UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Joiner dos Santos Sá

Desenvolvimento de Softwares e Algoritmo Baseado em Redes Neurais Artificiais para Suporte à Gestão da Mobilidade Urbana em Smart Campus com Característica Multimodal

DM: 19/2022

UFPA / ITEC / PPGEE

Campus Universitário do Guamá

Belém – Pará – Brasil

2022

Joiner dos Santos Sá

Desenvolvimento de Softwares e Algoritmo Baseado em Redes Neurais Artificiais para Suporte à Gestão da Mobilidade Urbana em Smart Campus com Característica Multimodal

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Pará, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica na área de Computação Aplicada.

Orientadora: Profa. Dra. Jasmine Priscyla Leite de Araújo

DM: 19/2022

UFPA/ITEC/PPGEE
Campus Universitário do Guamá
Belém – Pará – Brasil
2022

Joiner dos Santos Sá

Desenvolvimento de Softwares e Algoritmo Baseado em Redes Neurais Artificiais para Suporte à Gestão da Mobilidade Urbana em Smart Campus com Característica Multimodal

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Pará, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica na área de Computação Aplicada.

Conceito: _____

Belém, 20 de julho de 2022.

BANCA EXAMINADORA

_____

Profa. Dra. Jasmine Priscyla Leite de Araújo

Orientadora – PPGEE / UFPA

_____

Prof. Dr. Fabricio Jose Brito Barros

Avaliador interno – PPGEE / UFPA

_____

Prof. Dr. Maria Emília de Lima Tostes

Avaliador interno – PPGEE / UFPA

_____

Prof. Dr. Fabrício de Souza Farias

Avaliador Externo – Campus Universitário do Tocantins / UFPA-CAMETÁ

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S111d Sá, Joiner dos Santos.

Desenvolvimento de Softwares e Algoritmo Baseado em Redes Neurais Artificiais para Suporte à Gestão da Mobilidade Urbana em Smart Campus com Característica Multimodal / Joiner dos Santos Sá. — 2022.

128 f. : il. color.

Orientador(a): Profª. Dra. Jasmine Priscyla Leite de Araújo

Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2022.

1. Desenvolvimento de Software. 2. Smart Campus. 3. Mobilidade Urbana. 4. Redes Neurais Artificiais. 5. IoT. I. Título.

CDD 620

Dedico este trabalho ao meu filho Eduardo, como uma forma de demonstrar o quanto luto para levar dias melhores para ele. Que este trabalho sirva de exemplo para que um dia ele possa se tornar um grande profissional/cientista seja qual for sua área.

AGRADECIMENTOS

Primeiramente agradeço ao todo poderoso e soberano Deus, por me conceder vida, saúde, força, capacidade e oportunidade de chegar até aqui.

À minha esposa Lana Sá e ao meu filho Eduardo por terem me dado alegria diante de dias difíceis. Aos meus pais por me apoiarem no meu crescimento acadêmico e às minhas irmãs por serem exemplos pra mim de dedicação aos estudos.

Agradeço à minha orientadora professora Dra. Jasmine Priscyla Leite de Araujo por ter dedicado seu tempo em me ajudar na realização desta conquista, além de ter aceito o desafio de me orientar.

Ao professor Dr. Fabrício de Souza Farias, por sempre ter disposto seu tempo em me ajudar, além de ter me dado oportunidades de crescimento profissional e intelectual.

Ao Laboratório de Programação Extrema (LABEX) pelo apoio técnico durante toda minha trajetória acadêmica. Aos meus colegas de LABEX e mestrado Leonardo Gonçalves e Edinho Nascimento, por terem me ajudado durante o mestrado e na realização deste trabalho.

Aos professores do PPGEE por todo conhecimento científico de alta qualidade proporcionado durante o curso de mestrado.

A Deus toda honra e toda glória eternamente!

RESUMO

Este trabalho apresenta o desenvolvimento de duas soluções de *software* e um algoritmo baseado em redes neurais artificiais para suporte à gestão da mobilidade urbana em um *smart campus*. O primeiro *software*, intitulado Norte Rotas, é uma solução *web* cujo objetivo é dar suporte ao planejamento de rotas de pedestres, provendo informações relevantes sobre condições físicas das rotas de um *smart campus*. Já a segunda solução, é um *software mobile* Android que tem o objetivo de fazer a gestão de modais de transporte presentes em um *smart campus*. Testes em ambiente simulado e real foram realizados e os resultados apontam que as ferramentas propostas são boas soluções para o planejamento e gerenciamento de rotas de modais em um campus universitário inteligente. Além dos *softwares*, é proposto um algoritmo de inteligência computacional para determinação da melhor rota de viagem considerando as opções a pé, de ônibus e de barco em um sistema IoT de um *smart campus*. Dados foram coletados a partir de rotas do ônibus Circular da UFPA, e testes com diferentes parâmetros de uma RNA foram realizados. Os resultados apontam que a solução baseada em RNA é promissora para ser implantada em sistemas de auxílio à mobilidade urbana em um *smart campus.*

Palavras-chave: Desenvolvimento de Software; Smart Campus; Mobilidade Urbana; Redes Neurais Artificiais; IoT.

ABSTRACT

This work presents the development of two software solutions and an algorithm based on artificial neural networks to support the management of urban mobility in a smart campus. The first software, called Norte Rotas, is a web solution whose objective is to support the planning of pedestrian routes, providing relevant information about the physical conditions of the routes of a smart campus. The second solution is an Android mobile software that aims to manage transport modes present in a smart campus. Tests in simulated and real environments were carried out and the results indicate that the proposed tools are good solutions for the planning and management of modal routes in an intelligent university campus. In addition to the software, a computational intelligence algorithm is proposed to determine the best travel route considering the options on foot, by bus and by boat in an IoT system of a smart campus. Data were collected from UFPA's Circular bus routes, and tests with different parameters of an ANN were performed. The results show that the solution based on ANN is promising to be implanted in urban mobility aid systems in a smart campus.

Keywords: Software development; Smart Campus; Urban mobility; Artificial neural networks; IoT.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

4G	Quarta Geração de telefonia móvel
API	*Application Programming Interface*
BRT	*Bus Rapid Transit*
CEAMAZON	Centro de Excelência em Eficiência Energética da Amazônia
CO2	Dióxido de Carbono
CPF	Cadastro de Pessoa Física
CPQD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
CSS	*Cascading Style Sheets*
DER	Diagrama Entidade-Relacionamento
F1	Ponto final 1
F2	Ponto final 2
FESA	Faculdade de Engenharia Sanitária e Ambiental
GB	*Gigabyte*
GPS	*Global Positioning System*
GSM	*Global System for Mobile*
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
I1	Ponto inicial 1
I2	Ponto inicial 2
IC	Inteligência Computacional
ID	Identificador
INPI	Instituto Nacional da Propriedade Industrial

IoT	*Internet of Things*
JPEG	*Joint Photographics Experts Group*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
KNN	*K-Nearest Neighbors*
LABEX	Laboratório de Programação Extrema
LCT	Laboratório de Computação e Telecomunicações
LoRa	*Long Range*
MER	Modelo Entidade-Relacionamento
MQTT	*Message Queuing Telemetry Transport*
NEB	Núcleo de Educação Básica
NUMA	Núcleo de Meio Ambiente
OSM	*OpenStreetMap*
P&D	Projeto de Pesquisa e Desenvolvimento
PCT	Parque de Ciência e Tecnologia do Guamá
PHP	Acrônimo recursivo de *Hypertext Preprocessor*
PNG	*Portable Network Graphics*
PSO	*Particle Swarm Optimization*
RAM	*Random-Access Memory*
REST	*Representational State Transfer*
RF	Requisito Funcional
RFID	*Radio Frequency Identification*
RNA	Rede Neural Artificial
RNF	Requisito Não Funcional
RU	Restaurante Universitário

SGBD	Sistema de Gerenciamento de Banco de Dados
SIMA	Sistema Inteligente Multimodal da Amazônia
SQL	*Structured Query Language*
U1	Usuário 1
U2	Usuário 2
UFPA	Universidade Federal do Pará
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto da Pesquisa

Problemas de mobilidade urbana são realidades em quase todos os tipos de espaços urbanos, seja em grandes, médias ou pequenas cidades ou até mesmo em espaços menores como, por exemplo, em *campi* universitários. Os fatores que podem contribuir com a baixa qualidade no serviço de mobilidade urbana estão relacionados, por exemplo, com as condições precárias dos veículos públicos, trajetos sem infraestruturas, inexistência de logística para um planejamento estratégico de mobilidade urbana, entre outros, sendo esses problemas parte do contexto de várias cidades do Brasil (JÚNIOR; BAPTISTA, 2018).

Além dos diversos problemas que interferem na qualidade de locomoção de pessoas pelos espaços urbanos, vale ressaltar que existem os problemas ambientais causados pela mobilidade urbana baseada em transportes motorizados que utilizam combustíveis fósseis, que de certa forma interferem na qualidade de vida da população.

Diante deste contexto, surgiu o Projeto de Pesquisa e Desenvolvimento (P&D) do Sistema Inteligente Multimodal da Amazônia (SIMA), cujo o objetivo é o desenvolvimento de um sistema inteligente de mobilidade elétrica multimodal para a Amazônia, composto por dois ônibus e um barco, todos elétricos e alimentados por sistemas fotovoltaicos. O projeto SIMA tem uma infraestrutura composta por geração fotovoltaica, armazenamento de energia, eletropostos, medidores de energia, rede Long Range (LoRa); *middleware* de Internet of Things (IoT); aplicações de mobilidade; e sistemas de gestão e monitoramento de uma parte dessa infraestrutura (LOBATO, 2022).

O SIMA foi idealizado e implantado na Cidade Universitária Professor José Silveira Netto da Universidade Federal do Pará (UFPA), em Belém. O projeto colocou uma estrutura de dispositivos interconectados no campus universitário, que possibilita que ele seja considerado um campus inteligente, ou também conhecido como *smart campus*.

Diante do exposto, esta dissertação apresenta uma parte das soluções desenvolvidas para o projeto SIMA. As soluções propostas neste trabalho, são, um *software* voltado para o planejamento de rotas para pedestres; um aplicativo móvel para gestão dos modais elétricos; e

uma solução de Inteligência Computacional (IC) baseada em Rede Neural Artificial (RNA) para escolha da melhor rota de viagem para usuários do projeto SIMA.

1.2 Motivação

A principal motivação para o desenvolvimento das soluções propostas neste trabalho, é a possibilidade de ter um campus conectado, inteligente e com informações relevantes para que qualquer usuário consiga ter mobilidade urbana facilitada dentro do campus universitário da UFPA em Belém.

1.3 Justificativas

Apesar de existir soluções como o aplicativo Circular UFPA (MAIA, 2017) e o UFPA Digital (CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE DA UFPA, 2022) para o gerenciamento do ônibus circular da UFPA, uma das justificativas para o desenvolvimento deste trabalho é apresentar soluções mais abrangentes para o gerenciamento de mais de um tipo de transporte do campus, ou seja, transporte multimodal.

Outra justificativa desta dissertação é sua relevância para o planejamento de rotas para pedestres dentro da cidade universitária, principalmente para a comunidade de pessoas que necessitam de acessibilidade, onde a informação sobre infraestrutura de acessibilidade, condições de trafegabilidade, segurança das rotas, entres outros problemas podem ser gerenciados via um *software* apresentado neste trabalho.

Uma outra justificativa para o desenvolvimento deste trabalho é a importância que ele tem para comunidade no contexto geral de mobilidade urbana. Os *softwares* desenvolvidos, assim como o projeto SIMA de modo geral, contribuem de forma positiva na qualidade de vida da população universitária, isto é, ajudam alunos, servidores e visitantes, uma vez que essas pessoas podem utilizar meios de transporte menos poluentes.

Por fim, outra justificativa é a importância teórica das soluções apresentadas para comunidade científica das áreas de computação aplicada, mobilidade urbana e IoT, onde os *softwares* e o algoritmo de IC apresentado, são projetados como soluções de mobilidade urbana em um campus inteligente e com característica multimodal.

1.4 Objetivos

O objetivo geral desta dissertação é o desenvolvimento de dois *softwares* e um algoritmo de RNA para o apoio à gestão de mobilidade urbana multimodal em um *smart campus*.

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Levantar requisitos para os dois *softwares*;
- Fazer a modelagem lógica e física dos bancos de dados relacional (para ambos os *softwares*) e não relacional (para o aplicativo móvel);
- Desenvolvimento de uma *Application Programming Interface* (API) do tipo *Representational State Transfer* (REST) para ser consumida pelo aplicativo móvel;
- Criação de um *script* simulador para testes do aplicativo móvel proposto;
- Definição de cenários e arquiteturas para realização dos testes dos *softwares*;
- Seleção de uma base de dados para experimentos da RNA proposta; e
- Definição de cenários e criação de um painel *web* para realização dos experimentos da RNA proposta.

1.5 Organização do Trabalho

Esta dissertação é composta por sete Capítulos e as referências bibliográficas. Além deste Capítulo de introdução, o restante deste trabalho está estruturado da seguinte forma:

Capítulo 2 – Trabalhos Relacionados: apresenta trabalhos acadêmicos e *softwares* de mercado relacionados aos apresentados nesta dissertação;

Capítulo 3 – Referencial Teórico: apresenta toda parte teórica e conceitos sobre as tecnologias e ferramentas utilizados nesta pesquisa;

Capítulo 4 – Software Norte Rotas: apresenta o desenvolvimento, testes e resultados de um *software* para o planejamento de rotas para pedestres;

Capítulo 5 – Aplicativo Móvel SIMA Mobilidade Multimodal: apresenta o desenvolvimento, testes e resultados de um *software* móvel para a gestão de mobilidade elétrica multimodal de um *smart campus*;

Capítulo 6 – Rede Neural Artificial para Escolha de Rotas em um *Smart Campus*: apresenta o uso de um algoritmo de IC baseado em RNA para determinação da melhor rota de viagem em um *smart campus*;

Capítulo 7 – Conclusão: apresenta as considerações finais, trabalhos futuros e as produções acadêmicas publicadas durante esta pesquisa.

2 TRABALHOS RELACIONADOS

Este capítulo apresenta trabalhos semelhantes aos temas apresentados nesta dissertação, ele está organizado da seguinte forma, a Seção 2.1 faz uma introdução sobre este capítulo, em seguida, na Seção 2.2 são apresentados trabalhos que propõem ferramentas para o planejamento de rotas de pedestres. Na Seção 2.3 são apresentados *softwares* de apoio à mobilidade urbana. Já a Seção 2.4, apresenta trabalhos que utilizam inteligência computacional para auxiliar a mobilidade urbana. Por fim, este Capítulo termina na Seção 2.5 com uma conclusão.

2.1 Introdução

A mobilidade urbana é bastante discutida quando o assunto é o planejamento de centros urbanos, ela trata o modo como as pessoas transitam pelas cidades. Na literatura é possível encontrar diversos trabalhos que auxiliam no processo de planejamento de rotas e mobilidade urbana.

Este capítulo tem o objetivo de mostrar o que a literatura tem apresentado sobre os principais temas que basearam o autor a concluir esta dissertação. As próximas Seções apresentam trabalhos sobre planejamento de rotas para pedestres; bem como, *softwares* de apoio à mobilidade urbana, encontrados tanto na literatura quanto no mercado de aplicativos; e também, técnicas de IC utilizadas para auxiliar na mobilidade urbana.

2.2 Planejamento de Rotas de Pedestres

No trabalho de Yao *et al.* (2018), foi proposta uma estrutura hiper-heurística multiobjetivo para o planejamento de rotas seguras de caminhada em uma cidade inteligente. Os autores adotaram um mecanismo de aprendizado por reforço para selecionar boas heurísticas para acelerar a velocidade de busca por novas rotas. Experimentos foram conduzidos no mapa de índice de segurança construído a partir dos dados urbanos históricos da cidade de Nova York. Os resultados mostraram que a solução proposta é promissora e mais rápida que outros algoritmos comparados nos experimentos.

Os autores Novack *et al.* (2018) apresentam um sistema que gera rotas de pedestres customizadas inteiramente baseadas em dados do OpenStreetMap[1] (OSM). O sistema permite que os usuários definam até que ponto eles gostariam que a rota tivesse áreas verdes, lugares sociais e ruas mais tranquilas com menos tráfego rodoviário. O sistema foi avaliado 156 usuários, onde as perguntas estavam relacionadas a utilidade, usabilidade, controlabilidade e transparência do sistema. A maioria dos usuários concordou que o sistema é útil e fácil de usar e que lhes dá a sensação de ter o controle sobre a extração de rotas de acordo com suas preferências (áreas verde, sociais e tranquilas).

No estudo de Opach *et al.* (2020) é proposto um planejador de rotas que inclui informações de contexto em tempo real de sensores físicos e observações de cidadãos para apoiar a navegação de pedestres em áreas urbanas expostas a calor extremo e inundações. Os autores destacam que o planejador de rotas pode ser valioso para cidadãos individuais, especialmente aqueles com mobilidade limitada, bem como para profissionais de saúde e autoridades responsáveis pela resposta e gerenciamento de crises.

O trabalho de Opach *et al.* (2021), apresenta uma ferramenta *web* chamada WayFinder para o roteamento de pedestres, que considera água acumulada em estradas, calçadas e trilhas após chuva e derretimento da neve. A ferramenta foi testada por 56 participantes, onde para cada um deles, foi solicitado que ao utilizar a ferramenta na prática, relatar as áreas de inundações das vias e preencher três questionários de *feedback* sobre a ferramenta proposta. Os autores destacaram que embora a maioria dos participantes tenha obtido sucesso no uso da ferramenta, eles preferiram selecionar rotas que passassem por áreas passíveis de inundação e obter informações sobre os riscos do que selecionar uma única sugestão de rota que já evitasse áreas expostas.

Os autores Velázquez *et al.* (2018) propuseram um sistema de navegação vestível para pedestres com deficiência visual que combina um *Global Positioning System* (GPS) para localização externa do usuário e estimulação via tátil-pé para apresentação de informações. Dados de GPS em tempo real fornecidos por um *smartphone* foram usados em um aplicativo de navegação dedicado para determinar as direções de um determinado destino. As direções de navegação foram codificadas como vibrações e transmitidas ao usuário por meio de um solado tátil presente em seu sapato. Experimentos foram realizados e os resultados mostraram que os usuários foram capazes de reconhecer com alta precisão o *feedback* tátil fornecido aos seus pés.

[1] É um projeto de mapeamento colaborativo para criar um mapa livre e editável do mundo.

Os testes preliminares realizados em locais ao ar livre envolveram dois usuários cegos que foram guiados por caminhos pré-determinados de 380 a 420 metros, compartilhando o espaço com outros pedestres e enfrentando obstáculos urbanos típicos. Os sujeitos alcançaram com sucesso os destinos alvo. Os resultados sugerem que o sistema proposto melhora a mobilidade urbana, independência e segurança dos pedestres cegos.

Barczyszyn (2019) propõe um modelo baseado em calçadas com o objetivo de suprir as necessidades de cadeirantes através de um serviço de planejamento de rotas. O modelo foi definido como um grafo, em que os vértices são as esquinas e as arestas são as calçadas e cruzamentos. No modelo proposto, o *feedback* do usuário foi considerado para atualização de informações sobre problemas de acessibilidade e resolução de problemas pelo departamento de planejamento urbano da cidade. Além do modelo, o autor desenvolveu uma API para o fornecimento dos serviços de caminho mínimo em calçadas. Experimentos foram realizados em uma cidade brasileira a fim de validar o modelo e o serviço de planejamento de rotas.

2.3 Softwares de Apoio à Mobilidade Urbana

Na literatura existem diversos trabalhos que apresentam *softwares* para auxiliar a mobilidade urbana. Os autores Castro *et al*. (2018) propuseram um sistema cujo objetivo é apresentar informações de geolocalização e segurança dos ônibus. O sistema proposto era composto por uma aplicação *web* e duas para a plataforma móvel Android. O *software web* era para gerenciar informações de rotas e usuários. Um dos aplicativos móveis servia para o motorista/cobrador do ônibus compartilhar a localização e informação de segurança, por exemplo, no momento em que o veículo estivesse sendo assaltado ou em manutenção. O último aplicativo era voltado para o usuário final, onde a finalidade era visualizar as informações sobre os ônibus no mapa.

Ingle e Bagwan (2018) propuseram um sistema de rastreamento de ônibus em tempo real. A proposta apresentava ao usuário final, através de *smarphone* Android e de uma página *web*, a localização dos veículos de forma precisa e em tempo real. O projeto utilizou GPS, *Radio Frequency Identification* (RFID) e *Global System for Mobile* (GSM) para acessar a localização dos ônibus em tempo real. Testes de simulação foram realizados e resultados mostraram a eficiência da proposta. O sistema economiza tempo e aumenta a eficiência, pois reduz os esforços do usuário em viagens e evita o desperdício de tempo de espera dos ônibus.

Já os autores Deilami *et al.* (2020), apresentaram um aplicativo para *smartphone* para modelar a rota de transporte ativo termicamente mais confortável para um destino planejado usando informações de calor e sombreamento de árvores. Os autores fizeram uma revisão sistemática sobre trabalhos e *softwares* relacionados, e foi identificado que existe uma falta de atenção em relação ao uso de aplicativos de *smartphones* para abordar o conforto térmico urbano para transporte ativo por parte do governo e da iniciativa privada. Pesquisas foram realizadas com opiniões dos participantes de uma cidade na Austrália, sobre tema. Os resultados mostraram que há uma necessidade de um melhor gerenciamento de rotas com sombreamento.

Assim como em pesquisas acadêmicas, no mercado de *softwares* existem diversas soluções que auxiliam a mobilidade urbana. Na loja Google Play, podem ser encontrados diversos aplicativos para plataforma Android, por exemplo, o Moovit tem como principal objetivo auxiliar o planejamento de viagens através do transporte público. A plataforma apresenta itinerários de rotas de ônibus, trens, metrôs, barcos e bicicletas, além de mostrar dados de tarifas, tempo de viagem, alertas em tempo real sobre as condições do trânsito, mapas off-line, entre outros (MOOVIT, 2022).

O Google Maps é um aplicativo disponível para diversas plataformas, inclusive Android, com ele é possível conferir e encontrar locais e estabelecimentos, gerar rotas a pé ou de diversos modais de transportes, medir distância de percurso, estimar tempo de viagem, visualizar informações sobre o trânsito, entre outros. Este *software* é um dos mais completos do mercado, em termos de funcionalidades e dados geográficos (GOOGLE LLC, 2022).

Assim como o Google Maps, o aplicativo Waze auxilia pedestres e motoristas na mobilidade urbana. Ele oferece rotas otimizadas e informações sobre o trânsito em tempo real que auxiliam na tomada de decisão e no bom andamento da mobilidade urbana, como engarrafamento, bloqueios policiais, buracos na via, entre outros (WAZE, 2022).

O Cittamobi é um aplicativo móvel que auxilia na mobilidade urbana mostrando a localização de ônibus e horários de chegada em paradas em tempo real. Ele está mapeado para funcionar em diversas cidades brasileiras, como por exemplo, em São Paulo, Salvador, Porto Alegre, Juiz de Fora, Campinas, Sorocaba, Ribeirão Preto, Recife, Pelotas, entre outras (CITTAMOBI, 2022).

2.4 Inteligência Computacional na Mobilidade Urbana

Na literatura, diversos autores propõem a aplicação da IC para solucionar problemas de mobilidade urbana, por exemplo, os autores Ahmed *et al.* (2019) fizeram um estudo utilizando diversas hiper heurísticas como: *Greedy Selection*, *Simulated Annealing*, *Random Permutation*; para o problema de roteamento de veículos.

Os autores Liang *et al.* (2021) utilizaram o algoritmo de Colônia de Formigas para planejamento de rotas aplicado ao contexto de turismo, enquanto que no trabalho de Cohen e Dalyot (2020), foram utilizadas técnicas de *machine learning* para o problema de rotas de pedestres como: RNA, *Stochastic Gradient Descent*, e *Random Forest.*

Os autores Sá *et al*. (2019) fizeram o uso do algoritmo *K-Nearest Neighbors* (KNN) para ajustar pontos de geolocalização de rotas de ônibus. Neste trabalho, testes reais foram feitos com ônibus na cidade de Belém–PA, no Brasil. Os resultados utilizando o algoritmo de IC, apontaram resultados promissores para o ajuste de pontos de geolocalização em rotas de ônibus compartilhadas por usuários.

No trabalho de Sousa *et al.* (2019), foi proposta uma infraestrutura para o monitoramento em tempo real e predição, utilizando aprendizado de máquina, de rotas e paradas de ônibus em dois *smart campus* localizados na cidade de Quixadá, Ceará, Brasil. Testes foram realizados em rotas de ônibus reais e os resultados se mostraram atrativos e com potencial para pesquisa. Já no trabalho de Zhong *et al*. (2018), foi proposto um método otimizado para o planejamento de rotas de *Bus Rapid Transit* (BRT). Um dos principais objetivos era encontrar a melhor configuração para atingir o maior número de usuários atendidos pela rota de ônibus de BRT, no qual foi utilizado algoritmo de *Particle Swarm Optimization* (PSO).

2.5 Conclusão

Este Capítulo apresentou trabalhos correlatos aos *softwares* e algoritmos propostos para o suporte à gestão de mobilidade urbana desta dissertação.

Embora as soluções relatadas na Seção 2.2 sejam bastante benéficas, no contexto da mobilidade urbana, ainda há necessidade de uma coleta eficaz das possibilidades de rotas e logísticas em determinada área, o que gera o desafio de transformar os dados de geolocalização em informações relevantes para auxiliar os usuários, por exemplo, considerando a adição de

informações complementares como sobre a segurança da via, suas condições de uso, iluminação, equipamentos de acessibilidade, entre outros. O Capítulo 4 trata da proposta relacionada a esses trabalhos sobre planejamento de rotas de pedestres.

Todos os *softwares* apresentados na Seção 2.3, tanto os da literatura quanto os do mercado de aplicativos, tem um grande potencial para o suporte à mobilidade urbana. A proposta de *software* desta dissertação, relacionada aos apresentados na Seção 2.3, é apresentada no Capítulo 5. O *software* proposto nesta dissertação tem funções parecidas aos apresentados neste Capítulo 2, o que o diferencia é seu cenário específico de atuação, que é em um *smart campus* com característica multimodal.

Os trabalhos pesquisados e apresentados na Seção 2.4, fazem relação à proposta apresentada por esta dissertação no Capítulo 6, a qual vem para incrementar com a ciência sobre a utilização de IC para escolha de rotas multimodais em um ambiente de *smart campus*, possibilitando uma melhor qualidade na mobilidade urbana.

Vale ressaltar que os trabalhos detalhados neste Capítulo focam no contexto da mobilidade urbana de cidades. No entanto, esta dissertação foca em um ambiente menor em termos de complexidade e mais controlável, que é um *smart campus*.

3 REFERENCIAL TEÓRICO

Neste Capítulo são apresentados conceitos teóricos encontrados na literatura sobre os métodos e tecnologias utilizadas neste trabalho. Na Seção 3.1, o autor faz uma breve introdução sobre os assuntos e seus respectivos autores da literatura utilizada. Na Seção 3.2 são apresentados os conceitos que envolvem requisitos de *software*. A Seção 3.3 aborda a modelagem de projetos de banco de dados, enquanto que na Seção 3.4 são abordados conceitos que envolvem o desenvolvimento de APIs REST. Já na Seção 3.5 é apresentada uma ferramenta de armazenamento e gerenciamento de dispositivos em redes IoT. A Seção 3.6 apresenta conceitos sobre Redes Neurais Artificiais. Por fim, este capítulo termina com uma conclusão apresentada na Seção 3.7.

3.1 Introdução

No processo de desenvolvimento de *softwares*, existem diversas etapas até se chegar em um produto final, desde a conversa com o cliente para elicitar requisitos, até o processo de codificação e teste de um aplicativo utilizável. Todas as etapas descritas neste trabalho foram elaboradas com base na literatura, visando alcançar os melhores resultados no processo de desenvolvimento dos *softwares* propostos.

Neste sentido, as Subseções a seguir, apresentam conceitos baseados em Pressman (2011) e Sommerville (2018) para a etapa de levantamento de requisitos; Heuser (2009), Elmasri e Navathe (2011) e Puja *et al*. (2019) para a etapa do projeto de banco de dados; RedHat (2020), IBM Cloud Education (2021) e PHP Group (2022) para o desenvolvimento de API REST; CPQD (2017) e CPQD (2020) para o armazenamento e gerenciamento de dispositivos IoT; e por fim, Mishra e Srivastava (2014), Melo Junior *et al.* (2016), Werbos (1974), Parker (1982), Rumelhart *et al*. (1986), Ribeiro *et al*. (2008), Yijun *et al*. (2010), Nascimento *et al*. (2010) e Haykin (2002) para o entendimento de Redes Neurais Artificiais.

3.2 Requisitos de Software

Para o sucesso do planejamento e criação de um programa computacional, é importante entender as necessidades do cliente antes de começar a projetar e construir um

software. A engenharia de requisitos é um dos processos iniciais que antecedem a construção de um *software*.

Segundo Pressman (2011), a engenharia de requisitos é um conjunto de atividades especiais que permitem, a partir da comunicação entre os atores do sistema, identificar os requisitos visando obter a melhor compreensão e classificação de todas as necessidades do projeto. Deste modo, a engenharia de requisitos oferece um processo de modelagem de *software* com uma visão futura do projeto, isto é, tornando possível analisar o trabalho que será desenvolvido, prevendo as necessidades que o projeto demanda, assim como, as funcionalidades que deverão ser implementadas para o sucesso do projeto.

Segundo Sommerville (2018), os requisitos precisam ser delineados em distintos graus de descrição, pois os mesmos transmitem informações sobre o sistema para diferentes tipos de indivíduos que estão exercendo o papel de leitores. Podemos classificar os requisitos em dois níveis, isto é, um determinado nível de requisitos do usuário que apresenta uma condição de informação com padrões elevados, e outro que nível de requisitos do sistema que contém uma descrição mais meticulosa do serviço que o sistema deve realizar. Para melhor compreensão, o autor definiu os níveis como: Os requisitos estabelecidos pelo usuário são afirmações das características que o sistema deve possuir, são apresentados em uma linguagem natural e diagramas, no qual, são especificados os serviços que o sistema deve prover aos usuários do sistema, assim com as ressalvas sob as quais ele está restrito; E os requisitos instituídos em um sistema são um conjunto de passos que descrevem as funções, serviços e delimitações operacionais do produto de *software*. Portanto, o documento de requisitos, precisa especificar o que deve ser desenvolvido. Deste modo, os requisitos são a causa do sucesso de um *software* devido tornarem viável a realização de estimativas de tempo de entrega, execução e testes, assim como, por permitir maior controle na manutenção do *software* depois de concluído.

Os requisitos de sistema de *software* podem ser classificados como funcionais e não funcionais. De forma geral, os requisitos funcionais descrevem as funcionalidades e os serviços que o sistema desempenhará, isto é, delineando o comportamento do programa a partir dos interesses dos usuários. Seguindo esta vertente, Sommerville (2018) afirma que os requisitos funcionais descrevem as funções que o sistema realizará e para cumprir essas funções deve-se considerar os fatores relacionados ao sistema e a sua criação, dentre eles, o tipo de *software* e especificações do usuário.

Para que o sistema realize as funcionalidades definidas, os requisitos funcionais precisam ser delineados com precisão para evitar mudanças, principalmente quando o projeto está em desenvolvimento inicial ou em um estágio avançado, evitando assim, custos adicionais ou atrasos na entrega. O Quadro 1 apresenta três exemplos de requisitos funcionais de um sistema de informação de pacientes para cuidados com a saúde mental.

Quadro 1 – Requisitos funcionais de um sistema de pacientes para cuidados com a saúde mental.

Requisito funcional
1. Um usuário deve poder fazer uma busca na lista de consultas de todas as clínicas.
2. O sistema deve gerar, a cada dia e para cada clínica, uma lista de pacientes que devam comparecer às consultas naquele dia.
3. Cada membro da equipe que utiliza o sistema deve ser identificado exclusivamente por seu número de funcionário de oito dígitos.

Fonte: Sommerville (2018).

Já os requisitos não funcionais podem ser definidos como aqueles que não apresentam uma relação direta com funcionalidades do sistema. Para Sommerville (2018) os requisitos não funcionais não estão ligados diretamente às funções específicas dos sistemas e as suas características individuais, eles estão ligados a fatores resultantes dos sistemas, alguns desses requisitos podem restringir o desenvolvimento do sistema causando em alguns casos sua inviabilidade ou prejudicando o desenvolvimento de outras funcionalidades dentro do projeto. De modo geral, os requisitos não funcionais estão relacionados às restrições que o sistema deve ter. O Quadro 2 apresenta exemplos de requisitos não funcionais de um sistema de informação de pacientes para cuidados com a saúde mental, apresentados por Sommerville (2018).

Quadro 2 - Requisitos não funcionais de um sistema de pacientes para cuidados com a saúde mental.

Requisito não funcional
1. O sistema deve ficar disponível para todas as clínicas durante o expediente normal (segunda – sexta, 8h30 – 17h30).
2. O tempo que o sistema pode permanecer fora do ar no expediente normal não deve ultrapassar 5 segundos em qualquer dia.
3. Os usuários do sistema devem se identificar usando o cartão de identificação de autoridade de saúde.
4. O sistema deve implementar providências para a privacidade do paciente.

Fonte: adaptado de Sommerville (2018).

3.3 Projeto de Banco de Dados

Esta Subseção trata de alguns conceitos fundamentais para criação de projetos de bancos de dados de *softwares*. Um projeto de banco de dados bem planejado é de fundamental importância para o sucesso no processo de sua implantação em uma empresa.

Segundo Heuser (2009), um projeto de banco de dados é composto por duas fases: a modelagem conceitual e a modelagem lógica. Na modelagem conceitual é criado um modelo conceitual que apresenta modelos de negócios e suas associações, esta etapa dá uma visão geral sobre o negócio, e possibilita o entendimento entre usuários e a equipe de desenvolvimento. O principal objetivo desta etapa é apresentar uma descrição abstrata dos dados que deverão ser armazenados no banco de dados. O Modelo Entidade-Relacionamento (MER) é um meio pelo qual é definida a etapa da modelagem conceitual. A Subseção 3.4.1 apresenta mais detalhes sobre o MER.

De acordo com Heuser (2009), objetivo da fase de modelagem lógica é transformar o modelo conceitual em um modelo lógico que especifica como o banco de dados deverá ser implementado em um Sistema de Gerenciamento de Banco de Dados (SGBD). Nesta fase, são apresentadas as estruturas de dados que devem ser utilizadas para implementação do banco de dados.

Para Elmasri e Navathe (2011), existe uma terceira fase que compõe o projeto de banco de dados, que é a de modelagem física, cujo objetivo é especificar as estruturas de armazenamento internas, índices, parâmetros físicos do projeto para os arquivos do banco de dados, entre outros. Nesta etapa, a linguagem *Structured Query Language* (SQL) é utilizada para definição, manipulação e controle do banco de dados.

3.3.1 Modelo Entidade-Relacionamento

O MER descreve, de forma abstrata, como os dados de uma base de dados deverão ser organizados em um domínio de negócios. Ele é elaborado na fase de modelagem conceitual do banco de dados, após o levantamento e análise de requisitos do *software*, utilizando conceitos de entidade, atributos e relacionamento.

Para Elmasri e Navathe (2011), uma entidade no MER é a representação de algum objeto do mundo real, que pode ser de existência física (por exemplo, um computador, um ônibus ou um funcionário) ou existência conceitual (por exemplo, o cargo de um funcionário, marca de automóvel ou uma sala virtual de reuniões). Já os atributos são as propriedades que

descrevem as entidades, por exemplo, uma entidade ônibus pode ser descrita pelos atributos marca, modelo, cor, tipo de combustível, tamanho e capacidade de passageiros.

Além dos atributos comuns, que descrevem características das entidades, existem também os atributos-chave, onde seu objetivo é identificar cada entidade de maneira exclusiva, por exemplo, suponha uma entidade de nome “pessoa”, seu atributo-chave pode ser o número do Cadastro de Pessoa Física (CPF) (ELMASRI; NAVATHE, 2011).

De acordo com Elmasri e Navathe (2011), no MER, quando um atributo de um tipo de entidade faz referência a outro tipo de entidade, existirá um determinado relacionamento entre as entidades. Por exemplo, o atributo departamento de um Funcionário faz referência à entidade Departamento no qual o funcionário trabalha.

No MER, podem existir três tipos de relacionamentos: “um para um”, “um para muitos” e “muitos para muitos”. O primeiro é utilizado quando uma entidade faz referência a apenas uma unidade da outra entidade a qual está relacionada, e vice-versa. Já o segundo tipo de relacionamento é utilizado quando uma primeira entidade pode referenciar várias unidades de uma segunda entidade, em contrapartida, uma unidade da segunda entidade pode referenciar apenas uma unidade da primeira entidade. Por fim, o terceiro tipo de relacionamento, “muitos para muitos”, é utilizado quando, em um relacionamento entre duas entidades, a primeira pode referenciar muitas unidades da segunda, bem como uma unidade da segunda entidade pode referenciar várias unidades da primeira.

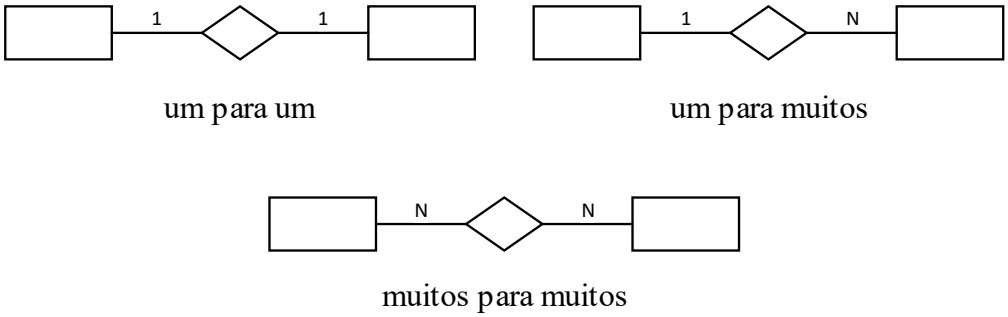
Com o objetivo de ilustrar o MER, Peter Chen propôs o Diagrama Entidade-Relacionamento (DER), que é a principal técnica pela qual a modelagem conceitual de um banco de dados pode ser ilustrada de forma gráfica (ELMASRI; NAVATHE, 2011). No DER, as entidades, atributos e relacionamentos, são representados respectivamente por retângulos, elipses e losangos, conforme Figura 1.

Figura 1 – Representação de entidade, atributo e relacionamento do DER.

Fonte: elaborada pelo autor.

A Figura 2 ilustra a forma de representação das cardinalidades que existem em um relacionamento, que podem ser 1:1 (um para um), 1:N (um para muitos) e N:N (muitos para muitos).

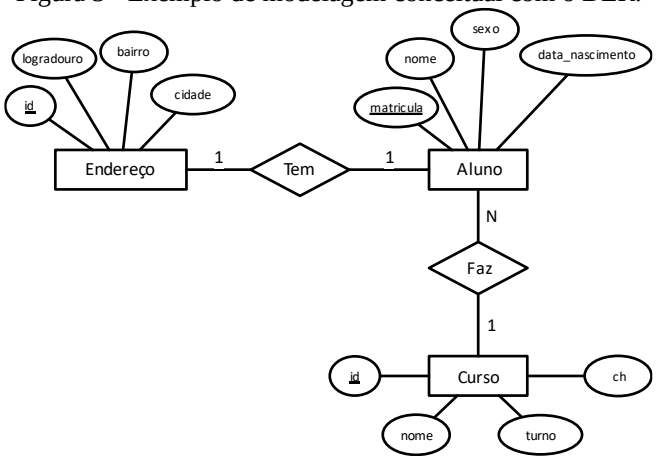
Figura 2 - Representação de cardinalidade no DER.



Fonte: elaborada pelo autor.

A Figura 3 ilustra um exemplo de modelagem conceitual de um banco de dados utilizando o DER, onde a entidade Aluno pode ter um Endereço e fazer um único curso, enquanto que um curso pode ser feito por vários alunos.

Figura 3 - Exemplo de modelagem conceitual com o DER.



Fonte: elaborada pelo autor.

3.3.2 Modelagem Lógica e Física

Embora a modelagem conceitual, através do MER e do DER, seja importante para o entendimento e concepção de uma estrutura inicial do banco de dados, em muitos casos, os projetos reais começam pelo modelo lógico, principalmente com o auxílio de ferramentas e diagramas que apresentam uma estrutura mais completa que o DER.

Para bancos de dados relacional, a modelagem lógica consiste em apresentar uma descrição do banco de dados na forma de tabelas, atributos, chaves primárias, chaves estrangeiras, relacionamentos e cardinalidades (HEUSER, 2009).

Supondo duas entidades, Curso e Aluno, onde a cardinalidade pode ser 1:N, no sentido de que um Curso pode ser feito por vários Alunos, a modelagem lógica na forma textual, baseada em Heuser (2009), pode ser a seguinte:

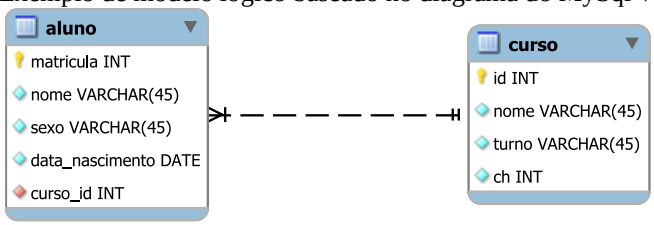
```
Curso(id, nome, turno, ch)
Aluno(matricula, nome, sexo, data_nascimento, id_curso)
        id_curso referencia Curso
```

Outra forma de ilustrar a modelagem lógica é através de diagramas que contém tabelas, atributos, chaves primárias e estrangeiras, e relacionamentos. A Oracle[2], através do MySql Workbench[3], apresenta um diagrama muito utilizado no meio acadêmico e na indústria de desenvolvimento de *softwares* para a modelagem de bancos de dados relacional, este diagrama pode ser considerado como um modelo lógico de banco de dados.

No diagrama do MySql Workbench, as tabelas (forma de retângulos com cantos arredondados) contêm o nome da entidade, chaves primária e estrangeira, e atributos. Neste, são apresentados os tipos e tamanhos de dados dos atributos, diferentemente do modelo lógico textual apresentado por Heuser (2009). A notação de relacionamento utilizada neste diagrama é a *Crow's foot*. Esta notação, também conhecido no português como "pé de galinha", tem como característica um símbolo parecido com um pé de galinha no lado da cardinalidade "muitos" (PUJA *et al*., 2019).

A Figura 4 ilustra o modelo lógico baseado no diagrama do MySql Workbench, para o mesmo exemplo apresentado no modelo lógico textual, com as entidades Curso e Aluno.

Figura 4 - Exemplo de modelo lógico baseado no diagrama do MySql Workbench.



Fonte: elaborada pelo autor.

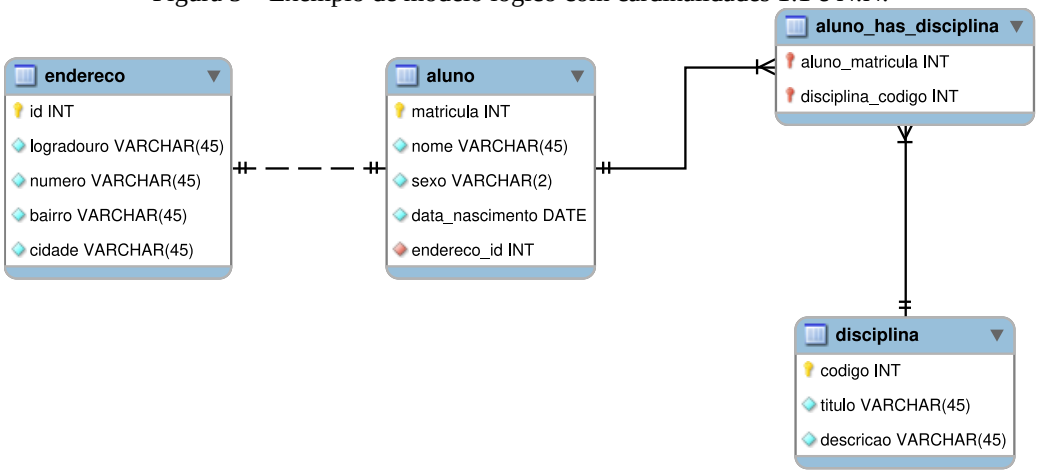
Supondo a modelagem de um banco de dados com três tabelas, onde a entidade **aluno** tem a cardinalidade 1:1 com a entidade **endereco**, e a entidade **disciplina** tem cardinalidade N:N entre a entidade **aluno**, a Figura 5 ilustra o modelo lógico deste banco. A relação 1:1 é caracterizada por uma linha tracejada com duas linhas paralelas nas extremidades que ligam as tabelas. Já para representar a cardinalidade N:N entre as tabelas **disciplina** e **aluno**, é gerada uma terceira tabela (**aluno_has_disciplina**) contendo as duas chaves primárias de cada tabela

[2] Empresa multinacional de tecnologia e informática norte-americana, especializada no desenvolvimento e comercialização de *hardware*, *softwares* e de banco de dados.

[3] Ferramenta visual unificada para arquitetos de banco de dados, desenvolvedores e DBAs. Ela fornece modelagem de dados, desenvolvimento de SQL e ferramentas de administração abrangentes para configuração de servidor, administração de usuários, backup e entre outros.

do relacionamento, formando assim, uma chave primária composta nesta nova tabela intermediária. As linhas que ligam as duas tabelas à tabela intermediária são contínuas e com as extremidades no formato *crow's foot*, do lado da nova tabela, conforme Figura 5.

Figura 5 – Exemplo de modelo lógico com cardinalidades 1:1 e N:N.



Fonte: elaborada pelo autor.

Para a modelagem física de um banco de dados, é utilizada alguma linguagem específica para fazer a manipulação dos dados de um SGBD específico. Este trabalho apresenta a utilização da linguagem SQL para o SGBD MySQL como ferramenta de modelagem física de banco de dados.

Supondo que o modelo lógico apresentado na Figura 5 seja de uma base de dados chamada **escola**, seu modelo físico, para o SGBD MySQL, será o seguinte:

```
CREATE SCHEMA IF NOT EXISTS `escola` DEFAULT CHARACTER SET utf8;
USE `escola`;

CREATE TABLE IF NOT EXISTS `endereco` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `logradouro` VARCHAR(45) NOT NULL,
  `numero` VARCHAR(45) NOT NULL,
  `bairro` VARCHAR(45) NOT NULL,
  `cidade` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `aluno` (
  `matricula` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `sexo` VARCHAR(2) NOT NULL,
  `data_nascimento` DATE NOT NULL,
  `endereco_id` INT NOT NULL,
  PRIMARY KEY (`matricula`),
  INDEX `fk_aluno_endereco_idx` (`endereco_id` ASC) VISIBLE,
  CONSTRAINT `fk_aluno_endereco`
    FOREIGN KEY (`endereco_id`)
    REFERENCES `endereco` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `disciplina` (
  `codigo` INT NOT NULL AUTO_INCREMENT,
  `titulo` VARCHAR(45) NOT NULL,
  `descricao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`codigo`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `aluno_has_disciplina` (
  `aluno_matricula` INT NOT NULL,
  `disciplina_codigo` INT NOT NULL,
  PRIMARY KEY (`aluno_matricula`, `disciplina_codigo`),
  INDEX `fk_aluno_has_disciplina_disciplina_idx` (`disciplina_codigo` ASC) VISIBLE,
  INDEX `fk_aluno_has_disciplina_aluno_idx` (`aluno_matricula` ASC) VISIBLE,
  CONSTRAINT `fk_aluno_has_disciplina_aluno`
    FOREIGN KEY (`aluno_matricula`)
    REFERENCES `aluno` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_aluno_has_disciplina_disciplina`
    FOREIGN KEY (`disciplina_codigo`)
    REFERENCES `disciplina` (`codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

3.4 Desenvolvimento de API REST

Em desenvolvimento de *softwares*, uma API pode ser entendida como um contrato entre aplicações que requerem determinadas informações e a parte que prover as informações. API é a parte que prover informações, ou seja, que as entrega a outras aplicações, e para isso, ela utiliza um conjunto de padrões e protocolos para construção e integração de *softwares* de aplicação (REDHAT, 2020).

Já o REST, consiste em um conjunto de princípios e regras que permitem o desenvolvimento de APIs bem definidas para aplicações de arquiteturas *web*.

APIs REST utilizam o protocolo de comunicação HTTP para transferência de dados entre as aplicações cliente e servidor. Os principais tipos de verbos HTTP utilizados em uma API REST são: GET para leitura de dados; POST para criação de dados no servidor; DELETE para exclusão de dados; e PUT para atualização de registros.

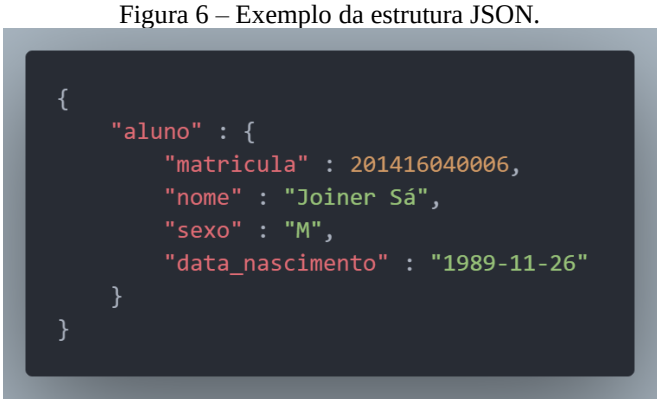
Segundo a IBM Cloud Education (2021), a arquitetura REST é definida por algumas restrições, tais como:

- **Interface uniforme:** garantir que qualquer solicitação para o mesmo recurso deve ter a mesma aparência, utilizando Identificador de Recurso Uniforme (do inglês, *Uniform Resource Identifier* - URI), independente da origem da solicitação;

- **Desacoplamento cliente-servidor:** a aplicação do servidor deve ser independente das aplicações do cliente. As aplicações do cliente devem saber apenas a URI de algum recurso solicitado;
- **Sem estado:** cada requisição deve ser feita de forma independente e precisa incluir todas as informações necessárias para processá-la;
- **Utilização de cache:** em alguns casos, os recursos podem ser armazenados em cache, com o objetivo de evitar chamadas recorrentes ao servidor e consequentemente melhorar o desempenho no lado do cliente e aumentar a escalabilidade do lado do servidor.

Quanto à estrutura dos dados em APIs REST, a aplicação do lado do servidor pode entregar dados formatados em diversos formatos, no entanto, o formato mais utilizado é o *JavaScript Object Notation* (JSON), cuja sua característica é baseada em chave-valor. Esse formato de dados é de simples entendimento por humanos e é suportado por diversas linguagens de programação. A Figura 6 ilustra a estrutura de um JSON contendo um exemplo de dados de um aluno.

Figura 6 – Exemplo da estrutura JSON.



Fonte: elaborada pelo autor.

Para construção de APIs no contexto de desenvolvimento *web*, utiliza-se linguagens de programação chamadas *server-side* ou *back-end*, as quais são entendidas pelo servidor, ou seja, suas aplicações são executadas no servidor. Javascript, Java, Python, Ruby e PHP, são algumas das linguagens de programação *server-side* mais utilizadas para o desenvolvimento de APIs.

Neste trabalho, o autor destaca o PHP (acrônimo recursivo de *Hypertext Preprocessor*), que é uma linguagem de *script* de uso geral, mas que é especialmente adequada para o desenvolvimento *web*. Ela é considerada uma linguagem de fácil entendimento e pode

ser utilizada junto da escrita de códigos em *HyperText Markup Language* (HTML) (PHP GROUP, 2022).

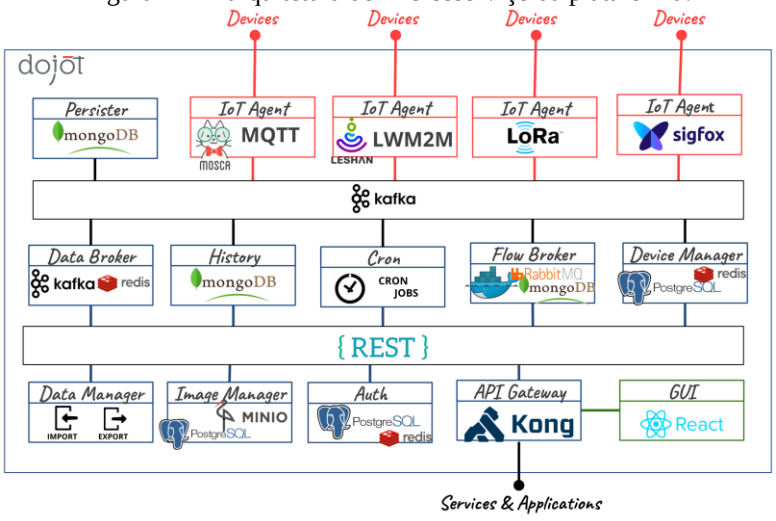
3.5 Plataforma Dojot

Desenvolvida no Brasil pelo Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPQD) em 2017, a Dojot é uma plataforma de código aberto que tem o objetivo de facilitar o desenvolvimento de soluções tecnológicas para cidades inteligentes, seguindo preceitos da IoT, e tendo como enfoque a mobilidade urbana, saúde e segurança pública (CPQD, 2017).

A Dojot facilita o acesso de diversas aplicações aos recursos de sua plataforma através de APIs abertas; é capaz de armazenar grandes volumes de dados em diferentes formatos; facilita a conexão com diferentes tipos de dispositivos; proporciona a construção de regras e fluxo de dados de forma visual, possibilitando de forma rápida a prototipação e validação de cenários IoT; e possibilita o processamento de eventos em tempo real (CPQD, 2020).

Em uma rede IoT, a Dojot exerce o papel de *middleware* IoT, o qual é responsável por receber e persistir dados de dispositivos e os fornecer para as demais aplicações. A arquitetura da Dojot é baseada em microsserviços, ou seja, ela possui pequenos serviços que são independentes e podem se comunicar através de APIs. A Figura 7 ilustra a arquitetura da plataforma.

Figura 7 – A arquitetura de microsserviço da plataforma.



Fonte: CPQD (2020).

Uma breve e resumida explicação de funcionamento da arquitetura da Dojot, pode ser: os dados dos dispositivos IoT são recebidos através dos *IoT Agents*; são persistidos através do componente *Persister*; e são fornecidos pelo microsserviço *History*. A consulta de dados disponíveis pelo *History* se dá através da *API Gateway* (utilizada por alguma aplicação externa à Dojot) ou diretamente para o usuário pelo sistema *web* da Dojot, utilizando o microsserviço *Graphical User Interface* (*GUI*).

Dentre os componentes ilustrados na Figura 7, os mais importantes para esta dissertação são: *DeviceManager*, *IoT Agent*, *Persister*, *History*, *API Gateway* e *GUI*. O Quadro 3 apresenta uma visão geral sobre cada função destes componentes. As funções dos demais componentes podem ser encontradas na documentação da Dojot.

Quadro 3 – Visão geral sobre alguns componentes da Dojot.

Componente	**Função**
IoT Agent	É um serviço de adaptação entre dispositivos físicos e componentes principais da *Dojot*. A plataforma *Dojot* pode ter vários *IoT Agent*, cada um deles especializado em um protocolo específico, como por exemplo, *Message Queuing Telemetry Transport* (MQTT), LoRa, HTTP, entre outros.
Persister	Funciona como um condutor de dados e eventos que devem ser persistidos em um banco de dados. Os dados são convertidos em uma estrutura de armazenamento que é enviada para o banco de dados correspondente, que neste caso é o MongoDB.
History	É um microsserviço que permite a consulta, através de API REST, dos dados dos devices persistidos no banco de dados MongoDB.
DeviceManager	É uma unidade responsável por manter as estruturas de dados de dispositivos e modelos, utilizando o PostgreSQL e o Redis.
API Gateway	É utilizado como um ponto de fronteira (*entry point*) entre as aplicações e serviços externos e os serviços internos da Dojot. Este componente utiliza a tecnologia *Kong*.
GUI	*Graphical User Interface* ou em português Interface Gráfica do Usuário. Na Dojot, este componente é um sistema *web* voltado para interação direta com usuário através de um navegador *web*. Este foi construído em *React*, e provê interfaces responsivas para gerenciamento da plataforma, incluindo funcionalidades como: gerenciamento de perfis de usuários; gerenciamento de usuários; gerenciamento de modelos de dispositivos; gerenciamento de dispositivos; gerenciamento de fluxos de processamento; e notificações.

Fonte: produzido pelo autor baseado em CPQD (2020).

Os componentes que conectam os dispositivos reais à Dojot, os *IoT Agents*, estão presentes na plataforma e suportam diferentes protocolos de comunicação, por exemplo, para conectar dispositivos que fazem parte de uma rede LoRa, é disponibilizado um *IoT Agent* específico que utiliza o protocolo LoRa/ATC. Já para dispositivos que precisem conectar-se através do protocolo MQTT, a Dojot disponibiliza um *IoT Agent* que faz o uso de dois *brockers* MQTT chamados *Mosca* e *VerneMQ*.

Como já especificado no Quadro 3, a Dojot disponibiliza interface gráfica (componente *GUI*) para criação de dispositivos virtuais e modelos (*templates*). Na Dojot, um dispositivo virtual é uma representação digital de um *gateway* ou dispositivo real com um ou mais sensores ou de um virtual com sensores/atributos inferidos de outros dispositivos (CPQD, 2020). Já um modelo (ou *template*) podem representar não apenas "modelos de dispositivos", mas também podem abstrair uma "classe de dispositivos" (CPQD, 2020). Um modelo pode ser considerado um molde para criação de vários dispositivos. Por exemplo, supondo que fosse necessário monitorar todos os sensores de umidade de um campus inteligente (*smart campus*), seria preciso criar um modelo de dispositivo como um atributo numérico para representar a umidade. A partir disto, poderiam ser criados vários dispositivos virtuais baseados no modelo criado, contendo um atributo para armazenar a umidade de cada dispositivo físico em campo.

A Figura 8 ilustra um *screenshot* da GUI da Dojot, ela mostra a lista de dispositivos cadastrados do usuário logado. Na GUI, o usuário pode acessar, através do menu lateral de opções, os dispositivos, o gerenciamento de modelos, fluxos, notificações, usuários e perfis de usuários.

Figura 8 - *Screenshot* da interface gráfica da Dojot.

Fonte: elaborada pelo autor.

Além da interface *web* disponibilizada para o usuário, a Dojot possibilita o acesso aos seus recursos através de uma API. Todas as requisições feitas à API, devem conter um *token* de acesso que seja válido (CPDQ, 2020). Supondo que a Dojot esteja em *localhost*, o nome do

usuário seja “Eduardo” e a senha “r3n10J”, a requisição para gerar um novo *token* teria as seguintes características:

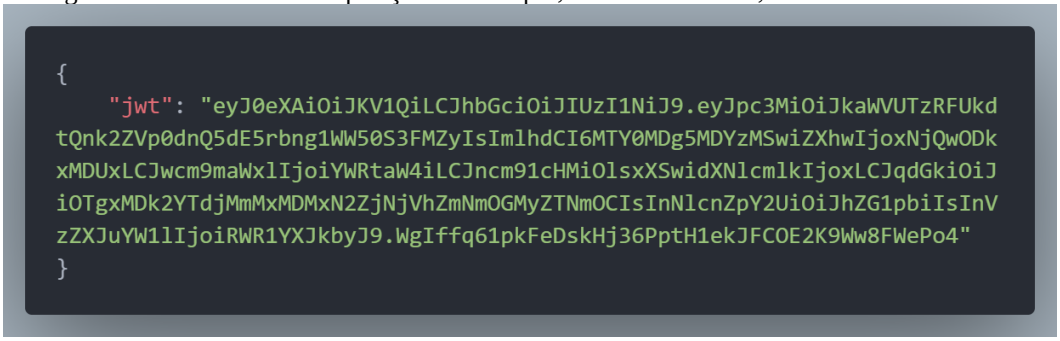
- Método HTTP: POST;
- Sem autenticação;
- Rota de API (URI): http://localhost:8000/auth;
- Parâmetros: JSON no *body* da requisição, contendo usuário e senha, conforme ilustra a Figura 9.

Figura 9 - JSON do *body* da requisição de exemplo.

Fonte: elaborada pelo autor.

A Figura 10 ilustra um possível resultado no formato JSON, da requisição anterior.

Figura 10 – Resultado da requisição de exemplo, no formato JSON, com o *token* de acesso.



Fonte: elaborada pelo autor.

Dentre os diversos recursos disponíveis pela API da Dojot, este trabalho destaca a requisição para obtenção de dados de dispositivos salvos no componente *History*. Supondo que a Dojot esteja instalada em *localhost* e o usuário esteja autenticado com o *token*, a requisição deve ter as seguintes características:

- Método HTTP: GET;
- Autenticação: Bearer Token;
- URI: http://localhost:8000/history/device/***device*_id**/history?lastN=1&attr=***atributo***.

onde, ***device*_id** é o identificador do dispositivo virtual a ser consultado, lastN são os *n* últimos registros gravados em um determinado dispositivo virtual (neste exemplo, o número de registros a ser consultado é 1), e ***atributo*** é o atributo a ser consultado.

Supondo um exemplo onde é preciso consultar os dois últimos registros de temperatura de um dispositivo cujo o ***device*_id** é "18S14", a URI da requisição seria conforme a Figura 11.

Figura 11 – URI da requisição de exemplo ao *history* da Dojot.

Fonte: elaborada pelo autor.

3.6 Redes Neurais Artificiais

As RNAs são sistemas de IC que são inspirados no funcionamento do cérebro e adquirem conhecimento através da experiência (MISHRA; SRIVASTAVA, 2014). De forma geral, uma RNA é dada por um vetor de entrada X com tamanho n, onde n é o número máximo de entradas. A partir da entrada, a RNA retorna um vetor de saída com tamanho m, onde m é o número de possíveis saídas.

Uma RNA pode ser composta de diversas camadas, por exemplo, a camada de entrada, a de saída, e as camadas intermediárias, também chamadas de camadas escondidas. Todas as camadas da RNA possuem nós que se conectam com as camadas vizinhas, e cada conexão possui um peso sináptico associado (MELO JUNIOR *et al.*, 2016).

O aprendizado de uma RNA começa com um processo iterativo de ajustes dos pesos sinápticos, também chamado de processo de treinamento. O objetivo do ajuste dos pesos é minimizar uma função erro a partir da repetição iterativa até que uma condição de parada seja satisfeita. Por fim, a RNA consegue aprender quando se atinge uma solução generalizada para uma classe de problemas (MELO JUNIOR *et al.*, 2016).

Na literatura, existem diversas técnicas utilizadas para o processo de treinamento de RNA, dentre elas, este trabalho destaca o algoritmo *backpropagation*, que tem um papel fundamental na correção de erros e ajustes de pesos sinápticos.

O algoritmo *backpropagation* foi desenvolvido de forma independente por diversos autores, dentre eles, Werbos em 1974 (WERBOS, 1974), em 1982 por Parker (PARKER,

1982), e em 1986 por Rumelhart, Hinton e Williams (RUMELHART *et al*., 1986). Desde sua criação o algoritmo *backpropagation* tem sido largamente utilizado como um algoritmo de aprendizado para RNA com topologia de múltiplas camadas (RIBEIRO *et al*., 2008; YIJUN *et al*., 2010; NASCIMENTO *et al*., 2010). No entanto, é comumente utilizado com três camadas (entrada, escondida e de saída).

O *backpropagation* é baseado no método do gradiente descendente para correção do erro, por meio da propagação das entradas para cálculo das saídas (respostas do algoritmo) e retropropagação para ajuste dos pesos sinápticos (processo de aprendizagem) (HAYKIN, 2002). Desta forma, é utilizado um aprendizado através de exemplos de entrada e saída, ou seja, caso a saída encontrada seja diferente da saída desejada o ajuste dos pesos sinápticos é retropropagado através da minimização de uma função custo obtida com o uso do gradiente descendente. O passo de propagação (*forward*) inicia pelo envio de sinais de entrada, representados pelas respostas do formulário.

Esse processo se repete até a camada de saída, depois o cálculo de um valor de saída é realizado. A retropropagação (*backpropagation*) calcula o erro e compara o valor calculado na saída com a saída desejada apresentada à rede. Novos conjuntos de pesos são iterativamente calculados, através da modificação dos pesos existentes, baseado nos valores de erros até um erro mínimo global ser alcançado. Para verificação e comparação do erro global obtido após cada etapa de treinamento, a entropia cruzada é normalmente usada como critério de medição.

3.7 Conclusão

Este Capítulo apresentou conceitos, baseados na literatura, sobre os temas abordados nesta dissertação. Ele serve como base de entendimento sobre os assuntos e tecnologias que o autor utilizou para elaboração deste trabalho.

Neste Capítulo, foram abordadas as fases do processo de desenvolvimento de *softwares*, tais como, levantamento de requisitos, modelagem de banco de dados relacional, desenvolvimento de APIs REST, bem como, entendimento e utilização da plataforma de IoT Dojot, e por fim, o entendimento sobre RNAs.

4 SOFTWARE NORTE ROTAS

Neste Capítulo são apresentadas as etapas para o desenvolvimento do produto de *software* Norte Rotas. A Seção 4.1 apresenta o contexto e os objetivos do *software*. A Seção 4.2 apresenta os requisitos levantados. Já na Seção 4.3 o autor apresenta como foi modelado o projeto de banco de dados do Norte Rotas.

Os resultados são apresentados na Seção 4.4, e por fim, na Seção 4.5, o autor conclui este Capítulo com as considerações finais sobre o *software* desenvolvido.

4.1 Introdução

Smart campus é uma estrutura baseada em redes de IoT para prover de forma eficiente comunicação e informação em tempo real para usuários e dispositivos eletrônicos instalados em um campus universitário (ZHAMANOV, 2017). A ideia de *smart campus* pode ser considerada um caminho para redução de emissões de Dióxido de Carbono (CO_2), e melhoria no dia a dia da população a partir de soluções aplicadas na mobilidade urbana dentro de uma área.

A partir deste contexto, este capítulo apresenta uma proposta de *software* intitulado "*Software* de Planejamento de Rotas para Pedestres em *Smart Campus* – Norte Rotas" (ou chamado somente de Norte Rotas), que é um produto de *software* que foi idealizado a partir da necessidade de informações de rotas de pedestres. Seu objetivo é atuar através da coleta e adição de informações sobre rotas a pé do campus da UFPA, sendo estas integradas de forma automática ao banco de dados do projeto SIMA.

4.2 Requisitos

A etapa inicial de levantamento de requisitos do Norte Rotas, se deu através de reuniões entre os membros das equipes do Centro de Excelência em Eficiência Energética da Amazônia (CEAMAZON) – UFPA, Laboratório de Computação e Telecomunicações (LCT) – UFPA e demais colaboradores.

Durante as reuniões, foram sugeridas adição de novas informações ao projeto SIMA, sendo elas relacionadas à consideração de rotas a pé como opção aos usuários e apresentação

de informações complementares sobre a qualidade da via a ser percorrida pelo ônibus ou pedestres. Desta forma, iniciou-se a o processo de desenvolvimento e registro do *software* Norte Rotas para o projeto SIMA.

Para o sucesso no desenvolvimento do Norte Rotas, foi necessário iniciar com o levantamento de requisitos que o sistema atende, em seguida, foi modelado o banco de dados relacional, e por fim, se fez necessário a implementação e teste do *software*. Esta Seção apresenta a etapa inicial que corresponde ao levantamento de requisitos.

No levantamento de requisitos, foram levadas em consideração as necessidades dos clientes deste *software*, os quais estão representados pelos membros do LCT e CEAMAZON, que por sua vez serão responsáveis em utilizar a plataforma para adicionar novos dados de geolocalização de possíveis rotas a pé ao banco de dados do projeto SIMA.

O Quadro 4 apresenta os requisitos funcionais do Norte Rotas. Estes estão organizados pelo identificador RF#, de 01 a 12, e possuem descrições detalhadas sobre o que foi projetado para implementação da aplicação e atendimento do usuário. Em linhas gerais, este *software* é do tipo *web*, com responsividade para dispositivos móveis, e tem como função principal a realização de tarefa administrativa, onde a equipe do projeto poderá automatizar a tarefa de elaboração de rotas a pé, assim como, poderá categorizar as rotas a pé a partir de elementos da infraestrutura do espaço e condições de trafegabilidade.

Quadro 4 – Requisitos funcionais do Norte Rotas.

RF#	**Descrição**
RF01	O sistema deve permitir ao usuário fazer seu cadastro através do e-mail.
RF02	O sistema deve permitir ao usuário a recuperação/alteração de sua senha de acesso.
RF03	O sistema deve permitir ao usuário fazer login e logout de sua conta.
RF04	O sistema deve permitir ao usuário selecionar o ponto inicial e final de uma rota a partir de uma lista composta por todos os prédios cadastrados no banco de dados do projeto SIMA.
RF05	O sistema deve permitir que o usuário busque por rotas a pé entre dois pontos (neste projeto entre dois prédios já cadastrados no banco de dados do projeto SIMA).
RF06	O sistema deve permitir ao usuário salvar no banco de dados do projeto a rota encontrada a partir do requisito RF05.
RF07	O sistema deve informar ao usuário se a rota já pertence ou ainda não ao banco de dados de rotas a pé.
RF08	O sistema deve armazenar, ao cadastrar uma nova rota entre dois pontos, os seguintes itens: autor da rota, se possui boa iluminação, se a rota está coberta ou descoberta, se a rota possui pavimentação, se possui piso tátil ou rampa ou corrimão, se a rota pode ser utilizada por bicicletas ou automóveis, se a rota oferece a sensação de segurança aos usuários.
RF09	O sistema deve apresentar ao usuário, de forma gráfica, o mapa e a rota sugerida entre o ponto inicial e final.

RF10	O sistema deve apresentar todas as rotas a pé cadastradas no banco de dados.
RF11	O sistema deve apresentar a distância a ser percorrida a pé e o tempo de viagem a pé entre os pontos inicial e final da rota.
RF12	O sistema deve permitir que usuários com o maior grau de hierarquia façam a gestão de outros usuários no *software*.

Fonte: elaborado pelo autor.

O Quadro 5 apresenta os requisitos não funcionais do *software* Norte Rotas, organizados com o identificador RNF#. Estes requisitos atendem as necessidades de uso da aplicação em relação a usabilidade, desempenho, segurança da informação, entre outros.

Quadro 5 – Requisitos não funcionais do Norte Rotas.

RNF#	**Descrição**
RNF01	O *software* deve ser compatível com os navegadores Google Chrome e Mozilla Firefox.
RNF02	O sistema deve possuir mecanismos de segurança para autenticação de usuário e controle de acesso a conteúdo.
RNF03	O *software* deve possuir interface gráfica que proporcione interatividade, dinamicidade e agilidade para os usuários.
RNF04	O sistema deve permitir que os dados sejam tratados e os resultados sejam coerentes. Assim como, deve permitir que a análise do usuário especialista decida o cadastro ou descarte da rota
RNF05	O sistema deve ter alta disponibilidade (24 horas/7 dias por semana), para que as informações sejam inseridas e acessadas a qualquer tempo.
RNF06	O sistema deve executar as tarefas em tempo hábil (respostas às requisições em segundos), tornando o uso agradável.
RNF07	O sistema deve proteger os dados de seus usuários, assim como criptografar as senhas e ter mecanismos *anti-hacking*.
RNF08	O sistema deve ser implementado dentro da arquitetura *web*, utilizando as linguagens PHP, Javascript, HTML5 e *Cascading Style Sheets* (CSS).
RNF09	O sistema deve ser responsivo e adaptar-se em diferentes dimensões de telas, tanto em *desktop* quanto em *mobile*.

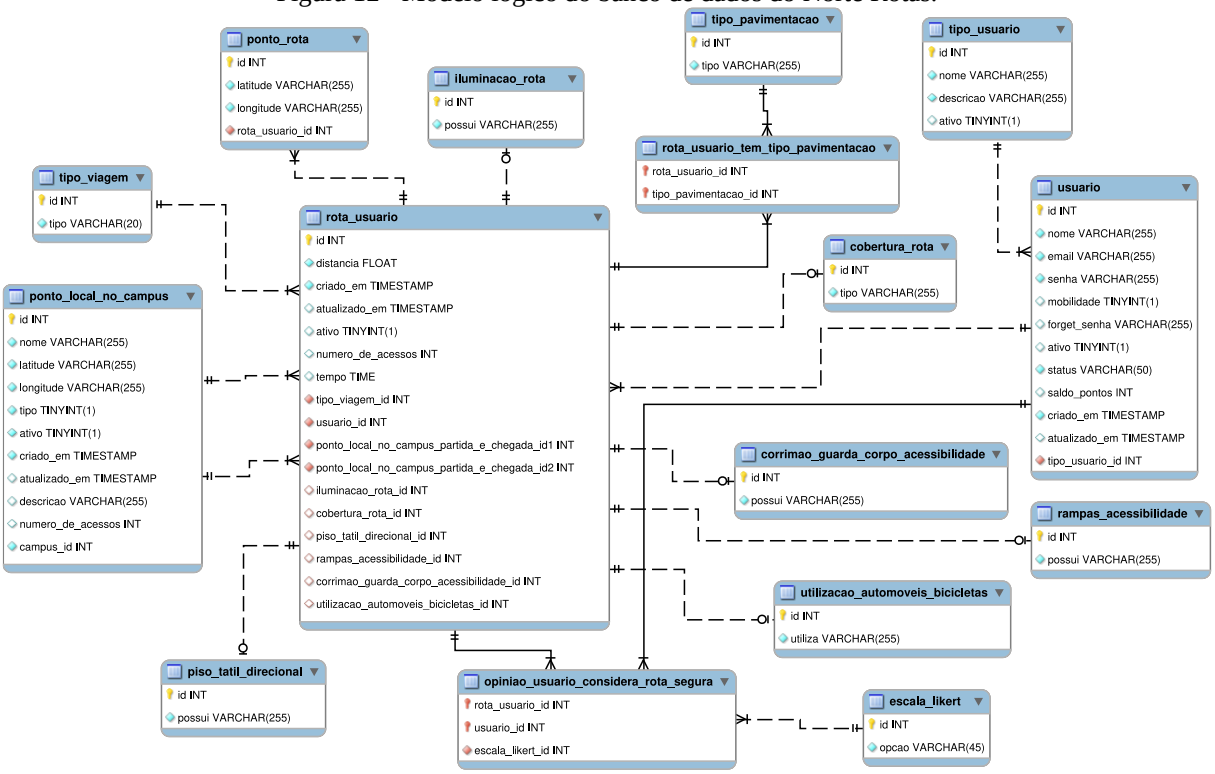
Fonte: elaborado pelo autor.

4.3 Modelagem do Banco de Dados

Esta Seção apresenta a modelagem lógica do banco de dados do *software* Norte Rotas. A modelagem lógica é uma etapa de grande importância para definir a forma como os dados do *software* serão armazenados.

O modelo lógico se deu após a fase de levantamento e refinamento dos requisitos do *software*, utilizando conceitos de tabela, atributo, relacionamento e cardinalidade. A Figura 12 ilustra o modelo lógico do banco de dados do Norte Rotas contendo as 16 tabelas, seus relacionamentos, atributos, chaves e tipos de dados.

Figura 12 - Modelo lógico do banco de dados do Norte Rotas.



Fonte: elaborada pelo autor.

Cada tabela da modelagem tem um objetivo específico no banco de dados, conforme listados a seguir:

- **usuario**: tabela responsável em salvar dados do perfil do usuário, tais como nome, e-mail, senha, etc.;
- **tipo_usuario**: tabela responsável em salvar o nome dos diferentes tipos de usuário do sistema, que podem ser, por exemplo “comum”, “moderador”, “admin” e “proprietário”;
- **rota_usuario**: tabela responsável em guardar dados referentes às rotas que podem ser utilizadas pelos usuários, tanto de ônibus, barco ou a pé. Neste contexto, esta tabela armazena somente rotas do tipo a pé. Os atributos salvos nesta tabela são identificadores (IDs) dos pontos de partida e chegada, do próprio registro da rota, do usuário que cadastrou a rota, do tipo de viagem, do tipo de iluminação da rota, do tipo de cobertura, entre outros;
- **ponto_rota**: tabela responsável em armazenar todos os pontos de geolocalização de uma determinada rota. Os atributos são, ID, latitude, longitude, e identificador que faz referência a tabela **rota_usuario**;

- **tipo_viagem**: tabela responsável em salvar os IDs dos registros e diferentes tipos de viagens, tais como ônibus, barco e a pé. Vale ressaltar que cada ID desta tabela é referenciado na tabela **rota_usuario**;
- **ponto_local_no_campus**: tabela que armazena todos os pontos de geolocalização de prédios, paradas de ônibus e cais de barcos, presentes no campus. Esta tabela é composta por ID, nome do ponto, latitude, longitude, tipo (que pode ser prédio, parada de ônibus, ou cais de barco), etc.;
- **cobertura_rota**: tabela que guarda o tipo de cobertura de uma determinada rota, esse tipo pode ser coberta, parcialmente coberta, ou descoberta. É importante ressaltar que cada ID desta tabela é referenciado na tabela **rota_usuario**;
- **iluminacao_rota**: tabela projetada para armazenar dados sobre a qualidade da iluminação da rota. O ID desta tabela é referenciado na tabela **rota_usuario**;
- **rampas_acessibilidade**: tabela projetada para armazenar dados sobre a existência de rampas de acessibilidade nas rotas cadastradas no sistema. O ID desta tabela é referenciado como chave estrangeira na tabela **rota_usuario**;
- **corrimao_guarda_corpo_acessibilidade**: tabela projetada para armazenar dados sobre a existência de corrimão e guarda corpo acessibilidade nas rotas cadastradas no sistema. O ID desta tabela é referenciado como chave estrangeira na tabela **rota_usuario**;
- **piso_tatil_direcional**: tabela projetada para armazenar dados sobre a existência de piso de alerta tátil direcional nas rotas cadastradas no sistema. O ID desta tabela é referenciado na tabela **rota_usuario**;
- **utilizacao_automoveis_bicicletas**: tabela usada para armazenar o valor lógico “sim” ou “não” quanto a utilização das rotas de pedestres por automóveis ou bicicletas. O ID desta tabela é referenciado na tabela **rota_usuario**;
- **tipo_pavimentacao**: tabela usada para armazenar diferentes tipos de pavimentação de uma rota, tais como “blocos”, “asfalto”, “concreto”, “seixo”, “grama” e “terra”;
- **rota_usuario_tem_tipo_pavimentacao**: tabela gerada da relação de cardinalidade N para N entre as tabelas **rota_usuario** e **tipo_pavimentacao**. Ela armazena como chaves estrangeiras os IDs das tabelas **rota_usuario** e **tipo_pavimentacao**, desta forma é possível relacionar vários tipos de pavimentação para uma rota, ou um tipo de pavimentação para várias rotas;
- **opiniao_usuario_considera_rota_segura**: tabela gerada da relação de cardinalidade N para N entre as tabelas **rota_usuario** e **usuario**. Ela armazena a opinião do usuário quanto o nível de segurança de determinada rota. Seus atributos são ID da rota, ID do

usuário e ID da resposta baseada na escala de *Likert* (JOSHI; *et al.* 2015). Os IDs fazem referência às tabelas **rota_usuario**, **usuario**, e **escala_likert**;

- **escala_likert**: tabela que armazena seus IDs e opções da escala de *Likert*, tais como "Concordo Completamente", "Concordo", "Indiferente", "Discordo", e "Discordo completamente".

A modelagem física do banco de dados a partir deste modelo lógico, foi gerada automaticamente pelo *software* MySql Workbench, utilizando a linguagem SQL padrão para o SGBD MySql. Este trabalho não apresenta a modelagem do banco de dados físico em SQL.

4.4 Cenário e Resultados

Esta Seção apresenta o cenário de coleta de rotas, o resultado da implementação do Norte Rotas através de suas telas, bem como resultados de testes de utilização e coleta de rotas.

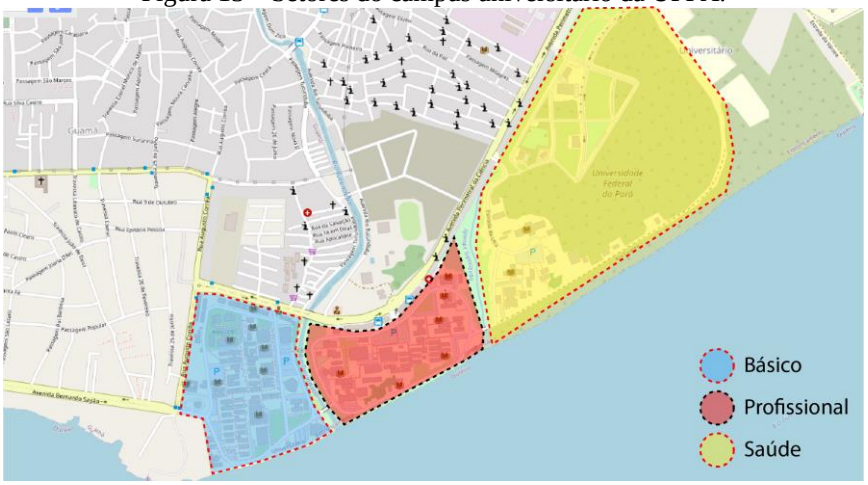
Após o levantamento de requisitos e modelagem do banco de dados, foi iniciado o processo de implementação. A lógica de negócios e instruções de processamento por detrás (o *back-end*) do *software* Norte Rotas foi desenvolvida com a linguagem de programação PHP (versão 8.0). As linguagens HTML, CSS e Javascript foram utilizadas para criação e interação do usuário com a interface gráfica. Testes foram feitos utilizando os navegadores de internet Mozilla Firefox (versão 94.0.1 – 64 bits) e Google Chrome (versão 96.0.4664.45 – 64 bits). Para o armazenamento dos dados, foi adotado o sistema de gerenciamento de banco de dados MySql.

4.4.1 Cenário de Coleta

O campus da UFPA em Belém foi o cenário de coleta de dados do *software* Norte Rotas. Ele tem uma área de aproximadamente 450 mil m^2, composta por cerca de 142 prédios (MENDES, 2018). Sua infraestrutura acolhe atualmente uma população universitária de mais de 56 mil pessoas (UFPA, 2021). A Figura 13 ilustra o campus universitário de Belém, o qual está dividido em três setores: básico, profissional e saúde.

A mobilidade de pedestres entre as 142 edificações do campus se dá através de rotas que passam por ruas e passarelas, o que resulta em aproximadamente 10 mil rotas possíveis, levando em conta que o caminho de ida pode ser o mesmo da volta.

Figura 13 – Setores do campus universitário da UFPA.



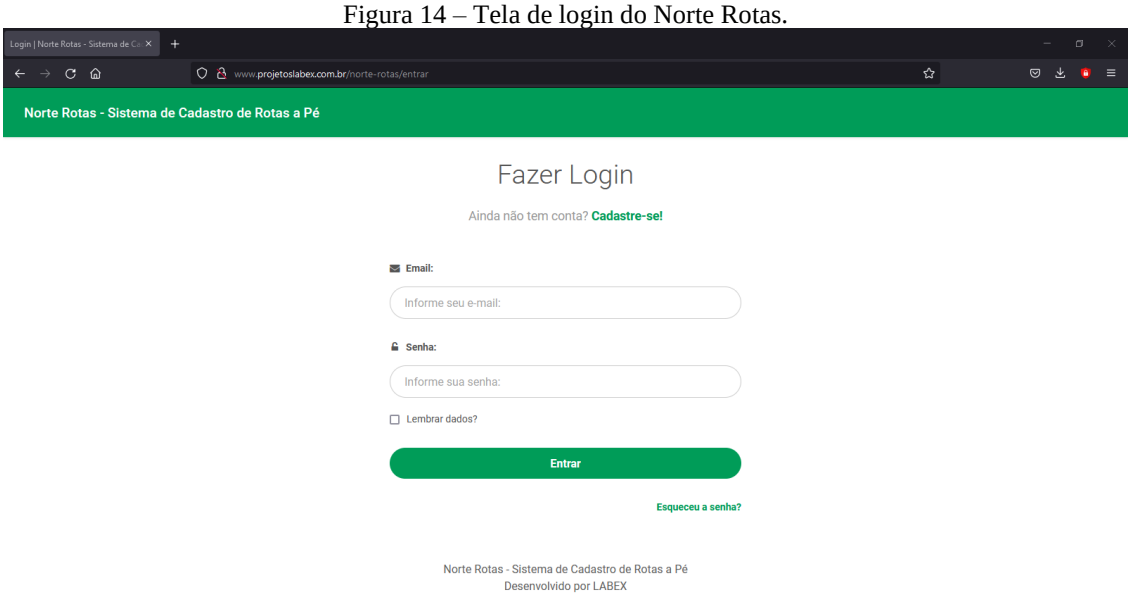
Fonte: elaborada pelo autor.

4.4.2 Telas do *Software*

As figuras apresentadas nesta Subseção são capturas de telas do Norte Rotas, ou seja, elas ilustram fielmente o resultado do produto de *software* produzido.

A Figura 14 ilustra a tela de *login*, onde o usuário pode entrar com seu endereço de e-mail e senha. Na mesma tela, é possível acessar *links* para recuperação de senha e cadastro de um novo usuário.

Figura 14 – Tela de login do Norte Rotas.



Fonte: elaborada pelo autor.

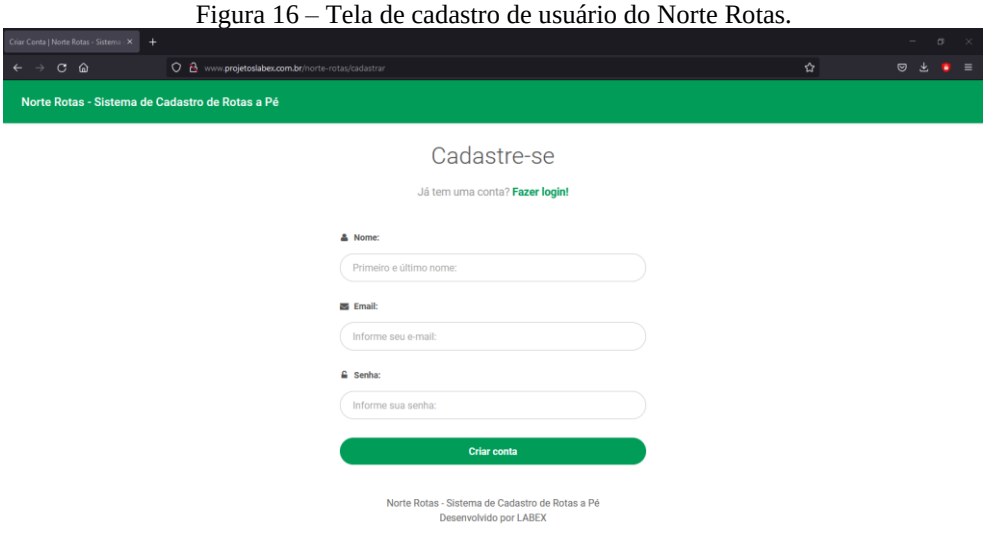
Para a recuperação de senha, o usuário pode informar ao sistema seu e-mail, e receber através de uma mensagem de e-mail o link para o processo criação de uma nova senha. A Figura 15 ilustra a tela de recuperação de senha.

Figura 15 – Tela de recuperação de senha do Norte Rotas.

Fonte: elaborada pelo autor.

O processo de cadastro de usuário no *software* é ilustrado na Figura 16, ele requer e-mail, senha e nome. A partir da liberação de acesso ao usuário, este poderá armazenar os resultados das buscas por rotas a pé, que posteriormente poderão ser consultados pelos usuários do projeto SIMA, assim como, também é possível: obter resultados gráficos através do mapa da localização inicial e final de pontos no mapa; e obter resultados numéricos sobre a distância percorrida e tempo entre os percursos. Por fim, é possível preencher as informações adicionais sobre a rota e sua infraestrutura.

Figura 16 – Tela de cadastro de usuário do Norte Rotas.

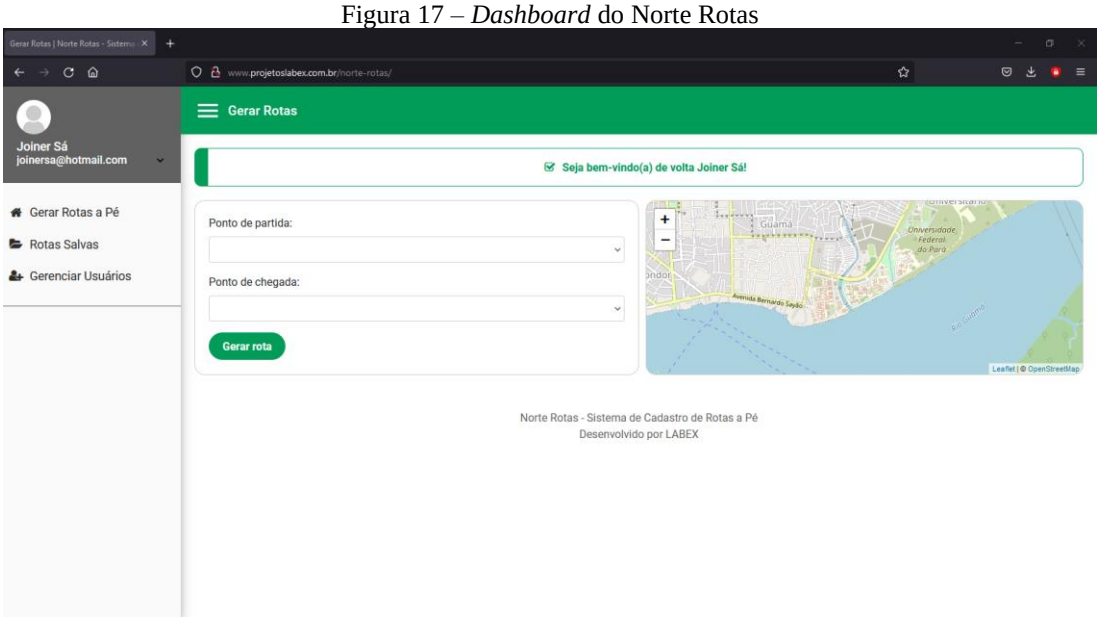


Fonte: elaborada pelo autor.

A Figura 17 ilustra o *dashboard* do Norte Rotas, onde é apresentado ao usuário um menu lateral esquerdo composto pela foto do usuário, seu nome e e-mail, além das opções de “Gerar Rotas a Pé” e “Rotas Salvas” para usuários de nível “admin”, e “Gerenciar Usuários” para usuários de nível “proprietário”. Na parte central e direita é apresentado a tela principal

com as opções correspondentes a funcionalidade “Gerar Rotas a Pé”. Nesta tela, o usuário pode selecionar um ponto de partida e um ponto de chegada, os quais são advindos do banco de dados do projeto SIMA.

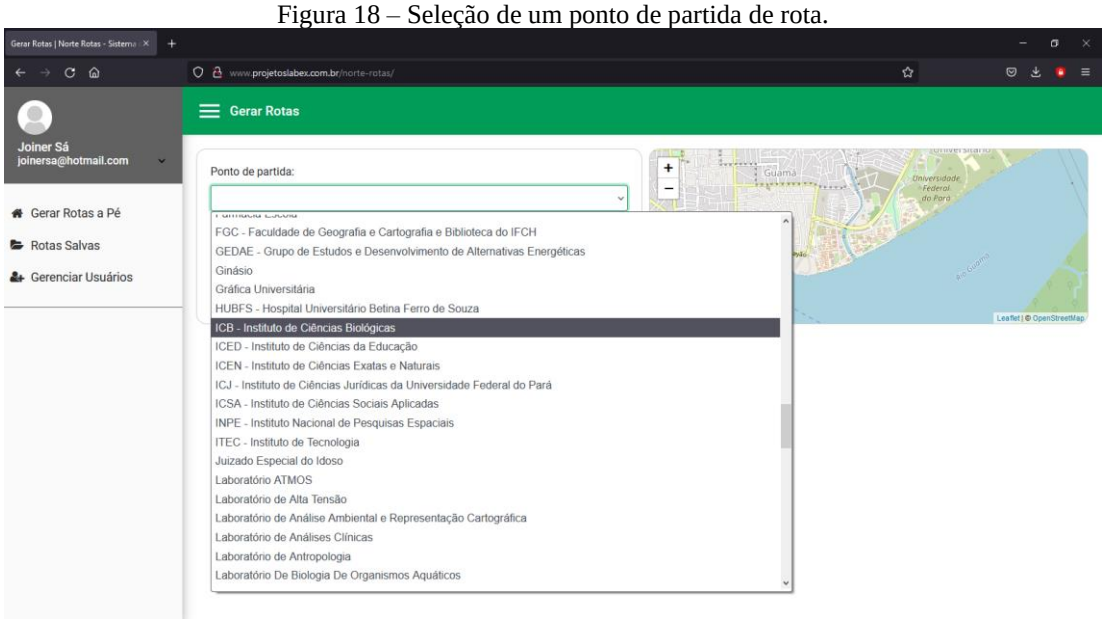
Figura 17 – *Dashboard* do Norte Rotas



Fonte: elaborada pelo autor.

A Figura 18 e Figura19 ilustram, respectivamente, a seleção das opções de partida e chegada. Após selecionar as opções, o usuário pode interagir com o botão “Gerar Rotas”. É válido reforçar que as informações de ponto de partida e chegada são advindas do banco de dados do projeto SIMA e correspondem a lista de prédios que serão mapeados para o projeto e estão localizados dentro do campus da UFPA.

Figura 18 – Seleção de um ponto de partida de rota.



Fonte: elaborada pelo autor.

Figura 19 – Seleção de um ponto de chegada de rota.

Fonte: elaborada pelo autor.

A Figura 20 ilustra o próximo passo a ser realizado após a seleção dos pontos de partida e chegada. É válido observar que os pontos selecionados são apresentados graficamente no mapa localizado no lado superior direito, sendo ambos representados por marcadores em cor verde.

Figura 20 – Botão de gerar rota e mapa com ponto inicial e final.

Fonte: elaborada pelo autor.

Ao clicar no botão “Gerar rota”, o sistema irá calcular a menor rota possível entre os dois pontos dispostos no mapa, para isso foi utilizada a API do *GraphHopper*, que por sua vez

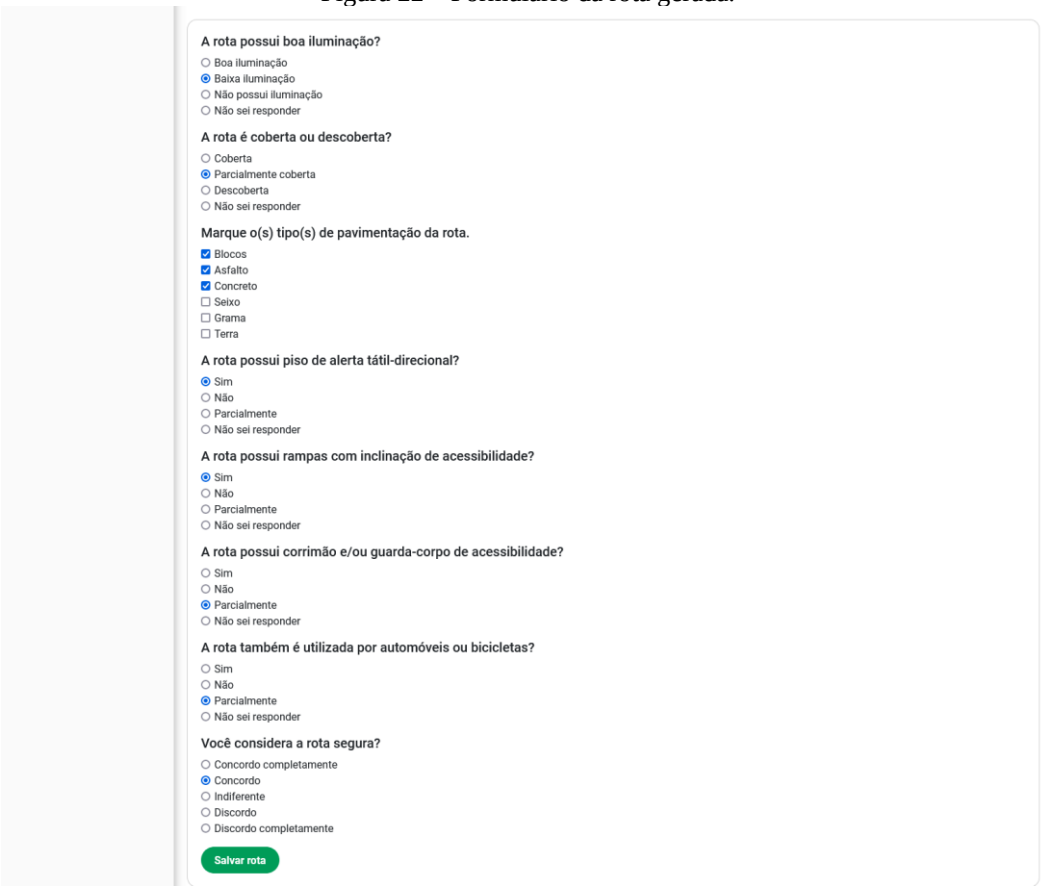
é uma biblioteca e um servidor de roteamento de código aberto escrito em Java que fornece uma API de roteamento HTTP. Como resultado, o usuário é redirecionado para uma nova tela composta por informações sobre a rota gerada (dados sobre o ponto de partida e chegada, distância percorrida e tempo de viagem a pé), representada pela Figura 21. Além das informações, nesta tela o usuário é convidado a preencher um formulário de múltipla escolha que corresponde a perguntas sobre infraestrutura e trafegabilidade na rota. Tais perguntas são baseadas no requisito funcional RF08, apresentado no Quadro 4 presente na Seção 4.2. Ao final do formulário, o usuário deve confirmar clicando no botão “Salvar Rota”, e após o clique, a rota passará a ficar disponível na área “Rotas Salvas”.

Figura 21 – Informações da rota gerada pelo *software* Norte Rotas.

Fonte: elaborada pelo autor.

A Figura 22 ilustra a tela com as oito perguntas do formulário da rota a ser salva, além do botão “Salvar rota”.

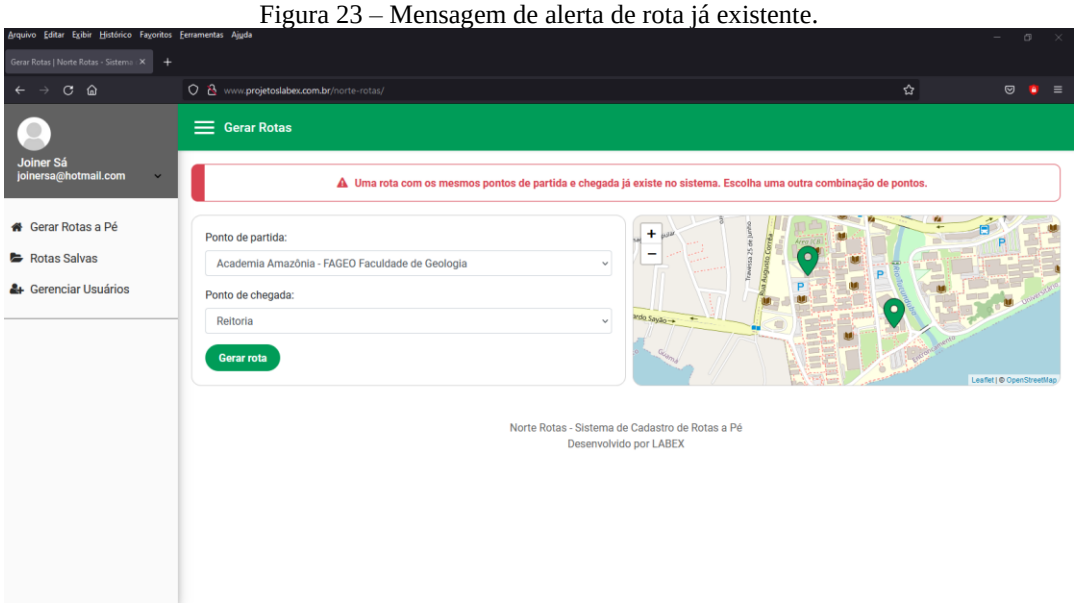
Figura 22 – Formulário da rota gerada.



Fonte: elaborada pelo autor.

Caso o usuário tenho escolhido pontos de partida e chegada de uma rota já existente no banco de dados, uma mensagem de alerta é emitida pelo *software* conforme ilustra a Figura 23.

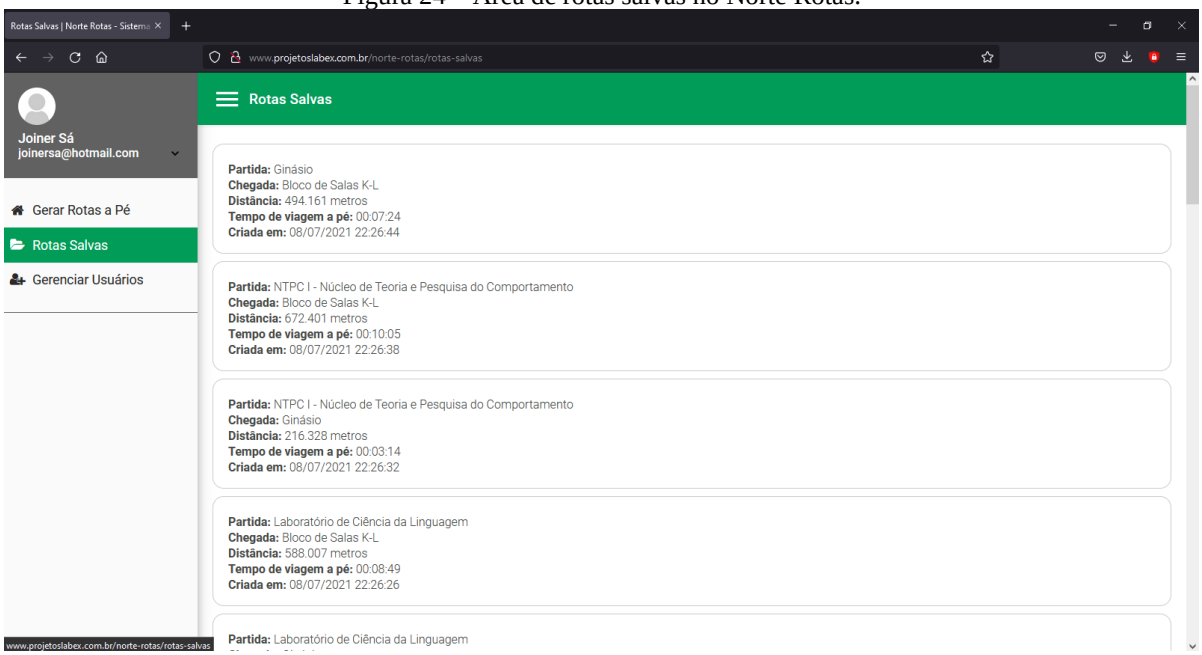
Figura 23 – Mensagem de alerta de rota já existente.



Fonte: elaborada pelo autor.

A Figura 24 ilustra a área de “Rotas Salvas”, onde todas as rotas criadas pelos envolvidos no projeto estarão disponíveis para rápida consulta, sendo que estas rotas e seus dados também estarão disponíveis em outros *softwares* projeto SIMA, no entanto, a forma de acesso irá se diferenciar. Neste *software*, o acesso se dá em lista, já em outros *softwares* será a partir da escolha do ponto de partida e chegada definido pelo usuário, onde a rota já cadastrada irá ser apresentada em conjunto com as informações de infraestrutura produzidas a partir do uso do Norte Rotas.

Figura 24 – Área de rotas salvas no Norte Rotas.

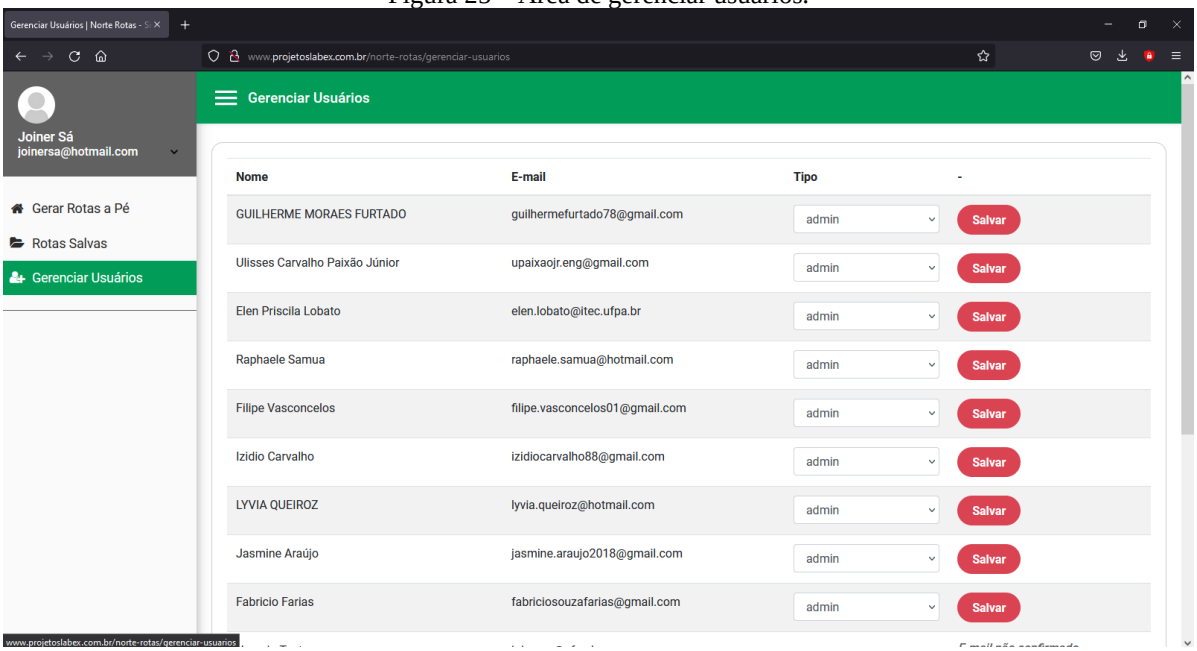


Fonte: elaborada pelo autor.

A Figura 25 ilustra a área “Gerenciar Usuários”, cuja a função é possibilitar a alteração do tipo do usuário, assim, escolher os usuários apropriados para o uso do Norte Rotas. Esta função está disponível apenas para usuários do tipo “proprietário”, que tem o grau mais alto na hierarquia de usuários do banco de dados do projeto SIMA.

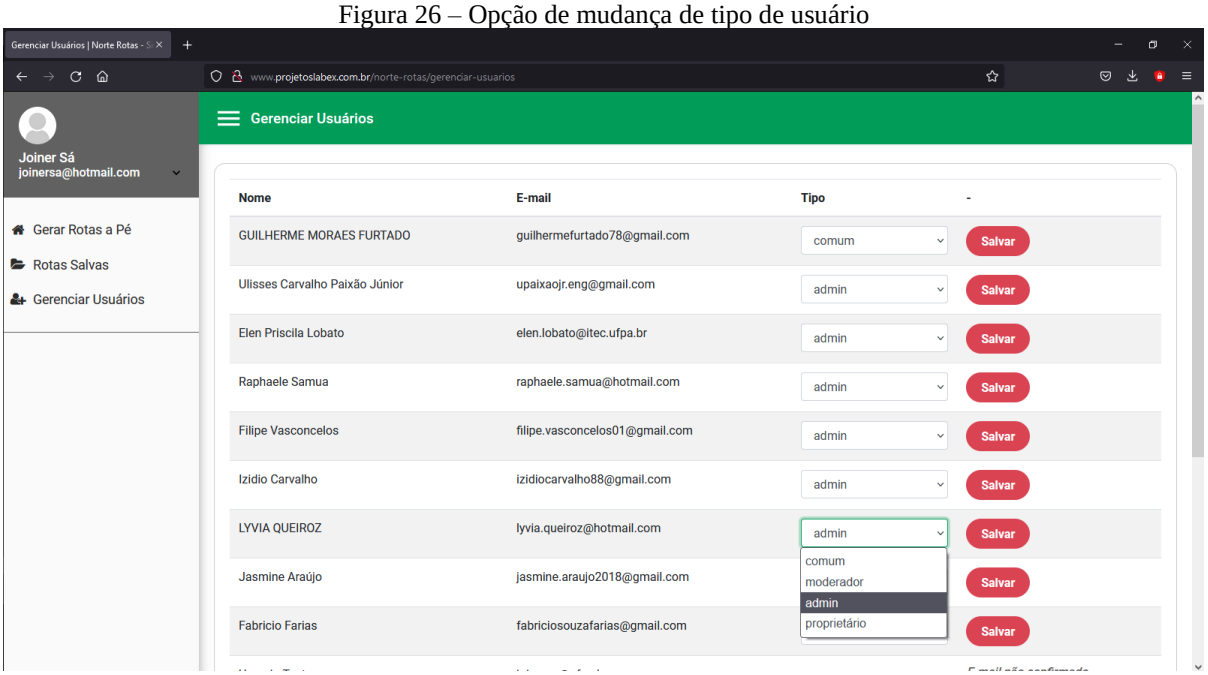
É importante ressaltar que os únicos usuários permitidos para utilização deste *software*, são do tipo “admin” e “proprietário”. Quando um novo usuário se cadastra no sistema, ele automaticamente será do tipo “comum”, cabendo apenas aos usuários “proprietários” liberarem seu acesso através da mudança do tipo de usuário, conforme ilustra a Figura 26.

Figura 25 – Área de gerenciar usuários.



Fonte: elaborada pelo autor.

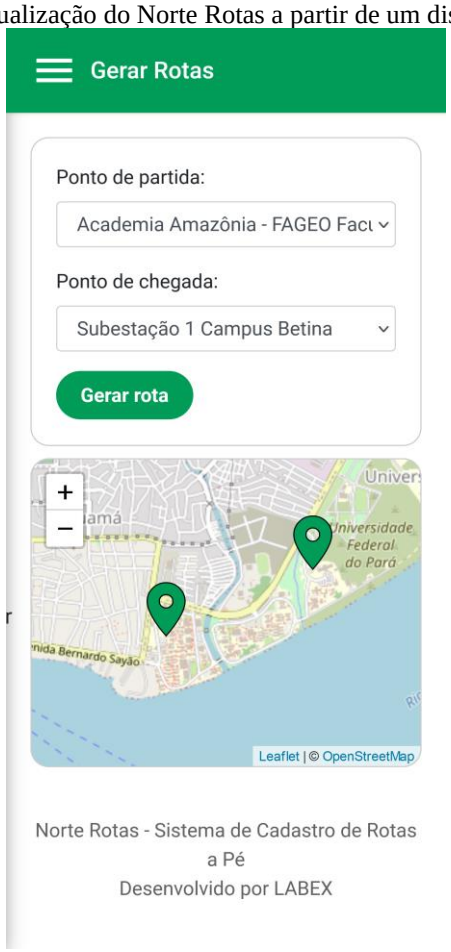
Figura 26 – Opção de mudança de tipo de usuário



Fonte: elaborada pelo autor.

A Figura 27 ilustra um exemplo de visualização a partir de um dispositivo móvel da área de “Gerar Rotas” do Norte Rotas. Esta visualização se deu através do navegador Google Chrome (96.0.4664.45), na versão móvel para o sistema operacional Android.

Figura 27 – Visualização do Norte Rotas a partir de um dispositivo móvel.



Fonte: elaborada pelo autor.

Na Figura 27 é possível ver a responsividade do Norte Rotas em se adaptar em uma tela pequena. Isso possibilita que o sistema seja usado em situações onde não se tem acesso a um computador *desktop*.

4.4.3 Resultados de Coletas de Rotas

O Norte Rotas foi utilizado, até o momento da escrita deste trabalho, para cadastrar 9987 rotas a pé distribuídas nos três setores do campus.

As Figuras desta Subseção ilustram seis amostras da utilização do *software* produzido. Estas amostras apresentam resultados de combinações de rotas dos três setores do campus.

A Figura 28 ilustra uma das rotas geradas pelo *software*, no setor básico da UFPA, que corresponde a trajetória iniciando no prédio da Faculdade de Geofísica e terminando no Restaurante Universitário (RU). Os resultados indicam que a menor distância entre os prédios é de 800 metros e o tempo previsto de caminhada entre os prédios é de 12 minutos.

Figura 28 – Resultados de uma rota no setor Básico do campus.

Fonte: elaborada pelo autor.

Na Figura 29 é ilustrado o resultado da utilização do Norte Rotas para gerar uma rota entre os setores Básico e Profissional, tendo como ponto de partida o Instituto de Ciências Biológicas, e ponto de chegada o RU do setor Profissional. Segundo resultados, a distância do percurso calculado é de um pouco mais de 1,2 km, o que dá em média 18 minutos de caminhada.

Figura 29 – Resultados de uma rota entre os setores Básico e Profissional do campus.

Fonte: elaborada pelo autor.

Na rota saindo da Faculdade de Artes Visuais em direção ao Bloco K – Serviço social, no setor profissional do campus, o *software* calculou a menor distância de 415 metros, com um tempo de caminhada de 6 minutos e 13 segundos, conforme Figura 30.

Figura 30 – Resultados de uma rota no setor Profissional do campus.

Fonte: elaborada pelo autor.

Já um resultado de uma rota entre os setores Profissional e Saúde, partindo do Ver-o-pesinho do profissional em direção à Faculdade de Odontologia, conforme Figura 31, apresentou a menor trajetória de aproximadamente 1 km e 43 metros, com um tempo de viagem caminhando de menos de 16 minutos.

Figura 31 – Resultado de uma rota entre os setores Profissional e Saúde do campus.

Fonte: elaborada pelo autor.

A Figura 32 ilustra a menor trajetória a pé entre a Faculdade de Nutrição e a Biblioteca Central da UFPA, situados respectivamente no setor Saúde e Básico do campus. Para esta rota, o *software* apresenta uma distância de um pouco menos que 1,7 km para um tempo de caminhada de aproximadamente 25 minutos.

Figura 32 – Resultado de uma rota entre os setores Saúde e Básico do campus.

Fonte: elaborada pelo autor.

Em um experimento onde o usuário escolhe sair do Centro de Atenção à Saúde da Mulher e da Criança em direção à Faculdade de Enfermagem, ambos no setor Saúde, o Norte Rotas apresenta como resultado a menor rota com uma distância de 454 metros e tempo de caminhada de 6 minutos e 48 segundos, conforme Figura 33. Além das informações geradas pelo sistema, o usuário utilizador do *software* informou condições relevantes sobre a estrutura física e de segurança da rota, alimentando, desta forma, o banco de dados do projeto SIMA.

Figura 33 – Resultado de uma rota dentro do setor Saúde do campus.

Fonte: elaborada pelo autor.

4.5 Conclusão

Este Capítulo apresentou a concepção dos requisitos, modelagem da base de dados e amostras da utilização de um *software* de provimento de informações relevantes sobre rotas de pedestres em um *smart campus*. A solução proposta é uma ferramenta importante para geração de conhecimento sobre condições estruturais e de segurança de ruas e passarelas de um *smart campus*, as quais poderão ser utilizadas para tomada de decisões de mobilidade urbana. A partir dos dados coletados, será possível adotar sistemas especialistas para tomada de decisões com o intuito de melhorar a qualidade de vida da comunidade acadêmica, além de ser fonte de dados para outras aplicações no contexto da IoT. É importante ressaltar que o sistema pode ser generalizado para provimento de informações de cidades inteligentes, não somente em um campus específico.

5 APLICATIVO MÓVEL SIMA MOBILIDADE MULTIMODAL

Este Capítulo apresenta o aplicativo móvel SIMA Mobilidade Multimodal e as etapas para seu desenvolvimento. A Seção 5.1 apresenta uma introdução e o objetivo deste *software*. Na Seção 5.2 é apresentada a fase de requisitos. Já a Seção 5.3 apresenta a modelagem do banco de dados do sistema móvel proposto.

Na Seção 5.4, o autor apresenta uma API REST desenvolvida para dar suporte à comunicação entre o aplicativo móvel e o banco de dados remoto. Já na Seção 5.5 são apresentados o cenário de aplicação e os resultados de utilização do *software*. Por fim, a Seção 5.6 apresenta as conclusões.

5.1 Introdução

O SIMA Mobilidade Multimodal é um *software* para plataforma Android que visa gerir dados de modais do projeto SIMA. O aplicativo pode apresentar informações como geolocalização, temperatura interna, número de passageiros, horário de chegada, trajetórias e distâncias das rotas dos ônibus e barco elétricos do campus de Belém da UFPA.

Para o sucesso no desenvolvimento do aplicativo, foi necessário iniciar com o levantamento de requisitos que o sistema atende, em seguida, foi necessário propor a modelagem do banco de dados para o recebimento de informações referentes aos usuários e às rotas dos modais, e por fim, se fez necessário a implementação e teste do *software*.

5.2 Requisitos

Nesta etapa, os requisitos foram considerados segundo as necessidades dos usuários finais deste *software*, os quais podem ser representados por alunos, servidores e visitantes do campus. O levantamento dos requisitos se deu a partir de uma reunião com os desenvolvedores desta aplicação com os demais membros do projeto SIMA, os quais fazem parte do laboratório LCT e CEAMAZON.

O Quadro 6 apresenta 16 requisitos funcionais do *software* móvel SIMA Mobilidade Multimodal, os quais estão organizados com o identificador RF#, e possuem a descrição detalhada sobre o que é esperado a ser implementado para o atendimento dos usuários.

Quadro 6 – Requisitos funcionais do *software* SIMA Mobilidade Multimodal.

RF#	Descrição
RF01	O aplicativo deve permitir ao usuário fazer seu cadastro através de seu e-mail e senha.
RF02	O aplicativo deve permitir ao usuário fazer *login* e *logout* de sua conta.
RF03	O aplicativo deve permitir ao usuário a redefinição de sua senha de acesso.
RF04	O aplicativo deve apresentar graficamente no mapa para o usuário as localizações, em tempo real, dos modais ônibus e barco. Esta funcionalidade pode ser acessada tanto por usuários conectados, quanto por usuários não conectados ou que não possuem cadastro.
RF05	O aplicativo deve permitir ao usuário selecionar o ponto partida e chegada de uma rota a partir de uma lista composta por todos os pontos de prédios cadastrados no banco de dados.
RF06	O aplicativo deve permitir ao usuário escolher o tipo de rota desejada, tais como rota de ônibus, barco ou a pé.
RF07	O aplicativo deve permitir ao usuário visualizar no mapa a trajetória da rota escolhida no RF06, bem como apresentar a distância e o horário de chegada do modal ao ponto de parada de ônibus ou barco mais próximo do usuário. Quando a rota escolhida for a pé, o *software* deve apresentar a trajetória (no mapa), a distância e a média do tempo de caminhada.
RF08	O aplicativo deve apresentar o nome do marcador quando o usuário o pressionar. Os marcadores podem conter o nome do ônibus, barco, parada de ônibus, parada de barco, prédio de partida, prédio de chegada e usuário.
RF09	O aplicativo deve permitir ao usuário favoritar ou desfavoritar a rota selecionada no RF6.
RF10	O aplicativo deve permitir ao usuário visualizar todas suas rotas favoritas.
RF11	O aplicativo deve permitir ao usuário selecionar uma de suas rotas favoritas do RF10, para apresentar rapidamente as funções do RF07, sem a necessidade de o usuário passar por RF05 e RF06.
RF12	O aplicativo deve permitir ao usuário visualizar no mapa, separadamente dos demais modais, a localização, o número de passageiros e a temperatura, em tempo real, do ônibus elétrico intermunicipal.
RF13	O aplicativo deve apresentar para o usuário, uma lista com todos os modais presentes no campus.
RF14	O aplicativo deve permitir ao usuário selecionar um item do RF13, e apresentar informações individuais do modal escolhido. As informações a serem apresentadas são: a localização (graficamente no mapa e textualmente), a temperatura (interna e externa) e o número de passageiros.
RF15	O aplicativo deve permitir ao usuário acessar a área de perfil, onde é possível visualizar a foto de perfil, o nome e o e-mail do usuário conectado, bem como um botão para usuário desconectar-se do aplicativo.
RF16	O aplicativo deve apresentar para o usuário um menu lateral com todas as opções de acesso às funcionalidades do aplicativo.

Fonte: elaborado pelo autor.

Durante a reunião, foram levantados 11 requisitos não funcionais do *software*, eles estão organizados pelo identificador RNF# no Quadro 7. Estes requisitos atendem as

necessidades de uso da aplicação em relação a usabilidade, desempenho, segurança da informação, entre outros.

Quadro 7 – Requisitos não funcionais do *software* SIMA Mobilidade Multimodal.

RNF#	Descrição
RNF1	O aplicativo deve ser compatível com sistemas operacionais Android, a partir da versão 5.0 (API 21).
RNF2	O aplicativo deve possuir mecanismos de segurança para autenticação de usuário e controle de acesso a conteúdo.
RNF3	O aplicativo deve possuir interface gráfica que proporcione interatividade, dinamicidade e agilidade para o usuário.
RNF4	O aplicativo deve ter alta disponibilidade (24 horas/7 dias por semana), para que as informações sejam inseridas e acessadas a qualquer tempo.
RNF5	O aplicativo deve executar as tarefas em tempo hábil (respostas às requisições em segundos), tornando o uso agradável.
RNF6	O aplicativo deve proteger os dados de seus usuários, assim como criptografar as senhas e ter mecanismos anti-*hacking*.
RNF7	O aplicativo deve ser implementado dentro da arquitetura móvel Android, utilizando a linguagem de programação Java para os cálculos matemáticos e a lógica de negócios, e a linguagem de marcação *eXtensible Markup Language* (XML) para a construção dos componentes visuais do aplicativo.
RNF8	O aplicativo deve utilizar um banco de dados remoto através de uma API Rest.
RNF9	O aplicativo deve ser compatível com diferentes densidades de pixel, usando unidades de medição independentes de resolução da tela do *smartphone*.
RNF10	O aplicativo deve ser codificado utilizando *design patterns* que visem a facilidade de testes, reutilização de código, separação de responsabilidades, manutenção e extensibilidade do aplicativo.
RNF11	O aplicativo deve ter versionamento de código utilizando o Git[4].

Fonte: elaborado pelo autor.

5.3 Modelagem dos Bancos de Dados

Esta Seção apresenta a modelagem lógica do banco de dados relacional e a modelagem do banco de dados não relacional, ambos consumidos pelo *software* móvel SIMA Mobilidade Multimodal.

5.3.1 Banco de Dados Relacional

O banco de dados relacional do aplicativo móvel SIMA Mobilidade Multimodal, foi modelado com o objetivo de armazenar dados exclusivamente de usuários, rotas dos modais, prédios, e demais informações dos setores do campus. Este banco de dados foi projetado para

[4] Git é um sistema de controle de versão gratuito e de código aberto.

funcionar em um servidor remoto, utilizando o SGBD MySql, onde uma API REST (descrita na Seção 5.4) é responsável pela comunicação com o aplicativo móvel.

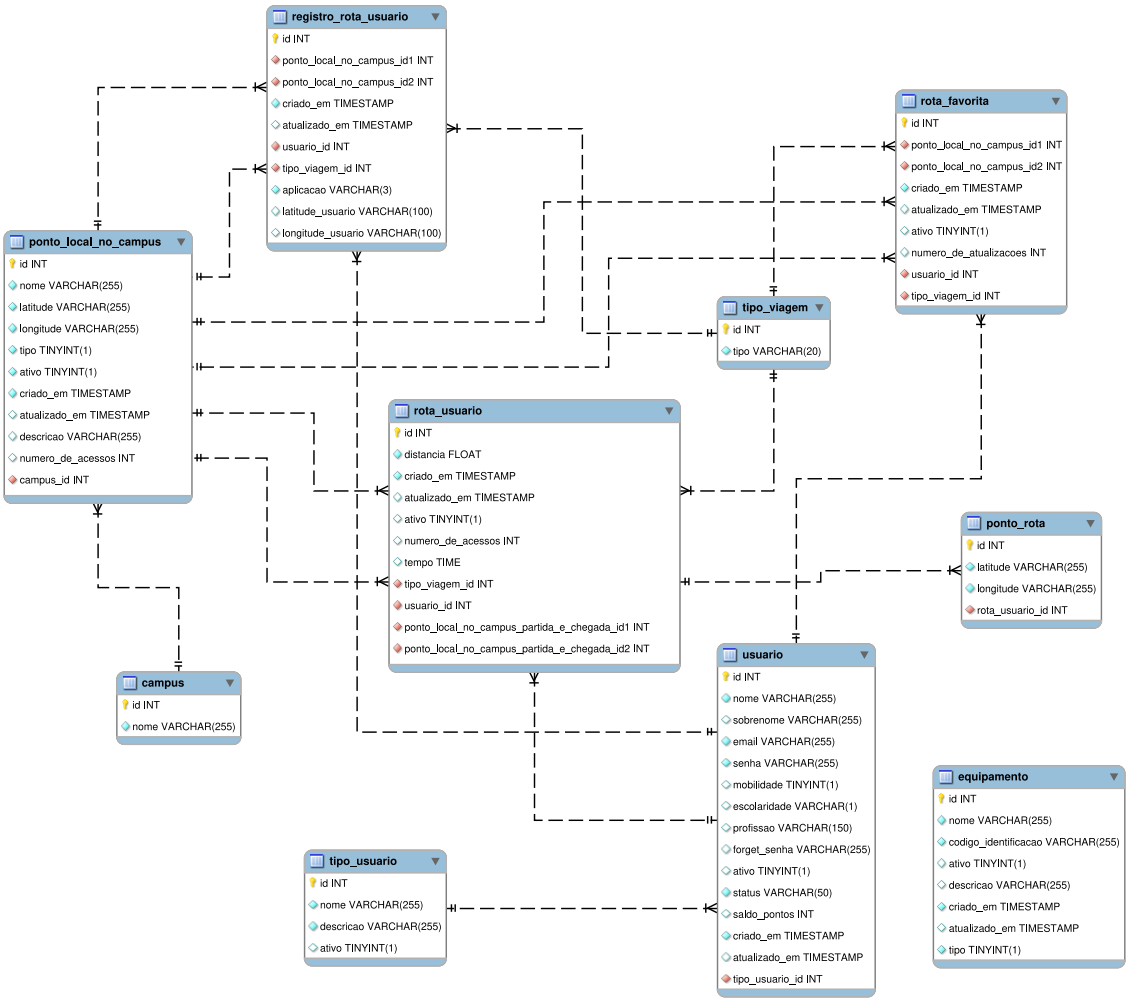
Para este banco de dados, o autor se concentrou em fazer a modelagem lógica, pois além dela proporcionar um simples entendimento entre desenvolvedores de *software*, ela é de grande importância para definir a forma como os dados do aplicativo serão armazenados.

O modelo lógico foi construído após o levantamento de requisitos. Nesta fase, foi utilizado o diagrama de modelo lógico baseado no MySql Workbench, conforme apresentado no Capítulo 3, Seção 3.3.

A Figura 34 ilustra o modelo lógico do banco de dados contendo as 11 tabelas (entidades), seus relacionamentos, atributos e tipos de dados. Cada tabela da modelagem tem um objetivo específico no banco de dados, conforme listados a seguir:

- **usuario**: tabela responsável em salvar dados do perfil do usuário, tais como nome, sobrenome, e-mail, senha, foto, tipo de mobilidade (reduzida ou não), escolaridade, profissão, ID do tipo do usuário (faz referência para a tabela **tipo_usuario**), entre outros;
- **tipo_usuario**: tabela responsável em salvar diferentes tipos de usuário do *software*, que podem ser, “comum”, “moderador”, “admin” e “proprietário”;
- **rota_usuario**: tabela armazena dados das rotas que podem ser utilizadas pelos usuários do sistema. Elas podem ser rotas de ônibus, barco ou a pé. No entanto, é importante ressaltar que esta tabela armazena somente rotas do tipo a pé (geradas pelo produto de *software* apresentado no Capítulo 4), sendo que as rotas de ônibus e barcos são concatenações de rotas a pé com outros tipos de rotas dos modais apresentadas no trabalho de Silva *et al*. (2020). Os atributos salvos nesta tabela são identificadores (IDs) dos pontos de partida e chegada, do próprio registro da rota, do tipo de viagem, entre outros;
- **ponto_rota**: tabela que armazena todos os pontos de geolocalização de uma determinada rota. Os atributos são: identificador (*PRIMARY KEY* e *AUTO_INCREMENT*), latitude, longitude, e identificador que faz referência (*FOREIGN KEY*) a tabela **rota_usuario**;
- **tipo_viagem**: tabela responsável em salvar os IDs dos registros e diferentes tipos de viagens, tais como ônibus, barco e a pé. Vale ressaltar que cada ID desta tabela é referenciado na tabela **rota_usuario**;

Figura 34 – Modelo lógico do banco de dados utilizado pelo SIMA Mobilidade Multimodal.



Fonte: elaborada pelo autor.

- **ponto_local_no_campus**: tabela que armazena todos os pontos de geolocalização de prédios, paradas de ônibus e cais de barcos, presentes no campus. Esta tabela é composta por ID, nome do ponto, latitude, longitude, tipo (que pode ser prédio, parada de ônibus, ou cais de barco), ID do campus (que faz referência a tabela **campus**), etc.;
- **campus**: tabela que armazena os nomes dos setores da UFPA Belém. Estes nomes podem ser: “Básico”, “Profissional” e “Saúde”;
- **registro_rota_usuario**: tabela responsável por salvar dados de uso do usuário no aplicativo. Os dados salvos são: ID (*PRIMARY KEY* e *AUTO_INCREMENT*), IDs do ponto de partida e do ponto de chegada escolhido pelo usuário ao selecionar uma rota no *software*, data de inserção do registro ao banco, ID do usuário, ID da tabela referente ao tipo de viagem, tipo de aplicação (que pode ser “mob” se o registro for proveniente do aplicativo móvel SIMA Mobilidade Multimodal, ou “web” se for dado a partir de um sistema *web* não especificado neste trabalho), latitude e longitude do usuário;

- **rota_favorita**: tabela que armazena a rota favorita selecionada pelo usuário no aplicativo SIMA Mobilidade Multimodal. Seus atributos são: ID do registro, IDs (chaves estrangeiras) do ponto de partida e do ponto de chegada escolhido pelo usuário ao selecionar uma rota no *software*, data de inserção do registro ao banco, data de atualização do registro, atributo "ativo" para identificar se aquela rota foi favoritada ou não, o atributo "número de atualizações" que serve como um contador para salvar a quantidade de vezes que um determinado usuário favorita (quando havia sido desfavoritada) uma mesma rota, ID do usuário, ID do tipo de viagem, estes dois últimos são chaves estrangeiras e fazem referência a tabela **usuario** e **tipo_viagem**, respectivamente; e
- **equipamento**: tabela que armazena dados referentes às estruturas dos dispositivos virtuais da Dojot. Suas colunas são: ID do registro, nome, codigo_identificacao (representa o *device_id* da Dojot), status de ativado, data de criação e atualização de registro e tipo (que pode ser ônibus ou barco).

A modelagem física do banco de dados a partir deste modelo lógico, foi gerada automaticamente pelo *software* MySql Workbench, utilizando a linguagem SQL padrão para o SGBD MySql. Este trabalho não apresenta a modelagem do banco de dados físico em SQL.

5.3.2 Banco de Dados Não Relacional

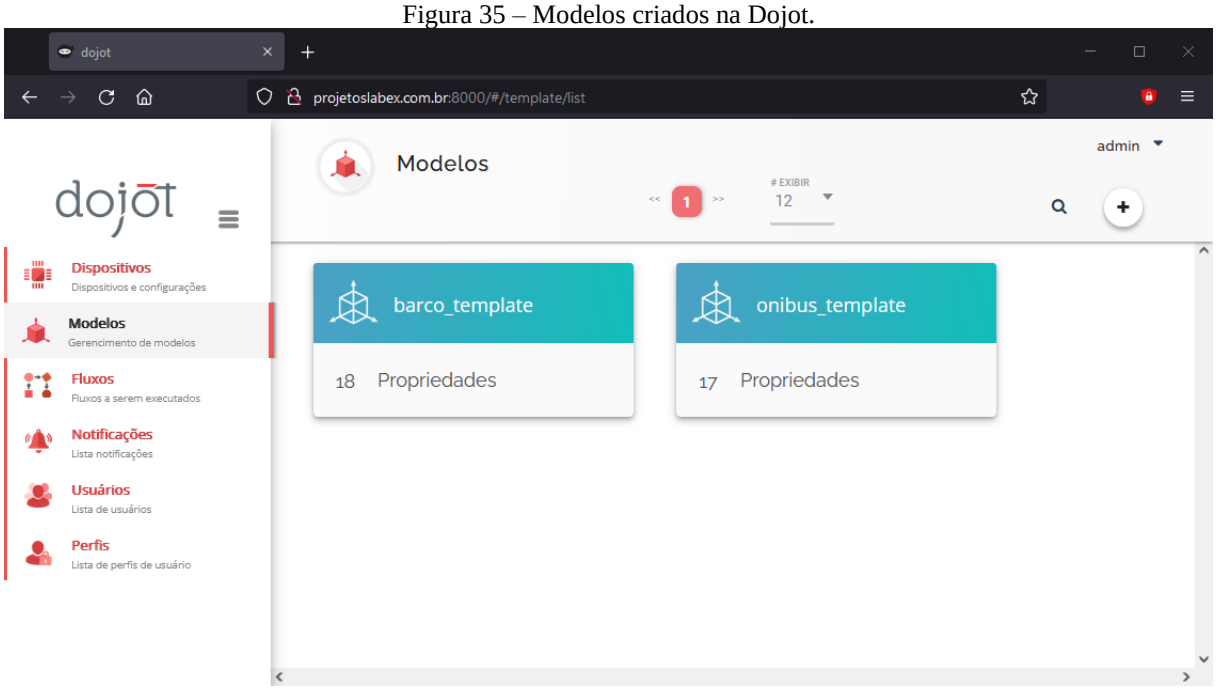
O banco de dados não relacional utilizado pelo *software* móvel SIMA Mobilidade Multimodal tem o objetivo de armazenar dados dos dispositivos IoT dos modais elétricos do campus, tais como geolocalização, temperatura e carga de baterias, temperatura interna e externa dos veículos, número de passageiros, entre outros.

A modelagem e armazenamento dos dados dos dispositivos dos modais, se deu através da plataforma Dojot (especificada na Seção 3.5 do Capítulo 3 deste trabalho). A Dojot por sua vez, proporciona uma interface gráfica apropriada para a criação de modelos e dispositivos virtuais que possibilitam a modelagem e o armazenamento de dados de dispositivos reais em um banco de dados não relacional específico, neste caso, ela encapsula o banco de dados MogoDB.

Antes da modelagem, foi necessária a instalação da plataforma Dojot em um servidor *online* com o sistema operacional Ubuntu 20.04, *Random-Access Memory* (RAM) de 4 GB, e configuração de 4 GB de memória *Swap*. O *host* de acesso à Dojot neste servidor é http://www.projetoslabex.com.br:8000.

Para o projeto SIMA, foi definido inicialmente que os modelos de dispositivos virtuais da Dojot fossem com 18 propriedades (atributos) para o barco e 17 propriedades para o ônibus. A Figura 35 mostra o *screenshot* do painel de modelos da Dojot, com os dois modelos de modais definidos, os quais foram intitulados de **barco_template** e **onibus_template**.

Figura 35 – Modelos criados na Dojot.



Fonte: elaborada pelo autor.

Das 17 propriedades definidas no **onibus_template** para o armazenamento de dados do modal ônibus, 4 (quatro) são utilizadas pelo aplicativo SIMA Mobilidade Multimodal, que são temperatura externa, temperatura interna, quantidade de usuários (passageiros) e localização. Toda a estrutura definida para o **onibus_template**, contendo nomes, IDs, tipos de dados, entre outros, está descrita no JSON ilustrado na Figura 36.

Na Figura 37 é ilustrada a estrutura (em formato JSON) do modelo (**barco_template**) definido para o armazenamento de dados do modal barco, contendo 4 principais propriedades utilizadas pelo SIMA Mobilidade Multimodal, das 18 definidas.

Figura 36 – Estrutura em JSON do modelo de ônibus criado na Dojot.


```
{
    "templates": [
    {
        "attrs": [
        {
            "id": 1,
            "label": "temperatura_interna",
            "static_value": "",
            "template_id": "1",
            "type": "dynamic",
            "value_type": "integer"
        },
        {
            "id": 2,
            "label": "temperatura_externa",
            "static_value": "",
            "template_id": "1",
            "type": "dynamic",
            "value_type": "integer"
        },
        {
            "id": 3,
            "label": "quantidade_usuarios",
            "static_value": "",
            "template_id": "1",
            "type": "dynamic",
            "value_type": "integer"
        },
        {
            "id": 4,
            "label": "localizacao",
            "static_value": "",
            "template_id": "1",
            "type": "dynamic",
            "value_type": "geo:point"
        }
        ],
        "id": 1,
        "label": "onibus_template"
    }
    ]
}
```


Fonte: elaborada pelo autor.

Figura 37 – Estrutura em JSON do modelo de barco criado na Dojot.


```
{
    "templates": [
    {
        "attrs": [
        {
            "id": 20,
            "label": "temperatura_interna",
            "static_value": "",
            "template_id": "2",
            "type": "dynamic",
            "value_type": "integer"
        },
        {
            "id": 21,
            "label": "temperatura_externa",
            "static_value": "",
            "template_id": "2",
            "type": "dynamic",
            "value_type": "integer"
        },
        {
            "id": 22,
            "label": "quantidade_usuarios",
            "static_value": "",
            "template_id": "2",
            "type": "dynamic",
            "value_type": "integer"
        },
        {
            "id": 23,
            "label": "localizacao",
            "static_value": "",
            "template_id": "2",
            "type": "dynamic",
            "value_type": "geo:point"
        }
        ],
        "id": 2,
        "label": "barco_template"
    }
    ]
}
```


Fonte: elaborada pelo autor.

Após os modelos de dispositivos definidos, foi necessário a definição dos dispositivos virtuais. Vale lembrar que cada dispositivo virtual criado na Dojot corresponde a um dispositivo

físico no campus, ou seja, cada dispositivo virtual é responsável por armazenar dados do seu dispositivo físico correspondente. É importante ressaltar que cada dispositivo físico é responsável por enviar à Dojot, dados de sensores instalados no modal.

Para realização de testes, foram definidos três dispositivos virtuais na Dojot, dois para o ônibus circular elétrico, e um para o barco elétrico, conforme ilustrado no *screenshot* da Dojot na Figura 38.

Figura 38 – Três dispositivos virtuais para os testes.

Fonte: elaborada pelo autor.

A estrutura em formato JSON dos três dispositivos criados para testes do SIMA Mobilidade Multimodal está ilustrada na Figura 39.

Na Figura 39 é possível ver que cada dispositivo virtual criado corresponde ao seu modelo específico através do ID, que nesse caso o ID do modelo (*template*) do ônibus é 1, e do barco é o 2, conforme ilustrado na Figura 36 e Figura 37.

Figura 39 – Estrutura JSON dos três dispositivos criados na Dojot para testes do SIMA Mobilidade Multimodal.


```
{
    "devices": [
    {
        "attrs": [],
        "id": "648337",
        "label": "circular_eletrico_2",
        "templates": [
            1
        ]
    },
    {
        "attrs": [],
        "id": "6765a0",
        "label": "circular_eletrico_1",
        "templates": [
            1
        ]
    },
    {
        "attrs": [],
        "id": "e623f8",
        "label": "barco_eletrico_1",
        "templates": [
            2
        ]
    }
    ]
}
```


Fonte: elaborada pelo autor.

5.4 API REST

Esta Seção apresenta a API REST projetada para ser consumida pelo aplicativo móvel SIMA Mobilidade Multimodal e outros *softwares* do projeto SIMA, ela utiliza um conjunto de boas práticas de requisições HTTP que possibilitam a comunicação entre uma aplicação *back-end* com diferentes aplicações *back-end* e *front-end*.

Esta API REST nada mais é do que um *software back-end*, ou seja, que está em um servidor *web*, e com ela é possível entregar a diferentes aplicações, dados de modais e usuários provenientes do banco de dados relacional do projeto (os detalhes deste banco de dados foram especificados na Subseção 5.3.1).

Assim como em outros tipos de *software*, no desenvolvimento de uma API, é necessário que seus requisitos estejam bem definidos, para que se possa alcançar seu principal objetivo, que é entregar recursos para as demais aplicações (neste caso para o aplicativo SIMA Mobilidade Multimodal). A Subseção 5.4.1 apresenta os requisitos funcionais e não funcionais definidos para API REST desenvolvida.

As especificações técnicas de desenvolvimento da API REST são apresentadas na Subseção 5.4.2.

5.4.1 Requisitos da API

Os requisitos funcionais e não funcionais foram definidos pela equipe de desenvolvimento com base nos requisitos apresentados na Seção 5.2 e no banco de dados relacional apresentado na Subseção 5.3.1. O Quadro 8 apresenta 12 requisitos funcionais da API REST, os quais estão organizados com o identificador RF#, e possuem a descrição detalhada sobre cada requisito implementado para o atendimento do aplicativo SIMA Mobilidade Multimodal e seus usuários.

Quadro 8 – Requisitos funcionais da API REST.

RF#	Descrição
RF01	A API deve permitir o cadastro do usuário através de nome, e-mail e senha.
RF02	A API deve permitir a autenticação do usuário, através do e-mail e senha.
RF03	A API deve permitir a redefinição de senha de acesso do usuário.
RF04	A API deve permitir a alteração dos dados do usuário.
RF05	A API deve permitir a consulta dos dados de um usuário autenticado.
RF06	A API deve permitir que uma rota do usuário seja favoritada.
RF07	A API deve permitir a consulta de todas a rotas favoritas de um usuário.
RF08	A API deve permitir a verificação de existência de uma rota favorita.
RF09	A API deve permitir a desativação de uma rota favorita.
RF10	A API deve permitir a gravação do *log* de uso de uma determinada rota escolhida pelo usuário.
RF11	A API deve permitir a consulta de todos os equipamentos.
RF12	A API deve permitir a consulta dos pontos de prédios do campus.

Fonte: elaborado pelo autor.

O Quadro 9 apresenta 11 requisitos não funcionais desta API REST, eles estão organizados pelo identificador RNF#. Estes requisitos atendem as necessidades de uso da API em relação à organização, desempenho, segurança da informação, entre outros.

Quadro 9 – Requisitos não funcionais da API REST.

RNF#	Descrição
RNF01	A API deve funcionar em um ambiente Linux, com servidor HTTP Apache e PHP 8 instalados.
RNF02	O API deve possuir mecanismos de segurança para autenticação de usuário e controle de acesso a conteúdo.
RNF03	A API deve utilizar os princípios e regras de arquitetura REST.
RNF04	A API deve ter alta disponibilidade (24 horas/7 dias por semana), para que as informações sejam inseridas e acessadas a qualquer tempo.
RNF05	A API deve executar as tarefas em tempo hábil (respostas às requisições em segundos).
RNF06	A API deve proteger os dados dos usuários da base de dados, assim como criptografar as senhas e ter mecanismos *anti-hacking*.
RNF07	A API deve ser implementada utilizando a linguagem de programação PHP versão 8.

RNF08	A API deve utilizar rotas de API de forma a tornar as *Uniform Resource Locator* (URL) amigáveis.
RNF09	A API deve ser implementada utilizando boas práticas de padrões de projeto, que visem a facilidade de testes, reutilização de código, separação de responsabilidades, manutenção e extensibilidade da API.
RNF10	A API deve ter versionamento de código utilizando o Git.
RNF11	A API deve utilizar JSON como resposta às requisições.

Fonte: elaborado pelo autor.

5.4.2 Especificações da API

Esta Subseção apresenta as ferramentas e configurações utilizadas para o desenvolvimento da API.

Diante dos requisitos e da modelagem banco de dados, foi iniciado o processo de implementação e implantação da API. Utilizou-se a linguagem de programação PHP (versão 8.0) para criação da lógica de negócios e instruções de processamento da API. O formato de dados utilizado para respostas às requisições da API foi o padrão JSON. Algumas páginas *front-end* necessárias para apresentar a confirmação de cadastro e alteração de senha do usuário, foram desenvolvidas utilizando HTML, CSS e Javascript. Para o armazenamento dos dados, foi adotado o SGBD MySql.

A API foi implantada em um servidor Ubuntu 20.04, com 4 GB de memória RAM, Apache HTTP *Server*, PHP 8.0 e MySql. O *host* definido para implantação do código-fonte da API REST foi: http://www.projetoslabex.com.br/sima-app-api.

As Subseções 5.4.2.1 a 5.4.2.16 apresentam as especificações técnicas de cada função desenvolvida para contemplar os requisitos definidos para esta API.

5.4.2.1 Cadastro de Usuário

Para a funcionalidade de cadastro de usuário, foram definidas as seguintes especificações da requisição:

- Método HTTP: POST;
- Sem autenticação;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/cadastrar;
- Parâmetros: nome, email, senha.

A Figura 40 apresenta um exemplo de resposta bem-sucedida para a URI da requisição desta funcionalidade.

Figura 40 – Resposta de uma requisição bem-sucedida de cadastro de usuário.


```
{
    "message": "Falta pouco! Enviamos o link de confirmação de cadastro para seu
 e-amil. Verifique na caixa de entrada ou de spam de seu e-mail."
}
```


Fonte: elaborada pelo autor.

5.4.2.2 Autenticação do Usuário

Na função de autenticação do usuário, é retornado como resultado: um *token* do tipo *JSON Web Token* (JWT), nome, e-mail, foto e alguns outros atributos do usuário. As especificações da requisição desta função são:

- Método HTTP: POST;
- Sem autenticação;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/autenticar;
- Parâmetros: email, senha.

A Figura 41 ilustra o resultado de um exemplo de requisição bem-sucedida para a rota desta funcionalidade.

Figura 41 – Resposta de uma requisição bem-sucedida de autenticação de usuário.


```
{
    "nome": "Joiner",
    "email": "joinerssa@gmail.com",
    "mobilidade": 0,
    "escolaridade": "P",
    "profissao": "Programador",
    "status": "confirmado",
    "foto": "5282717680901e2d2c8b623d87b417bc.png",
    "saldo_pontos": null,
    "tipo_usuario_id": 1,
    "jwt": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbWFpbCI6ImpvaW5lcnNzYUBnbWFp
bC5jb20ifQ.oui4Mg7EanAFyAHfUzHd4_0qsUqMVEkTTCy4DDNe5FI"
}
```


Fonte: elaborada pelo autor.

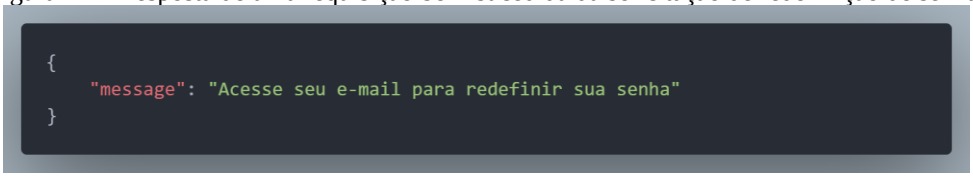
5.4.2.3 Solicitação de Redefinição de Senha do Usuário

Esta função faz a solicitação de redefinição de senha do usuário enviando um link de redefinição para o e-mail do usuário cadastrado. Suas especificações são:

- Método HTTP: POST;
- Sem autenticação;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/recuperar;
- Parâmetros: email.

Um exemplo de resultado de requisição bem-sucedida à API, é ilustrado na Figura 42.

Figura 42 – Resposta de uma requisição bem-sucedida da solicitação de redefinição de senha.



Fonte: elaborada pelo autor.

5.4.2.4 Redefinição de Senha do Usuário

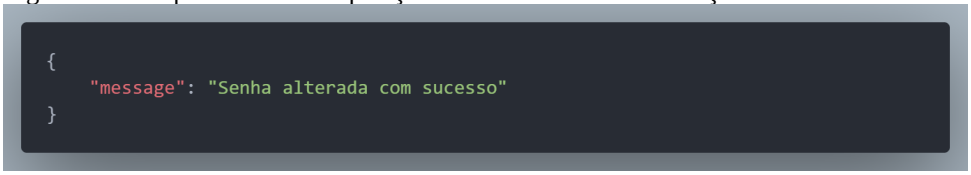
Esta funcionalidade redefine a senha do usuário a partir de dados de formulário. As especificações de requisição desta função são:

- Método HTTP: POST;
- Sem autenticação;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/recuperar/resetar;
- Parâmetros: code, email, senha, senha_repetida.

Onde os parâmetros são respectivamente, o código de verificação presente no *link* enviado para o e-mail do usuário; o e-mail do usuário; sua nova senha; e a repetição da nova senha informada através de um *Form URL Encoded*.

Um exemplo de resultado de requisição bem-sucedida é ilustrado na Figura 43.

Figura 43 – Resposta de uma requisição bem-sucedida de redefinição de senha do usuário.



Fonte: elaborada pelo autor.

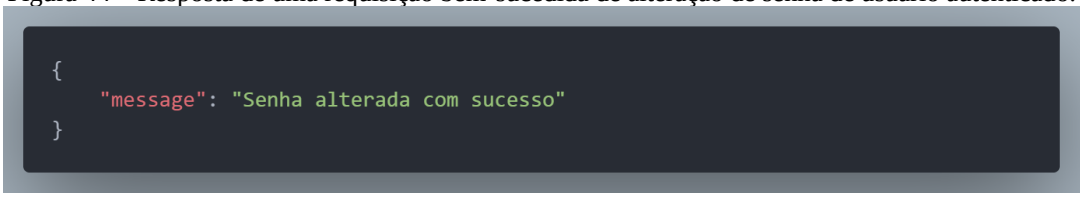
5.4.2.5 Alteração de Senha de Usuário Autenticado

Esta função altera a senha de um usuário já autenticado, através do uso de seu *token* JWT. Seus detalhes técnicos são:

- Método HTTP: PUT;
- Autenticação: Bearer Token;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/usuario/senha;
- Parâmetros: senha, nova_senha, senha_repetida.

Um exemplo de resultado de requisição bem-sucedida é ilustrado na Figura 44.

Figura 44 – Resposta de uma requisição bem-sucedida de alteração de senha de usuário autenticado.



Fonte: elaborada pelo autor.

5.4.2.6 Alteração de Dados do Usuário Autenticado

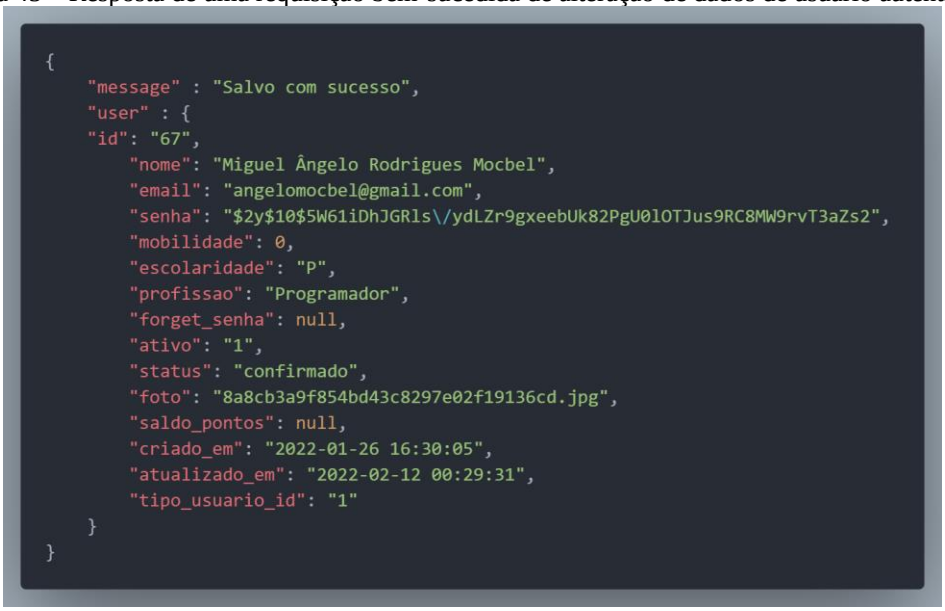
Função que altera dados referente ao usuário autenticado com seu token, seus detalhes técnicos são:

- Método HTTP: POST;
- Autenticação: Bearer Token;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/usuario/editar;
- Parâmetros: nome, mobilidade, escolaridade, profissao e foto (arquivo de imagem *Joint Photographics Experts Group* (JPEG) ou *Portable Network Graphics* (PNG)).

Os parâmetros desta requisição são passados através de um formulário do tipo *multipart/form-data*, para que seja possível enviar arquivos de imagem.

A Figura 45 ilustra o resultado em JSON de uma requisição bem-sucedida feita à API.

Figura 45 – Resposta de uma requisição bem-sucedida de alteração de dados de usuário autenticado.



```
{
    "message" : "Salvo com sucesso",
    "user" : {
    "id": "67",
        "nome": "Miguel Ângelo Rodrigues Mocbel",
        "email": "angelomocbel@gmail.com",
        "senha": "$2y$10$5W61iDhJGRls\/ydLZr9gxeebUk82PgU0lOTJus9RC8MW9rvT3aZs2",
        "mobilidade": 0,
        "escolaridade": "P",
        "profissao": "Programador",
        "forget_senha": null,
        "ativo": "1",
        "status": "confirmado",
        "foto": "8a8cb3a9f854bd43c8297e02f19136cd.jpg",
        "saldo_pontos": null,
        "criado_em": "2022-01-26 16:30:05",
        "atualizado_em": "2022-02-12 00:29:31",
        "tipo_usuario_id": "1"
    }
}
```


Fonte: elaborada pelo autor.

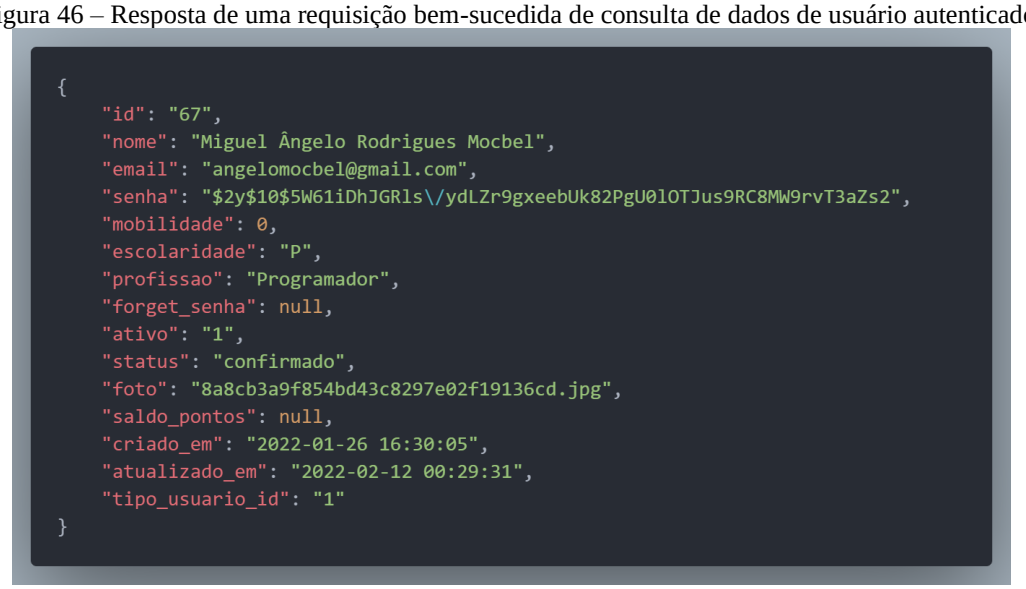
5.4.2.7 Consulta de Dados do Usuário Autenticado

Esta função faz a consulta de dados de um determinado usuário que possui o token JWT de autenticação. Os detalhes técnicos da requisição desta função são:

- Método HTTP: GET;
- Autenticação: Bearer Token;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/usuario.

Um exemplo de resultado de uma requisição bem-sucedida é ilustrado na Figura 46.

Figura 46 – Resposta de uma requisição bem-sucedida de consulta de dados de usuário autenticado.



```
{
    "id": "67",
    "nome": "Miguel Ângelo Rodrigues Mocbel",
    "email": "angelomocbel@gmail.com",
    "senha": "$2y$10$5W61iDhJGRls\/ydLZr9gxeebUk82PgU0lOTJus9RC8MW9rvT3aZs2",
    "mobilidade": 0,
    "escolaridade": "P",
    "profissao": "Programador",
    "forget_senha": null,
    "ativo": "1",
    "status": "confirmado",
    "foto": "8a8cb3a9f854bd43c8297e02f19136cd.jpg",
    "saldo_pontos": null,
    "criado_em": "2022-01-26 16:30:05",
    "atualizado_em": "2022-02-12 00:29:31",
    "tipo_usuario_id": "1"
}
```


Fonte: elaborada pelo autor.

5.4.2.8 Consulta uma Imagem a Partir do Nome do Arquivo

Esta rota retorna uma imagem a partir do nome do arquivo, ela é usada para recuperação da foto de perfil do usuário. Suas especificações são:

- Método HTTP: GET;
- Sem autenticação;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/arquivo/**NOME-DO-ARQUIVO**.

Onde, **NOME-DO-ARQUIVO** é o atributo *filename*. O resultado da requisição é um arquivo do tipo JPEG ou PNG.

5.4.2.9 Favorita uma Rota de Modal

Esta função da API possibilita que um usuário selecione uma rota de modal como sua favorita. As especificações técnicas são:

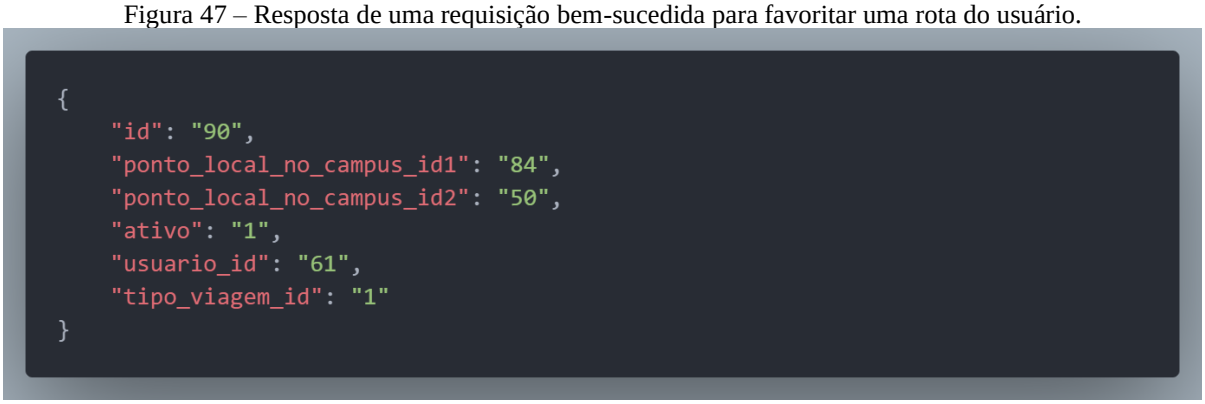
- Método HTTP: POST;
- Autenticação: Bearer Token;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/rota-favorita;
- Parâmetros: ponto_local_no_campus_id1, ponto_local_no_campus_id2, tipo_viagem_id.

Onde os parâmetros representam respectivamente o ID do ponto de partida da rota de modal, o ID do ponto de chegada da rota de modal, e o ID do tipo de viagem.

É importante ressaltar que a palavra “rota” tem dois significados diferentes nesta dissertação. No contexto de desenvolvimento de *software web* e APIs, o termo rota se refere a um *link* ou URI por onde se acessa recursos de dados. Já no contexto de mobilidade urbana, uma rota se refere a um percurso que um modal ou uma pessoa pode fazer.

A Figura 47 ilustra o resultado de uma requisição bem-sucedida quando o usuário solicita que uma rota de modal seja favoritada.

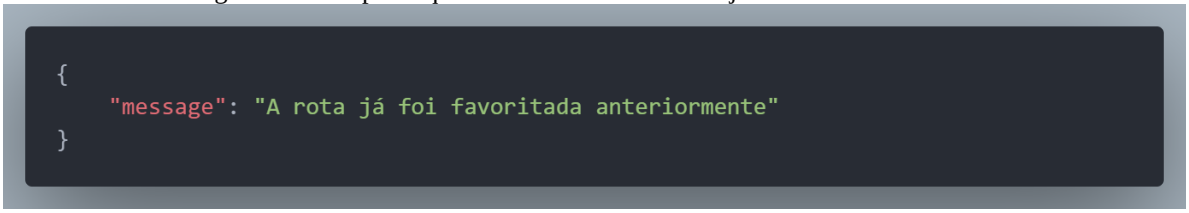
Figura 47 – Resposta de uma requisição bem-sucedida para favoritar uma rota do usuário.



Fonte: elaborada pelo autor.

A Figura 48 ilustra a mensagem de retorno da API caso a rota de modal já esteja favoritada.

Figura 48 – Resposta quando uma rota do usuário já se encontra favoritada.



Fonte: elaborada pelo autor.

5.4.2.10 Consulta Todas as Rotas de Modais Favoritas do Usuário

Função que traz uma lista de todas as rotas favoritas de um determinado usuário autenticado. Seus detalhes são:

- Método HTTP: GET;
- Autenticação: Bearer Token;
- Rota (URI): http://www.projetoslabex.com.br/sima-app-api/rota-favorita.

O resultado de uma requisição de exemplo bem-sucedida é ilustrado na Figura 49.

Figura 49 – Resultado de uma requisição bem-sucedida para consulta de todas as rotas de modais favoritas do usuário

```
[
    {
        "id": "58",
        "tipo_viagem": "Ônibus",
        "nome_ponto_partida": "Bloco B",
        "latitude_ponto_partida": "-1.474231",
        "longitude_ponto_partida": "-48.450907",
        "nome_ponto_chegada": "Academia Amazônia - FAGEO Faculdade de Geologia",
        "latitude_ponto_chegada": "-1.474267",
        "longitude_ponto_chegada": "-48.457892"
    },
    {
        "id": "60",
        "tipo_viagem": "A pé",
        "nome_ponto_partida": "Bloco de Salas K-L",
        "latitude_ponto_partida": "-1.4763356",
        "longitude_ponto_partida": "-48.4572428",
        "nome_ponto_chegada": "FAAD - Faculdade de Administração",
        "latitude_ponto_chegada": "-1.473041",
        "longitude_ponto_chegada": "-48.449279"
    },
    {
        "id": "64",
        "tipo_viagem": "Barco",
        "nome_ponto_partida": "Academia Amazônia - FAGEO Faculdade de Geologia",
        "latitude_ponto_partida": "-1.474267",
        "longitude_ponto_partida": "-48.457892",
        "nome_ponto_chegada": "CAME - Centro Acadêmico de Engenharia Mecânica",
        "latitude_ponto_chegada": "-1.473835",
        "longitude_ponto_chegada": "-48.452925"
    }
]
```


Fonte: elaborada pelo autor.

5.4.2.11 Verifica a Existência de uma Rota de Modal Favorita

É possível verificar a existência de uma rota de modal favorita de um usuário. Para isso, são usadas as seguintes especificações da requisição:

- Método HTTP: GET;
- Autenticação: Bearer Token;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/rota-favorita/**ID-DO-PONTO-DE-PARTIDA/ID-DO-PONTO-DE-CHEGADA/ID-DO-TIPO-DE-VIAGEM**.

Onde, **ID-DO-PONTO-DE-PARTIDA** é o atributo **ponto_local_no_campus_id1**, **ID-DO-PONTO-DE-CHEGADA** é o atributo **ponto_local_no_campus_id2**, e **ID-DO-TIPO-DE-VIAGEM** é o atributo **tipo_viagem_id**.

Supondo o que o *link* de uma requisição seja http://www.projetoslabex.com.br/sima-app-api/rota-favorita/84/50/1, e que a rota de modal exista para um usuário autenticado, o resultado seria conforme ilustrado na Figura 50.

Figura 50 – Resposta da requisição de verificação de existência de uma rota de modal favorita.


```
{
    "id": "90",
    "ponto_local_no_campus_id1": "84",
    "ponto_local_no_campus_id2": "50",
    "criado_em": "2021-12-28 08:12:09",
    "atualizado_em": "2021-12-28 08:12:09",
    "ativo": "1",
    "numero_de_atualizacoes": "1",
    "usuario_id": "61",
    "tipo_viagem_id": "1"
}
```


Fonte: elaborada pelo autor.

5.4.2.12 Desativa uma Rota Favoritada pelo Usuário

Esta função da API desabilita uma rota de modal favorita de um usuário. Os detalhes técnicos desta requisição são:

- Método HTTP: PUT;
- Autenticação: Bearer Token;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/rota-favorita/desativar/**ID-DA-ROTA-FAVORITA**.

Onde **ID-DA-ROTA-FAVORITA** é o ID da rota de modal que está favoritada na tabela *rota_favorita* do banco de dados.

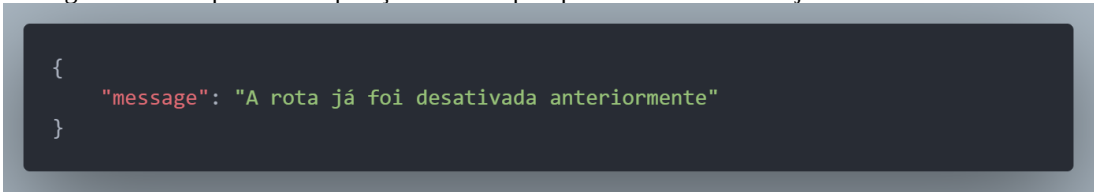
Supondo uma requisição cuja seu *link* seja http://www.projetoslabex.com.br/sima-app-api/rota-favorita/desativar/75, seu resultado é um JSON com uma mensagem informando se a rota de modal foi ou não favoritada, conforme ilustrado na Figura 51 e Figura 52.

Figura 51 – Resposta da requisição de exemplo bem-sucedida para desativar uma rota favorita de um usuário.


```
{
    "message": "Favorita desativada"
}
```


Fonte: elaborada pelo autor.

Figura 52 – Resposta da requisição de exemplo quando a rota favorita já havia sido desativada.



Fonte: elaborada pelo autor.

5.4.2.13 Registra a Atividade do Usuário ao Escolher uma Rota de Modal

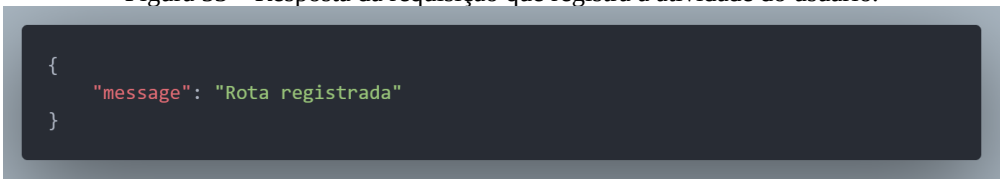
Esta funcionalidade registra, através da API, a atividade de um usuário ao escolher uma rota de modal no *software* SIMA Mobilidade Multimodal ou qualquer outra aplicação. Estes dados são gravados com o objetivo de gerar conhecimento a partir de rotas de modais mais utilizadas pelo usuário. As especificações da requisição são:

- Método HTTP: POST;
- Autenticação: Bearer Token;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/registro-rota-usuario;
- Parâmetros: ponto_local_no_campus_id1, ponto_local_no_campus_id2, tipo_viagem_id, aplicacao, latitude_usuario e longitude_usuario.

Onde os parâmetros representam respectivamente o ID do ponto de partida, o ID do ponto de chegada, o ID do tipo de viagem, o tipo de aplicação (*mob* ou *web*), e por fim, a latitude e longitude do usuário, que são campos não obrigatórios.

A Figura 53 ilustra um exemplo de resultado de uma requisição bem-sucedida.

Figura 53 – Resposta da requisição que registra a atividade do usuário.



Fonte: elaborada pelo autor.

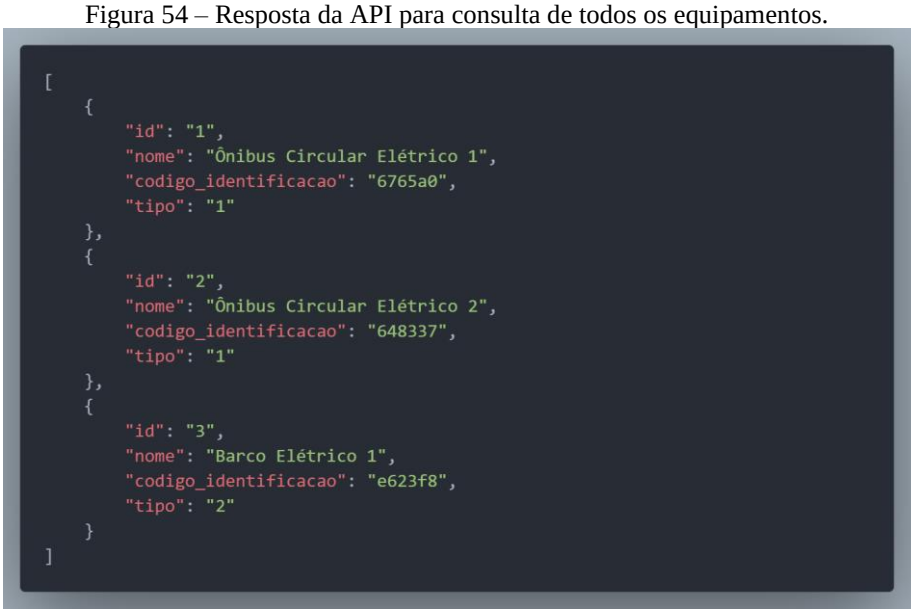
5.4.2.14 Consulta Todos os Equipamentos

Esta função consulta todos os equipamentos presentes na base de dados. Vale lembrar que estes equipamentos são dados de referências para os dispositivos virtuais de cada modal da Dojot. A requisição para esta funcionalidade é composta pelas seguintes especificações:

- Método HTTP: GET;
- Sem autenticação;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/equipamentos.

Vale destacar que esta rota de API não precisa de autenticação por *Bearer Token*, pois, é através dela que um usuário não autenticado poderá consultar o *codigo_identificacao* que representa o *device_id* dos dispositivos da Dojot. Esta é uma observação importante para lógica de negócios do SIMA Mobilidade Multimodal. Um exemplo de resultado de uma requisição bem-sucedida é ilustrado na Figura 54.

Figura 54 – Resposta da API para consulta de todos os equipamentos.



```
[
    {
        "id": "1",
        "nome": "Ônibus Circular Elétrico 1",
        "codigo_identificacao": "6765a0",
        "tipo": "1"
    },
    {
        "id": "2",
        "nome": "Ônibus Circular Elétrico 2",
        "codigo_identificacao": "648337",
        "tipo": "1"
    },
    {
        "id": "3",
        "nome": "Barco Elétrico 1",
        "codigo_identificacao": "e623f8",
        "tipo": "2"
    }
]
```


Fonte: elaborada pelo autor.

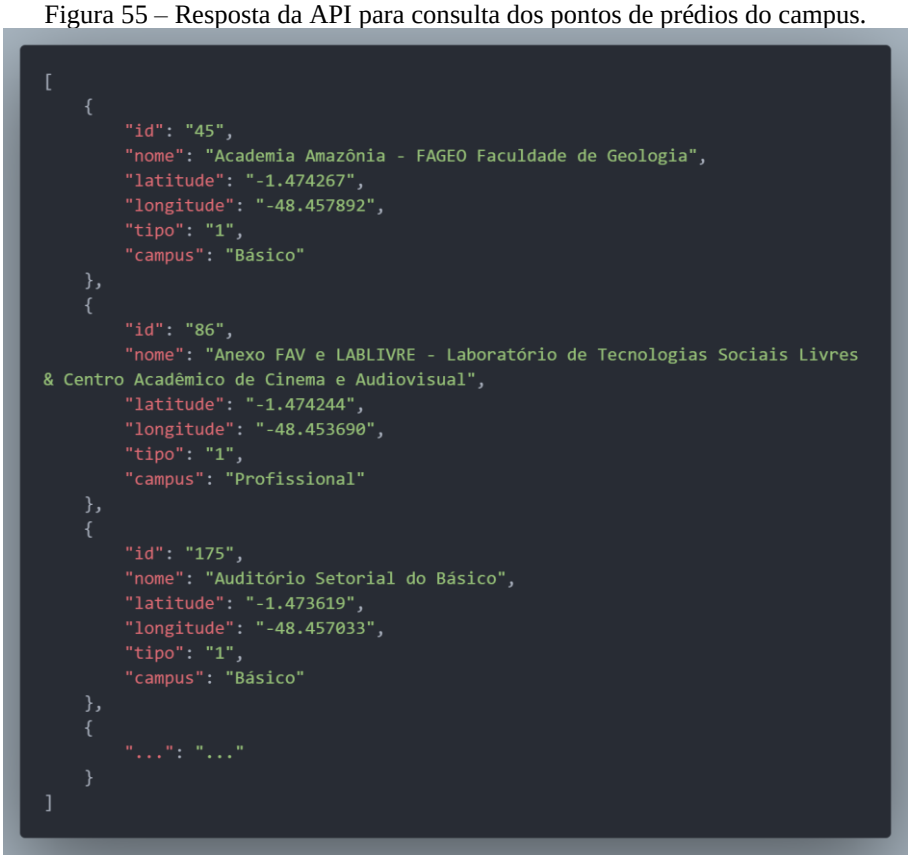
5.4.2.15 Consulta os Pontos de Prédios do Campus

Esta função faz a consulta somente dos pontos de prédios do campus. Estes são referentes à tabela *ponto_local_no_campus* do banco de dados relacional deste projeto. A requisição tem as seguintes especificações:

- Método HTTP: GET;
- Autenticação: Bearer Token;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/ponto-local-no-campus/predios.

A Figura 55 ilustra um exemplo de resultado de uma requisição bem-sucedida. Neste resultado, todos os dados estão ordenados pelo atributo **nome**.

Figura 55 – Resposta da API para consulta dos pontos de prédios do campus.



```
[
    {
        "id": "45",
        "nome": "Academia Amazônia - FAGEO Faculdade de Geologia",
        "latitude": "-1.474267",
        "longitude": "-48.457892",
        "tipo": "1",
        "campus": "Básico"
    },
    {
        "id": "86",
        "nome": "Anexo FAV e LABLIVRE - Laboratório de Tecnologias Sociais Livres
& Centro Acadêmico de Cinema e Audiovisual",
        "latitude": "-1.474244",
        "longitude": "-48.453690",
        "tipo": "1",
        "campus": "Profissional"
    },
    {
        "id": "175",
        "nome": "Auditório Setorial do Básico",
        "latitude": "-1.473619",
        "longitude": "-48.457033",
        "tipo": "1",
        "campus": "Básico"
    },
    {
        "...": "..."
    }
]
```


Fonte: elaborada pelo autor.

5.4.2.16 Consulta Todos os Pontos de Locais no Campus

Esta função consulta todos os pontos de locais no campus, tanto prédios quanto pontos de ônibus e barco. Estes pontos são referentes à tabela *ponto_local_no_campus* do banco de dados relacional deste projeto. Esta requisição tem as seguintes especificações:

- Método HTTP: GET;
- Autenticação: Bearer Token;
- Rota de API (URI): http://www.projetoslabex.com.br/sima-app-api/ponto-local-no-campus/todos.

A Figura 56 ilustra o resultado de uma requisição bem-sucedida. Neste resultado, os dados apresentados estão ordenados pelo atributo **nome**.

Figura 56 – Resposta da API para consulta de todos os tipos pontos de locais no campus.

```
[
    {
        "id": "45",
        "nome": "Academia Amazônia - FAGEO Faculdade de Geologia",
        "latitude": "-1.474267",
        "longitude": "-48.457892",
        "tipo": "1",
        "campus": "Básico"
    },
    {
        "id": "86",
        "nome": "Anexo FAV e LABLIVRE - Laboratório de Tecnologias Sociais Livres
& Centro Acadêmico de Cinema e Audiovisual",
        "latitude": "-1.474244",
        "longitude": "-48.453690",
        "tipo": "1",
        "campus": "Profissional"
    },
    {
        "id": "88",
        "nome": "Atelier de Artes",
        "latitude": "-1.473879",
        "longitude": "-48.453797",
        "tipo": "1",
        "campus": "Profissional"
    },
    {
        "id": "175",
        "nome": "Auditório Setorial do Básico",
        "latitude": "-1.473619",
        "longitude": "-48.457033",
        "tipo": "1",
        "campus": "Básico"
    },
    {
        "...": "..."
    }
]
```


Fonte: elaborada pelo autor.

5.5 Cenários de Teste e Resultados

Esta Seção apresenta o resultado da implementação do aplicativo SIMA Mobilidade Multimodal através de suas telas, os cenários de teste e os resultados de testes reais e por simulação.

A implementação do aplicativo iniciou após as fases de levantamento de requisitos, modelagem do banco de dados e desenvolvimento da API REST. O Java foi a linguagem de programação utilizada para codificação da lógica de negócios e instruções de processamento do aplicativo, enquanto que a linguagem XML foi utilizada para criação dos *layouts* da interface gráfica do *software*. Os dados geográficos utilizados no aplicativo, foram coletados do OSM, através da biblioteca[5] osmdroid[6].

[5] Estruturalmente, uma biblioteca Android é igual a um módulo de app Android. Ela pode conter tudo o que é necessário para criar um app, inclusive código-fonte, arquivos de recursos e um manifesto do Android.

[6] Biblioteca utilizada para criação de mapas em aplicativos Android com dados do OSM.

O consumo de dados de dispositivos IoT, referente aos modais ônibus e barco, pelo SIMA Mobilidade Multimodal, se deu através da API da Dojot, cujo o *host* é http://www.projetoslabex.com.br:8000.

Além dos recursos consumidos da API da Dojot e da API REST (apresentada na Seção 5.4), o aplicativo SIMA Mobilidade Multimodal consome recursos de uma API proposta por Silva *et al.* (2020). Esta API disponibiliza informações de rotas dos ônibus e barco do *smart campus* da UFPA, referente ao projeto SIMA. Estas informações são trajetórias de modais, horários de chegada e distâncias entre um determinado modal e seus pontos de parada.

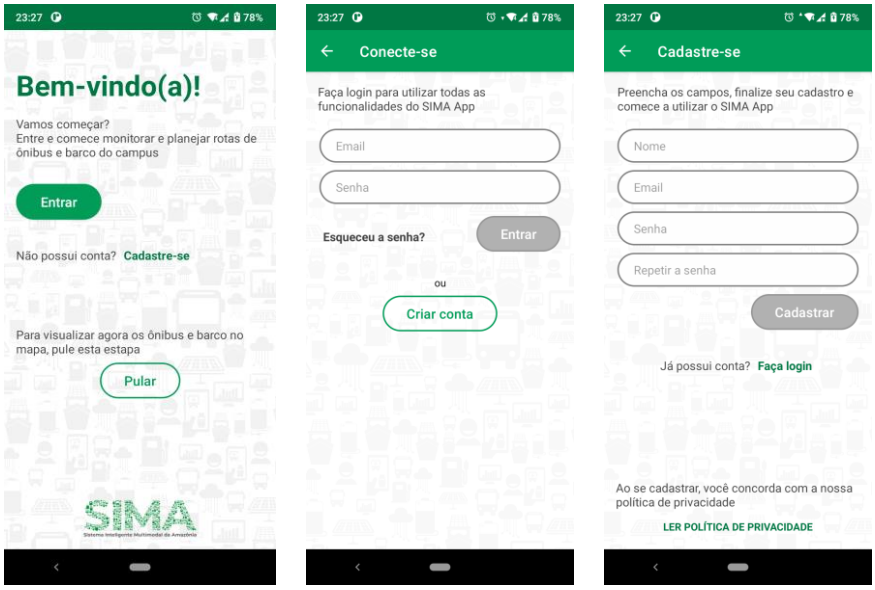
5.5.1 Telas do Aplicativo

Esta Subseção apresenta o resultado da implementação do aplicativo SIMA Mobilidade Multimodal através de suas telas de navegação. As telas apresentadas, representam fielmente o resultado final do produto de *software* produzido neste projeto. Além disso, elas foram elaboradas de acordo com os requisitos propostos na Seção 5.2 deste Capítulo.

A Figura 57 ilustra três primeiras telas que o usuário pode acessar no *software*. Ao ser utilizado pela primeira vez, o aplicativo inicia pela tela de apresentação ilustrado pela Figura 57(a), onde é possível o usuário ter acesso, através de botões, às telas de login, cadastro e tela de visualização dos modais.

Na tela de Login da aplicação, Figura 57(b), o usuário pode entrar com seu endereço de e-mail e senha. Na mesma tela, é possível acessar botões que levam à redefinição de senha e cadastro de usuário. A Figura 57(c) ilustra a área de cadastro, onde é possível que o usuário informe seu nome, e-mail e senha.

Figura 57 – Três primeiras telas do SIMA Mobilidade Multimodal. (a) Tela de apresentação. (b) Tela de *login*. (c) Tela de cadastro de usuário.

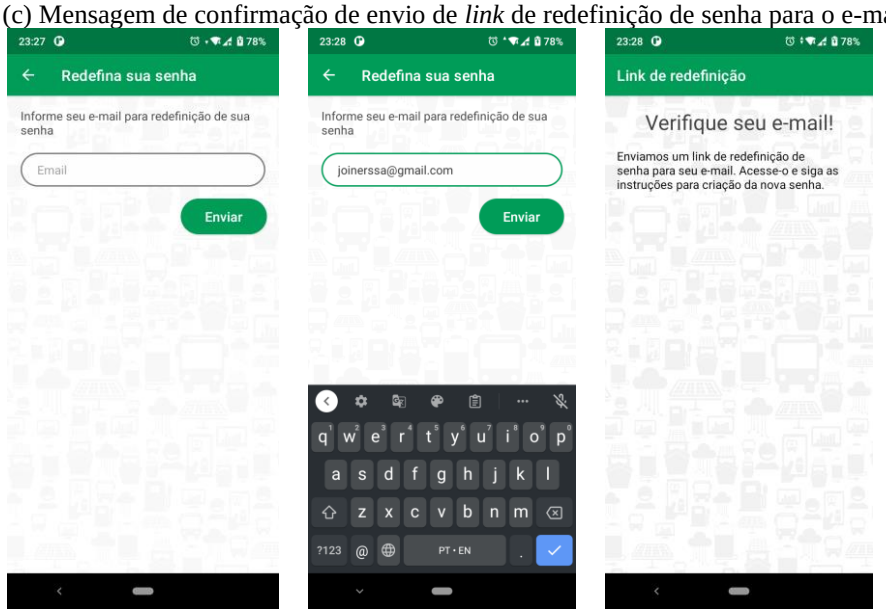


(a) (b) (c)

Fonte: elaboradas pelo autor.

Na utilização do aplicativo, caso o usuário pressione a opção “Esqueceu a senha?”, ilustrado na Figura 57(b), o mesmo será convidado a inserir seu e-mail para que o *software* o envie um *link* de redefinição de sua senha, conforme Figura 58.

Figura 58 – Solicitação de redefinição de senha. (a) Campo de inserção de e-mail. (b) Inserção de e-mail pelo teclado virtual. (c) Mensagem de confirmação de envio de *link* de redefinição de senha para o e-mail do usuário.



(a) (b) (c)

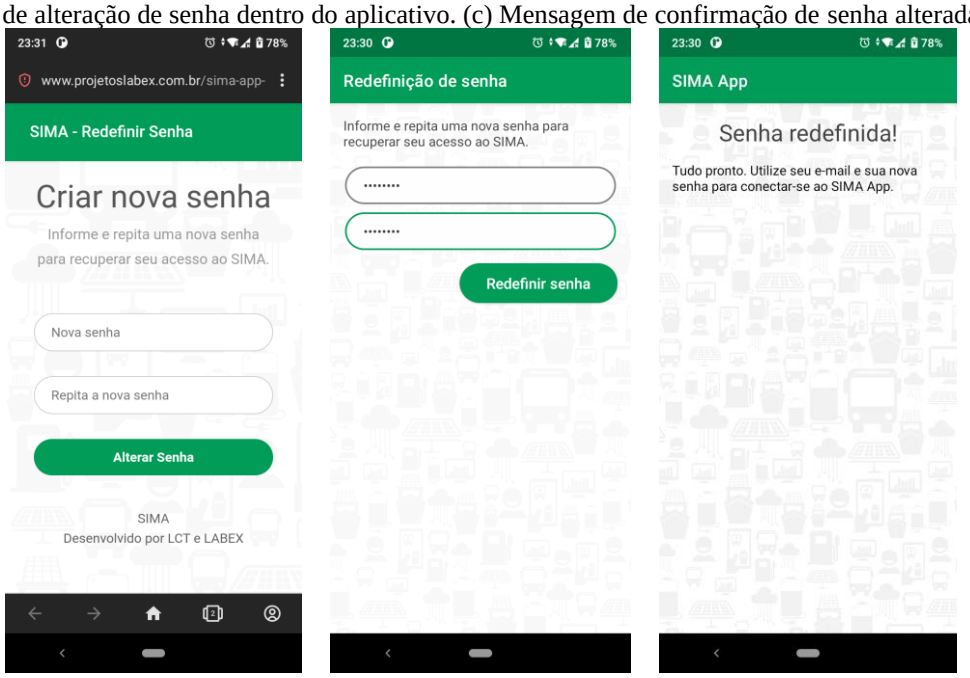
Fonte: elaboradas pelo autor.

A tela ilustrada na Figura 58(a) contém o campo de inserção de e-mail e o botão enviar. Já a Figura 58(b) ilustra um teste de inserção do e-mail de um usuário através do teclado virtual

do Android. A Figura 58(c) apresenta a tela de confirmação de envio de *link* de redefinição de senha.

O *link* de redefinição de senha enviado para caixa de entrada do e-mail do usuário, pode ser aberto a partir de um navegador *web* ou do SIMA Mobilidade Multimodal, conforme ilustrado na Figura 59, onde a Figura 59(a) é um exemplo da área de redefinição de senha acessada a partir de um navegador *web*, na Figura 59(b) o formulário de redefinição é dentro do SIMA Mobilidade Multimodal. A Figura 59(c) ilustra a tela de confirmação de alteração de senha realizada pelo usuário no SIMA Mobilidade Multimodal.

Figura 59 – Alteração de senha. (a) Formulário de alteração de senha acessado a partir de um navegador *web*. (b) Formulário de alteração de senha dentro do aplicativo. (c) Mensagem de confirmação de senha alterada.

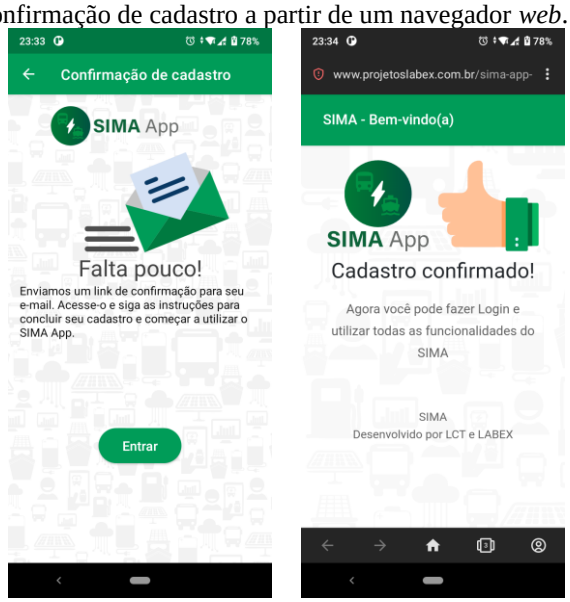


(a) (b) (c)

Fonte: elaboradas pelo autor.

Para o caso em que o usuário realize seu cadastro, conforme Figura 57(c), ao pressionar o botão “cadastrar”, um *link* será enviado para o e-mail informado no formulário, este serve para a confirmação do cadastro do usuário. A Figura 60(a) ilustra a tela de confirmação do envio do *link* para o e-mail do usuário. Já a Figura 60(b) mostra, no navegador, a página *web* de confirmação de cadastro após o usuário abrir o *link*.

Figura 60 – Telas referentes à função de cadastro de usuário. (a) Mensagem de envio de *link* para confirmação de cadastro. (b) Mensagem de confirmação de cadastro a partir de um navegador *web*.



(a) (b)

Fonte: elaboradas pelo autor.

Na Figura 61 é ilustrado o monitoramento dos modais ônibus e barco, e opões de escolha de rota. A Figura 61(a) ilustra a tela inicial do aplicativo, onde é possível visualizar os ônibus e barco em tempo real. Vale ressaltar que para visualizar os modais no mapa, não é obrigatório que o usuário esteja autenticado. As demais funcionalidades ilustradas na Figura 61 devem ser acessadas apenas por usuários autenticados.

A Figura 61(b) e Figura 61(c) ilustram a lista de prédios onde o usuário poderá selecionar o ponto de partida e chegada de rota. A Figura 61(d) ilustra dois pontos de rota selecionados.

Figura 61 – Telas de monitoramento de modais e escolha de rotas. (a) Monitoramento de modais e opções de escolha de rotas. (b) Pontos de partida. (c) Pontos de chegada. (d) Dois pontos selecionados pelo usuário.

(a) (b) (c) (d)

Fonte: elaboradas pelo autor.

A Figura 62 ilustra a visualização das rotas de modais e usuário no mapa. A Figura 62(a) ilustra três opções de rota para o usuário, estas são apresentadas quando o usuário pressiona o botão “Encontrar rota” ilustrado na Figura 61(d).

Figura 62 – Rotas de modais. (a) Opções de escolha. (b) Rota de ônibus. (c) Rota de barco. (d) Rota a pé.

(a) (b) (c) (d)

Fonte: elaboradas pelo autor.

Já a Figura 62(b) ilustra a rota do usuário a ser feita usando o ônibus. Nesta tela são apresentadas informações de distância e horário de chegada do ônibus na parada de ônibus mais próxima do usuário.

A Figura 62(c) ilustra a rota do usuário usando barco, esta tela apresenta informações de distância e horário de chegada. A tela da Figura 62(d) ilustra a opção de rota a pé do usuário,

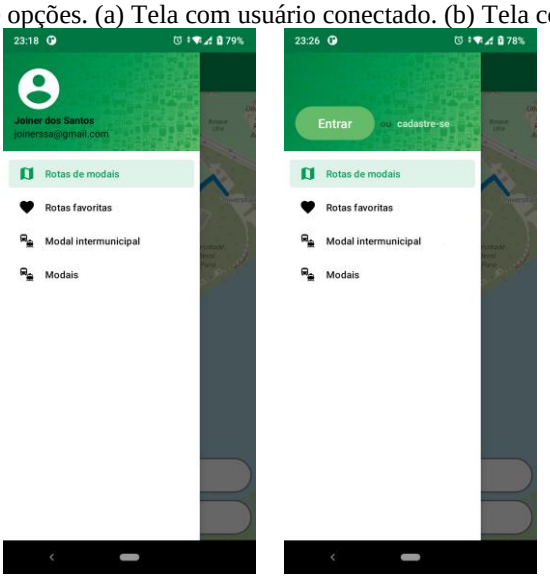
ela apresenta a distância do percurso em metros e o tempo de viagem no formato h:m:s, onde h, m e s são respectivamente horas, minutos e segundos.

A Figura 62(b), Figura 62(c) e Figura 62(d) apresentam um botão (no formato de coração) para o usuário favoritar a rota escolhida, assim como um botão para fechar informações da rota escolhida.

O *software* compõe um menu de opções lateral que contém botões que dão acesso às principais funcionalidades do aplicativo, conforme Figura 63.

A Figura 63(a) ilustra a tela do aplicativo com o menu lateral composto pela foto, nome e e-mail do usuário, a foto tem a função de um botão e serve para dar acesso à área de perfil do usuário. Os demais componentes do menu são os botões “rotas de modais”, “rotas favoritas”, “modal intermunicipal” e “modais”. Já a Figura 63(b) ilustra o menu lateral do SIMA App quando nenhum usuário estiver conectado. Esta tela é composta por mais dois botões, o de entrar e cadastrar-se, os demais componentes visuais são os mesmos da Figura 63(a), com exceção dos dados do usuário. Vale ressaltar que todas as funcionalidades dos botões do menu lateral são especificadas ao longo deste relatório.

Figura 63 – Menu lateral de opções. (a) Tela com usuário conectado. (b) Tela com usuário não conectado.



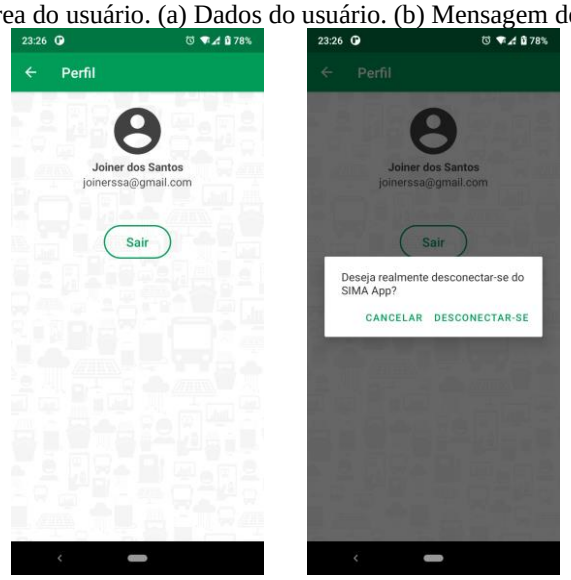
(a) (b)

Fonte: elaboradas pelo autor.

A partir do menu lateral, o usuário pode acessar sua área que contém informações do usuário, tais como nome, e-mail e foto, conforme Figura 64. Em sua área, o usuário pode desconectar-se do *software* pressionando o botão “Sair” ilustrado na Figura 64(a). O *software*

apresenta uma mensagem de confirmação antes de desconectar o usuário, conforme Figura 64(b).

Figura 64 – Área do usuário. (a) Dados do usuário. (b) Mensagem de confirmação.



(a) (b)

Fonte: elaboradas pelo autor.

Outra funcionalidade que o SIMA Mobilidade Multimodal disponibiliza para o usuário é a visualização das rotas favoritas. A Figura 65 ilustra um exemplo de lista de rotas favoritadas por um usuário, onde os itens da lista são compostos pelos tipos (Ônibus, Barco e A pé) e os nomes dos prédios de partida e chegada das rotas. Cada item da lista tem um evento de clique que leva o usuário à tela de rotas de modais, conforme já apresentado na Figura 62(b), Figura 62(c) e Figura 62(d).

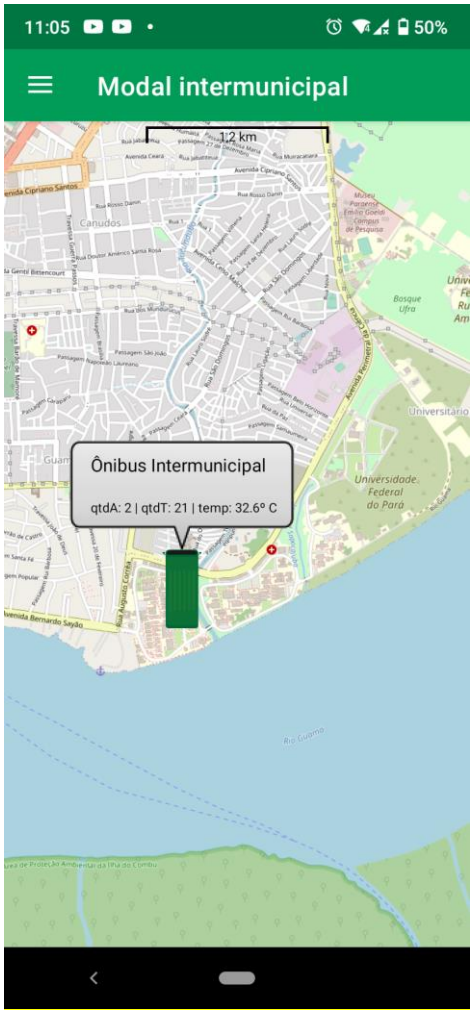
Figura 65 – Tela de rotas favoritas do usuário.

Fonte: elaborada pelo autor.

Vale lembrar que a tela de rotas favoritas só pode ser acessada a partir do menu apresentado na Figura 63.

A partir do menu de opções, Figura 63, é possível que o usuário monitore o ônibus intermunicipal cuja a rota de ida tem o ponto de partida na UFPA campus Belém e ponto de chagada na UFPA campus de Castanhal. A Figura 66 ilustra a tela de monitoramento do modal intermunicipal, que é composta por um mapa, o marcador do ônibus e as informações de quantidade de usuário, temperatura interna e nome do modal.

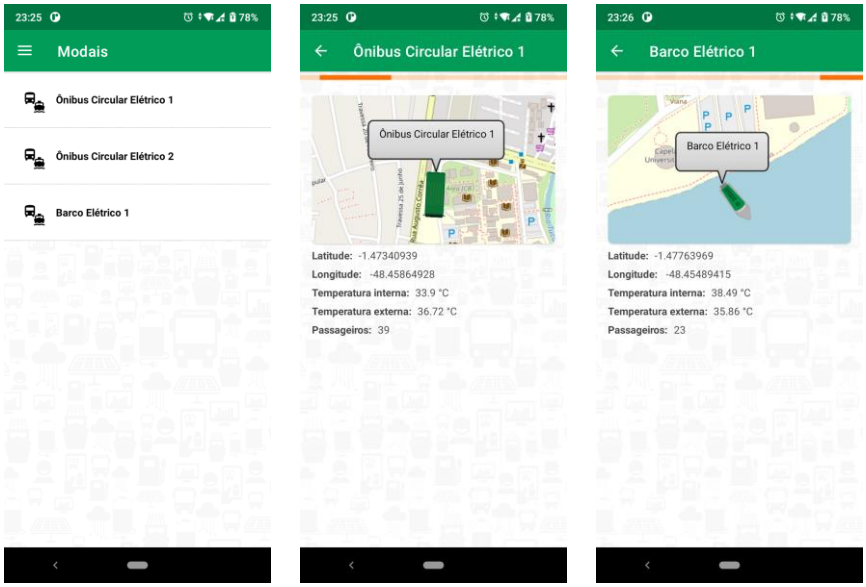
Figura 66 – Tela de monitoramento do modal intermunicipal.



Fonte: elaborada pelo autor.

O *software* SIMA Mobilidade Multimodal também disponibiliza dados individuais de cada modal do campus, conforme Figura 67. A Figura 67(a) ilustra uma lista de modais, presentes no campus, composta por seus ícones e nomes. Já a Figura 67(b) ilustra a tela de dados individuais de um modal, exemplo de um ônibus, composta por um mapa com o marcador do modal, nome, latitude, longitude, temperatura interna e externa, e quantidade de passageiros. Na Figura 67(c) é ilustrado um exemplo usando um modal do tipo barco.

Figura 67 – Dados individuais dos modais. (a) Lista de modais. (b) Exemplo de um ônibus. (c) Exemplo de um barco.



(a) (b) (c)

Fonte: elaboradas pelo autor.

5.5.2 Cenários e Arquiteturas dos Testes

O *software* móvel SIMA Mobilidade Multimodal foi testado em ambiente real e simulado. Os cenários de testes reais da aplicação SIMA Mobilidade Multimodal se deram nos *campi* da UFPA nas cidades de Belém-PA e Castanhal-PA. Enquanto que o cenário simulado foi feito através de um *script* Node.js[7] com dados de geolocalização do campus de Belém. As Subseções 5.5.2.1 e 5.5.2.2 apresentam os cenários e as arquiteturas de testes do *software*.

5.5.2.1 Cenário e Arquitetura do Teste por Simulação

Para realização dos testes do aplicativo SIMA Mobilidade Multimodal, foi elaborada uma arquitetura que simula dados de dispositivos presentes nos ônibus e barco elétricos do campus da UFPA Belém.

O primeiro passo para a elaboração da arquitetura, foi coletar as rotas do ônibus circular e do barco elétrico. Estas rotas são compostas por vários dados formados por posições de geolocalização (latitude e longitude). Em seguida, foi criado um *script* em linguagem de programação Javascript para simular os dados da trajetória do ônibus e do barco pelo campus.

[7] Node.js é um *software* interpretador de códigos Javacript.

A Figura 68 ilustra o cenário de teste por simulação do ônibus e barco, ela mostra a trajetória por onde o ônibus e barco, simulados pelo *script*, devem percorrer. A rota do barco foi definida para passar pelos três setores do campus (básico, profissional e saúde). Já a rota do ônibus passa pelos três setores e pelo Parque de Ciência e Tecnologia do Guamá (PCT).

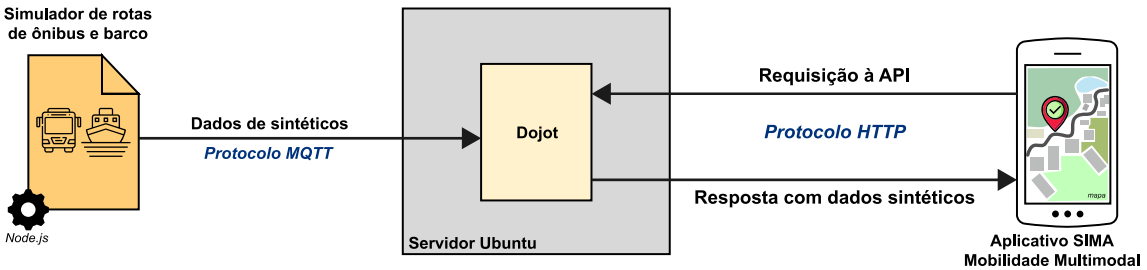
Figura 68 – Cenário de teste por simulação da rota do ônibus e barco no campus de Belém.

Fonte: elaborada pelo autor.

O *script* foi programado para a cada 3 segundos enviar uma posição de geolocalização para a Dojot, simulando assim, a viagem dos modais pelo campus. Os dados de número de passageiros e valores de temperatura foram gerados de forma aleatória utilizando funções nativas da linguagem Javascript.

O *script* foi executado com o Node.js em um *host* local, ele aponta para o *host* da Dojot instalado em um servidor Ubuntu, conforme especificado na Subseção 5.3.2. A Figura 69 ilustra a arquitetura de funcionamento da simulação, onde o *script* envia dados sintéticos dos modais para a Dojot, e o aplicativo SIMA Mobilidade Multimodal os recupera através da API da Dojot.

Figura 69 – Arquitetura de teste por simulação



Fonte: elaborada pelo autor.

5.5.2.2 Cenários e Arquiteturas dos Testes Reais

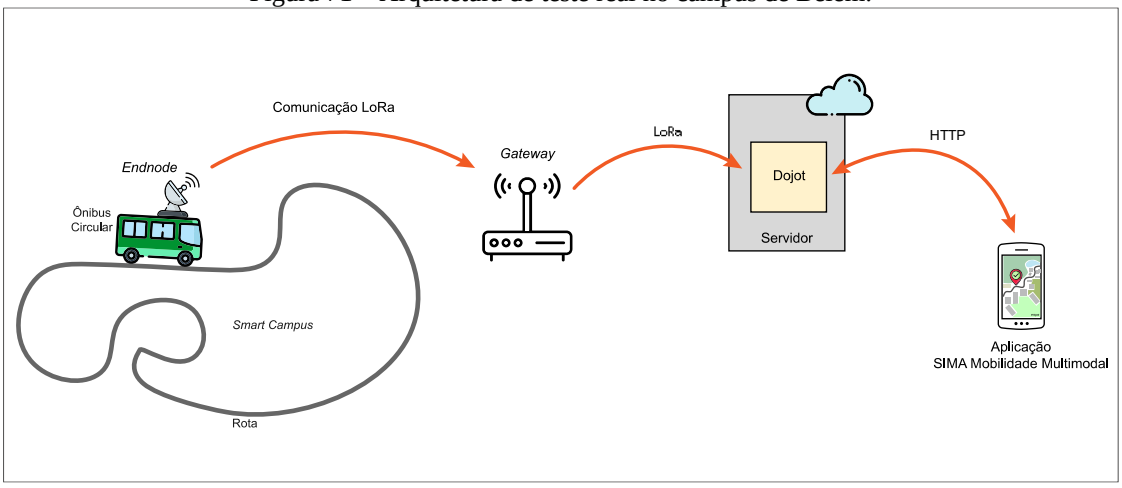
Para realização dos testes do aplicativo com dados reais, foram configuradas duas arquiteturas com dois cenários distintos. O primeiro cenário escolhido foi o campus da UFPA em Belém. A rota de teste real do ônibus elétrico tem um comprimento de 9,38 km, a qual visita os três setores do campus, além do PCT. A Figura 70 ilustra o mapa do campus e a trajetória de teste real do ônibus circular.

Figura 70 – Cenário e trajetória do ônibus para o teste real no campus de Belém.

Fonte: elaborada pelo autor.

A arquitetura do teste real no campus de Belém foi composta por um ônibus elétrico equipado com um dispositivo físico (*endnode*) que capturava sua geolocalização e a enviava a cada 5 segundos para a Dojot, a comunicação entre o *endnode* e *gateway* se deu através da rede LoRa. Com os dados reais do modal enviados para Dojot, o aplicativo SIMA Mobilidade Multimodal pôde os consultar e apresentar graficamente em tempo real para o usuário. A Figura 71 ilustra esta arquitetura projetada.

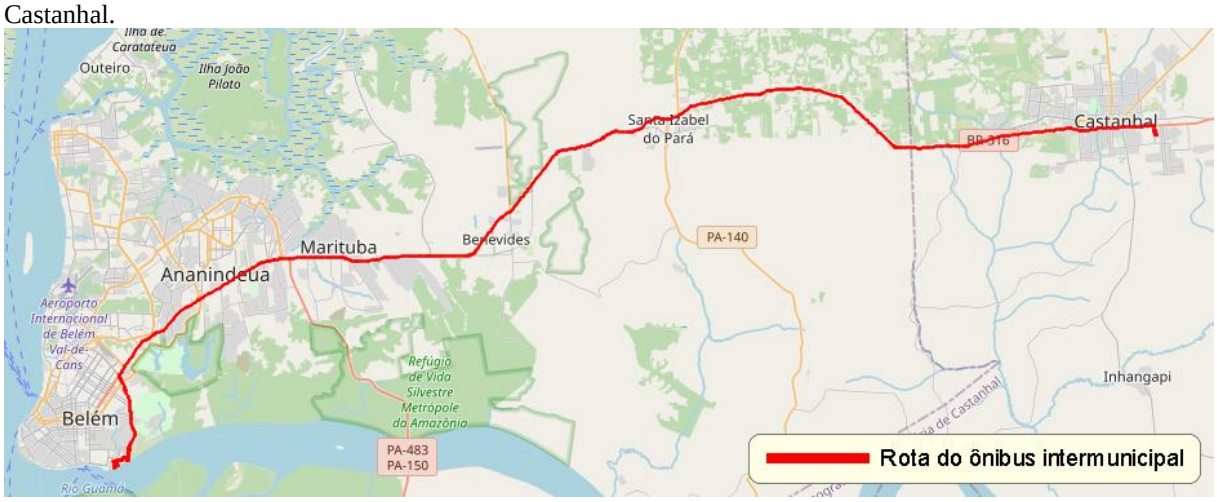
Figura 71 – Arquitetura de teste real no campus de Belém.



Fonte: elaborada pelo autor.

O segundo cenário de teste real, se deu em uma rota entre os *campi* da cidade de Belém e Castanhal, conforme ilustrado na Figura 72. Esse cenário tem uma rota de aproximadamente 77 km de comprimento entre os dois *campi*. Além das cidades de partida e chegada, a trajetória passa pelos municípios de Ananindeua, Marituba, Benevides e Santa Izabel do Pará.

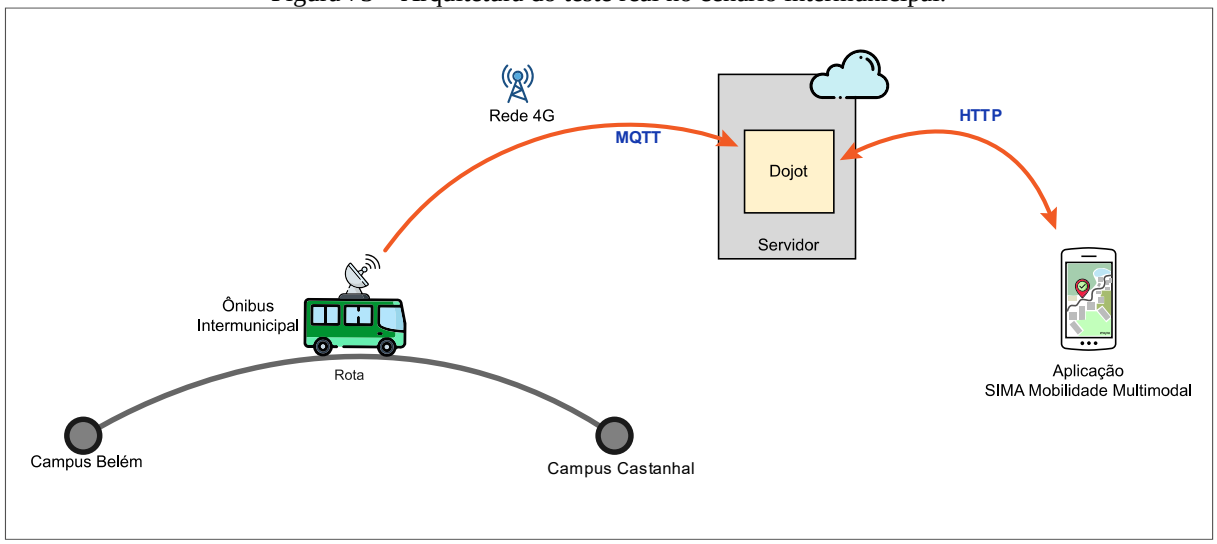
Figura 72 – Cenário e trajetória do ônibus intermunicipal para o teste real nos *campi* das cidades de Belém e Castanhal.



Fonte: elaborada pelo autor.

A arquitetura montada para o envio de dados da trajetória do ônibus intermunicipal foi composta por um *hardware* instalado no ônibus com o objetivo de recuperar sua geolocalização e as enviar para Dojot usando o protocolo MQTT através da rede de Quarta Geração de telefonia móvel (4G). Por sua vez, o aplicativo proposto recuperava os dados do modal em tempo real e os apresentava para o usuário, conforme ilustra a Figura 73.

Figura 73 – Arquitetura do teste real no cenário intermunicipal.

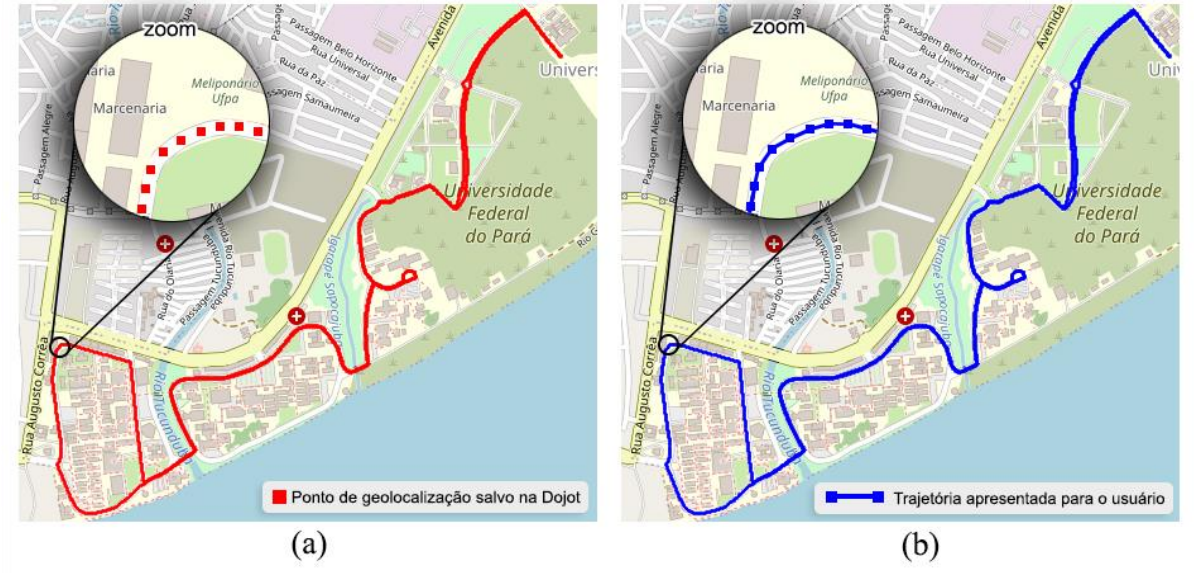


Fonte: elaborada pelo autor.

5.5.3 Resultados do Teste Via Simulação

Durante o teste de simulação da rota do ônibus circular elétrico no cenário do campus da UFPA em Belém, os dados sintéticos de geolocalização foram acompanhados em tempo real via mapa no aplicativo SIMA Mobilidade Multimodal. Ao final do teste, percebeu-se que o ônibus completou toda sua trajetória que havia sida definida no experimento. A Figura 74 ilustra o teste por simulação do ônibus, onde a Figura 74(a) apresenta graficamente todos os pontos discretos de geolocalização recebidos pela Dojot. Já a Figura 74(b) ilustra, de forma contínua, toda a trajetória apresentada para o usuário que acompanhou o teste via aplicativo. Desta forma, todos os dados sintéticos de teste que chegaram à Dojot, foram lidos em tempo real pelo *software* proposto.

Figura 74 - Resultados do teste por simulação do ônibus.



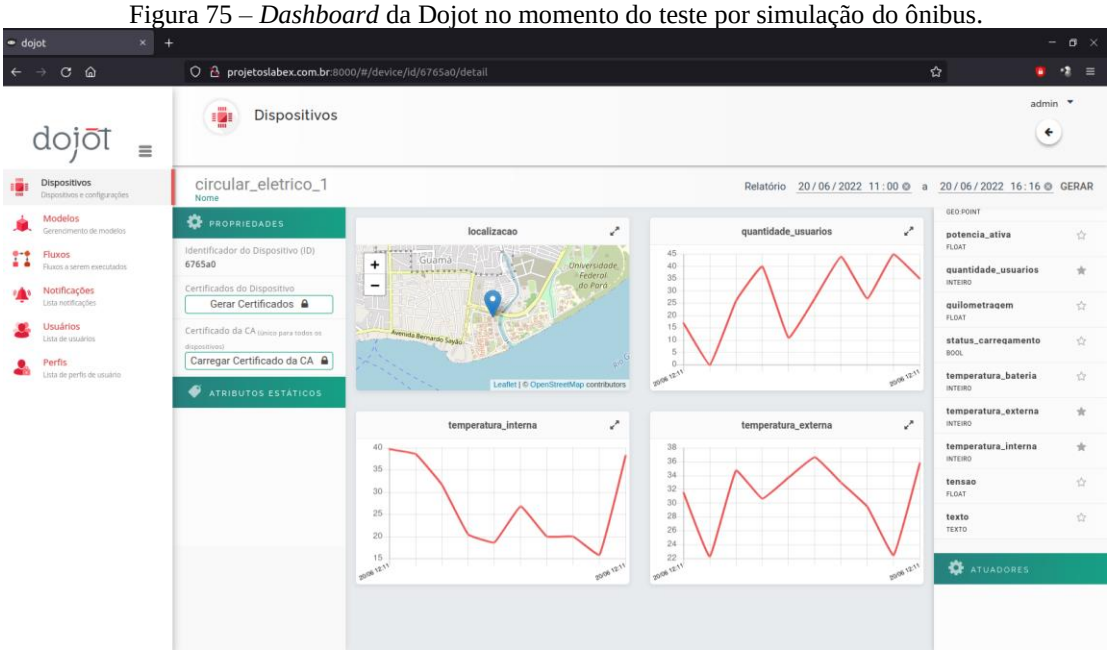
Fonte: elaboradas pelo autor.

O teste de simulação do ônibus circular durou cerca de 1 hora e 34 minutos. Nele, foram enviados 1883 registros para Dojot, 1 a cada 3 segundos, conforme definido no simulador. Além do acompanhamento via o *software* proposto, o teste foi acompanhado através da *dashboard* da Dojot, conforme ilustra a Figura 75.

Já o teste do barco, durou cerca de 15 minutos percorrendo uma rota de 1,47 km. Durante a execução do simulador, o barco com dados sintéticos foi acompanhado através do SIMA Mobilidade Multimodal. A Figura 76 ilustra o resultado do experimento de simulação com o barco, onde a Figura 76(a) apresenta a rota com dados discretos de todos os pontos de geolocalização recebidos e salvos pela Dojot. Já a Figura 76(b) ilustra a trajetória por onde o

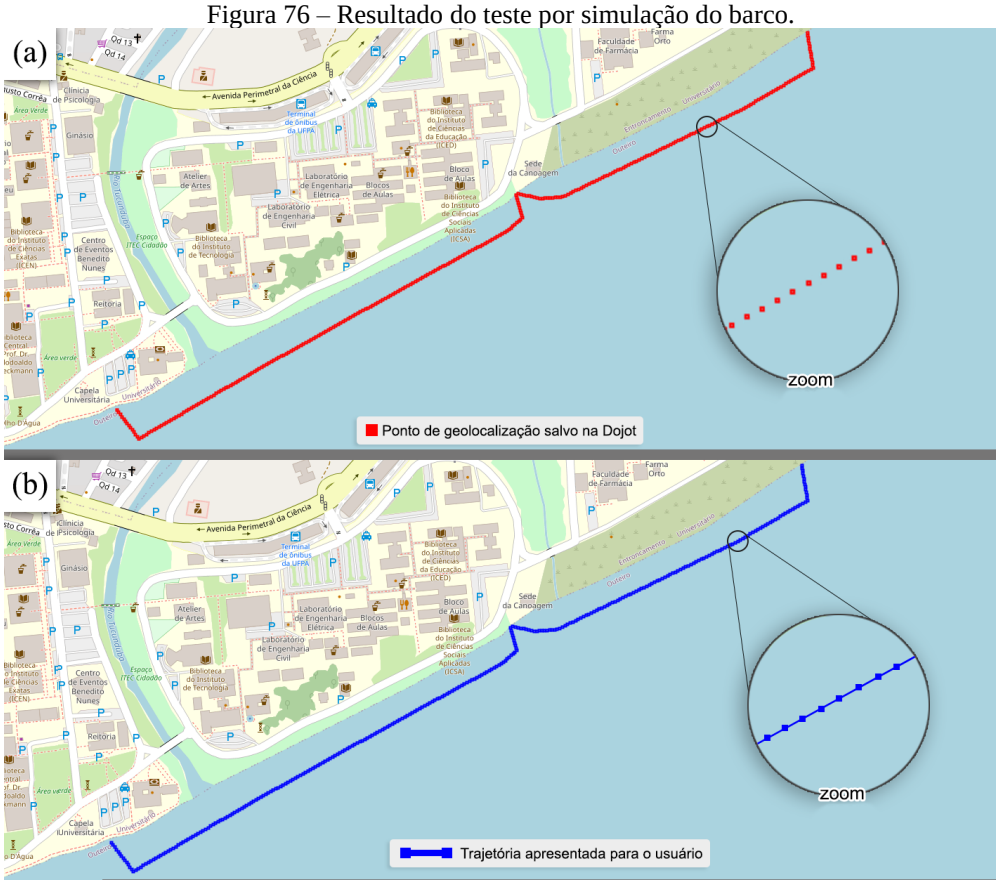
barco passou no mapa do aplicativo. Neste experimento, todos os dados salvos na Dojot, foram lidos e apresentados para o usuário em tempo real na aplicação SIMA Mobilidade Multimodal.

Figura 75 – *Dashboard* da Dojot no momento do teste por simulação do ônibus.



Fonte: elaborada pelo autor.

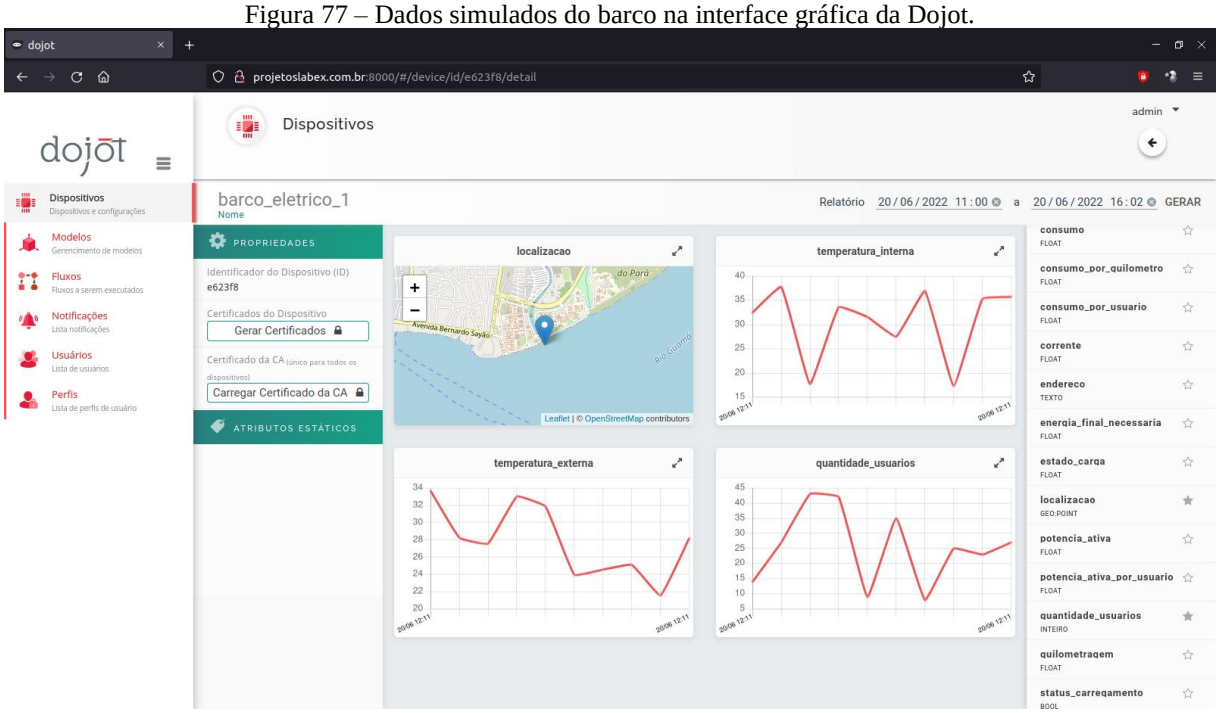
Figura 76 – Resultado do teste por simulação do barco.



Fonte: elaboradas pelo autor.

Além do aplicativo, o experimento foi acompanhado via interface gráfica da Dojot, conforme ilustrado na Figura 77.

Figura 77 – Dados simulados do barco na interface gráfica da Dojot.



Fonte: elaborada pelo autor.

5.5.4 Resultados dos Testes Reais

No primeiro teste real, realizado no campus da UFPA em Belém, o ônibus real elétrico percorreu uma rota de um pouco mais de 9 km. Os dados reais foram enviados em tempo real para Dojot via uma rede LoRa, e foram lidos pelo *software* proposto e apresentados para o usuário. Durante este teste, houve falhas de comunicação na rede LoRa, assim como também falhas de GPS.

A falta de sinal da rede LoRa impossibilitou que o dispositivo físico enviasse dados do ônibus em tempo real para Dojot, isso foi percebido ao visualizar o ônibus pelo aplicativo, onde o marcador do ônibus no mapa do *software* parou no ponto de geolocalização que ocorreu a falta de sinal.

As falhas de GPS foram observadas através do aplicativo quando o ônibus se comportou de forma inesperada, saindo de sua rota, ou seja, quando ele andou por cima de gramados, calçadas, casas, entre outros.

A Figura 78 ilustra graficamente os dados de geolocalização salvos na Dojot, lidos e apresentados para o usuário no aplicativo. A Figura 78(a) apresenta todos os dados de GPS

capturados e enviados para Dojot pelo dispositivo físico instalado no ônibus. É possível observar a irregularidade dos pontos de geolocalização para esse teste real em comparação com os dados dos testes simulados. Nesta Figura 78(a) existem alguns pontos fora das ruas provocados por erros de GPS. Existe ainda, um erro onde o ponto está muito distante da rota, ficando fora do campus. Este erro foi observado em tempo real pelo aplicativo SIMA Mobilidade Multimodal durante o teste. A Figura 78(b) ilustra a trajetória do ônibus pelo mapa do aplicativo, nela é possível observar o ponto e a trajetória quando o ônibus se afastou da rota, chegando a ir para fora do campus (próximo ao setor básico) e logo em seguida retornando quando o GPS fez a correta leitura de localização.

Figura 78 – Resultados do teste real do ônibus elétrico no campus de Belém.

(a) (b)

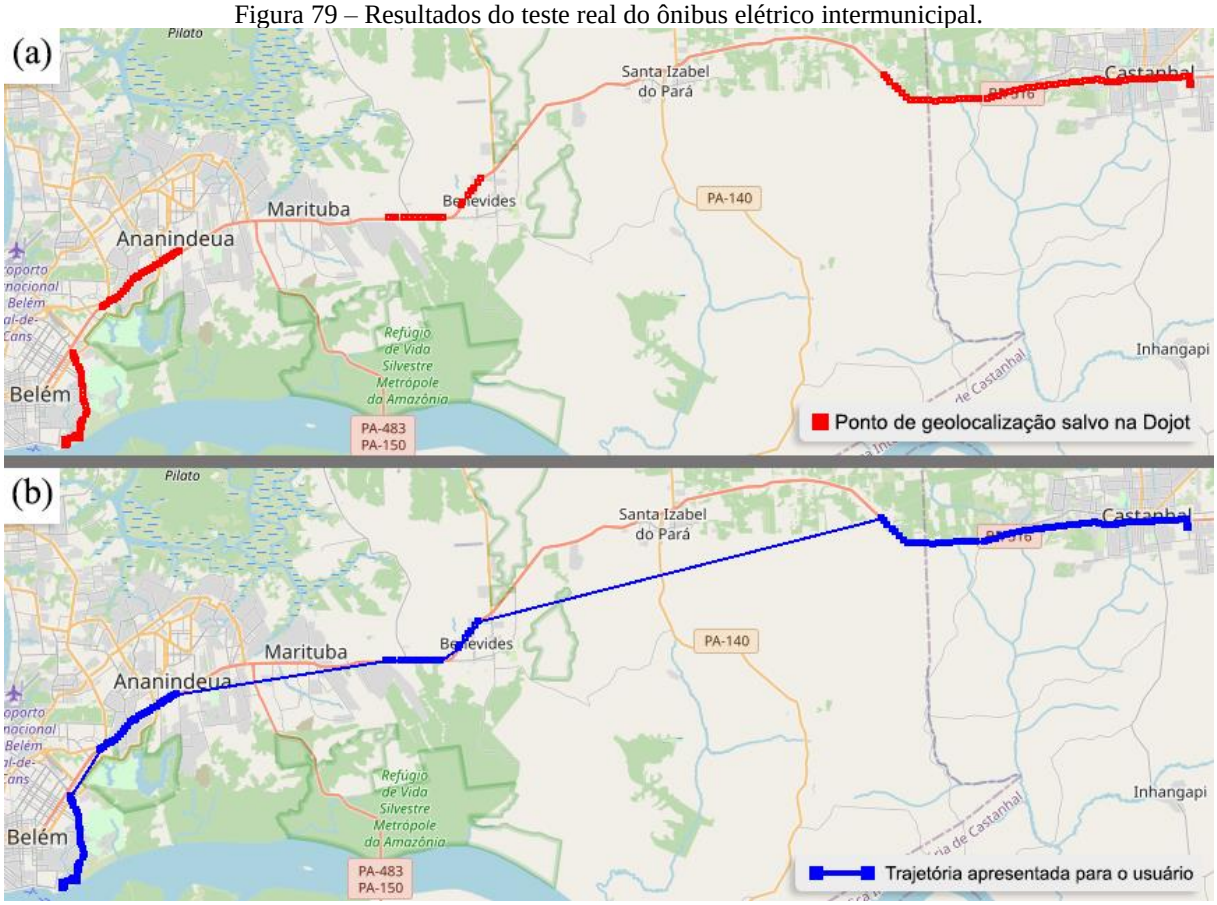
Fonte: elaboradas pelo autor.

No segundo teste real, o intermunicipal, o ônibus saiu da cidade de Belém em direção a cidade de Castanhal. A maior parte do trajeto dentro da cidade de Belém, foi bem-sucedido, no entanto, em um trecho da avenida Almirante Barroso, houve falha de comunicação com a rede 4G, o que impossibilitou o envio da geolocalização do ônibus para a Dojot. Além dessa falha dentro do município de partida da rota, houve falhas de comunicação com a rede durante boa parte do percurso. A Figura 79 ilustra os dados do teste real da rota intermunicipal.

A Figura 79(a) ilustra todos os pontos enviados para Dojot quando o dispositivo físico do ônibus teve conectividade com a rede 4G. Já a Figura 79(b) ilustra o rastro de rota feita pelo ônibus apresentado para o usuário no aplicativo SIMA Mobilidade Multimodal. Apesar da falta de conectividade e envio de dados à Dojot, o aplicativo proposto foi satisfatório em fazer a leitura dos dados presentes na Dojot.

Na Figura 79(b) é possível visualizar as linhas retas que o ônibus fez no mapa do aplicativo quando houve falta de conectividade, principalmente nas cidades de Marituba e Santa Izabel do Pará.

Figura 79 – Resultados do teste real do ônibus elétrico intermunicipal.



Fonte: elaboradas pelo autor.

5.6 Conclusão

Este Capítulo apresentou toda a estrutura para construção e teste de um *software* móvel cujo objetivo é gerir dados de modais de transporte de um *smart campus*. Para isso, foram apresentados os requisitos, a modelagem do banco de dados relacional e não relacional, a API REST construída para comunicação do aplicativo com o banco de dados relacional, os cenários e estruturas de testes, bem como, os resultados dos testes do *software*.

A partir dos resultados, o *software* se mostrou eficiente em apresentar a informação presente na plataforma Dojot. No ambiente controlado, os resultados foram melhores devido a regularidade dos pontos de geolocalização e por não haver falta de conectividade do *script* simulador com o servidor. Já nos resultados dos testes reais, o aplicativo também se mostrou eficaz no que se propôs a fazer, onde mesmo em momentos em que o sinal (LoRa ou 4G) do

dispositivo/ônibus se perdeu, o *software* continuou procurando os pontos na Dojot, e quando aparecia, ele mostrava para o usuário.

Problemas de perda de sinal 4G ou LoRa, que afeta nos dados mostrados para o usuário no aplicativo (como visto na Subseção 5.5.4), podem ter soluções, no entanto foge do escopo deste trabalho. Já os problemas causados por erros de GPS, podem ser solucionados utilizando algoritmos de IC, tal como o KNN por exemplo, proposto por Sá *et al.* (2019), que ajusta todos os pontos de geolocalização para dentro da rota do modal ônibus.

6 REDE NEURAL ARTIFICIAL PARA ESCOLHA DE ROTAS EM UM *SMART CAMPUS*

Este Capítulo apresenta uma proposta de Rede Neural Artificial para a classificação da melhor rota de viagem em um *smart campus*.

O restante deste Capítulo está organizado da seguinte forma, na Seção 6.1 o autor faz uma introdução contextualizando e apresentando o objetivo da proposta. Na Seção 6.2 é apresentado o algoritmo de IC e a base de dados utilizada. Já na Seção 6.3, são expostos os resultados, e por fim, a Seção 6.4 apresenta a conclusão.

6.1 Introdução

A IoT é um paradigma que surgiu para conectar objetos do cotidiano à Internet, com a capacidade de coletar e transmitir dados, facilitando processos industriais e atividades domésticas. Neste contexto surgem os *smart campus*, que utilizam IoT para solucionar diversos problemas, tais como garantir a coleta e transmissão de dados ou permitir o gerenciamento de recursos de forma eficiente em um campus universitário (ZHAMANOV, 2017).

Diante do grande número de dados gerados e coletados por dispositivos IoT instalados em um *smart campus*, torna-se necessário a utilização de algoritmos inteligentes com o objetivo de extrair conhecimentos a partir de bases de dados obtidas. Deste modo, aplicar IC no contexto de *smart campus*, possibilita solucionar problemas de diversas áreas, uma delas é a mobilidade urbana, onde é possível através dos algoritmos, por exemplo, prever o tempo de chegada de ônibus, planejar e sugerir rotas de menor caminho para pedestres, entre outras aplicações.

A partir deste contexto, este Capítulo tem como objetivo apresentar o uso de um algoritmo de IC baseado em RNA para determinação da melhor rota de viagem considerando as opções a pé, de ônibus e de barco em um sistema IoT de um *smart campus*.

6.2 Base de Dados e RNA Proposta

A base de dados utilizada na RNA proposta, foi previamente coletada por Silva *et al.* (2020) e pelo *software* Norte Rotas apresentado no Capítulo 4 desta dissertação.

Silva *et al*. (2020), no âmbito do projeto SIMA, fez a coleta das rotas de ônibus, barco, e trajetórias a pé saindo de prédios em direção às paradas de ônibus e cais de barco. Já as rotas a pé entre prédios da UFPA, como visto no Capítulo 4, foram coletadas pelo Norte Rotas. Ambas as fontes de dados utilizaram um banco de dados relacional para o armazenamento das rotas.

As rotas coletadas e armazenadas em uma base de dados, formam uma matriz composta de latitude e longitude, conforme a Equação (1).

$$r_i = \{(lat_1, lng_1), (lat_2, lng_2), \dots, (lat_n, lng_n)\} \quad (1)$$

onde r_i, lat, lng representam respectivamente a *i-ésima* rota coletada, latitude e longitude coletadas para cada um dos *n* pontos.

Os dados das rotas coletadas foram transformados para um formato legível pela RNA. Para isso, todas as *m* posições/registros presentes na tabela que guarda as rotas cadastradas são percorridas, sendo retornada uma matriz composta pelas *m* posições iniciais e finais presentes em cada rota cadastrada, conforme Equação (2).

$$transf\_r_i = \{(lat_1, lng_1, lat_n, lng_n)\} \quad (2)$$

onde $transf\_r_i$ representa o *i-ésimo* registro de entrada para a RNA, lat_1 e lng_1 representam a latitude e longitude do ponto inicial da rota, por fim, lat_n e lng_n representam a latitude e longitude do ponto de chegada final.

Cada registro é composto pela latitude e longitude do ponto de partida, latitude e longitude do ponto de chegada, e as saídas que representam os neurônios com a melhor rota. A matriz de neurônios de saída é composta por *m* registros e por *r* classes de saída. A Equação (3) representa o modelo de saída de um neurônio ativado em $r_{i,1}$.

$$neurônio\_r_{i,1} = \{1,0,0, \dots .,0\} \quad (3)$$

onde $neurônio\_r_{i,1}$ representa o *i-ésimo* registro de entrada correspondente à primeira classe de saída em estado ativo para a RNA. De forma visual, a Tabela 1 ilustra um exemplo geral da combinação dos registros de entrada e da classe de saída para uma representação da base de dados ajustada para o uso da RNA.

Tabela 1 – Representação dos *m* registros da base de dados utilizada.

Latitude (partida)	**Longitude (partida)**	**Latitude (chegada)**	**Longitude (chegada)**	**Neurônio 1 (Rota 1)**	**...**	**Neurônio r (Rota r)**

-1.47597	-48.45649	-1.473511	-48.45763	1	...	0
-1.47650	-48.45683	-1.473511	-48.45763	1	...	0
-1.47459	-48.45742	-1.474759	-48.456541	0	...	0
-1.47463	-48.45741	-1.474759	-48.456541	0	...	0
...	...	...	...	...	...	...
-1.47633	-48.45624	-1.473465	-48.45831	0	...	0
-1.47619	-48.45627	-1.473465	-48.45831	0	...	1

Fonte: elaborada pelo autor.

Para este trabalho, foram transformados dados de classe de saída de geolocalização (latitude e longitude) para $r = 209$ rotas distintas, totalizando o número de registros $m = 2297$ na base de dados. Deste modo, cada neurônio de saída representa uma possível rota a ser selecionada pela RNA.

A RNA proposta neste trabalho é formada por uma camada de entrada, uma camada escondida, uma camada de saída e o algoritmo *backpropagation* para correção do erro e ajustes dos pesos sinápticos. Sua estrutura de funcionamento é descrita nas próximas Equações.

Na Equação (4) a matriz a ser treinada é normalizada.

$$x_j = normaliza(transf\_r_i) \qquad 1 \leq i \leq m \tag{4}$$

onde x_j representa a matriz com dados de entrada normalizada. Em seguida, a matriz normalizada é multiplicada por uma matriz formada por pesos sinápticos gerados aleatoriamente, conforme a Equação (5).

$$net_k = \sum_{j=1}^{k-1} W_{kj} x_j \qquad m \leq k \leq N + m \tag{5}$$

onde W_{kj} são os elementos da matriz de pesos sinápticos, nos quais, o tamanho depende do número de neurônios na camada adjacente correspondente da RNA. Como resultado é gerado o resultado de net_k que corresponde ao resultado do produto dos pesos e dados da matriz de entrada normalizada.

Depois, a net_k é apresentada à função de ativação *sigmoid* da RNA, conforme Equação (6).

$$f(net_k) = \frac{1}{1 + e^{-net_k}} \tag{6}$$

onde a função de ativação *sigmoid* é mais popular e aceita na literatura (FAYYAD, et al. 1996) (MISHRA; SRIVASTAVA; 2014) (YIJUN, 2010). A partir da saída obtida na Equação (6), esta é reapresentada para a Equação (5) tendo como valor de $x_j = f(net_k)$ como entrada para o próximo neurônio entre a camada escondida e camada de saída da RNA. Em seguida, o novo resultado é reapresentado a Equação (6), assim, sendo obtido o resultado de saída da RNA, que por sua vez, representa a rota a ser selecionada pelo usuário, conforme exemplificado na Equação (3). Como próximo passo, o erro é calculado a partir da Equação (7).

$$Erro = \frac{1}{2}\left(neurônio\_r_{i,j} - f(net_k)\right)^2 \tag{7}$$

onde $neurônio\_r_{i,j}$ e $-f(net_k)$ representam respectivamente a saída desejada para o *i-ésimo* registro da base de treinamento e o resultado da RNA obtido a partir do treinamento da RNA. Caso o *Erro* não atenda às condições de parada, os pesos sinápticos são atualizados a partir do cálculo do gradiente descendente, conforme Equação (8).

$$W_{ij}^{+} = W_{ij} - \eta \, \nabla_{Erro} \tag{8}$$

onde W_{ij}^{+}, η e ∇_{Erro} representam respectivamente a nova matriz de pesos sinápticos, a taxa de aprendizagem do algoritmo e o gradiente do erro.

6.3 Resultados

Esta Seção apresenta uma descrição detalhada do cenário de teste, dos parâmetros utilizados para alimentar o algoritmo de classificação baseado na RNA, os resultados alcançados, e um modelo de aplicação da RNA utilizando um painel *web*, para auxiliar processos de identificação da melhor rota entre dois pontos.

6.3.1 Cenário de Teste

O cenário de teste foi o *smart campus* implantado na UFPA, localizado na cidade de Belém-PA. Para o teste, foi realizado o mapeamento da rota do barco e do ônibus denominado circular. Durante o mapeamento, foram coletados e armazenados em um banco de dados relacional os pontos de geolocalização da rota percorrida pelo barco e ônibus circular, e também foram mapeados os percursos entre os prédios e os pontos de parada que estão localizados dentro da área do campus, conforme especificado no trabalho de Silva *et al.* (2020). É válido

reforçar que durante os testes, para a recuperação das informações sobre as rotas, foi utilizado um painel *web*. Este painel *web* possibilitou que o usuário fizesse os testes da RNA, onde ao inserir os pontos de partida e chegada, o sistema apresentava a melhor rota a partir da saída da RNA.

6.3.2 Parâmetros de Simulação e Resultados da RNA

Durante as simulações o número de neurônios na camada escondida foi variado de 10 até 1000, sendo para cada valor realizado uma repetição de 10 simulações. Em seguida, o número de neurônios foi aumentado de 10 em 10, isto é, 10, 20, 30, 40, ..., 980, 990, 1000. A Tabela 2 apresenta as combinações realizadas.

Tabela 2 – Variação dos parâmetros por simulação.

Parâmetros de Simulação	**Número de Amostras**
Taxa de aprendizagem	0.01 – 0.99
Número de neurônios na camada escondida	10 – 1000
Divisão da base de dados em porcentagem de Treinamento, Validação e Teste, respectivamente	50, 20, 30
	50, 25, 25
	60, 20, 20
	70, 15, 15

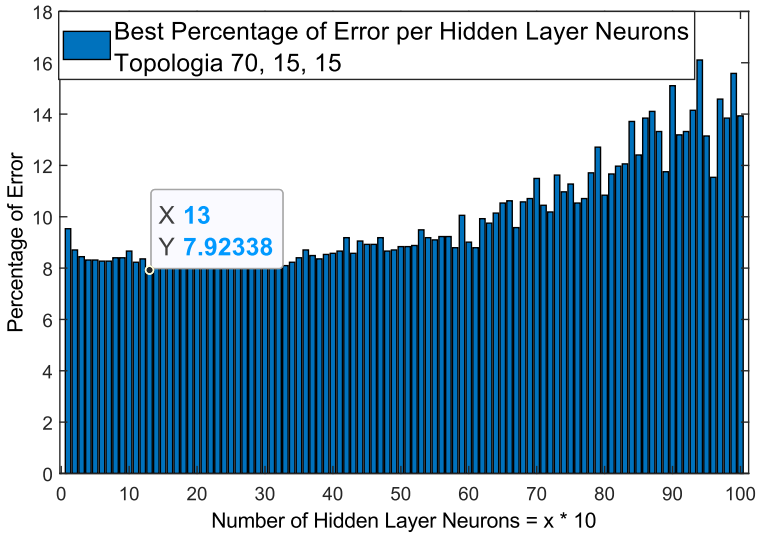
Fonte: elaborada pelo autor.

A Figura 80 ilustra os resultados obtidos a partir da variação dos parâmetros de simulação, sendo apresentado no eixo *x* o número de neurônios na camada escondida (N_{hidden}), o qual é dado pela Equação (9).

$$N_{hidden} = 10 * x_{axis} \tag{9}$$

onde x_{axis} representa o valor no eixo *x* variando de 1 até 100. Enquanto que, o eixo *y* apresenta a menor porcentagem de erro obtida a partir de cada uma das 10 simulações com o número de pesos sinápticos.

Figura 80 – Melhor resultado obtido para a topologia 70, 15, 15 considerando a variação do número de neurônios de 10 – 1000.



Fonte: elaborada pelo autor.

Dentre todas as combinações dos parâmetros de simulação, o melhor resultado foi obtido com uma η = 0.01 e topologia 70, 15, 15.

Para obtenção do resultado geral de porcentagem de performance geral ($P_{general}$), foi utilizada a Equação (10).

$$P_{general} = 100\frac{(S_{train} + S_{val} + S_{test})}{m} \tag{10}$$

onde S_{train}, S_{val} e S_{test} que representam respectivamente, o número de acertos no treinamento, validação e teste.

A Tabela 3 apresenta a distribuição dos registros aplicados para a obtenção da melhor RNA treinada, sendo adotada a divisão de 1607, 345 e 345 registros para, respectivamente, treinamento, validação e teste. A partir da divisão, o melhor resultado retornou um $P_{general}$ superior a 92%, o que demonstra sua capacidade de generalização para o problema.

Tabela 3 – Divisão da base de dados e melhor resultado obtido nas etapas de treinamento, validação e teste.

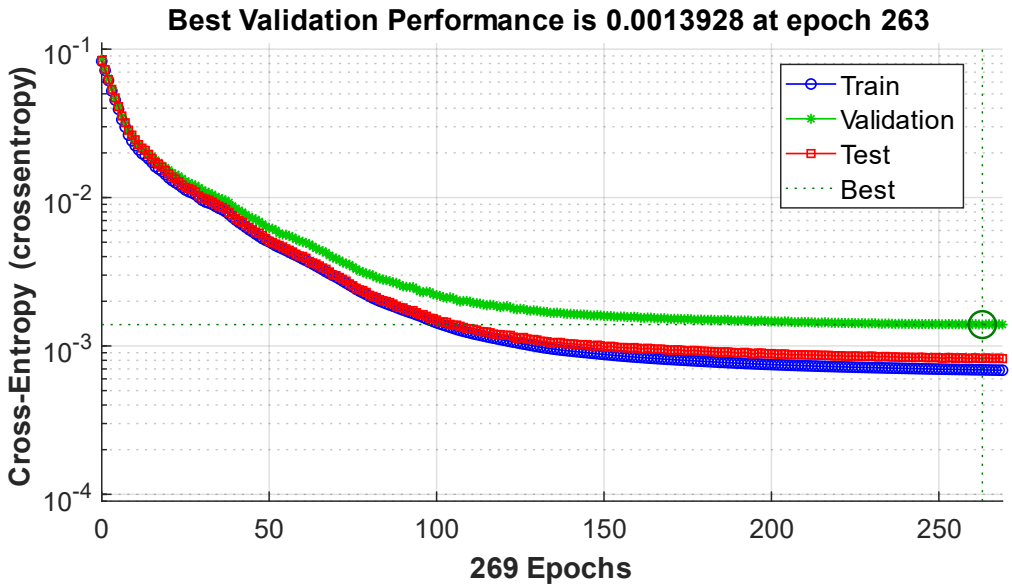
Etapa	Registros	Divisão (%)	Acurácia (%)
Treinamento	1607	70	92,72
Validação	345	15	90,15
Teste	345	15	91,02
Total	2297		

Fonte: elaborada pelo autor.

De forma detalhada, a melhor simulação é ilustrada na Figura 81, onde é apresentada a relação entre o número de épocas do algoritmo e seu resultado de performance por época simulada. Para obtenção deste resultado, foi utilizada a entropia cruzada em relação a base de dados de treinamento, validação e teste, sendo obtidos os resultados de acertos de 1490 no treinamento, 311 na validação, e 314 no teste.

Como critério de parada, foi utilizada a performance da validação, sendo obtido o melhor resultado na época 263 da simulação.

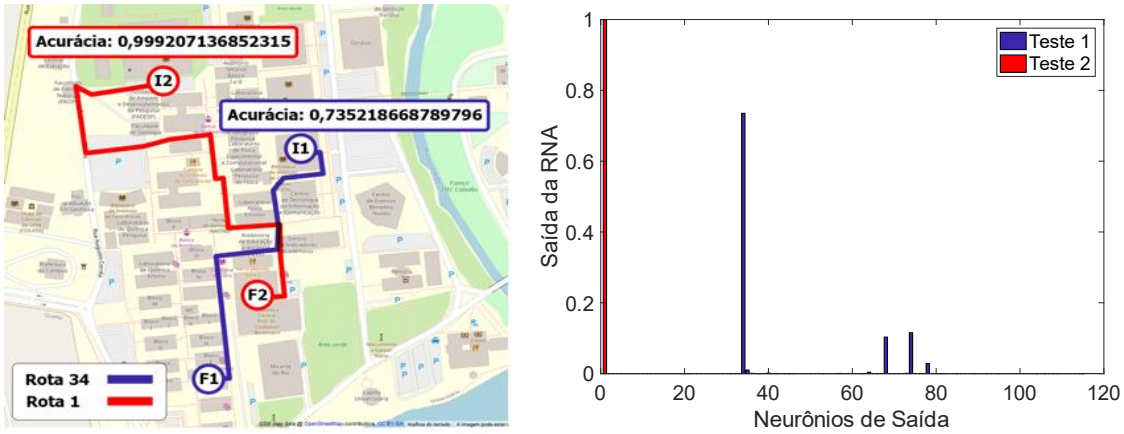
Figura 81 – Melhor resultado das simulações com 130 neurônios na camada escondida, η = 0.01 e topologia 70, 15, 15.



Fonte: elaborada pelo autor.

Para exemplificar os resultados, a Figura 82 ilustra os resultados visuais de dois experimentos a partir da integração dos pesos sinápticos treinados como uma solução de mobilidade urbana do *smart campus* da UFPA. A partir desse resultado é possível observar duas rotas diferentes, sendo estas percorridas a partir da escolha determinada por diferentes usuários (U1, U2), sendo dado como entrada os pontos iniciais (I1, I2) e finais (F1, F2) para cada um dos usuários. A Figura 82(a) ilustra as duas rotas apresentadas pela RNA a partir dos pontos de entrada informados por U1 e U2. Já a Figura 82(b) ilustra a ativação dos neurônios 1 (no experimento/teste 2) e 34 (no experimento/teste 1), com a acurácia de 0,999207136852315 e 0,735218668789796, respectivamente.

Figura 82 – Resultado de dois testes a pé. (a) Duas rotas de saída da RNA. (b) Gráfico dos neurônios que representam as rotas.



(a) (b)

Fonte: elaborada pelo autor.

Em um terceiro experimento, no painel *web*, o usuário selecionou como ponto inicial o prédio da Faculdade de Fisioterapia e ponto final a Faculdade de Engenharia Civil. A RNA conseguiu identificar e classificar os pontos de ônibus para partida (“Parada: Nutrição-Ida”) e chegada (“Parada: Portão 3-Volta”), como também, a rota do ônibus (marcada em vermelho). Para chegar ao ponto de partida, basta se deslocar do ponto inicial pelo caminho complementar (em preto), quando estiver no destino o usuário deve descer na parada de chegada, e seguir a pé o caminho complementar até o ponto final marcado por ele, conforme ilustrado na Figura 83.

Figura 83 – Percurso entre a Faculdade de Fisioterapia e Faculdade de Engenharia Civil.

Fonte: elaborada pelo autor.

Em um quarto experimento, utilizando o painel *web*, o usuário selecionou como ponto inicial o prédio do Núcleo de Educação Básica (NEB) e ponto final Faculdade de Engenharia Sanitária e Ambiental (FESA). A RNA conseguiu identificar e classificar os pontos de ônibus para partida (“Parada: ICJ-Volta”) e chegada (“Parada: Eng. Ambiental”), bem como, a rota do

ônibus (marcada em vermelho). Para chegar ao ponto de partida, basta se deslocar do ponto inicial pelo caminho complementar (marcado em preto), quando estiver no destino o usuário deve descer na parada de chegada, e seguir o caminho complementar até o ponto final marcado por ele, a Figura 84 ilustra toda trajetória a ser percorrida pelo usuário a partir do resultado da RNA.

Figura 84 – Percurso entre o NEB e a FESA

Fonte: elaborada pelo autor.

Já em um quinto experimento, o usuário selecionou com o clique um local próximo ao prédio do Restaurante Universitário (RU) (ponto inicial) e um local próximo ao prédio do Núcleo de Meio Ambiente (NUMA) (ponto final). A RNA foi responsável em identificar e classificar o ponto de partida (“Parada: Cais 01”) e chegada (“Cais 02”), deste modo, retornando a rota de barco (em vermelho), além de demarcar em preto a rota complementar que vai do ponto inicial até a parada de partida, da mesma forma, traçando o caminho complementar a pé que o usuário deve seguir, depois que o barco chegar na parada final até o ponto final, conforme ilustrado na Figura 85.

Figura 85 – Percurso entre o RU e NUMA.

Fonte: elaborada pelo autor.

Por fim, em um sexto experimento, utilizando o painel *web*, o usuário selecionou com o clique um local do prédio da Faculdade de Farmácia (ponto inicial) e um local próximo ao Chalé de Ferro (ponto final). A RNA conseguiu identificar e classificar o ponto de partida (“Parada: Cais 03”) e chegada (“Parada: Cais 02”), posto isso, demarcando o caminho complementar (rota em preto) até chegar na parada de embarque no Cais 03, cujo o barco, seguirá o seu trajeto (rota em vermelho) até sua parada de desembarque no Cais 02, por fim, o usuário deve seguir a trajetória complementar até o ponto de chegada conforme ilustrado na Figura 86.

Figura 86 – Percurso entre a Faculdade de Farmácia e o Chalé de Ferro.

Fonte: elaborada pelo autor.

Em todos os experimentos a RNA conseguiu cumprir a sua tarefa de identificar e classificar o ponto de partida e chegada, desta forma, retornando sempre a rota mais eficiente.

6.4 Conclusão

Este Capítulo apresentou um algoritmo de inteligência computacional para definição da melhor rota de viagem, em um *smart campus*, de acordo com a escolha das geolocalizações dos pontos de partida e chegada.

A partir dos resultados, é possível concluir que a RNA é uma alternativa para ser integrada no sistema de mobilidade urbana do *smart campus* da UFPA, como alternativa para determinação da melhor rota a ser indicada ao usuário.

7 CONCLUSÃO

7.1 Considerações Finais

Esta dissertação alcançou seu objetivo principal de desenvolvimento de dois *softwares* e um algoritmo de RNA para o apoio à gestão de mobilidade urbana multimodal em um *smart campus*.

Para alcançar o objetivo de apresentar o produto de *software* Norte Rotas, foi preciso seguir alguns passos, como o levantamento e análise de requisitos; a concepção e modelagem do banco de dados relacional através da modelagem lógica e física; a implementação do sistema *web*; e os testes em um cenário do *smart campus* da UFPA. O Norte Rotas mostrou-se uma ferramenta importante para geração de conhecimento sobre condições estruturais e de segurança de rotas de pedestres em um ambiente de *smart campus*, o que possibilita sua integração à sistemas especialistas para tomadas de decisões no contexto da mobilidade urbana. Além de sua aplicabilidade em um cenário de campus universitário, a proposta pode ser generalizada para ambientes maiores como *smart cities*.

Já o alcance do objetivo da proposta do *software* móvel SIMA Mobilidade Multimodal se deu pelas fases de reuniões para o levantamento de requisitos; modelagem dos bancos de dados relacional e não relacional; construção de uma API REST para gerir dados do banco de dados relacional e os disponibilizar para o aplicativo móvel; implementação do aplicativo móvel para dispositivos Android; e Testes em cenário simulado e real no *smart campus* da UFPA em Belém, bem como um teste intermunicipal entre as cidades de Belém e Castanhal no Pará. Os resultados dos experimentos mostraram a eficiência do aplicativo em apresentar as informações salvas dos modais na plataforma Dojot. Nos experimentos por simulação, os resultados se mostraram melhores, devido ao ambiente ser controlado, que proporcionou regularidade dos dados de geolocalização e conectividade constante do simulador com a Dojot. O aplicativo também se mostrou eficaz na leitura dos dados da Dojot nos experimentos reais, mesmo com problemas de GPS e de conexão entre os modais e a Dojot.

Apesar do *software* móvel se mostrar eficiente em fazer a leitura de dados dos modais do *smart campus*, para que haja uma melhor experiência de uso do SIMA Mobilidade Multimodal, existe a necessidade de solucionar problemas de comunicação da rede 4G e LoRa, no entanto, este trabalho foca apenas nas soluções apresentadas para o usuário final. Já

problemas causados imprecisão de GPS, podem ser solucionados utilizando algoritmos de IC, como por exemplo, o proposto por Sá *et al.* (2019), que corrige todos os erros de GPS em um sistema de rotas de ônibus.

Em relação à RNA proposta, a base de dados de rotas de ônibus e barcos utilizada para o treinamento, validação e teste, foi coletada por Silva *et al*. (2020), o qual faz parte do projeto SIMA, o mesmo projeto desta dissertação. Já os dados de rotas a pé utilizadas na RNA proposta, foram coletados a partir do *software* Norte Rotas. Experimentos com diferentes parâmetros de simulação foram realizados e os resultados apontaram que a proposta é uma alternativa para ser integrada no sistema de mobilidade urbana do *smart campus* da UFPA, como alternativa para determinação da melhor rota a ser indicada ao usuário.

Diante deste trabalho, é possível concluir que os *softwares* e algoritmo proposto, são soluções viáveis para a gestão de informações sobre mobilidade urbana em um *smart campus* com característica de transporte multimodal, e podem ser generalizados para ambientes maiores como cidades inteligentes e com mais opções de modais.

7.2 Trabalhos Futuros

Para trabalhos futuros, são propostos os seguintes itens:

- Disponibilizar o *software* SIMA Mobilidade Multimodal na Google Play para usuários da UFPA;
- Integrar os *softwares* propostos ao sistema de gestão e ao banco de dados final do projeto SIMA; e
- Expandir a base de dados e integrar a RNA proposta à versão final do *software* de mobilidade urbana SIMA Mobilidade Multimodal.

7.3 Produções Acadêmicas

Durante a realização deste trabalho, no âmbito do projeto SIMA, foram aprovados alguns artigos e registrados alguns *softwares*, inclusive os propostos nesta dissertação. Destaca-se que uma parte do desenvolvimento desta dissertação foi publicada em dois artigos. O primeiro intitulado "**Redes Neurais Artificiais Aplicada na Mobilidade Urbana de um Smart Campus**" e o segundo intitulado "**Norte Rotas: Um Software para Planejamento de Rotas de Pedestres em Cenário de Smart Campus**", ambos apresentados e publicados nos

Anais da XI Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2021. Suas citações são:

- **SÁ, J. S.**; SILVA, E. N.; FURTADO, G. M.; GONÇALVES, L. N.; TOSTES, M. E.; CAVALCANTE, G. P. S.; NASCIMENTO, A. A.; BARROS, F. J. B.; ARAÚJO, J. P. L.; FARIAS, F. S. **Redes Neurais Artificiais Aplicada na Mobilidade Urbana de um Smart Campus**. Anais da XI Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2021. Campina Grande-PB, p. 7-8, 2021.
- **SÁ, J. S.**; SILVA, E. N.; GONÇALVES, L. N.; TOSTES, M. E.; CAVALCANTE, G. P. S.; NASCIMENTO, A. A.; BARROS, F. J. B.; ARAÚJO, J. P. L.; FARIAS, F. S. **Norte Rotas: Um Software para Planejamento de Rotas de Pedestres em Cenário de Smart Campus**. Anais da XI Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2021. Campina Grande-PB, p. 5-6, 2021.

Outras duas produções aprovadas e apresentadas, foram os artigos intitulados "**Algoritmos para Determinação de Rotas e Previsão de Chegada de Ônibus em um Smart Campus**" e "**Estudo de Caso Sobre a Comunicação Entre Dojot e Aplicações Android ou Web**", também publicados nos Anais da XI Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2021. Eles podem ser citados como:

- SILVA, E. N.; **SÁ, J. S.**; GONÇALVES, L. N.; TOSTES, M. E.; CAVALCANTE, G. P. S.; NASCIMENTO, A. A.; BARROS, F. J. B.; ARAÚJO, J. P. L.; FARIAS, F. S. **Algoritmos para Determinação de Rotas e Previsão de Chegada de Ônibus em um Smart Campus**. Anais da XI Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2021. Campina Grande-PB, p. 21-22, 2021.
- **SÁ, J. S.**; FURTADO, G. M.; PORTILHO, E. J. G.; GONÇALVES, K. R. G.; TOSTES, M. E.; CAVALCANTE, G. P. S.; NASCIMENTO, A. A.; BARROS, F. J. B.; ARAÚJO, J. P. L.; FARIAS, F. S. **Estudo de Caso Sobre a Comunicação Entre Dojot e Aplicações Android ou Web**. Anais da XI Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2021. Campina Grande-PB, p. 26-27, 2021.

Nos Anais da X Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2020, também foi publicado um artigo, cujo o título é "**Internet das Coisas com Banco de Dados Relacional e em Tempo Real**", e é citado como:

- SILVA, E. N.; **SÁ, J. S.**; GONÇALVES, L. N.; TOSTES, M. E.; CAVALCANTE, G. P. S.; NASCIMENTO, A. A.; ARAÚJO, J. P. L.; BARROS, F. J. B.; FARIAS, F. S.

Internet das Coisas com Banco de Dados Relacional e em Tempo Real. Anais da X Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2020. Natal-RN, p. 161–162, 2020.

Além dos artigos publicados, foram feitos os registros dos dois *softwares* apresentados nesta dissertação. Eles foram registrados pelo Instituto Nacional da Propriedade Industrial (INPI). Eles podem ser citados como:

- **SÁ, J. S.**; GONCALVES, L. N.; MELO, L. A.; SILVA, E. N.; MACHADO, C. A. S.; TOSTES, M. E. L.; BEZERRA, U. H.; CAVALCANTE, G. P. S.; BARROS, F. J. B.; ARAUJO, J. P. L.; FARIAS, F. S.; ALCANTARA NETO, M. C.; CRUZ, H. A. O.; BATALHA, I. S. **Software de Planejamento de Rotas para Pedestres em Smart Campus - Norte Rotas**. Titular: Norte Energia S.A; Universidade Federal do Pará; BYD Energy do Brasil LTDA; ABB Eletrificação LTDA. Procurador: José Carlos Vaz e Dias. Número do registro: BR512022000518-2, data de registro: 11/02/2022, Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial.
- GONCALVES, L. N. ; MELO, L. A. ; SILVA, E. N. ; MACHADO, C. A. S. ; **SÁ, J. S.**; TOSTES, M. E. L. ; BEZERRA, U. H. ; CAVALCANTE, G. P. S. ; BARROS, F. J. B. ; ARAUJO, J. P. L. ; FARIAS, F. S. ; ALCANTARA NETO, M. C. ; CRUZ, H. A. O. ; BATALHA, I. S. ; MOCBEL, M. A. R. ; CARDOSO, C. M. M. ; CARVALHO, J. A. R. ; MACHADO, A. A. ; BARBOSA, Y. H. S. ; LOPES, A. V. R. ; LIMA, W. G. . **SIMA Mobilidade Multimodal**. Titular: Norte Energia S.A; Universidade Federal do Pará; BYD Energy do Brasil LTDA; ABB Eletrificação LTDA. Procurador: José Carlos Vaz e Dias. Número do registro: BR512022001201-4, data de registro: 28/04/2022, Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial.

No projeto SIMA, houve ainda um outro *software* registrado com a participação do autor desta dissertação, que foi:

- GONCALVES, L. N.; MELO, L. A.; **SÁ, J. S.**; SILVA, E. N.; MACHADO, C. A. S.; TOSTES, M. E. L.; BEZERRA, U. H.; CAVALCANTE, G. P. S.; BARROS, F. J. B.; ARAUJO, J. P. L.; ALCANTARA NETO, M. C.; BARBOSA, Y. H. S.; BATALHA, I. S.; CRUZ, H. A. O.; LOPES, A. V. R.; FARIAS, F. S.; LIMA, W. G. **Software de Planejamento de Redes IoT para Tecnologia de Longo Alcance**. Titular: Norte Energia S.A; Universidade Federal do Pará; BYD Energy do Brasil LTDA; ABB Eletrificação LTDA. Procurador: José Carlos Vaz e Dias. Número do registro:

BR512021003138-5, data de registro: 30/09/2021, Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial.

REFERÊNCIAS

AHMED, L.; *et al*. **Solving urban transit route design problem using selection hyper-heuristics**. European Journal of Operational Research, v. 274, n. 2, p. 545-559, 2019.

BARCZYSZYN, G. L. **Sistema colaborativo para planejamento de rotas para cadeirantes**. Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná, 2019.

CASTRO, V. C.; *et al.* **Online Monitoring For Public Transport Using Mobile Applications**. IEEE Latin America Transactions, v. 16, n. 6, p. 1780-1786, 2018.

CENTRO DE COMPETÊNCIA EM SOFTWARE LIVRE DA UFPA. **UFPA Digital**. Google Play. Disponível em: <https://play.google.com/store/apps/details?id=br.ufpa.ccsl.ufpadigital>. Acesso em: 27 de junho de 2022.

CITTAMOBI. **Cittamobi: Horários de ônibus**. Google Play. Disponível em: <https://play.google.com/store/apps/details?id=br.com.cittabus>. Acesso em: 25 de junho de 2022.

COHEN, A.; DALYOT, S. **Machine-learning prediction models for pedestrian traffic flow levels: Towards optimizing walking routes for blind pedestrians**. Transactions in GIS, v. 24, n. 5, p. 1264-1279, 2020.

CPQD. **1. Arquitetura**. Documentação da dojot, 2020. Disponível em: <https://dojotdocs.readthedocs.io/pt_BR/latest/architecture.html>. Acesso em: 15 de junho de 2022.

_____. **2. Utilizando a API da dojot**. Documentação da dojot, 2020. Disponível em: <https://dojotdocs.readthedocs.io/pt_BR/stable/using-api-interface.html>. Acesso em: 17 de junho de 2022.

_____. **7. Dúvidas Mais Frequentes**. Documentação da dojot, 2020. Disponível em: <https://dojotdocs.readthedocs.io/pt_BR/latest/faq/faq.html>. Acesso em: 14 de junho de 2022.

_____. **Você conhece a dojot?**. Dojot, 2017. Disponível em: <https://dojot.com.br/sobre-a-dojot-iot/>. Acesso em: 14 de junho de 2022.

DEILAMI, K.; *et al.* **Allowing users to benefit from tree shading: Using a smartphone app to allow adaptive route planning during extreme heat**. Forests, v. 11, n. 9, p. 998, 2020.

FAYYAD, U.; *et al*. **From Data Mining to Knowledge Discovery in Databases**. in Artificial Intelligence Magazine, American Association for Artificial Intelligence, 1996.

GOOGLE LLC. **Google Maps**. Google Play. Disponível em: <https://play.google.com/store/apps/details?id=com.google.android.apps.maps>. Acesso em: 25 de junho de 2022.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2ª edição. Porto Alegre: Bookman, 2002.

HEUSER, C. A. **Projeto de banco de dados**. 6ª edição, vol. 4. Porto Alegre: Bookman Editora, 2009.

IBM CLOUD EDUCATION. **REST APIs**. Publicado em: 6 de abril de 2021. Disponível em: <https://www.ibm.com/cloud/learn/rest-apis>. Acesso em: 31 de maio de 2022.

INGLE, D.; BAGWAN, A. B. **Real-Time Analysis and Simulation of Efficient Bus Monitoring System**. In: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE, p. 128-133, 2018.

JOSHI, A.; *et al*. **Likert scale: Explored and explained**. British journal of applied science & technology, v. 7, n. 4, p. 396, 2015.

JÚNIOR, A. S. C.; BAPTISTA, V. F. **Mobilidade urbana, políticas públicas e o Plano Diretor do município de São Gonçalo**. Revista Baru-Revista Brasileira de Assuntos Regionais e Urbanos, v. 4, n. 1, p. 31-46, 2018.

LIANG, S.; *et al.* **An improved ant colony optimization algorithm based on context for tourism route planning**. PloS one, v. 16, n. 9, p. e0257317, 2021.

LOBATO, E. P. S. **Implantação de um middleware IoT escalável para aplicações de Mobilidade Elétrica Multimodal**. Dissertação de Mestrado. Mestrado em Engenharia Elétrica. Universidade Federal do Pará, 2022.

MAIA, C. **Aplicativo monitora a rota do ônibus circular pela Universidade**. Assessoria de Comunicação da UFPA. Universidade Federal do Pará. Publicado em: 29 de dezembro de 2017. Disponível em: <https://www.portal.ufpa.br/index.php/ultimas-

noticias2/7812-aplicativo-monitora-a-rota-do-onibus-circular-pela-universidade>. Acesso em: 27 de junho de 2022.

MELO JUNIOR, J. G.; *et al.* **Assessment of the Sustainability of Agroecosystems in the Amazon Region Using Neural Artificial Networks**. In: IEEE Latin America Transactions, vol. 14, no. 8, p. 3804-3810, 2016.

MENDES, V. **Diretoria de Segurança Cataloga Objetos e Documentos Perdidos para Facilitar Devolução aos Donos**. Assessoria de Comunicação da UFPA, Publicado em: 02 de agosto de 2018. Disponível em: <https://portal.ufpa.br/index.php/ultimas-noticias2/8739-diretoria-deseguranca-da-universidade-cataloga-objetos-e-documentos-perdidopara-facilitar-devolucao-aos-donos>. Acesso em: 08 de junho de 2022.

MISHRA, M.; SRIVASTAVA, M. **A View of Artificial Neural Network**. In: 2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014), IEEE, p. 1-3, 2014.

MOOVIT. **Moovit: Horários de Ônibus**. Google Play. Disponível em: <https://play.google.com/store/apps/details?id=com.tranzmate>. Acesso em: 25 de junho de 2022.

NASCIMENTO, C. F.; *et al*. **Neural Network-Based Approach for Identification of the Harmonic Content of a Nonlinear Load in a Single-Phase System**. In: IEEE Latin America Transactions, v. 8, n. 1, p. 65-73, 2010.

NOVACK, T.; *et al.* **A system for generating customized pleasant pedestrian routes based on OpenStreetMap data**. Sensors, v. 18, n. 11, p. 3794, 2018.

OPACH, T.; *et al*. **Pedestrian routing and perspectives: WayFinder's route down the lane—Come on with the rain**. ISPRS International Journal of Geo-Information, v. 10, n. 6, p. 365, 2021.

_____. **Towards a Route Planner Supporting Pedestrian Navigation in Hazard Exposed Urban Areas.** In: Proceedings of the 17th International Conference on Information Systems for Crisis Response and Management, ISCRAM, 2020.

PARKER, D. **Learning Logic**. Intention Report, Stanford University, File 1, Office of Technology Licensing, p. S81-64, Stanford, 1982.

PHP GROUP. **What is PHP?**. Documentation. Disponível em: <https://www.php.net/manual/en/intro-whatis.php>. Acesso em: 11 de junho de 2022.

PUJA, I.; POSCIC, P.; JAKSIC, D. **Overview and Comparison of Several elational Database Modelling Metodologies and Notations**. 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2019.

REDHAT. **What is a REST API?**. Publicado em: 08 de maio de 2020. Disponível em: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>. Acesso em: 31 de maio de 2022.

RIBEIRO, H. M. C.; *et al.* **Water Quality Monitoring in Large Reservoirs Using Remote Sensing and Neural Networks**. In: IEEE Latin America Transactions, vol. 6, n. 5, p. 419-423, 2008.

RUMELHART, D. E.; *et al.* **Learning representations by back-propagating errors**. Nature, vol. 323, p. 533–536, 1986.

SA, J.; *et al*. **Online monitoring of buses information using KNN, ATR and DMC algorithms**. IEEE Latin America Transactions, p. 564-572, 2019.

SILVA, E. N.; *et al*. **Internet das Coisas com Banco de Dados Relacional e em Tempo Real**. Anais da X Conferência Nacional em Comunicações, Redes e Segurança da Informação – Encom 2020. Natal-RN, 2020.

SOUSA, P. M. S.; *et al.* **Uma Infraestrutura para o Monitoramento e Predição de Rotas e Paradas de Ônibus no Transporte Universitário**. In: Workshop De Computação Urbana (COURB), Porto Alegre: Sociedade Brasileira de Computação, p. 111-124, 2019.

UFPA. **UFPA em Números 2021**. Universidade Federal do Pará, Belém–Pará, 2021. Disponível em: <http://www.ufpanumeros.ufpa.br/>. Acesso em: 14 de junho de 2022.

VELÁZQUEZ, R.; *et al*. **An outdoor navigation system for blind pedestrians using GPS and tactile-foot feedback**. Applied Sciences, v. 8, n. 4, p. 578, 2018.

WAZE. **Waze – GPS e Trânsito ao vivo**. Google Play. Disponível em: <https://play.google.com/store/apps/details?id=com.waze>. Acesso em: 25 de junho de 2022.

WERBOS, P. **Beyond Regression: New Tools for Prediction and Analysis**. Ph.D. Thesis in the Behavioral Sciences. Harvard University, Cambridge, 1974.

YAO, Y.; *et al*. **Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city**. IEEE Transactions on Vehicular Technology, v. 67, n. 11, p. 10307-10318, 2018.

YIJUN, L.; *et al.* **Artificial neural networks applied in environmental quality assessment.** In: 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), p. 546-549, Chengdu, 2010.

ZHAMANOV, A.; *et al*. **IoT smart campus review and implementation of IoT applications into education process of university**. 2017 13th International Conference on Electronics, Computer and Computation (ICECCO). IEEE, 2017.

ZHONG, S. *et al*. **The optimization of bus rapid transit route based on an improved particle swarm optimization**. Transportation Letters, v. 10, n. 5, p. 257-268, 2018.