais segurança no transporte escolar. O sistema é intuitivo e de fácil coleta de dados. Por meio dele, os responsáveis poderão receber notificações através de um aplicativo *Android*, sendo possível localizar o veículo, acessar imagens disponibilizadas pelo sistema e acompanhar toda a viagem em tempo real. Portanto, de acordo com AHMED et al., os usuários terão a disposição um sistema altamente confiável, seguro e inteligente para utilizar.

AHMED et al. desenvolveram uma aplicação que permite transmitir áudio, vídeo e localização do veículo em tempo real, além de enviar SMS na entrada e saída de um

determinado aluno identificado de forma individual através do leitor RFID. Entretanto, disponibilizar a lista de alunos embarcados e não embarcados no veículo, pode não ser uma tarefa tão simples e fácil, principalmente se a consulta por tal informação depender das condições de conectividade dos pais, tendo em vista que funcionalidades como áudio e vídeos podem demorar para serem carregadas dependendo da velocidade e tipo de conexão.

O trabalho intitulado "Sistema de notificação e monitoramento de ônibus escolar inteligente baseado em IoT", proposto por RAJ; SANKAR, destaca a importância de oferecer um serviço de transporte escolar confiável para garantir a segurança dos alunos. Os autores também ressaltam que é difícil saber o que acontece com as crianças no trajeto entre a escola e a casa, e vice-versa. Além disso, as crianças podem acabar pegando o ônibus errado, e ainda, os motoristas podem não ser capazes de identificar todos os alunos. De acordo com RAJ; SANKAR, algumas escolas implementaram o rastreamento GPS de ônibus, porém, para os autores, este sistema não garante a segurança absoluta. Pois, alguns destes dispositivos não transmitem informações em tempo real, além de não serem capazes de oferecer meios para identificar os alunos individualmente (RAJ; SANKAR, 2017).

No sistema proposto por RAJ; SANKAR, o rastreamento do veículo é feito através da leitura de coordenadas geográficas a partir do GPS. Um módulo carrega essas informações para um servidor de banco de dados remoto, através de redes *WI-FI*, utilizando um microcontrolador (RAJ; SANKAR, 2017). Estas informações podem, então, ser acessadas por usuários, pais e administradores da escola através de um aplicativo móvel (RAJ; SANKAR, 2017). O sistema de notificação alerta os pais quando o filho utiliza o leitor de *RFID*. Ao utilizar o leitor, um *script* é invocado no servidor, e uma notificação é enviada para o pai do aluno que contém identificação correspondente (RAJ; SANKAR, 2017). O sistema proposto pelos autores é composto por três domínios, que são eles:

- Administrador: pode adicionar informações sobre os horários dos ônibus, modificar detalhes do ônibus, motorista do ônibus e estudantes. Pode ver detalhes, como a localização, velocidade e lista de passageiros de cada ônibus (RAJ; SANKAR, 2017).
- Motorista: pode ver a rota, velocidade localização e as paradas programadas do ônibus, além de poder contactar os serviços de um administrador em caso de emergência (RAJ; SANKAR, 2017).
- Pais: podem acompanhar a viagem de seus filhos em tempo real. É possível ver informações do ônibus, como localização, rota e velocidade. Caso necessário, podem entrar em contato com o motorista do ônibus ou administração, além de receberem notificações sempre que seus filhos entram e saem do ônibus (RAJ; SANKAR, 2017).

Para RAJ; SANKAR, o sistema garante que o ônibus está na rota correta e com

uma velocidade segura, além de ser possível visualizar desvios e atrasos. Tudo isso faz com que os pais fiquem mais tranquilos, e mais convictos que seus filhos vão chegar em segurança (RAJ; SANKAR, 2017).

RAJ; SANKAR desenvolveram uma aplicação que mostra os requisitos básicos para possibilitar que os pais acompanhem a viagem dos filhos em tempo real. Entretanto, essa aplicação parece não disponibilizar a lista de alunos embarcados no veículo para os pais, o que faz com que os pais confiem apenas no sistema de notificação que é enviado quando os alunos entram e saem do veículo.

O trabalho intitulado "Sistema de rastreamento de ônibus escolar baseado em IOT", proposto por GODE; DESAI, destaca que, devido ao crescimento populacional, as viagens de transporte escolar estão ficando cada vez mais longas, e a segurança das crianças no trajeto escolar é a principal preocupação dos pais e das escolas.

O sistema proposto por GODE; DESAI tem três módulos; o primeiro é o GPS, que lê as coordenadas de localização do ônibus; o segundo é o sensor de álcool para evitar que o motorista dirija embriagado; e por fim, o sensor de gás e interruptor de emergência para alerta de situação de emergência. As informações adquiridas pelo veículo são enviadas para a nuvem através de um módulo *Wi-Fi*.

Para mostrar informações em tempo real, foi utilizado o aplicativo *Blynks IoT* [2]. Os dados são enviados para a *Blynk Cloud* e no aplicativo *Blynk* existirá uma caixa *widget*, onde é possível ver os dados recebidos do módulo *Wi-Fi*. Como o que foi enviado para a nuvem foram coordenadas geográficas, basta selecionar o *widget* de mapa para mostrar a localização do veículo (GODE; DESAI, 2020).

No trabalho de GODE; DESAI, de acordo com os resultados descritos, o módulo construído está disponível para ser utilizado. Ainda de acordo com os autores, o uso do aplicativo *Blynks* torna mais fácil rastrear o veículo a qualquer momento, sendo que o ponto de falha do aplicativo é quando o mesmo perde conexão com a internet.

Essa aplicação desenvolvida por GODE; DESAI conta com vários dispositivos voltados para a segurança. Entretanto, os autores esqueceram de alguns aspectos importantes como, por exemplo, como identificar os alunos individualmente, e fornecer uma listagem de quem está no veículo no momento.

O *Mapix*[3] é uma plataforma de rastreamento e monitoramento de veículos de transporte escolar. É possível utilizar o sistema em várias plataformas, como, *android* e *IOS*. As principais funcionalidades do *Mapix* são: exibir a localização do veículo em tempo real; estimar o tempo previsto para a chegada do transporte; facilitar a comunicação entre

[2] Blynk é uma plataforma IoT independente de hardware com aplicativos móveis de marca branca, nuvens privadas, gerenciamento de dispositivos, análise de dados e aprendizado de máquina <https://blynk.io/>.

[3] Site oficial do aplicativo Mapix. <https://mapixapp.com>

transportador e passageiros por chat e comunicados; e estimar o tempo previsto para a chegada à escola e no retorno.

Os usuários do *Mapix* ainda podem contar com ferramentas como: monitoramento em tempo real; sistema de chat; histórico do trajeto; comunicados instantâneos; e sistema de notificações. O sistema de notificações é um dos principais componentes do sistema, pois, ele pode enviar informações como tempo estimado que falta para chegar ao ponto de embarque e desembarque de alunos. É possível também, enviar notificação de desistência, fazendo com que o transporte não perca tempo, e possa remanejar sua rota, para pegar alunos em outros pontos, o que ajuda a reduzir custos. O sistema é pago, mas oferece um acesso gratuito por 30 dias.

O software *Mapix* oferece quase todas as funcionalidades que seriam interessantes de se implementar no trabalho proposto em questão, principalmente devido ao sistema de notificação, que parece ser bastante completo. Entretanto, se torna muito custoso desenvolver todas as funcionalidades do *Mapix* neste trabalho. Além disso, ele é pago. Porém, considera-se interessante propor algumas de suas funcionalidades para trabalhos futuros.

Na Tabela 1 foram apresentados as dez principais funcionalidades que neste trabalho considera-se serem as principais tecnologias que um software de Rastreamento e Monitoramento de Transporte Escolar pode oferecer. Essas funcionalidades foram levantadas com base na discussão do parágrafo anterior. Na Tabela 1 a seguir buscou-se destacar as funcionalidades implementadas nos trabalhos correlatos, bem como no trabalho proposto.

Tabela 1 – Comparação com Trabalhos Relacionados e Trabalho Proposto.

Funcionalidades	(GHAREEB et al., 2017)	(AHMED et al., 2019)	(RAJ; SANKAR, 2017)	(GODE; DESAI, 2020)	Mapix	Proposto
Identificação individual de Alunos	-	X	X	-	X	X
Lista de passageiros embarcados	-	-	X	-	-	X
Monitoramento em tempo real	-	X	X	X	X	X
Velocidade atual do veículo	-	-	X	-	X	X
Verificação de percurso	-	-	X	-	X	X
Histórico do Trajeto	-	-	-	-	X	-
Gravação de áudio	-	X	-	-	-	-
Gravação de vídeo	-	X	-	-	-	-
Rastreamento	X	X	X	X	X	X
Notificação	X	X	X	-	X	-

2.2 Tecnologias Utilizadas

Esta seção apresenta as tecnologias e as ferramentas que foram utilizadas no desenvolvimento deste trabalho, destacando os pontos específicos que determinaram a escolha. São feitas algumas comparações entre as ferramentas que foram utilizadas ou não, buscando mencionar os motivos, sejam eles técnicos, relacionados a custos ou por preferência do projeto.

Como ponto de partida, se faz necessário entender um pouco da arquitetura do *software*, que basicamente foi dividida entre *Backend* e *Frontend*, como é possível ver na Figura 2. Segundo PORTELA; SANTOS; PRADO, o *Frontend* é responsável pela interface e interação com o usuário; é onde ocorre a coleta de dados. Já o *Backend*, é responsável por toda a regra de negócio da aplicação, além de ser onde os dados são validados e persistidos.

Como é possível ver na Figura 2, a arquitetura do software está dividida em duas partes: *Front-end* e Back-end. A sigla *SRM* descrita na Figura 2 é um acrônimo para Sistema de Rastreamento e Monitoramento. O *Front-end* é composto por um sistema web onde é utilizado as seguintes tecnologias: *TypeScript*; *React*; *Material UI* e *Leaflet*. No *Back-end*, tem-se a *API Restful*, que utiliza as seguintes tecnologias: *Java*; *Spring* e *PostgreSQL*. Tem-se ainda o *WebSocket* que utiliza tecnologias como: *TypeScript*; *NodeJS*; *Socket.IO* e *PostgreSQL*. Todas as tecnologias citadas até aqui serão detalhadas posteriormente.

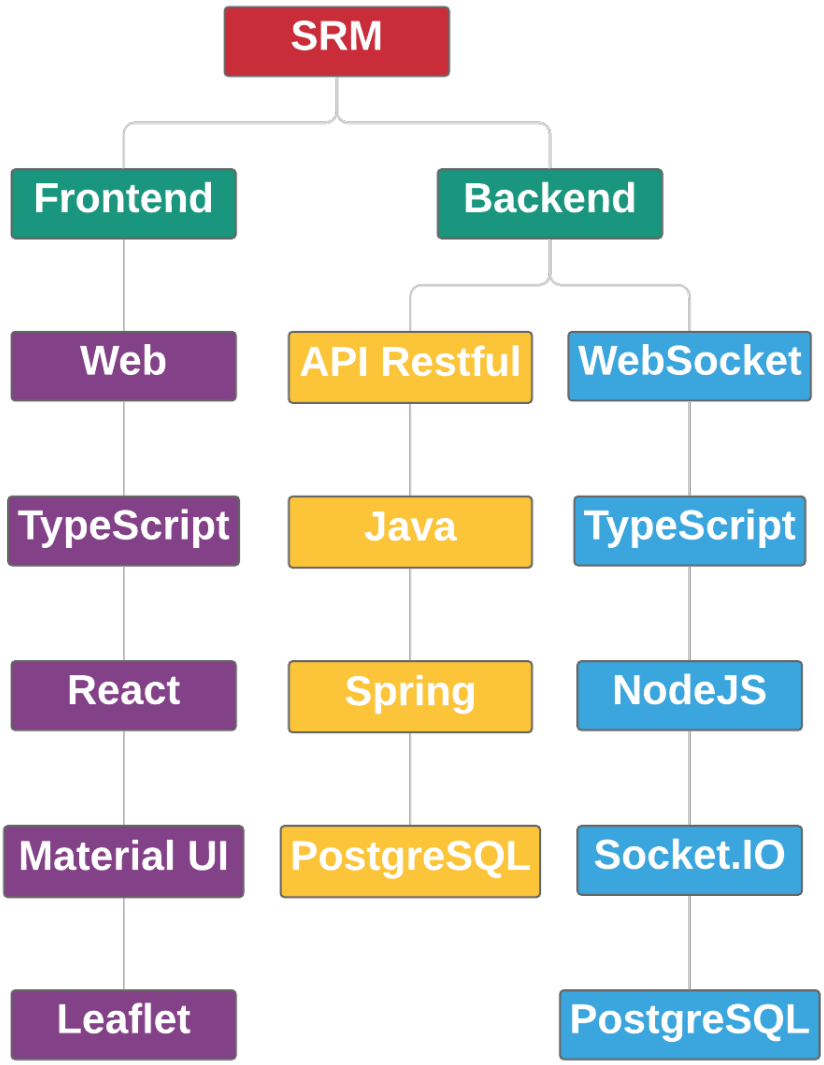


Figura 2 – Organograma de tecnologias

2.2.1 API RESTful

As APIs Restful são largamente utilizadas em aplicações distribuídas pela facilidade de integração com vários tipos de aplicação. Segundo SALVADORI et al., API utilizam o protocolo HTTP como meio de comunicação, sendo assim, é preciso obedecer a semântica do protocolo utilizando os verbos (GET, POST, PUT e DELETE), e os códigos que definem o status da requisição.

De acordo com SALVADORI et al., REST define vários princípios e restrições arquiteturais para o desenvolvimento de aplicações distribuídas. Dentre as principais restrições destaca-se o comportamento *stateless*, onde cada requisição é tratada de forma isolada. Pode-se também armazenar dados em cache para otimizar as interações entre cliente servidor, além da possibilidade de manipulação dos dados através de uma interface uniforme, podendo-se destacar aqui, a utilização de controles de hipermídia em JavaScript

Object Notation (JSON) (SALVADORI et al., 2015). Segundo SALVADORI et al., uma API se torna Restful quando ela segue todos os princípios e restrições proposto pela arquitetura REST.

Neste trabalho, a API foi construída em *Java*. Com a utilização do *Spring* e alguns dos seus componentes, foi possível fazer com que ela se torne uma API Restful capaz de seguir os princípios e restrições da arquitetura REST, que já foram citados no parágrafo anterior.

2.2.2 Websocket

Segundo SANTOS; SOUZA; ANDERLE, o *Websocket* permite a comunicação bidirecional e *full-duplex*, a partir de uma conexão TCP - do inglês Transmission Control Protocol. A comunicação via *Websocket* permite o tráfego e a troca de dados em tempo real, diminuindo a latência[4] da conexão (SANTOS; SOUZA; ANDERLE, 2015). Ainda segundo os autores, o *Websocket* faz uma ligação direta entre o navegador e o servidor, permitindo que os dados sejam enviados em qualquer direção.

2.2.3 TypeScript

Segundo BIERMAN; ABADI; TORGERSEN, *TypeScript* é uma extensão do *JavaScript* cujo objetivo é facilitar a construção de aplicativos *JavaScript* em larga escala. Todo aplicativo construído em *TypeScript* no final é *transpilado* [5] para *JavaScript*, porém, o *TypeScript* oferece mais recursos como classes, interfaces e tipagem de dados estática opcional (BIERMAN; ABADI; TORGERSEN, 2014).

2.2.4 React

O *React* [6] é uma biblioteca *JavaScript* para criação de interfaces de usuário, que pode ser utilizada tanto para web, quanto para dispositivos móveis. Através do *React* é possível criar componentes customizáveis utilizando a notação JSX, que é uma extensão do *JavaScript*. O *JSX* adiciona uma sintaxe elegante aos componentes do *React*, e a maioria dos desenvolvedores prefere utilizá-lo, por evidenciar melhor mensagens de erros e avisos ReactJS (2020).

Segundo LEMOS, o *React* usa o *Virtual Document Object Model (DOM)*, onde ele utiliza uma representação virtual da interface do usuário que é mantida na memória e

[4] Latência: definição de quanto tempo leva para um pacote de dados ir de uma ponto inicial a outro final. <https://pt.wikipedia.org/wiki/Lat%C3%AAncia>

[5] Transpilador: é uma categoria de tradutor que recebe o código-fonte de um programa escrito em uma linguagem de programação como entrada e produz um código-fonte equivalente no mesmo ou uma linguagem de programação diferente. <https://en.wikipedia.org/wiki/Source-to-source_compiler>

[6] <https://pt-br.reactjs.org/>

sincronizada com o DOM real. Quando o mesmo sofre alguma alteração, apenas o elemento alterado reflete o resultado para o usuário (LEMOS, 2019, p. 21).

O gráfico da Figura 3 mostra as bibliotecas *JavaScript* mais utilizadas atualmente. É possível perceber que o *React* vem crescendo bastante desde 2016, e se comparado com as demais bibliotecas no gráfico, em 2020 o *React* lidera o topo da lista. Outra vantagem do *React*, é que ele fica alocado no *browser* do usuário, o que diminui o processamento dos dados no lado servidor (LEMOS, 2019, p. 21).

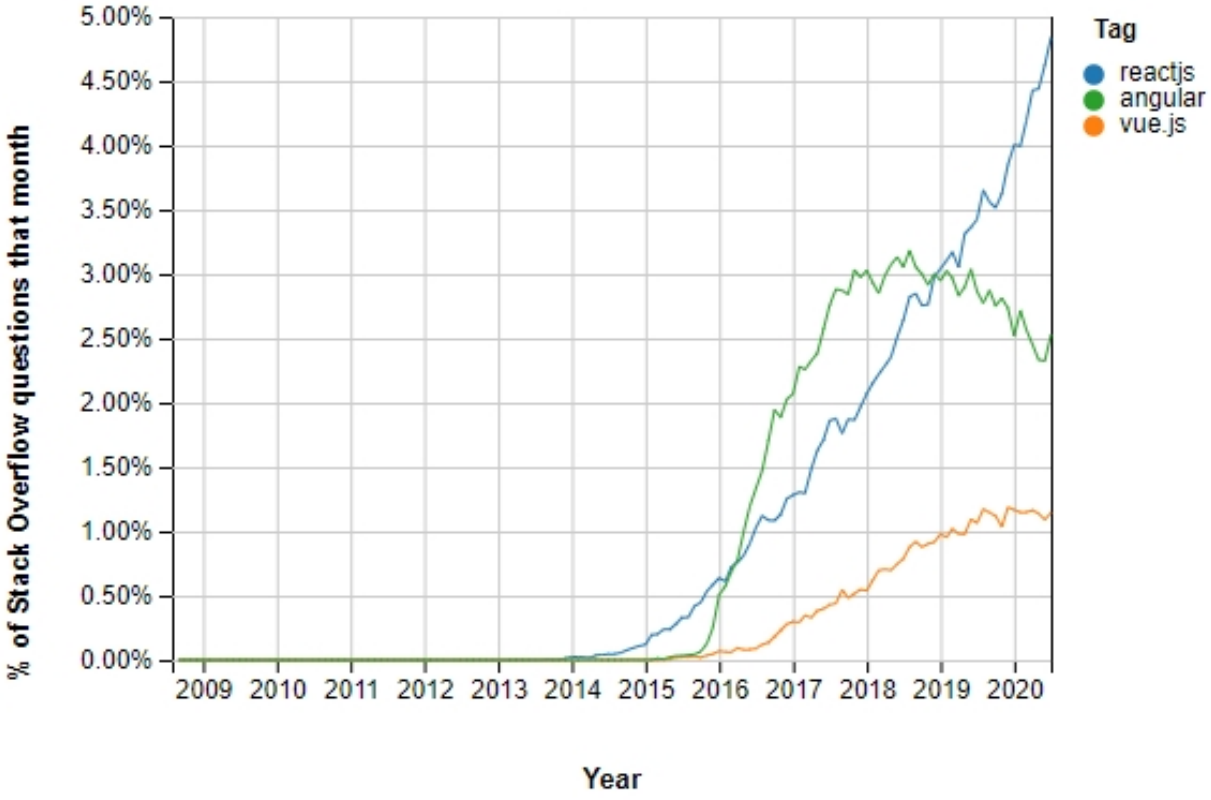


Figura 3 – Comparação dos *Frameworks Front-End* mais utilizados atualmente.

Fonte: Overflow (2020)

2.2.5 *Material UI*

O *Material-UI*[7] é uma biblioteca baseada em componentes *React*, que ajuda a desenvolver interfaces de forma fácil e ágil. Seus componentes são bem customizáveis, e a partir deles é possível criar várias interfaces diferences. Pode-se também, utilizar os componentes prontos que ele oferece para montar uma interface, desde a mais simples até a mais complexa, de forma fácil, elegante e responsiva, ou seja, adaptável à maioria dos tamanhos de telas de dispositivos que existem atualmente (MATERIAL-UI, 2020). Toda a interface deste trabalho foi construída com o *Material-UI*.

[7] <https://material-ui.com/pt/>

2.2.6 Leaflet OpenStreetMap

O *OpenStreetMap*[8] é um projeto *Open Source*, que conta com vários voluntários ao redor do mundo que contribuem para manter os dados sobre estradas, caminhos, estações ferroviárias em todo o mundo, atualizados (OpenStreetMap, 2020).

Para implementar o *OpenStreetMap* no sistema, existe uma biblioteca que se chama *Leaflet* [9]. Ela foi desenvolvida e deve ser implementada em *JavaScript* puro, e isso dificulta um pouco para se trabalhar com o *React*. Porém, existe uma variação dessa biblioteca, criada e adaptada para o *React* que se chama *React Leaflet*[10], que já conta com vários componentes prontos para serem utilizados juntamente com o *React*. Neste trabalho a biblioteca utilizada é o *React Leaflet*.

2.2.7 Java

Java é uma linguagem de programação orientada a objeto; mais que uma linguagem de programação, também é uma plataforma de desenvolvimento, onde é possível utilizá-la em vários cenários (INDRUSIAK, 1996). Neste trabalho, o *backend* foi construído em java; diversos *frameworks* são construídos em *java*, dos quais o escolhido para este trabalho foi o *Spring*[11] (ESBALTAR, 2018).

2.2.8 Spring

O *Spring* torna a programação em *Java* mais fácil e segura, pois, é possível seguir um padrão proposto pelo próprio *framework*. Ao escolher desenvolver uma aplicação com o *Spring*, o foco se torna a velocidade, simplicidade e a produtividade do desenvolvimento, porque todo o seu ecossistema é baseado no nível de configuração. O *Spring* conta com várias bibliotecas, que ao serem instaladas via gerenciador de dependência, o mesmo já os identifica, isso se deve ao seu sistema de Dependency Injection (DI). Mesmo que uma biblioteca do seu ecossistema não seja identificada, configurada, e integrada ao sistema automaticamente, bastará realizar poucas configurações e/ou inserções de código, para que o componente seja identificado pelo *Spring*, e funcione corretamente (REDDY, 2017, p.1, 221). Este trabalho foi criado utilizando o *Spring Initializr*[12], que é um site desenvolvido pela comunidade do *Spring*, para criar o projeto inicial do *Spring* com tudo que é necessário.

Para este trabalho em especial, foram utilizados alguns componentes oficiais do ecossistema *Spring*, e outros componentes desenvolvidos por terceiros, que são recomendados, pela própria comunidade. Todos são listados e detalhados a seguir.

[8] Link da documentação oficial do OpenStreetMap <https://www.openstreetmap.org/about>
[9] Link da documentação oficial do leaflet <https://leafletjs.com/>
[10] Link da documentação oficial do React Leaflet <https://react-leaflet.js.org/en/>
[11] Link da documentação oficial do Spring <https://spring.io/>
[12] <https://start.spring.io/>

- **Spring Boot:** facilita o processo de configuração dos componentes. Utiliza a convenção sobre configuração, onde basta incluir algum módulo específico no Gradle [13] ou no Maven[14] do ecossistema *Spring*, que ele o reconhece e o configura automaticamente (SOUZA; PINTO, 2020).

- **Spring MVC:** Determina uma série de componentes que podem ser estendidos pelas aplicações *Spring*. Esses componentes podem pertencer a diferentes camadas e ter diferentes responsabilidades como, por exemplo, componentes para tratar interface com o usuário, tratar requisições, regras de negócio e persistência de dados. O desenvolvedor tem a flexibilidade de trabalhar da forma que quiser (SOUSA et al., 2017).

- **Spring Data JPA:** É responsável por mapear os objetos, não havendo a necessidade de criar as Data Access Object (DAO), pois, ele as abstrai automaticamente (SOUSA et al., 2017). O *Spring Data JPA*[15] também abstrai o Object Relational Mapper (ORM) que está sendo utilizado, podendo utilizar outros ORMs diferentes. Neste trabalho, o ORM utilizado é o *Hibernate* (CUSTÓDIO, 2020).

- **Spring Security:** Possibilita implementar várias categorias diferentes de autenticação por ser altamente personalizável, além de oferecer uma estrutura de controle de acesso muito robusta, e que inclusive, é utilizada em aplicativos baseados no *framework Spring* (LOPES; MARCHI, 2013).

- **Spring Devtools:** É um módulo do *Spring Boot*. Necessita de um *plugin* instalado no IDE para funcionar. Um dos principais benefícios do *Devtools*, é conter um servidor embarcado, que faz o serviço de *LiveReload*, ou seja, ao identificar uma mudança no código ou em alguma biblioteca, ele reinicializa a aplicação, comparando, e aplicando somente as mudanças que foram feitas (NICOLL; WILKINSON; FREDERICK, 20–, p.44).

- **Spring Swagger:** O *Swagger*[16] ajuda a documentar a aplicação. Ele lê as *tags* de anotações do projeto, e cria a documentação seguindo o fluxo da aplicação. Fornece uma interface onde é possível ver informações sobre os serviços criados, além de poder realizar requisições com os *endpoints* criados no projeto, sem ter que utilizar software de terceiros para testar as requisições (SURYOTRISONGKO; JAYANTO; TJAHYANTO, 2017).

- **Spring Test:** O *Spring Boot* vem preparado para realizar testes. É possível escrever testes de unidade, integração, realizando poucas configurações (ANTONOV, 2015).

[13] <https://gradle.org/>
[14] <https://maven.apache.org/>
[15] <https://spring.io/projects/spring-data>
[16] <https://swagger.io/>

- **Spring Mail:** O *Spring Mail*[17] é um componente para envio de e-mail (HE, 2020, p.15-22).
- **Thymeleaf:** O *Thymeleaf* [18] é um *Template engine* para criação de páginas web. Foi utilizado para criar o modelo do *e-mail* (MITTAPALLI; ARTHUR, 2021, p.2).

2.2.9 NodeJS

Segundo PEREIRA, linguagens de programação como .NET, Java, PHP, Ruby ou Python têm uma característica em comum, são linguagens bloqueantes, ou seja, eles enfileiram as requisições, e executam uma por vez, colocando as outras em uma fila de espera. Diante de tal fato, à medida que aumenta os acessos ao sistema, podem haver muitos problemas na aplicação, como por exemplo, a indisponibilidade de recurso.

Um dos principais pontos que influenciaram na criação do *NodeJS*[19] é tentar construir uma arquitetura não bloqueante. Tal funcionalidade influencia na performance em relação ao consumo de memória e processamento, o que contribui para que os computadores possam usar o seu poder máximo de processamento de forma mais eficiente. Como o *NodeJS* é *single thread*, ele consegue isolar as requisições, ou seja, cada requisição terá uma única instância de um único processo. O *NodeJS* segue o *design Event Loop*, ele é orientado a eventos, e segue a mesma filosofia do *JavaScript*, porém, não existem eventos de mouse ou teclado como no *JavaScript*, mas, por exemplo, a entrada e saída de dados pode ser considerado um evento. Ele é um *loop* infinito, e a cada interação, ele verifica a sua fila de eventos, de modo a se certificar que um evento foi emitido (PEREIRA, 2014). Neste trabalho, o NodeJS, foi utilizado com o *Socket.IO*, para criar o *Websocket*.

2.2.10 Socket.IO

Em vez de trabalhar somente com o *Websocket*, neste trabalho foi utilizado o *socket.io*[20]. Atualmente, existem vários tipos de navegadores e dispositivos novos e antigos, e nem sempre um navegador estará apto a utilizar *Websocket*, principalmente por se tratar de uma tecnologia mais nova. Segundo PEREIRA, diante desse problema, nasceu o *socket.io*. Ele oferece compatibilidade com navegadores antigos, emulando outros tipos de *transports* de comunicação em navegadores que não possuem *Websocket*, tudo isso de forma totalmente abstraída para o desenvolvedor (PEREIRA, 2014).

O *socket.io* fica no cliente, ele busca informações sobre o *browser*, e a partir disso, define qual será o melhor tipo de comunicação com o servidor (PEREIRA, 2014). Segundo PEREIRA, o socket.io executa os seguintes *transports* de comunicação:

[17] <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mail.html>
[18] <https://www.thymeleaf.org/>
[19] <https://nodejs.org/pt-br/>
[20] Site oficial e documentação do socket.io <https://socket.io/>

- *WebSocket*;
- *Adobe Flash Socket*;
- *Ajax long polling*;
- *Ajax multpart streaming*;
- *Forever iframe*;
- *JSONP Polling*;

Se o navegador do usuário possuir compatibilidade com *WebSocket* ou *Adobe Flash Socket*, iniciará uma conexão bidirecional (PEREIRA, 2014). Casó contrário, a conexão será emulada e unidirecional, e enviará dados para o servidor em um pequeno intervalo de tempo, isso deixa o desempenho um pouco inferior, mas consegue entregar para o usuário a experiência de *real-time*. O fator importante é que toda a complexidade fica por conta do *socket.io*, facilitando a vida do desenvolvedor (PEREIRA, 2014).

2.2.11 PostgresSQL

Segundo MILANI, o PostgreSQL é um sistema gerenciador de banco de dados relacional, sendo utilizado para armazenar informações, além de oferecer a possibilidade de administrar o acesso às informações armazenadas (MILANI, 2008, p.25).

De acordo com (MILANI, 2008, p.23, 29, 167, 179), o *PostgreSQL* é gratuito, e é um dos bancos de dados mais utilizados atualmente, por ser seguro e oferecer vários mecanismos de recuperação de dados. Durante o desenvolvimento do trabalho, foi utilizado o *DBeaver*[21] para visualizar e gerenciar os dados da aplicação. Tal ferramenta foi de grande auxílio durante a modelagem do sistema e criação de *endpoints* para a *API*. Com o *PostgreSQL*, também foi utilizado neste trabalho o *plugin PostGIS*[22], para que fosse possível trabalhar com dados geoespaciais (OBE; HSU, 2011).

[21] <https://dbeaver.io/>
[22] <https://postgis.net/>

3 Desenvolvimento

Esta seção descreve como o sistema foi desenvolvido do início ao fim. Assim como no trabalho LEMOS, para gerenciar as tarefas de criação deste software em questão, foi utilizado o *scrum*.

Para organizar, gerenciar e desenvolver este trabalho, foi preciso definir três principais papeis. Neste caso, tem-se a figura do *Product Owner (PO)*, que é a representação do cliente; o *Scrum Master (SM)*, que é a pessoa responsável por resolver, negociar e tirar os impedimentos da equipe; e por fim, o grupo de desenvolvimento. Para aplicar o *scrum*, existe alguns rituais que necessitam ser feitos, como por exemplo, reuniões diárias para acompanhar as tarefas da equipe e, se preciso, ajudar a resolver alguns impedimentos. Logo, neste trabalho não foi possível trabalhar desta forma, devido à incompatibilidade de horários. Assim como descrito no trabalho LEMOS, neste trabalho só havia duas pessoas disponíveis, de modo que as funções foram divididas da seguinte forma: o orientador deste trabalho assumiu a função de *Product Owner* e *Scrum Master*, enquanto o autor do trabalho assumiu o papel de desenvolvedor. Foi adicionada a ferramenta *Trello*[1] para ajudar a gerenciar as tarefas do projeto.

O *Trello* é muito parecido com o método *Kanban*, que é uma ferramenta de gerenciamento de processos que visa aumentar a eficiência da produção e otimizar seus sistemas de movimentação, produção, realização de tarefas e conclusão de demandas. O *Trello* segue os princípios do método *Scrum*. Ele tem um aspecto importante que é a capacidade de estabelecer um gerenciamento compartilhado entre todos os membros da equipe. É possível criar diversos quadros, que referenciam projetos diferentes; para isso, ele conta com colunas, que são divididas em quatro status: a Fazer; em andamento; revisão; e concluído. Dentro de cada uma das colunas, é possível adicionar *cards*, que contêm a descrição das tarefas a serem realizadas. Também é possível anexar fotos, vídeos e documentos para ajudar a detalhar o que foi feito nas tarefas.

Ter um equilíbrio entre o desenvolvido do software e a documentação do mesmo é essencial, principalmente em ambiente universitário. Testes automatizados e aspectos relacionados à segurança e possíveis atualizações são importantes, mas é preciso ficar atento também às mudanças constantes que podem acontecer no projeto. Neste trabalho, não foi seguido um plano a risca, apenas respondemos às mudanças e o que achávamos que seria melhor para que o software tivesse todo o foco na qualidade. Prova disso foi a reconstrução do *back-end* utilizando o *spring*, e a remodelagem do banco de dados.

[1] Site oficial do Trello <https://trello.com/pt-BR>

3.1 Back-End

O *back-end* é responsável por cuidar das regras de negócio da aplicação. Ele recebe solicitações do *front-end* para tratar dados que estão encapsulados em uma pluralidade de objetos de negócio (BRUNSWIG et al., 2010). O *back-end* compõe *API RESTful*, detalhes de seu funcionamento estão presentes na sessão 2.2.1, e na sessão 2.2.8, onde é possível entender como funciona, de forma geral, a arquitetura do *back-end* deste trabalho.

Cada entidade representada na modelagem da aplicação na Figura 4, representa uma classe no *back-end*, que é objeto de negócio que foi citado no parágrafo anterior. Tais objetos, quando solicitados, passam por várias validações, que buscam preservar a integridade dos dados. O *Spring* e seus módulos facilitam o processo de validação do modelo, que é muito custoso, mas apesar disso, garante que os dados sejam persistidos e disponibilizados da forma correta. É importante ressaltar que o *Spring* é um *framework Java*, e tal ferramenta facilita o desenvolvimento de aplicações *Java* como foi descrito na sessão 2.2.7.

Neste trabalho, o *Spring* é utilizado juntamente com o *PostgreSQL*, devido à facilidade de uso e consistência dos seus comandos (SQL). O *PostgreSQL* é muito seguro, e é totalmente transacional, incluindo suas mudanças estruturais e destrutivas. Aceita vários tipos de dados, como, JavaScript Object Notation (JSON), Extensible Markup Language (XML), objetos genéricos, *tags* e matrizes. Pode-se escrever funções em *Phyton*, *Perl* e até C. É confiável e muito rápido, pois, faz uso estratégicos das indexações e consultas de otimização procurando trabalhar com o menor esforço possível (CARVALHO, 2017).

Apesar de ser um projeto a parte, e acessar somente o banco de dados *PostgreSQL*, o *websocket* também faz parte do *back-end*, porém, ele é utilizado somente para retornar dados em tempo real para aplicação. Neste trabalho, a base do *websocket* é a tabela *monitoring* mostrada na Figura 4. Tal tabela recebe coordenadas do veículo em questão, além de ser o ponto inicial para buscar demais dados. O *NodeJS* foi utilizado na construção do *websocket*, porque utilizando o mesmo, juntamente com o *socket.io*, foi possível ter um *websocket* escalável, poderoso e bem estruturado rodando em pouco tempo, seguindo apenas o tutorial disponível na obra de (PEREIRA, 2014), e quando necessário, buscando demais informações na internet.

Como explicado no Capítulo 1, este trabalho tem como principal objetivo desenvolver a parte de rastreamento e monitoramento de veículos de transporte escolar do projeto SBRP. Os softwares a serem desenvolvidos são: *API Restful*, *Websocket* e a interface *web*. O sistema embarcado faz parte do projeto SBRP, e não foi desenvolvido neste trabalho, porém, o mesmo será adaptado para ser utilizado no trabalho em questão, pelo fato de ser importante para o funcionamento do sistema.

A modelagem do sistema conforme representa a Figura 4 pode ser dividida segundo o funcionamento de cada aplicação. Para criar o planejamento de uma viagem através do dispositivo embarcado, basta ter acesso às tabelas *trip*, *vehicle* e *city*, pois, a partir delas é possível ter acesso às rotas e os veículos disponíveis conforme a cidade do motorista. Após preencher os dados, é gerado uma informação na tabela *dailyplanning* com o status "TNS", que significa Turno não Iniciado, do inglês *Turn Not Started*. Se o Motorista clicar no botão Iniciar Turno, o status da tabela *dailyplanning* mudará para "TS", que significa Turno Iniciado, do inglês *Turn Started*. É nesse momento que a tabela *monitoring* começará a receber informações como horário e coordenadas geográficas de cada ponto em um intervalo de cada dois segundos.

Após finalizar o turno, a tabela *dailyplanning* fica com status de "FT", que significa Turno Finalizado, do inglês *Finished Turn*. A tabela *busstop* é responsável por armazenar todos os pontos de parada que o veículo deve fazer na rota. Sobre a interface Web através da tabela *dailyplanning*, é possível ter acesso a todos os veículos que estão com os turnos iniciados ou não, assim como a rota a ser percorrida, que está armazenada na tabela *trip*. Os pontos de parada do veículo, como dito anteriormente, estão disponíveis através da tabela *busstop*; a tabela school fornece acesso aos pontos geográficos das escolas, e por fim, as informações dos veículos em tempo real podem ser encontradas na tabela *monitoring*. A tabela city irá garantir que o sistema está buscando as informações segundo a cidade que o usuário está logado. Todas as operações descritas são controladas pela API Restful e o Websocket. A demais tabelas que não foram citadas, não influenciam diretamente no funcionamento do Sistema de Rastreamento e Monitoramento escolar, mas fazem parte for projeto SBRP.

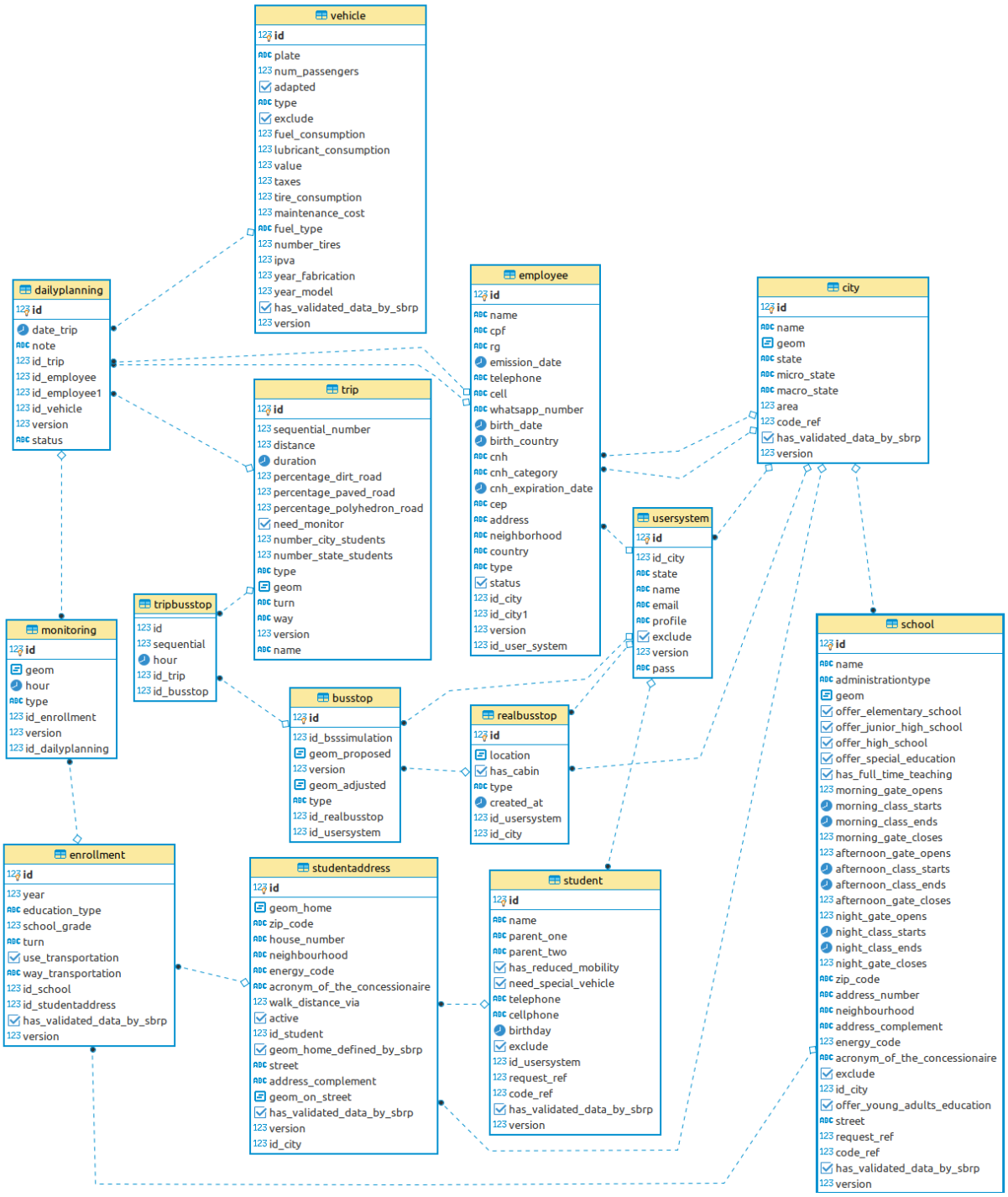


Figura 4 – Modelagem da aplicação.

As próximas seções detalham e exemplificam como foi construída a arquitetura da API Restful e do Websocket.

3.1.1 Arquitetura da API Restful

A Figura 5 exemplifica como foi construída a *API Restful*, e pode ser explicada da seguinte forma: o *Controller* recebe uma requisição do usuário, que passa para a camada *Service* para receber tratamentos relacionados à regra de negócio. Para retornar esses objetos, o *Repository* é chamado e verifica todas as *Entities* e seus relacionamentos, e aciona o *Spring JPA* para acessar o banco de dados e realizar as operações. O resultado é retornado para o *Service*, que por sua vez, chama o Data Transfer Object (DTO) para tratar os dados, e por fim, os mesmos são retornados para o *Controller*, que envia o resultado para o cliente através do navegador. Para cada tabela criada no banco de dados, como é possível ver na Figura 5, é preciso criar um domínio e um repositório para ela.

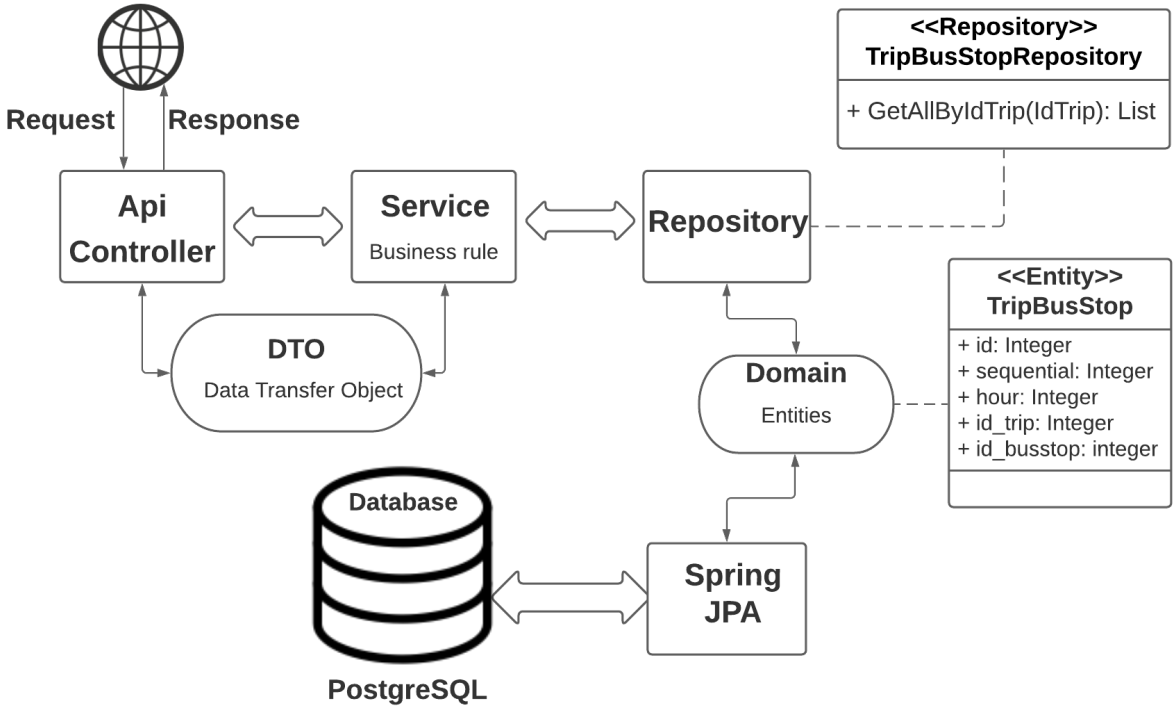


Figura 5 – Arquitetura da *API Restful*.

3.1.2 Arquitetura do Websocket

A Figura 6 exemplifica como foi construído o *Websocket*, e pode ser explicada da seguinte forma: um cliente *socket.io* faz uma requisição ao servidor, se o servidor aceitar essa requisição, a mesma se torna única e persistente. A requisição chega ao *Socket.io Server*, que aciona o *controller*; logo, o *controller* aciona a *model*, que é onde fica a regra de negócio do sistema. A *model* requisita as operações necessárias no banco de dados e retorna os dados para o *controller*, onde os mesmos serão tratados e enviados para o *Socket.io Server*, que por sua vez, envia os dados para o cliente em questão. Isso acontece

continuamente a cada dois segundos, ou até o usuário fechar a conexão. Também não é preciso fazer um tratamento para separar o cliente, porque por trás dos panos, o *socket.io* trata cada conexão de forma isolada.

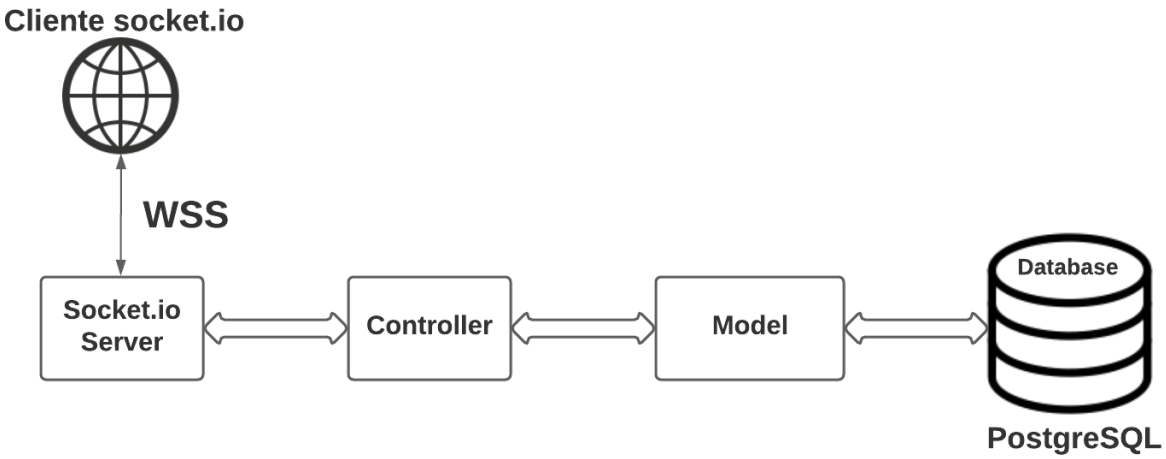


Figura 6 – Arquitetura do *Websocket*.

3.2 Front-End

O dispositivo *mobile* e o *front-end*, compartilham da mesma tecnologia; ambos foram desenvolvidos utilizando o *react*, que como foi detalhado na sessão 2.2.4, é uma biblioteca para desenvolver interfaces de usuário. Como já explicado anteriormente, o *front-end* foi desenvolvido neste trabalho, já o dispositivo embarcado, faz parte de um dos módulos do SBRP, sendo essencial para o funcionamento da aplicação.

Na Figura 1, apresentada no capítulo 1, é possível perceber que o *front-end* faz requisições na *API Restful*. Isso acontece para que seja possível realizar login na aplicação, e montar os dados iniciais nas telas, além de se conectar ao *websocket*. A Figura 7, é uma versão reduzida do projeto, para que seja possível visualizar melhor como funciona o *front-end* e o mobile. Como é possível ver na Figura 7, o dispositivo móvel é utilizado para mandar as coordenadas para o *back-end* através da *API Restful*, que por sua vez, grava os dados no *PostgreSQL*. Já o *front-end (web)* se conecta ao *Websocket* e traz as últimas coordenadas do veículo inseridas no banco de dados; logo depois, são apresentadas ao usuário utilizando as bibliotecas *leaflet* e *react-leaflet*, para apresentar as informações no *OpenStreetMap*.

O *front-end* foi construído com *react* e *Material UI* para agilizar o desenvolvimento da interface. Para aplicar conceitos de orientação a objetos buscando criar módulos e obter mais qualidade e coesão no desenvolvimento da interface, foi utilizado o *TypeScript*.

A interface *mobile*, foi desenvolvida em *JavaScript* puro, e para desenhar a interface não foi utilizado nenhum sistema de *template* com componentes já prontos, como no *front-end (web)*, foi utilizado apenas Cascading Style Sheets (CSS).

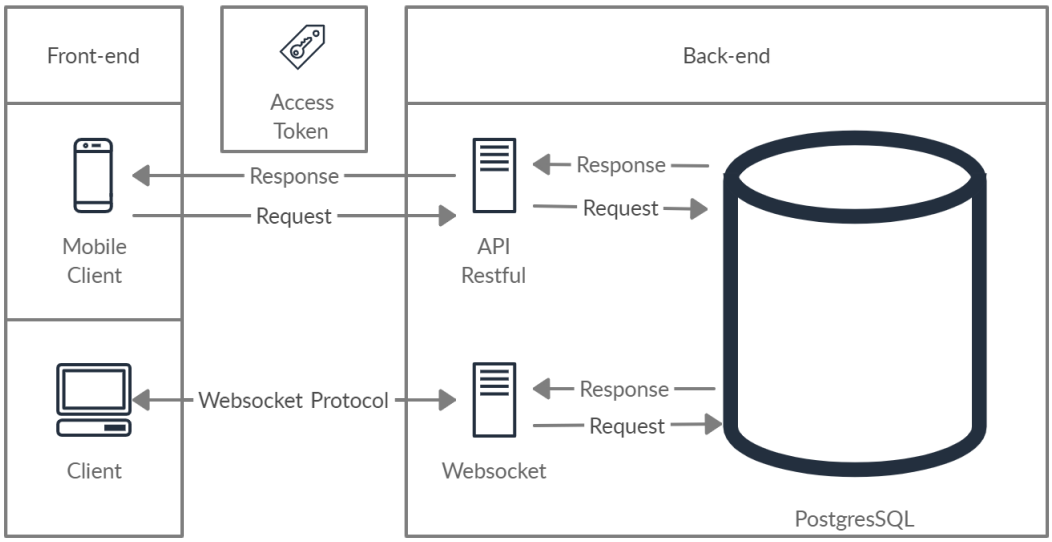


Figura 7 – Demonstração da arquitetura reduzida do Sistema.

4 Resultados

Neste capítulo, apresenta-se os resultados obtidos com o desenvolvimento do sistema. É importante ressaltar novamente, que a interface do sistema mobile foi desenvolvida no contexto de outro trabalho do projeto SBRP, e aqui utilizou-se apenas os *endpoints* previamente propostos. Em casos específicos, apenas as adaptações necessárias para que o mesmo pudesse funcionar foram feitas neste trabalho.

Na seção 2.2 foram apresentadas as tecnologias utilizadas no trabalho, sem dar ênfase em como foram utilizadas. Já no capítulo 3, foi detalhado de forma clara e direta como todas as tecnologias foram utilizadas, bem como o quanto foram importantes para este trabalho.

Neste capítulo, em especial na seção 4.1, serão detalhadas as metodologias de testes empregadas para garantir o correto funcionamento do sistema desenvolvido. Os resultados serão apresentados através de figuras conforme com o fluxo de utilização da aplicação na seção 4.2. Por fim, na seção 4.3, será apresentada a conclusão geral sobre a metodologia dos testes e dos resultados obtidos.

4.1 Metodologia para testes e validação

Para LUFT, os testes de software são fundamentais, pois, contribuem para melhorar a qualidade, diminuir os custos, evitar o retrabalho, aumentar a segurança e a confiabilidade, além de antecipar a descoberta de falhas e incompatibilidades no software. Dessa forma, nesse trabalho, optou-se pelo uso de metodologias de testes que têm como principal objetivo garantir que as funcionalidades implementadas estejam funcionando, mesmo após eventuais alterações de requisitos. Para isso, é importante dar ênfase aos testes de integração, que como descrito por LUFT, consistem em vários módulos da aplicação, sendo necessário testá-los em grupo, simulando o fluxo de trabalho. Outro ponto importante é a sua abrangência, e o pouco tempo que é preciso para a sua implementação.

Na metodologia de teste de integração, o software é construído e testado em pequenos módulos, tornando o teste mais eficiente. Os erros podem ser encontrados e isolados mais facilmente de modo a serem corrigidos mais rapidamente, além de diminuir a probabilidade de serem refletidos nas partes superiores do software (LUFT, 2013, p.28). Já os testes unitários que buscam testar uma unidade mínima de um software, como, por exemplo, um método de uma classe, são caros e demorados para serem implementados, pois, demandam uma grande consciência e experiência do desenvolvedor, o que o faz ser muito custoso para trabalhos com pouco tempo de prazo para implementação (SILVA;

MORENO, 2012).

Para apresentar os testes realizados, será utilizada uma técnica que se chama teste de caixa-preta. De acordo com BRUNELI et al., neste tipo de teste, o software se torna uma caixa preta, ou seja, somente interessa encontrar uma situação em que o programa não funciona conforme as especificações. O principal objetivo dos testes caixa-preta é validar os requisitos funcionais do software, não se importando com a estrutura interna ou com o código do programa. O que importa é a entrada e saída de dados do programa, ou seja, apenas com as respostas que o sistema dá à determinadas ações (BRUNELI et al., 2006) e (KOSCIANSKI; SOARES, 2007).

O primeiro passo para se realizar o teste caixa-preta é entender como funciona cada módulo do sistema e as conexões entre eles. Depois disso, basta definir uma série de testes, que verifica se todos os módulos do software têm as reações esperadas, como combinado uns com outros (BRUNELI et al., 2006) e (KOSCIANSKI; SOARES, 2007).

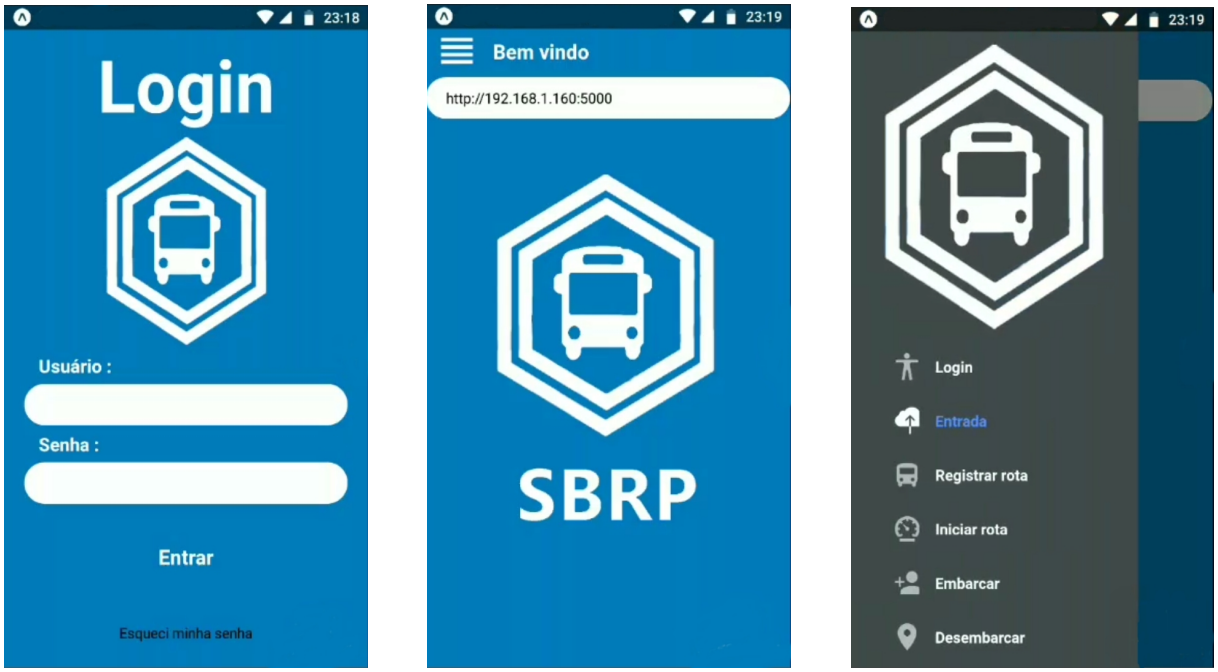
4.2 Resultados obtidos

Para ilustrar os resultados obtidos com o desenvolvimento desse trabalho, foi realizada a simulação de seu funcionamento. Portanto, nessa seção, será apresentada uma simulação do funcionamento do *software*, desde a inserção dos dados no sistema embarcado até a apresentação dos dados no sistema *web* proposto. Esse é o caso de teste escolhido de forma a passar por todos os módulos do sistema utilizado a metologia de testes de caixa-preta, descrita na seção 4.1.

É importante ressaltar que o teste foi realizado traçando toda a rota escolhida andando. O sistema *web* utilizou o tempo todo a conexão *Wi-Fi*. O sistema embarcado durante o trajeto utilizou as redes *Wi-Fi*, *4g* e *3g* em alguns pontos onde a recepção de sinal do dispositivo não era muito boa.

4.2.1 Planejamento da viagem

A primeira ação a executar é o registro de uma possível rota que o veículo deverá percorrer. Esta rota é composta por pontos de ônibus que o veículo deverá visitar, cada um desses pontos terão alunos a serem embarcados, e os alunos são previamente cadastrados em cada um deses pontos. Para o registro da rota, foi utilizado o aplicativo do *SBRP Mobile*, que faz parte do projeto SBRP. Na tela de login basta fornecer o *e-mail* e senha de usuário cadastrado, como demonstra a Figura 8a. Ao realizar o login, será possível apresentar a tela inicial do aplicativo, como mostra a Figura 8b. Como é possível ver na Figura 8c, o menu de opções do aplicativo fornece algumas opções como, por exemplo, registrar uma nova rota, sinalizar o início da execução da rota e, também, o registro de embarque e desembarque de alunos.

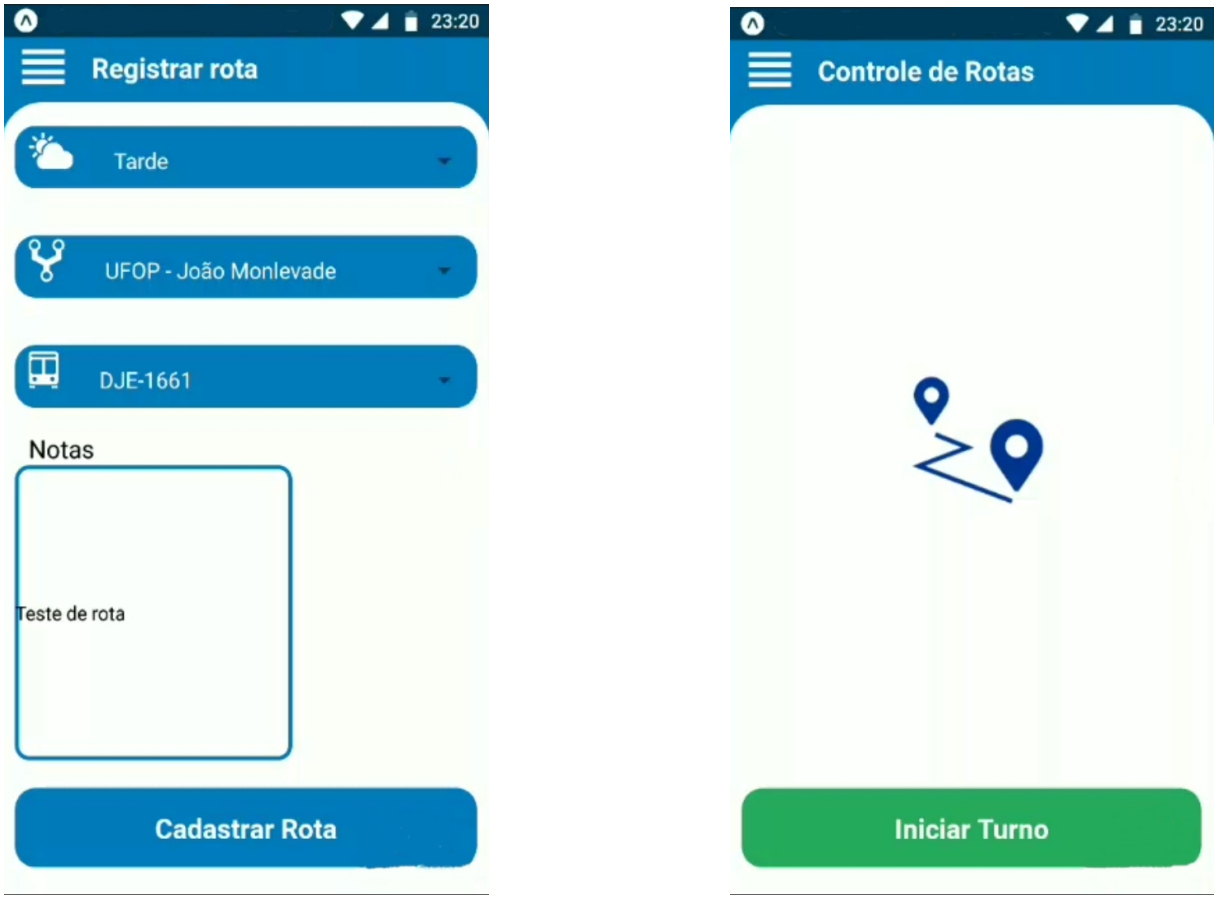


(a) Tela de login (b) Tela inicial (c) Tela de menu

Figura 8 – Entrando no sistema embarcado para acessar o menu de Registrar rota.

Ao acessar a opção Registrar rota, será apresentado um formulário onde é possível escolher o turno da viagem e a rota a ser executada, bem como identificar o veículo que deverá executar a viagem, como é possível ver na Figura 9a. É preciso ressaltar que tanto a rota quanto o veículo, estarão disponíveis de acordo com a cidade do usuário cadastrada no banco de dados da aplicação. Ainda na Figura 9a, após o preenchimento das informações necessárias na tela de Registrar rota, basta clicar no botão "Cadastrar Rota"e o usuário será direcionado para a tela que permite iniciar o turno de trabalho, como é possível observar na Figura 9b, que tem o botão "Iniciar Turno", e ao ser pressionado o veículo começará a ser monitorado.

Quando o turno é inciado, o dispositivo embarcado inicia o processo de registro de informações do veículo durante a viagem, portanto, passa a enviar os dados de localização, como também, as demais informações para o banco de dados utilizando a *API Restful* disponível.



(a) Tela de cadastrar rota

(b) Tela de iniciar turno

Figura 9 – Cadastrando rota e sendo direcionado para tela de iniciar turno.

Antes de iniciar um turno, o sistema *web* para o monitoramento veicular apresenta a listagem de todos os veículos disponíveis para operação, porém, em modo *offline*. Para tanto, basta realizar a autenticação de usuário no sistema *web* proposto, como ilustra a Figura 10.

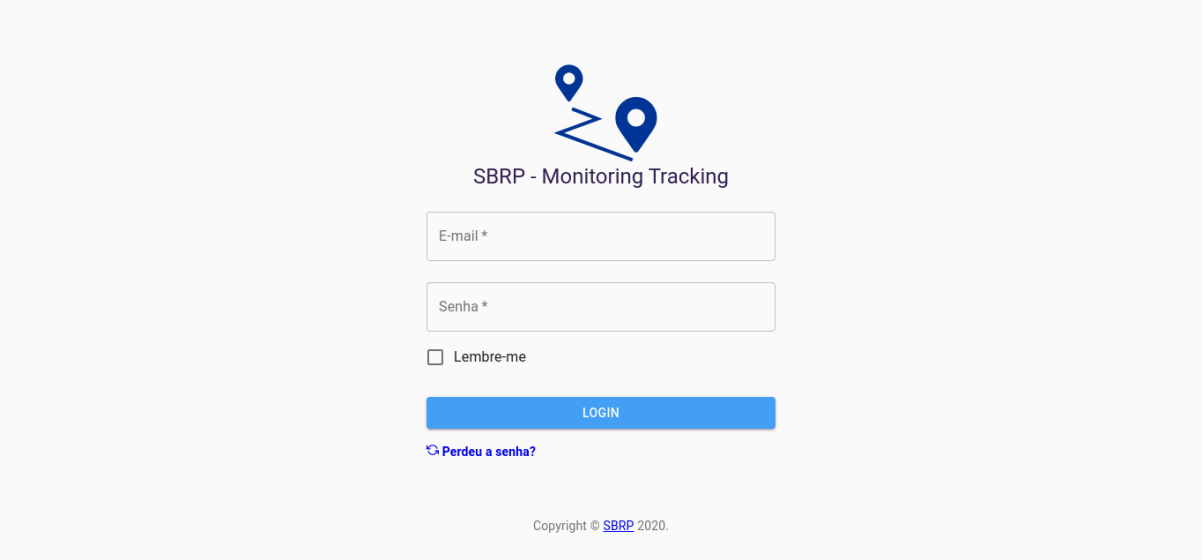


Figura 10 – Tela de login do sistema web.

Após a autenticação, o usuário é direcionado para a tela inicial do sistema *web*. Para visualizar a rota registrada, basta acessar o menu do lado esquerdo e clicar na opção "Monitoramento", como é possível ver na Figura 11.

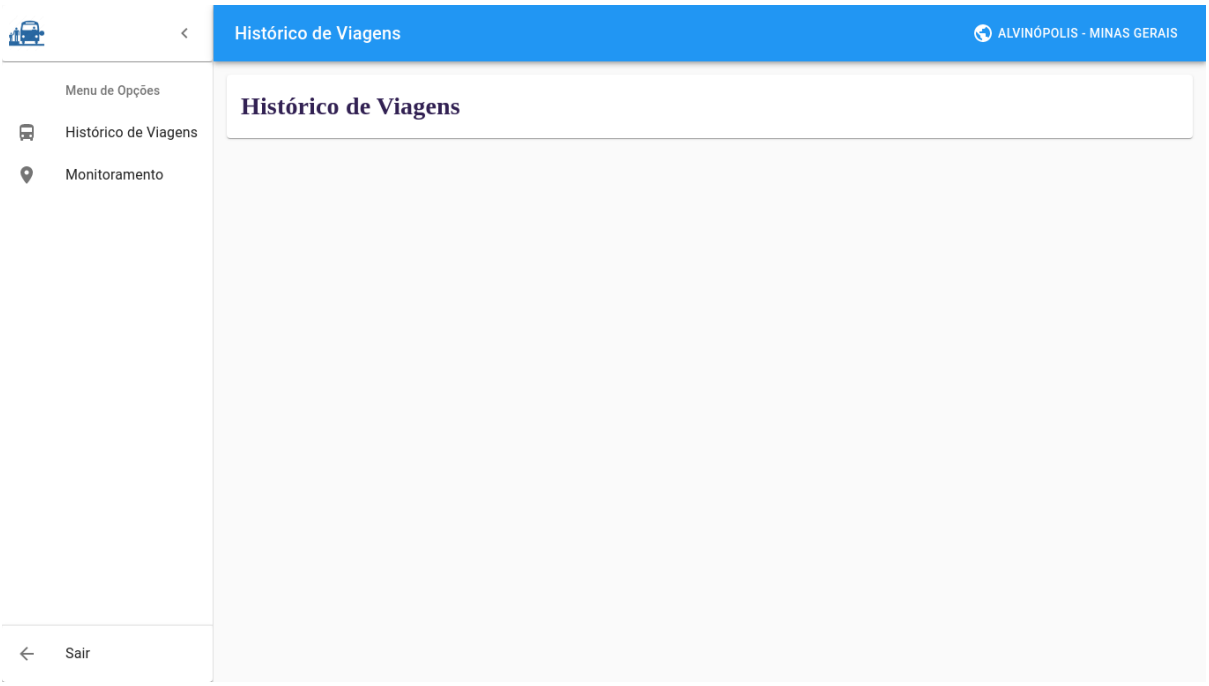


Figura 11 – Tela de inicial do sistema web.

Em seguida, ainda no menu do lado esquerdo da tela, ao clicar em *offline*, será possível visualizar o veículo que foi programado para a execução da rota. Ao clicar sobre o veículo desejado, será apresentado um mapa com a sua localização em tempo real, como demonstra a Figura 12. Para tratar a performance do sistema, com o objetivo de reduzir

o processamento, nesse momento será apresentado apenas o posicionamento do veículo, portanto, não será possível visualizar demais informações sobre o mesmo. Em outras palavras, nesse instante do teste realizado, o dispositivo embarcado não disponibiliza as demais informações para o sistema *web*. Esse fato ocorre, pois, o turno ainda não foi iniciado.

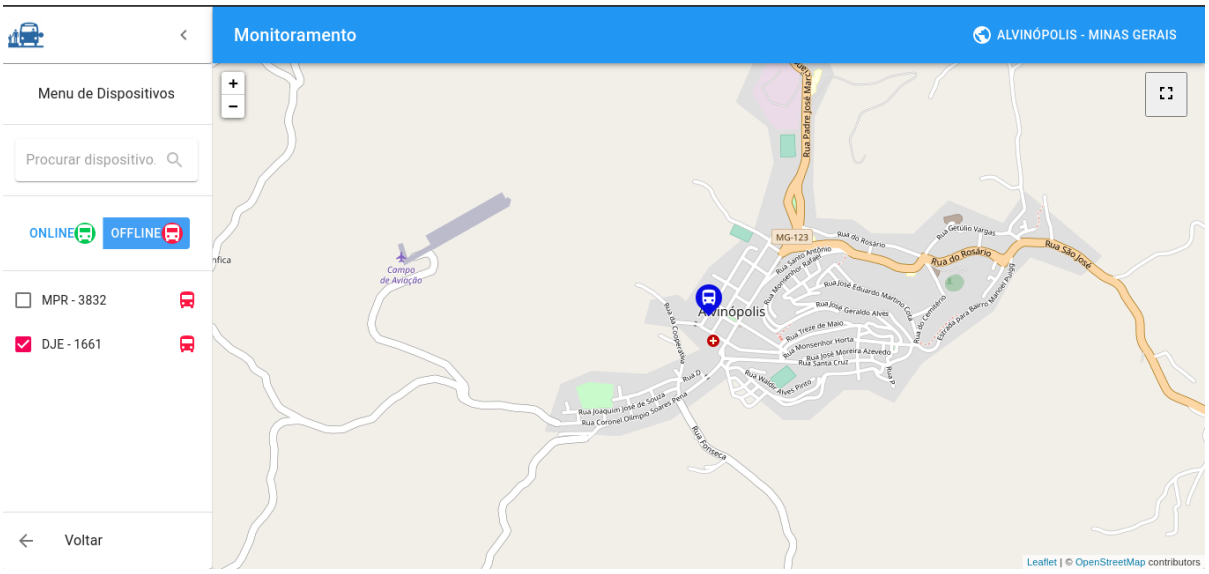


Figura 12 – Tela de monitoramento do sistema *web*.

4.2.2 Iniciar Turno

Para iniciar um turno, basta que o motorista clique no botão "Iniciar Turno"no dispositivo embarcado, como ilustrado pela Figura 9b. A partir desse momento, o sistema mobile inicia o envio de dados para o banco de dados através da *API Restful*. Também é possível apresentar os movimentos do veículo com uma taxa de atualização de dois segundos na própria tela do sistema mobile, como é possível ver na Figura 13. Além disso, também é disponibilizado uma linha na cor azul, que indica a rota que o veículo deve percorrer. Dessa forma, o condutor do veículo poderá consultar a rota que deverá ser executada caso o mesmo não conheça o trajeto.

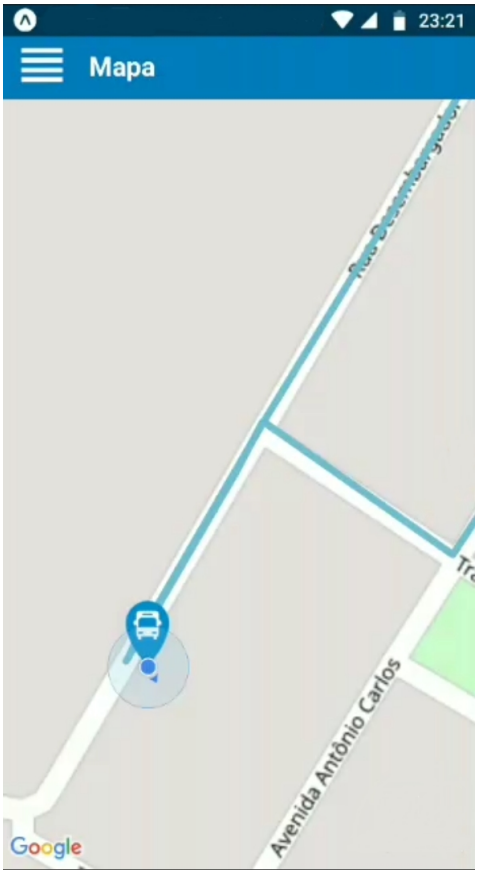


Figura 13 – Tela de monitoramento do sistema *mobile*.

Por outro lado, no sistema *web* proposto nesse trabalho, é possível visualizar a rota que o veículo deverá executar, como é possível observar pela Figura 14. Com base nas informações enviadas pelo SBRP *mobile*, através do uso de *websocket*, é possível atualizar o mapa indicando a posição atual do veículo em tempo real.

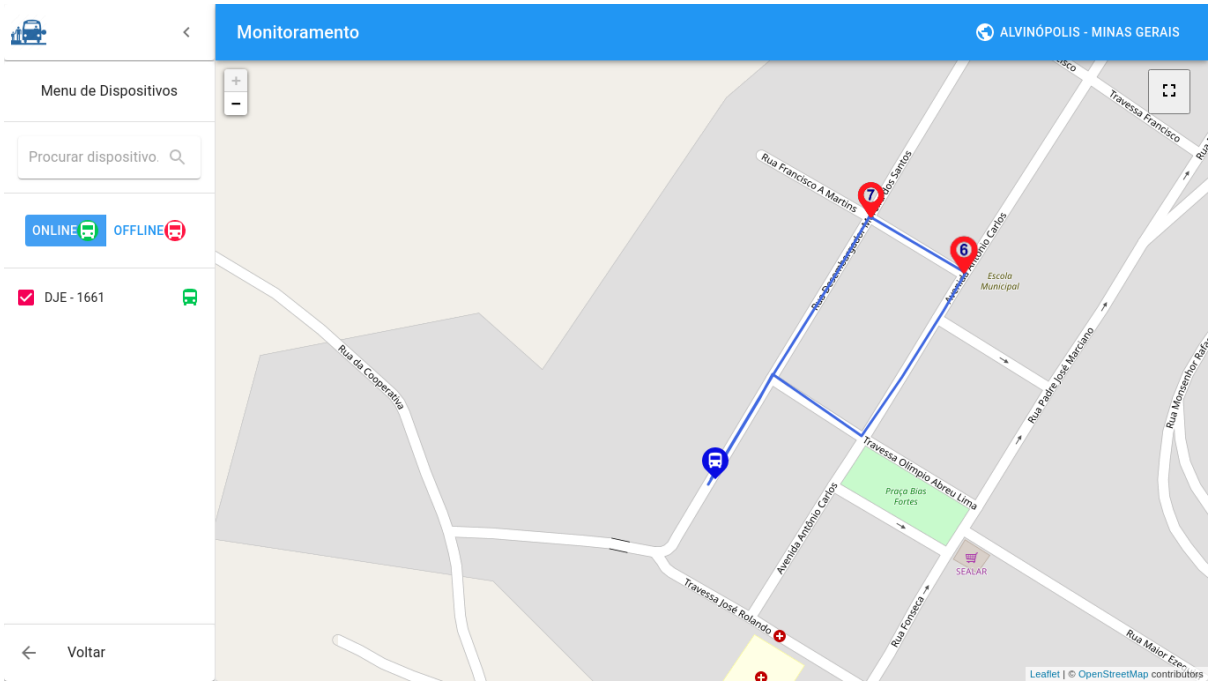


Figura 14 – Sistema *web*, tela que mostra que o monitoramento do veículo foi iniciado.

Algumas informações complementares são disponibilizadas ao clicar no ícone do veículo no mapa, onde será apresentado um *pop-up*, como é possível observar através da Figura 15. Entre tais informações, destacam-se: placa do veículo; velocidade; tempo estimado de viagem; tempo percorrido de viagem; ícone de status que mostra se o veículo está na rota correta ou não (se o ícone estiver verde, o veículo está na rota correta, se estiver vermelho, está percorrendo a rota errada.); e outro ícone, que ao ser clicado, leva para lista de alunos embarcados e não embarcados no veículo.

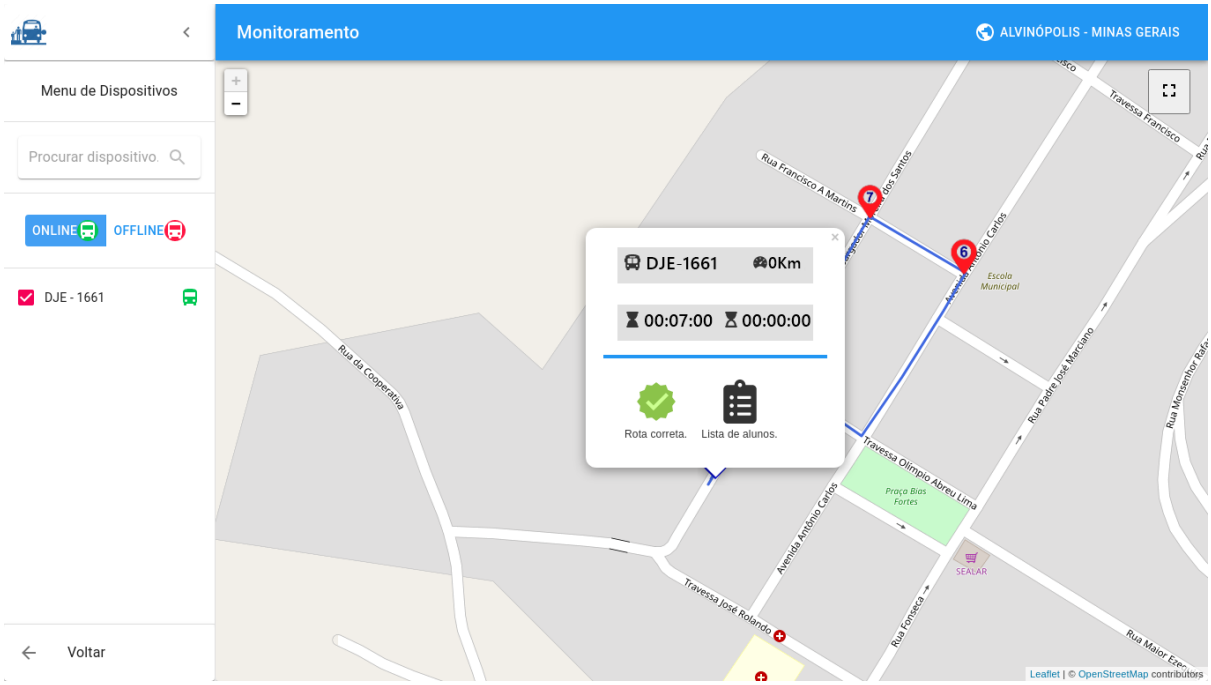


Figura 15 – Sistema *web*, tela com pop-up de informações do veículo monitorado.

Como é possível ver na Figura 16, quando o veículo está fora de rota, o ícone do *pop-up* fica vermelho, e logo abaixo terá uma frase informando que o veículo está na rota errada. Para realizar os testes, o veículo foi simplesmente tirado de rota, mas basta que o veículo se afaste 25 metros da rota correta, para que a *pop-up* apareça com o ícone vermelho.

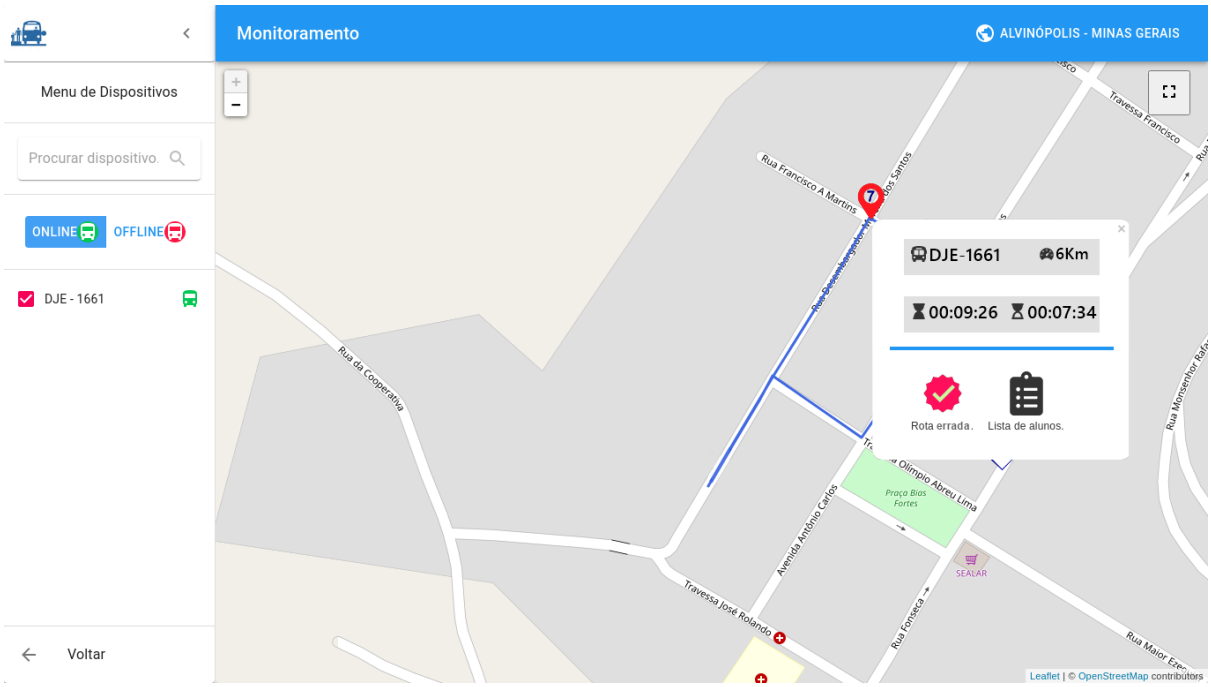


Figura 16 – Sistema *web*, tela com veículo fora de rota.

Os testes realizados, principalmente o da Figura 17, foram feitos andando, por isso, a quilometragem está baixo e o percurso é relativamente pequeno. Apesar de saber que os testes não seriam suficientes para um ambiente produtivo, o objetivo do teste foi provar que o software é funcional.

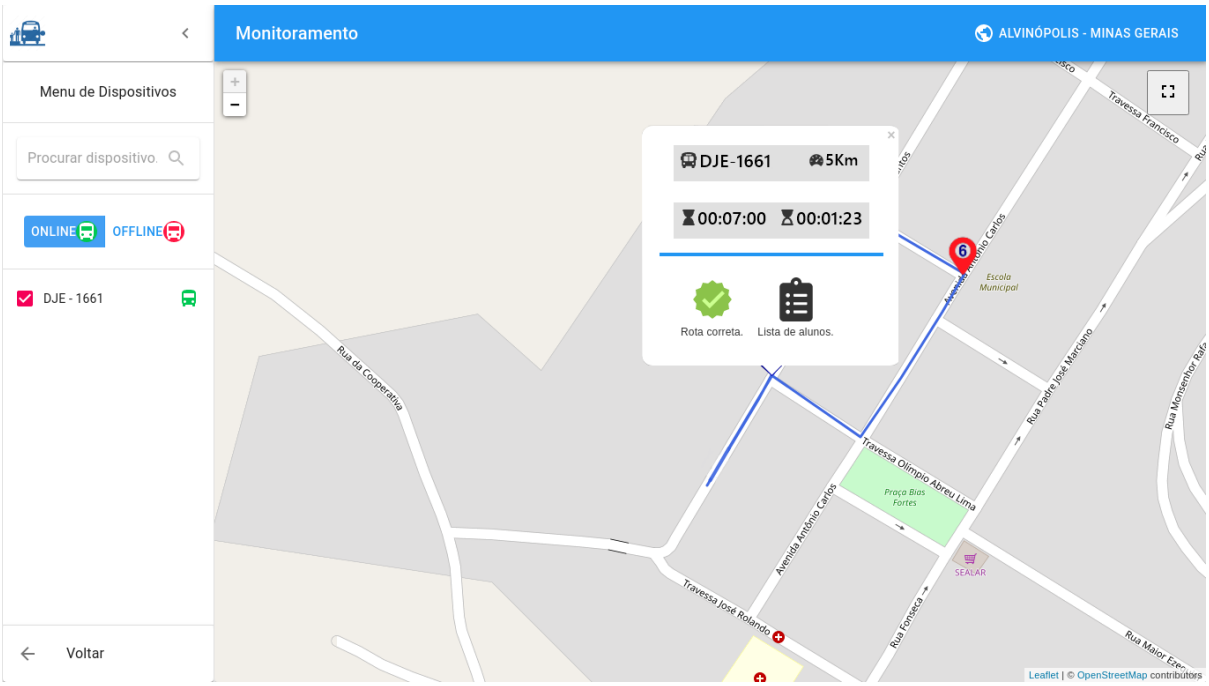
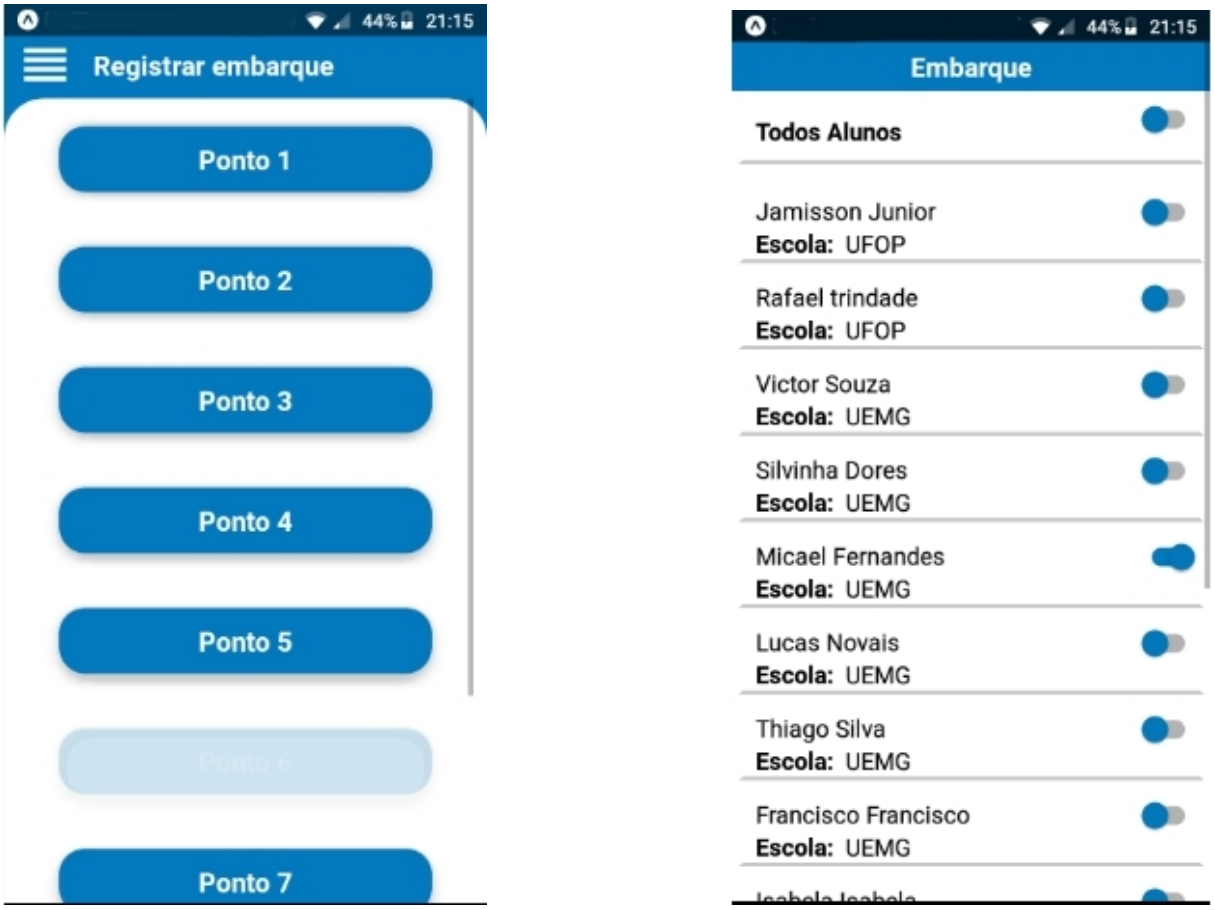


Figura 17 – Sistema *web*, tela de demonstração de movimento na rota.

4.2.2.1 Embarque e desembarque de alunos

O motorista é o responsável pelo embarque e desembarque de alunos, e isso é feito através do dispositivo embarcado. Para isso, basta ir ao menu do sistema *mobile*, e clicar na opção embarcar. O motorista será levado para uma tela, onde ele deverá selecionar o ponto que o aluno está esperando, como é possível ver na Figura 18a. Logo depois, o motorista será levado para a tela de embarque e desembarque de alunos. Para embarcar ou desembarcar um aluno, é só deslizar o botão, como é possível ver na Figura 18b. Cada ponto, terá um grupo de alunos a ser embarcado ou desembarcado.



(a) Tela de ponto de parada (b) Tela de registrar embarque de aluno

Figura 18 – Embarcando um aluno no veículo.

Também é possível ver os alunos embarcados, no sistema *web*. Para isso, basta clicar em cima do veículo que está sendo monitorado, e logo depois clicar no ícone de lista de usuário, para ver a listagem, como é possível ver na Figura 19.

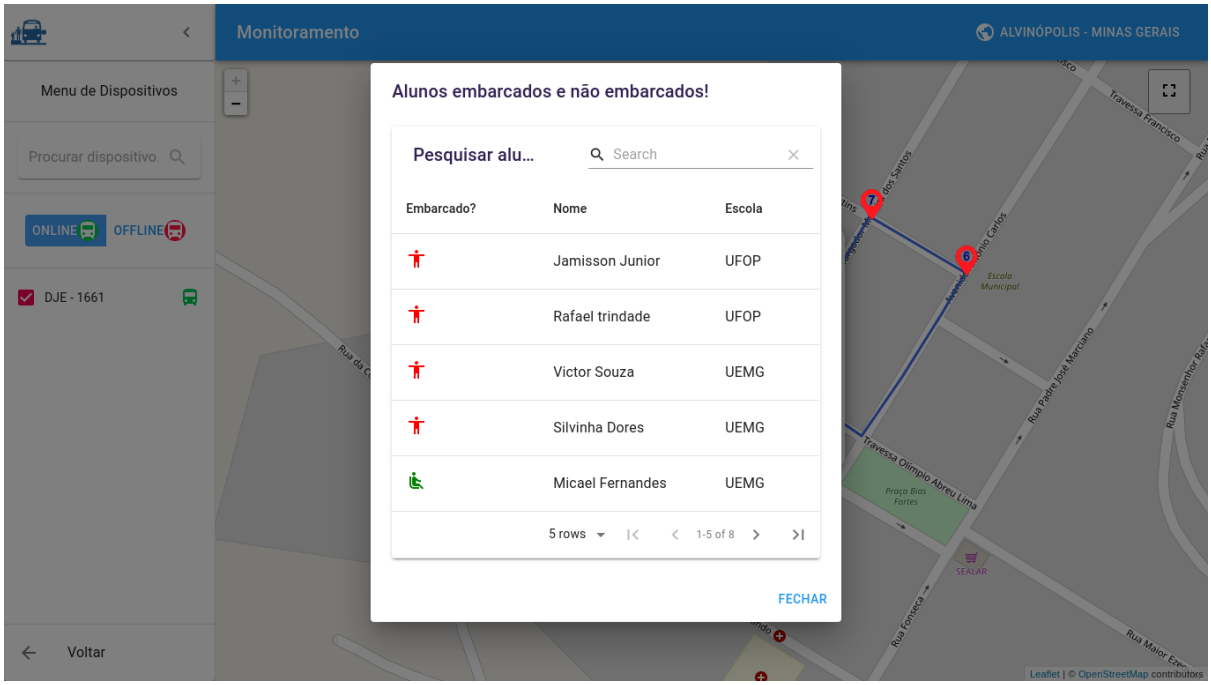


Figura 19 – Modal do sistema *web* de alunos embarcados e não embarcados no veículo.

4.2.2.2 Funcionalidades Extras

Para oferecer a melhor experiência possível para o usuário, é possível encolher a *sidebar* (menu lateral) do sistema, clicando na seta, que fica ao lado do ícone de ônibus no canto superior esquerdo, cujo resultado é possível ver na Figura 20. Ainda na Figura 20, também é possível ver um botão dentro do mapa, no canto superior direito, clicando nele, o mapa fica em modo *full screen*.

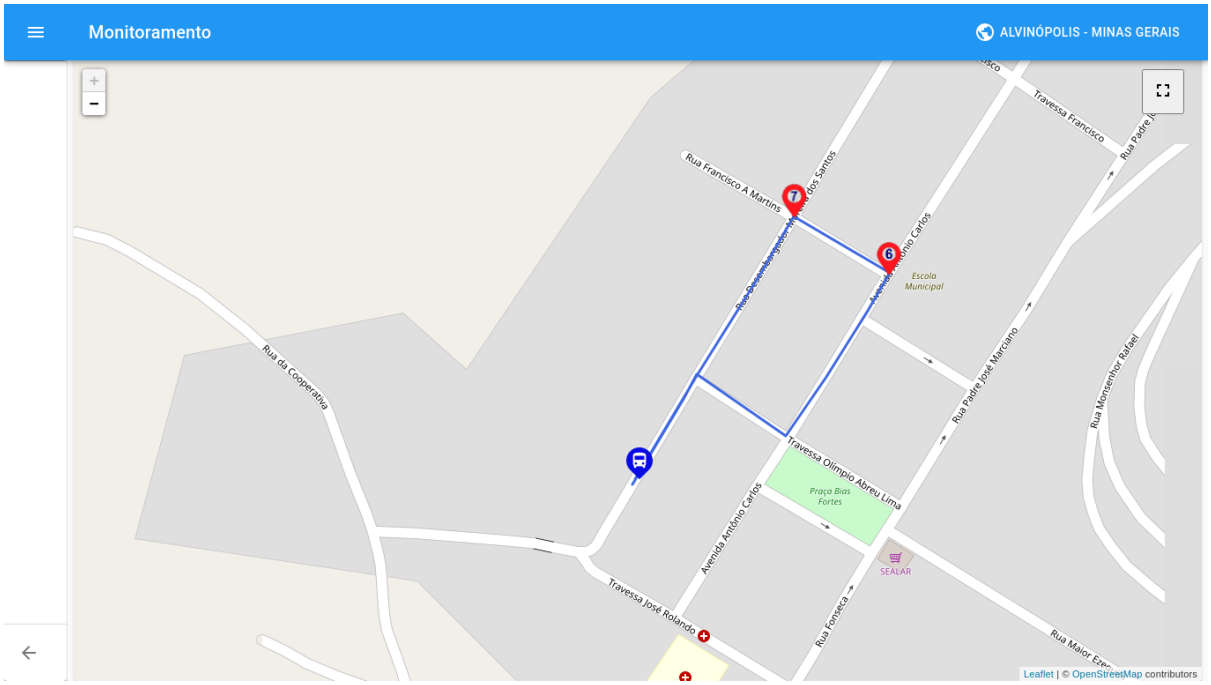


Figura 20 – Tela do sistema *web* com a *sidebar* oculta.

Quando a tela fica em modo *full screen*, aparece outro botão logo abaixo do botão citado anteriormente. Ao clicar nele, aparece um menu no lado esquerdo da tela para que o usuário manuseie os veículos que ele quer monitorar, como é possível ver na Figura 21. Para sair do modo *full screen*, basta clicar novamente no primeiro botão do canto superior direito.

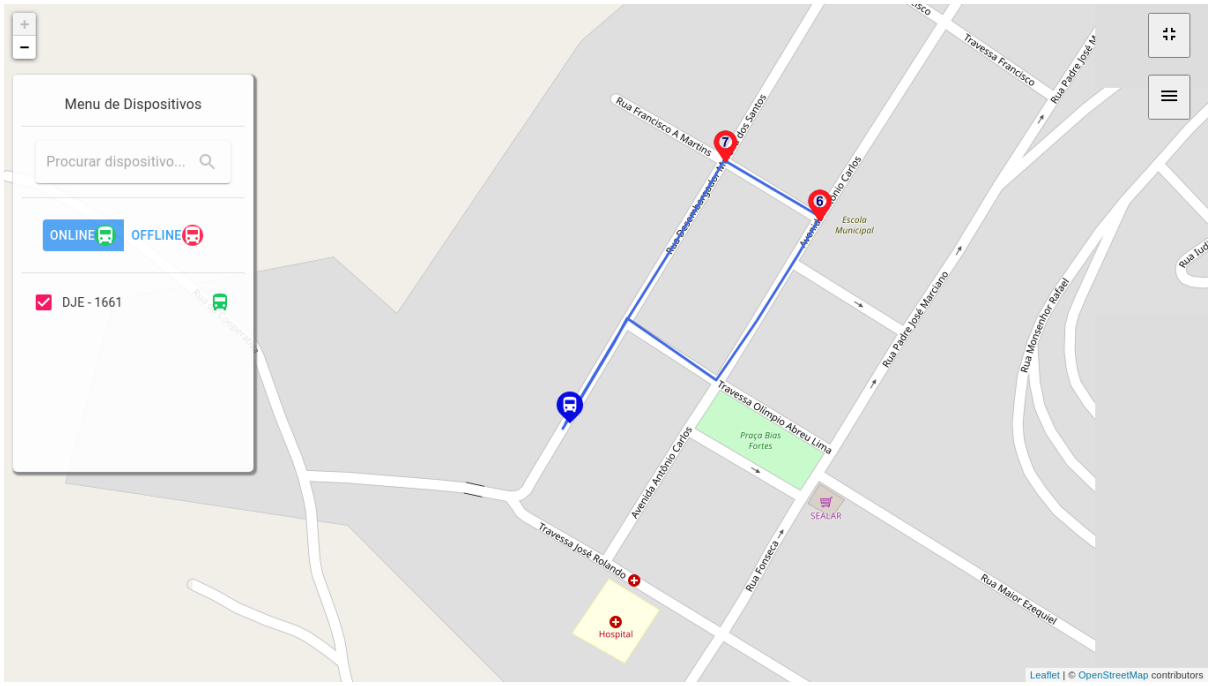


Figura 21 – Tela do sistema *web* em modo *full screen*.

4.2.3 Finalizar Turno

Para finalizar um turno, basta que o motorista vá ao menu do dispositivo embarcado e clique em registrar rota. A tela será um pouco diferente, da mostrada na seção 4.2.1, como é possível ver na Figura 22. Isso acontece porque o turno já foi iniciado, e agora é preciso finalizá-lo. Para isso, basta clicar no botão finalizar turno, e o sistema mobile não enviará mais informações sobre o veículo para a *API*. Além disso, o mesmo não irá aparecer mais no menu do sistema *web*, e não será possível vê-lo no mapa. Logo depois, o usuário será direcionado para tela inicial do sistema *mobile*.

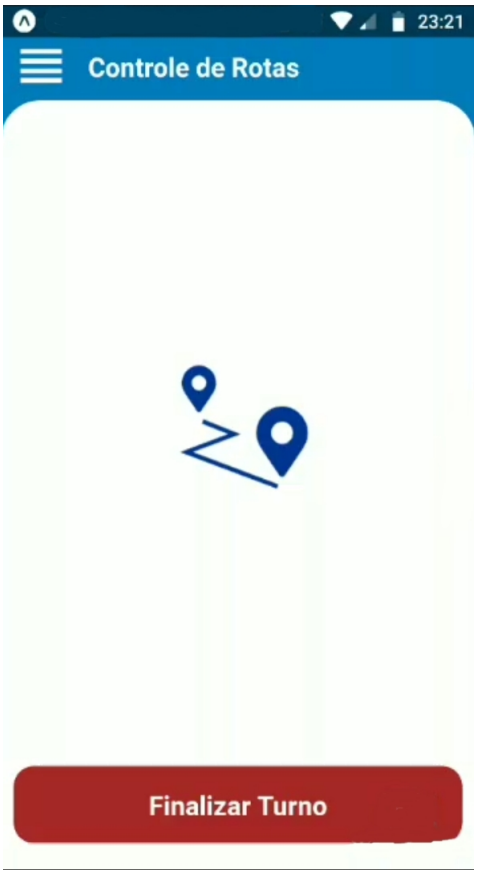


Figura 22 – Tela do dispositivo embarcado para finalizar um turno.

4.3 Conclusão sobre os resultados

Após finalizar os testes e verificar todas as funcionalidades do sistema proposto, pode-se perceber que o dispositivo embarcado contém apenas as funcionalidades que serão utilizadas no dia a dia do motorista. Por fim, o Sistema web é responsivo e adaptável à maioria dos tamanhos de telas que existem atualmente, devido ao uso do *Material UI* descrito na seção 2.2.5 , assim como a tela do mapa, que pode ser expandida em modo tela cheia, e ainda assim, oferecer os mesmos recursos da tela em modo comum, o que facilita bastante o trabalho de quem está monitorando os veículos.

Os testes foram realizados com apenas uma instância do aplicativo, portanto, os resultados apresentados não podem ir além. Todas as operações realizadas no aplicativo embarcado ofereceram respostas instantâneas, sendo assim, o aplicativo só dependerá do tempo que o banco de dados irá demorar para responder. O sistema web também apresentou respostas instantâneas em todas as operações que foram realizadas.

Ao considerar outros trabalhos como, por exemplo, os descritos na seção 2.1, todos apresentam as funcionalidades básicas do software proposto, que é a capacidade de monitorar e rastrear veículos, mesmo que não seja em tempo real. O que pode fazer diferença é a forma que as informações são enviadas para o servidor e são refletidas para o usuário. Neste caso, acredita-se que o sistema proposto neste trabalho pode levar vantagem, já que o mesmo utiliza *websocket*, o que faz com que a conexão entre cliente servidor fique mais leve e persistente, além de poder trafegar dados de forma simultânea.

5 Conclusão

Neste trabalho, foi possível apresentar o processo de desenvolvimento de uma ferramenta web para rastreamento e monitoramento de veículos de transporte escolar. O trabalho apresentou uma *interface mobile*, que utiliza uma *API Restful* para autenticar usuários e enviar informações ao banco de dados, inclusive as referentes à localização do veículo. Além disso, a ferramenta desenvolvida disponibiliza uma *interface Web*, que também utiliza a mesma *API Restful* para apresentar informações para os usuários no mapa em tempo real utilizando o *Websocket*.

A partir dos resultados obtidos, foi possível perceber que o uso de tecnologias pode ajudar a proporcionar maior segurança no transporte escolar. O veículo pode se afastar até 25 metros da rota percorrida, e caso o mesmo ultrapasse este limite, o *pop-up* do veículo será aberto automaticamente, apresentando a bandeira vermelha, que indica que o veículo está fora de rota. Outro fator importante é o monitoramento do veículo, pois, caso tenha um agente executando esta função, ele terá acesso a várias informações em tempo real, como, por exemplo, velocidade e tempo que falta para o veículo chegar ao destino. Também terá a possibilidade de verificar imediatamente se um aluno está embarcado ou não no veículo, e como descrito anteriormente, saberá se o veículo está na rota correta ou não. Tais informações podem ser úteis em caso de emergência. Nos testes demonstrados no capítulo 4 de resultados, o sistema se apresentou bastante funcional, visto que o recebimento dos dados apresentados ao usuário foram rapidamente renderizados na tela. O *Websocket* pode ter ajudado bastante na fluidez dos dados, porém, não é possível determinar como o sistema irá se comportar em ambiente de produção.

Pode-se concluir que o objetivo inicialmente proposto por esse trabalho, que era desenvolver um módulo de software para rastreamento e monitoramento veicular em tempo real utilizando as principais tecnologias que existem na década de 2020, foi atingido de acordo. Além disso, cabe destacar que o uso de metodologias de desenvolvimento de software como *Scrum* e a utilização do *Trello* para auxiliar na gestão das atividades de desenvolvimento, foram fundamentais para a conclusão deste trabalho.

5.1 Trabalhos futuros

Entende-se que este trabalho tem um grande potencial prático, sendo que diversas novas funcionalidades poderão ser incorporadas ao software desenvolvido. Além disso, alguns aspectos cruciais, que não foram possíveis de serem implementados neste projeto, serão descritos como possíveis futuras e melhorias. São eles:

1. Utilizar algum método para criptografar os dados de sessão do usuário, no sistema Web e Mobile.

2. Criar um sistema de históricos, como: rota percorrida pelo veículo; pontos de parada do veículo; pontos de entrada e saída de alunos.

3. Criar um Sistema de notificação.

Referências

AHMED, A. et al. An intelligent and secured tracking system for monitoring school bus. In: IEEE. *2019 International Conference on Computer Communication and Informatics (ICCCI)*. [S.l.], 2019. p. 1–5. Citado 3 vezes nas páginas 21, 22 e 26.

AMARAL, H. Q. Students clustering approaches aiming to minimize the bus stop number: a comparative study in real problems. 2019. Citado na página 14.

ANTONOV, A. *Spring Boot Cookbook*. [S.l.]: Packt Publishing Ltd, 2015. Citado na página 31.

BIERMAN, G.; ABADI, M.; TORGERSEN, M. Understanding typescript. In: SPRINGER. *European Conference on Object-Oriented Programming*. [S.l.], 2014. p. 257–281. Citado na página 28.

BRUNELI, M. V. d. Q. et al. A utilização de uma metodologia de teste no processo de melhoria da qualidade de software. [sn], 2006. Citado na página 42.

BRUNSWIG, F. et al. *Backend service adaptation*. [S.l.]: Google Patents, 2010. US Patent App. 12/246,247. Citado na página 35.

CARVALHO, V. *PostgreSQL: Banco de dados para aplicações web modernas*. [S.l.]: Editora Casa do Código, 2017. Citado na página 35.

CUSTÓDIO, R. S. Sg-serf: Sistema gerenciador de serviços externos e rastreio de funcionários. *Ciência da Computação-Tubarão*, 2020. Citado na página 31.

ESBALTAR, T. dos S. Linguagem java para iniciantes. *Revista Ada Lovelace*, v. 2, p. 102–105, 2018. Citado na página 30.

GHAREEB, M. et al. Smart bus: A tracking system for school buses. In: IEEE. *2017 Sensors Networks Smart and Emerging Technologies (SENSET)*. [S.l.], 2017. p. 1–3. Citado 3 vezes nas páginas 20, 21 e 26.

GODE, C. S.; DESAI, S. A. Iot based school bus tracking system. *Zeichen Journal*, 2020. Citado 2 vezes nas páginas 24 e 26.

GOLDEN, B. L.; RAGHAVAN, S.; WASIL, E. A. *The vehicle routing problem: latest advances and new challenges*. [S.l.]: Springer Science & Business Media, 2008. v. 43. Citado na página 14.

HE, K. Improvements in accessibility of people with disabilities using telepresence robots to participate in social activities. 2020. Citado na página 32.

INDRUSIAK, L. S. Linguagem java. *Grupo JavaRS JUG Rio Grande do Sul*, p. 19, 1996. Citado na página 30.

JISHA, R.; JYOTHINDRANATH, A.; KUMARY, L. S. Iot based school bus tracking and arrival time prediction. In: IEEE. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.], 2017. p. 509–514. Citado na página 17.

KOSCIANSKI, A.; SOARES, M. dos S. *Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. [S.l.]: Novatec Editora, 2007. Citado na página 42.

LEMOS, M. T. Web solution for the management of data related to the school bus routing problem. 2019. Citado 5 vezes nas páginas 14, 16, 28, 29 e 34.

LOPES, G. B.; MARCHI, K. R. Segurança no desenvolvimento web utilizando framework spring security. *XV SINPAR-Semana de Informática e XII Mostra de Trabalhos de iniciação científica de Paranavaí, Paranavaí*, 2013. Citado na página 31.

LUFT, C. C. Teste de software: uma necessidade das empresas. 2013. Citado na página 41.

MATERIAL-UI. *Material-UI: Um framework popular de React UI*. 2020. Disponível em: <https://material-ui.com/pt/>. Citado na página 29.

MESSIAS, E. Ferramenta web para visualização de rotas aplicados ao problema de roteamento de veículos escolares em ambientes reais. 2018. Citado na página 14.

MILANI, A. *PostgreSQL-Guia do Programador*. [S.l.]: Novatec Editora, 2008. Citado na página 33.

MITTAPALLI, J. S.; ARTHUR, M. P. Survey on template engines in java. In: EDP SCIENCES. *ITM Web of Conferences*. [S.l.], 2021. v. 37, p. 01007. Citado na página 32.

NICOLL, S.; WILKINSON, A.; FREDERICK, S. Spring boot maven plugin documentation. *spring.iohttps://docs.spring.io/spring-boot/docs/current/maven-plugin/reference/html*, 20–. Citado na página 31.

OBE, R.; HSU, L. Postgis in action. *GEOInformatics*, CMedia BV, v. 14, n. 8, p. 30, 2011. Citado na página 33.

OpenStreetMap. 2020. Disponível em: <https://www.openstreetmap.org/about>. Citado na página 30.

Overflow, S. *Stack Overflow Trends*. 2020. Disponível em: <https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular>. Citado na página 29.

PEREIRA, C. R. *Aplicações web real-time com Node.js*. [S.l.]: Editora Casa do Código, 2014. Citado 3 vezes nas páginas 32, 33 e 35.

PORTELA, E. D. W.; SANTOS, L. D. C.; PRADO, E. F. do. Lecard: Web aplicativo para comandas virtuais. *Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica*, v. 9, n. 2, 2018. Citado na página 26.

RAJ, J. T.; SANKAR, J. Iot based smart school bus monitoring and notification system. In: IEEE. *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. [S.l.], 2017. p. 89–92. Citado 3 vezes nas páginas 23, 24 e 26.

REACTJS. *React – Uma biblioteca JavaScript para criar interfaces de usuário*. 2020. Disponível em: <https://pt-br.reactjs.org/>. Citado na página 28.

REDDY, K. S. P. *Beginning Spring Boot 2: Applications and Microservices with the Spring Framework*. [S.l.]: Apress, 2017. Citado na página 30.

SALVADORI, I. L. et al. Desenvolvimento de web apis restful semânticas baseadas em json. 2015. Citado 2 vezes nas páginas 27 e 28.

SANTOS, J. V. dos; SOUZA, M. A. S. de; ANDERLE, D. F. Controlando dispositivos em tempo real através do websocket. *Organizadores Vanderlei Freitas Junior Lucyene Lopes da Silva Todesco Nunes Thales do Nascimento da Silva Gerson Luis da Luz*, v. 88960, p. 206, 2015. Citado na página 28.

SILVA, M. F da; MORENO, A. G. Automação em testes ágeis. *Revista de Sistemas e Computação-RSC*, v. 1, n. 2, 2012. Citado na página 42.

SOUSA, E. P. de et al. Arquitetura de aplicações spring mvc: Uma análise baseada no acoplamento lógico. 2017. Citado na página 31.

SOUZA, W. G. de; PINTO, G. S. A utilização do vue. js com uma api rest em spring boot. *Revista Interface Tecnológica*, v. 17, n. 2, p. 155–167, 2020. Citado na página 31.

STEINER, M. T. A. et al. O problema de roteamento no transporte escolar. *Pesquisa Operacional*, SciELO Brasil, v. 20, n. 1, p. 83–99, 2000. Citado na página 14.

SURYOTRISONGKO, H.; JAYANTO, D. P.; TJAHYANTO, A. Design and development of backend application for public complaint systems using microservice spring boot. *Procedia Computer Science*, Elsevier, v. 124, p. 736–743, 2017. Citado na página 31.

XIA, F. et al. Internet of things. *International journal of communication systems*, v. 25, n. 9, p. 1101, 2012. Citado na página 21.