UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DAINF - DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

LINCOLN BATISTA
RENAN DE SOUZA ANTUNES

MAPA INTERNO INTERATIVO DO CAMPUS UTFPR-CT

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2021

LINCOLN BATISTA
RENAN DE SOUZA ANTUNES

MAPA INTERNO INTERATIVO DO CAMPUS UTFPR-CT

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Robson Ribeiro Linhares
DAINF - Departamento Acadêmico de Informática - UTFPR

CURITIBA
2021

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO - TCC

MAPA INTERNO INTERATIVO DO CAMPUS UTFPR-CT

Por

Lincoln Batista e Renan de Souza Antunes

Monografia apresentada às **16** horas do dia **20 de Maio de 2021** como requisito parcial para conclusão do Curso de **Bacharelado em Sistemas de Informação** da Universidade Tecnológica Federal do Paraná, Câmpus Curitiba. O(a)(s) aluno(a)(s) foi(ram) arguido(a)(s) pelos membros da Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho **APROVADO.**

Banca examinadora:

Paulo Cézar Stadzisz	Membro
João Alberto Fabro	Membro
Robson Ribeiro Linhares	Orientador
Leyza Elmeri Baldo Dorini	Professor(a) responsável TCCII

RESUMO

BATISTA, Lincoln; ANTUNES, Renan de Souza. Mapa Interno Interativo do Campus UTFPR-CT. 2021. 47 f. Trabalho de Conclusão de Curso – Curso de Bacharelado em Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

A utilização de mapas interativos aumentou consideravelmente com a popularização de smartphones, sendo comum depender de mapas para localização e definição de trajetos para deslocamentos de médias e grandes distâncias, porém também existe uma categoria de mapas que tratam de curtas distâncias em ambientes como shoppings, aeroportos e universidades, esses mapas são conhecidos como mapas internos ou *indoor* e representam pequenas áreas com um alto nível de detalhe. Diversos problemas cotidianos reforçam a importância para a elaboração deste trabalho, por exemplo, estudantes e professores podem se atrasar devido a falta de informações sobre a localização de uma determinada sala. Neste trabalho é apresentado o desenvolvimento e as funcionalidades de um protótipo de mapa interativo interno do câmpus Curitiba Sede Centro da Universidade Tecnológica Federal do Paraná. Os resultados obtidos demonstram que o desenvolvimento de mapas internos ainda é uma tarefa trabalhosa, porém possível de ser feita por meio de software livre e de código aberto, sem depender de ferramentas comerciais.

Palavras-chave: Mapas Interativos. Mapas Internos. Web GIS. Grafos. Problema do Menor Caminho.

ABSTRACT

BATISTA, Lincoln; ANTUNES, Renan de Souza. Campus UTFPR CT's Interactive Indoor Map. 2021. 47 f. Trabalho de Conclusão de Curso – Curso de Bacharelado em Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

The use of indoor maps increased considerably with the popularization of smartphones, becoming common to depend on maps for localization and route planning for medium and long distance trips, however there is also a map category that addresses short distances in places like shopping malls, airports and universities. Those maps are known as indoor maps and represent small areas with a high level of detail. Several daily problems reinforce the importance of this work's elaboration, for instance, students and teachers may be delayed due to the lack of information about the location of a particular room. In this work we present a prototype of an interactive indoor map for the Curitiba (CT) campus of the Federal University of Technology - Paraná. The results obtained show that the development of indoor maps is still a laborious task, however it can be done without relying on commercial tools, through free and open source software.

Keywords: Interactive Maps. Indoor Maps. Web GIS. Graphs. Shortest Path Problem.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BLE	*Bluetooth Low Energy*
CLI	*Command Line Interface*
GIS	*Geographic Information System*
GPS	*Global Positioning System*
OSM	*OpenStreetMap*
PBF	*Protocol Buffers File*
REST	*Representational State Transfer*
RF	*Radio Frequency*
RFID	*Radio Frequency Identification*
RSS	*Received Signal Strength*
VGI	*Volunteered Geographic Information*

SUMÁRIO

1 INTRODUÇÃO

Tecnologias para produção de mapas têm sido úteis para a humanidade por muitos séculos. À medida que foi aumentando a demanda por esse tipo de informação, técnicas cada vez mais sofisticadas de mapeamento tomaram forma, possibilitando diversas evoluções, do mapa esculpido em rocha a sistemas como o Google Maps. A maioria dos mapas interativos são desenvolvidos com a finalidade de fornecer informações de ambientes externos, em contrapartida, o mesmo público atendido por essas tecnologias também trafega por edificações o que pode aumentar a demanda por aplicações dedicadas a esse tipo de ambiente.

A complexidade dos espaços internos de grandes edificações como aeroportos, universidades e shoppings podem dificultar que pessoas encontrem seus destinos causando diversos tipos de problemas, por exemplo um cliente que não encontrou a loja que queria pode desistir de fazer a compra, um passageiro pode perder seu voo por não encontrar o portão de embarque em tempo hábil ou um visitante em uma universidade pode não encontrar a sala de uma palestra e desistir de participar do evento. Por isso, desenvolver e disponibilizar mapas internos interativos para este tipo de espaço pode auxiliar os usuários a encontrar e definir o melhor trajeto até o seu destino, minimizando ou até mesmo resolvendo estes problemas.

A necessidade de mapear o interior das edificações com todos os andares, salas, corredores e interligações entre os andares como escadas, elevadores e rampas resulta em uma grande exigência por representações com maior nível de detalhe, o qual não pode ser obtido por um mapa comum, portanto são utilizados os mapas de interiores, também conhecidos como mapas *indoor*, que são o principal assunto deste trabalho. Diferentemente das representações de mapas comuns que apresentam vastas áreas como uma cidade, um estado ou até mesmo o planeta inteiro, os mapas *indoor* geralmente tratam de uma pequena área como um aeroporto, um shopping ou um campus de universidade.

Hodge (2020) apresenta a tecnologia da MazeMap de mapas internos aplicados a universidades como uma solução para atenuar a disseminação do COVID-19 pois, além de ajudar os estudantes e professores a transitarem pelo local, permite informar a disponibilidade de salas, visualizar onde as pessoas estão situadas dentro do prédio, por meio de mapas de calor e encoraja as pessoas a navegarem diretamente ao seu destino, evitando aglomerações em corredores e lugares pequenos.

Os mapas interativos são importantes em diversas áreas do conhecimento como na medicina para mapear doenças (GAO et al., 2008) e estudar pandemias como a do SARS-CoV-2 (BOULOS; GERAGHTY, 2020) inclusive com mapas sendo disponibilizados para o público em geral como o *COVID-19 Map* da Johns Hopkins University[1]. Órgãos

[1]<https://coronavirus.jhu.edu/map.html>

governamentais também utilizam mapas interativos, por exemplo o Instituto Nacional de Pesquisas Espaciais (INPE) desenvolve o projeto Terrabrasilis que monitora o desmatamento no território brasileiro[2]. O *United States Geological Survey* (USGS)[3] possui diversos projetos com dados geográficos e espaciais que incluem mapas da qualidade da água, atividade vulcânica, inundações, entre outros.

O aprimoramento da tecnologia e internet permitiu a transformação dos Sistemas de Informação Geográfica (GIS) em Web GIS que se tornou parte fundamental de tecnologias de mapeamento interativo. A atual popularidade dos Web GIS pode ser percebida pela quantidade de diferentes aplicações comerciais que utilizam essas tecnologias como por exemplo Google Maps, Waze, Uber, Pokémon GO, iFood, entre outras. Também existem diversas iniciativas de mapeamento colaborativo como OpenStreetMap (OPENSTREETMAP, 2020a), OpenAerialMap (OPENAERIALMAP, 2021) e Mapillary (MAPILLARY, 2021). O projeto OSM se tornou uma das maiores fontes de VGI (Volunteered Geographic Information), em 2011 contava com mais de cem mil contribuidores globais (NEIS; ZIPF, 2012).

Um problema específico de mapas internos é a navegação. A maioria dos serviços de navegação disponíveis atualmente utilizam o GPS, ou seja, são dedicados para ambientes externos. Isso ocorre pela facilidade do uso desse sinal, tornando essas tecnologias ainda mais acessíveis (VERMA et al., 2016). No entanto, devido a problemas de precisão ocasionados pela diminuição da qualidade do sinal, não é recomendado o uso do GPS para localização em ambientes internos (RUSTAGI; YOO, 2018a).

De acordo com Ohrt e Turau (2013), não há tecnologia gratuita que possibilite a fácil criação de mapas internos que incluem a localização atual e rotas. O autor ainda ressalta que sistemas que utilizam GPS não são adequados para estruturas granulares como salas dentro de edifícios, além disso, os dados são armazenados em servidores de terceiros, dificultando as restrições de segurança e fazendo com que as atualizações e manutenções se tornem ainda mais problemáticas ou lentas.

Apesar das dificuldades existem alternativas para esse problema. Um exemplo é a solução desenvolvida por Rustagi e Yoo (2018b) que consiste no desenvolvimento de um aplicativo de navegação interna por realidade aumentada que utiliza *vector tile sets* numa plataforma de objetos 3D de jogos (Unity) para criar a navegação. Parâmetros pré-definidos como pontos de sincronização (*sync points*) e pontos de destino (*destination points*) foram utilizados para possibilitar a localização. Em sua conclusão, o autor afirma que sua solução é mais efetiva que sistemas de navegação interna baseados em sinal de GPS no interior de prédios.

[2] <http://terrabrasilis.dpi.inpe.br/>
[3] <https://www.usgs.gov/>

1.1 Objetivo geral

O objetivo geral deste trabalho de conclusão de curso é desenvolver um protótipo de mapa interativo interno do campus UTFPR Curitiba - Sede Centro para auxiliar a navegação de pessoas pelo campus.

1.2 Objetivos específicos

- Realizar pesquisa, leitura e filtragem de material teórico;
- Pesquisar, analisar e definir as ferramentas que serão utilizadas;
- Mapear, total ou parcialmente, o campus UTFPR Curitiba - Sede Centro utilizando uma ferramenta escolhida no objetivo anterior;
- Implementar ou adaptar um algoritmo de busca de melhor caminho em áreas internas com diversos edifícios;
- Modificar o algoritmo de busca para possibilitar o roteamento de caminhos sem obstáculos para pessoas com mobilidade reduzida;
- Pesquisar, analisar a viabilidade e, caso determinado que seja possível, utilizar uma técnica de navegação interna para apresentar o posicionamento em tempo real;
- Implementar e testar um protótipo que utilize os dados, algoritmos e técnicas pesquisados;

1.3 Organização do Trabalho

Este trabalho está organizado em 6 capítulos. O capítulo 2 apresenta os conceitos teóricos, técnicas e especificações necessários para o desenvolvimento do trabalho, além de uma análise de trabalhos correlatos e do estado da arte. O capítulo 3 apresenta a metodologia e os recursos de software e hardware. O capítulo 4 descreve as etapas do processo de desenvolvimento do protótipo. No capítulo 5 são apresentados os resultados obtidos com o desenvolvimento. Finalmente, o capítulo 6 apresenta as considerações finais do trabalho e sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os principais conceitos teóricos, técnicas e especificações que foram utilizados no desenvolvimentos deste trabalho. Também são apresentados trabalhos correlatos e o estado da arte.

2.1 Mapas

Os mapas são utilizados em qualquer etapa do processo de manipulação de dados geoespaciais, além disso, o desenvolvimento da tecnologia vem trazendo a área de mapeamento cada vez mais próximos de outras disciplinas que, por sua vez, influenciam as abordagens de mapeamento (KRAAK, 2004). Segundo Agrawal e Gupta (2017), mapas são utilizados há vários séculos como fonte de informação espacial com a finalidade de solucionar problemas e seu uso aumentou muito nas últimas décadas devido a avanços tecnológicos na área.

Para representar a superfície terrestre, seja em papel ou em formato digital, é necessário utilizar uma projeção cartográfica. As projeções definem formas de representar a curvatura da terra em uma superfície plana, porém este processo pode causar diferentes tipos de distorções nas áreas do mapa. A projeção mais comumente utilizada em mapas digitais é a Web Mercator, que foi desenvolvida no início do século 21 e é uma variação da projeção Mercator tradicional. Essa projeção é utilizada pelos principais serviços de mapas web, como Google Maps e Microsoft Bing Maps (JENNY, 2012). Battersby et al. (2014) ressalta as vantagens da projeção Web Mercator, dentre elas o fato dessa projeção ser não conformal e da distorção de ângulos locais ser mínima e, além disso, para mapas web a distorção angular varia de acordo com a ampliação aplicada em uma determinada localização.

Como o objetivo deste trabalho é o desenvolvimento de um mapa interno que representa uma pequena área, os cálculos de distância entre dois pontos do mapa irão considerar a superfície da área como sendo perfeitamente plana, não sendo levado em conta a interferência da curvatura da Terra no cálculo (KARNEY, 2013).

2.2 GIS

Os Sistemas de Informação Geográfica (SIG) ou, em inglês, *Geographic Information Systems* (GIS) apresentam diversas definições, entre elas Devine e Field (1986) define GIS como “simplesmente uma tecnologia para expandir o uso de mapas”. Já Cooperative e Collins (1988) define GIS como “uma tecnologia da informação que armazena, analisa e exibe dados espaciais e não espaciais”. De acordo com Rigaux, Scholl e Voisard (2002), GIS “armazena dados geográficos, recupera e combina estes dados para criar novas representações

do espaço geográfico, fornece ferramentas para análises espaciais e realiza simulações para auxiliar usuários a organizar seu trabalho em várias áreas [...]". Cowen (1988) afirma que GIS é "um sistema de apoio à decisão envolvendo a integração de dados espaciais referenciados num ambiente de resolução de problemas". Já Burrough (1986), define GIS como sendo "um poderoso conjunto de ferramentas para coletar, armazenar, recuperar, transformar e apresentar dados espaciais do mundo real".

Atualmente, os GIS tornaram-se ainda mais acessíveis com opções gratuitas e de código aberto como QGIS (QGIS, 2021), GeoServer (GEOSERVER, 2021) e PostGIS (POSTGIS, 2021) e com soluções proprietárias que funcionam em arquiteturas *cloud* que possuem interfaces modernas e intuitivas como o ArcGIS (ARCGIS, 2021), além de soluções de mapeamento 3D como o AutoCAD Map 3D (AUTODESK, 2021). Originalmente, os GIS eram ferramentas monolíticas, de custo elevado e com um alto grau de complexidade para implementação, usados principalmente em grandes corporações e por usuários altamente treinados. Com o avanço da internet, essas tecnologias que até então eram utilizadas exclusivamente em um contexto offline, foram adaptadas para também funcionar na web, surgindo assim os Web GIS (AGRAWAL; GUPTA, 2017).

O autor também define a arquitetura dos Web GIS como sendo aplicações que utilizam os navegadores web como cliente para enviar requisições e servidores web para respondê-las. Um exemplo disso seria a aplicação Web GIS para visualização de dados geoespaciais do Pantanal apresentada por CRUZ, SILVA e MACÁRIO (2014). A arquitetura dessa aplicação consiste no uso de um banco de dados geoespacial, integrado a web services RESTful de acesso a dados geoespaciais no padrão OGC (Open Geospatial Consortium) e ferramentas para visualização gráfica cujos dados são referenciados em endereços URI (Uniform Resource Identifier), assim é feito um mapeamento de cada endereço em consultas SQL ou a requisições específicas nos servidores OGC. Os resultados destas consultas ou requisições são recebidos pelo servidor e, caso necessário, convertidos para formatos adequados e retornados à ferramenta solicitante do dado.

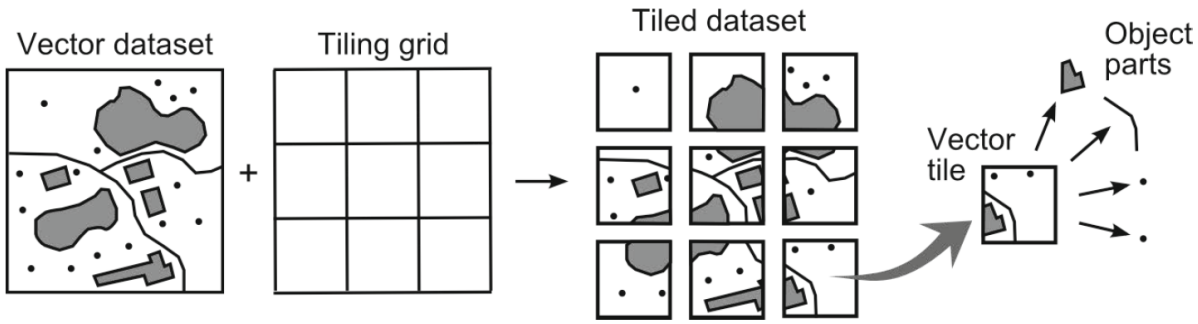
2.3 Tiled Web Maps

Os *Tiled Web Maps* são mapas web que são transferidos do servidor para a aplicação cliente utilizando um método que divide o mapa em áreas, chamadas de *tiles*, e diferentes níveis de detalhe (zoom). Cada *tile* é subdividido em novos *tiles*, formando o próximo nível de detalhe até o nível de detalhe mais alto. O benefício de usar *tiles* no lugar de outros métodos como o *Web Map Service* (WMS)[1] é a diminuição da transferência de dados entre cliente e servidor e tempo de carregamento da aplicação dado que só é preciso requisitar os *tiles* que serão efetivamente exibidos e não o mapa completo. Existem dois tipos de *tiles* os *raster tiles* que utilizam imagens bitmap e os *vector tiles* que utilizam

[1] <https://www.ogc.org/standards/wms>

gráficos vetoriais (ANTONIOU; MORLEY; HAKLAY, 2009).

Figura 1 – Ilustração da geração de *vector tiles*.



Fonte: Gaffuri (2012)

2.3.1 Raster Tiles e Vector Tiles

Os *raster tiles* utilizam rasters ou bitmaps que são um formato de dados utilizado para gerar imagens que contém uma matriz ou mapa de bits (bitmap) que representam os valores de cada pixel da imagem. Por esse tipo de *tile* ser uma imagem, não é possível interagir diretamente com os elementos do mapa, limitando a interatividade que muitas vezes é essencial para a análise de dados espaciais (ANTONIOU; MORLEY; HAKLAY, 2009).

Os *vector tiles* utilizam dados vetoriais, que são armazenados como pontos, linhas e polígonos em um plano Cartesiano que são renderizados na aplicação cliente ou no servidor. Segundo Gaffuri (2012), alguns dos benefícios de utilizar *vector tiles* no lugar de *raster tiles* são:

- Interagir e consultar dados de um objeto específico do mapa;
- Processamentos geográficos podem ser realizados diretamente na aplicação cliente, unificando Mapeamento Web e Web GIS;
- Alterar o estilo visual do mapa como um todo ou de objetos específicos diretamente na aplicação cliente;
- Possibilidade de customizar legendas trocando o idioma, por exemplo e melhorias na usabilidade dado que mudanças na orientação do mapa podem manter as legendas da mesma forma, sem atrapalhar na sua leitura;
- Criar *mash-ups* de mapas com dados de diversas fontes diferentes;
- Auxiliar no desenvolvimento de mapas colaborativos e VGI, dado que a interação entre usuário e fonte de dados é aperfeiçoada;
- Publicações e atualizações mais rápidas de conteúdo para mapas, dado que não é necessário pré-processar e fazer cache dos *tiles*;
- Desenvolvimento de novas técnicas de visualização cartográfica, que não eram possíveis com os *raster tiles*;

2.3.2 Especificações

O Mapbox Vector Tile Specification (MAPBOX, 2016b) é uma especificação desenvolvida pela empresa Mapbox que define um padrão para o armazenamento e codificação de *vector tiles*. A especificação é disponibilizada na licença Creative Commons Attribution 3.0 United States License. A especificação conta com dezenas de ferramentas, integrações e aplicações já desenvolvidas[2], especialmente por sua licença permitir o uso sem restrições, pagamento de royalties ou outros requisitos especiais.

A especificação define que os *vector tiles* sejam codificados utilizando a tecnologia *Protocol Buffers* (PBF) desenvolvida pelo Google que permite a serialização de dados estruturados, similar aos formatos XML e JSON, porém com arquivos menores. O sufixo de arquivo utilizado pela especificação é *.mvt*.

O Mapbox Style Specification (MAPBOX, 2016a) é outra especificação desenvolvida pela Mapbox que define os estilos que serão aplicados aos mapas renderizados a partir dos *vector tiles*. A especificação define o formato de arquivo utilizado (JSON) e os diversos objetos e atributos aceitos. É possível definir estilos para diferentes classes de objetos do mapa inclusive com a opção de alterar o estilo de acordo com o nível de detalhe (zoom) do mapa.

2.4 OpenStreetMap

O projeto OpenStreetMap (OSM) foi iniciado em 2004 na University College of London por Steve Coast, com o objetivo de disponibilizar acesso gratuito a dados geográficos. O projeto utiliza o conceito de *crowdsourcing* em que um grande número de indivíduos se une para realizar uma tarefa que é difícil de ser automatizada ou com implementação dispendiosa. No caso do projeto OSM essa tarefa é mapear o planeta (HAKLAY; WEBER, 2008).

No momento da escrita deste trabalho o projeto OSM conta com mais de 6.7 milhões de usuários registrados e mais de 8 bilhões de pontos de GPS (OPENSTREETMAP, 2020c). O projeto utiliza a licença Open Data Commons Open Database License (ODbL) que permite a cópia, transmissão e adaptação dos dados desde que os créditos do projeto OSM e seus contribuidores sejam devidamente atribuídos.

Os dados do OSM seguem um padrão similar dos dados vetoriais dos *vector tiles* com pontos (*nodes*), linhas e curvas (*ways*) e polígonos (*areas*) sendo usados para descrever todos os objetos da superfície terrestre como edificações, árvores, praças, ruas, trilhas, rios, lagos, etc. Para diferenciar os objetos são utilizados pares chave-valor que definem atributos como tipo da área (construções, parques, lagos, etc.), uso da área (universidade, restaurante, etc.), nome do local, endereço, recursos de acessibilidade, entre outros. Por exemplo, para mapear uma construção de dois andares que tem um restaurante de nome “Restaurante

[2] <https://github.com/mapbox/awesome-vector-tiles>

Universitário" seriam adicionadas as tags *building=yes; levels=2; amenity=restaurant; name=Restaurante Universitário;* na área que define a construção do restaurante.

O arquivo com todos os dados do projeto OSM, chamado de *Planet OSM*, tem tamanho de mais de 1 Terabyte quando descompactado (OPENSTREETMAP, 2020b), para evitar trabalhar com arquivos muito grandes é possível utilizar extratos do mapa, que são regiões específicas como continentes ou países (GEOFABRIK, 2020). Também é possível exportar pequenas regiões de tamanho customizado diretamente pelo site do projeto.

Existem diversas ferramentas que fazem a conversão do formato OSM para o formato Mapbox Vector Tile seja diretamente ou realizando a criação de uma base de dados GIS durante o processo.

2.5 Problema do Menor Caminho

Um recurso muito comum em mapas web é o cálculo de rota, os principais mapas como Google Maps, Apple Maps, Bing Maps e OpenStreetMap oferecem esta funcionalidade, alguns inclusive oferecem opções extras como previsão do tempo de deslocamento com análise em tempo-real do trânsito, trajeto passo a passo, entre outras funcionalidades todas disponíveis com tempos de resposta baixos. O que permite esses tempos de resposta baixos são os algoritmos de busca de menor caminho (*shortest path search*) altamente otimizados. A busca de menor caminho consiste em encontrar o menor caminho ótimo, ou seja, o caminho em que a soma das arestas seja a menor possível, entre dois vértices de um grafo. Todos os algoritmos analisados neste trabalho consideraram somente grafos com arestas de peso positivo, direcionados ou não.

Para realizar a busca de menor caminho em mapa é preciso transformar os dados do mapa em um grafo G = (V, E) onde os cruzamentos são os vértices ou nós, os segmentos das ruas entre os cruzamentos são as arestas e os pesos das arestas são o tempo de deslocamento ou comprimento do segmento. No caso de mapas *indoor* é possível considerar os corredores como arestas e os "cruzamentos" dos corredores como os nós, elevadores, escadas e rampas também podem ser transformados em nós.

O problema do menor caminho é antigo na computação, uma das soluções mais conhecidas é o algoritmo de Dijkstra et al. (1959) que consiste em visitar os nós vizinhos até que todas as possibilidades sejam esgotadas ou que outro caminho tenha um custo menor e repetir este processo até encontrar o nó de destino. Outro algoritmo bastante conhecido é o A* desenvolvido por Hart, Nilsson e Raphael (1968) que utiliza um método similar ao Dijkstra, porém conta com uma heurística que é adicionada ao custo para guiar a busca. Dado que a heurística nunca seja maior que o custo real, o caminho encontrado será ótimo. Muitos dos algoritmos desenvolvidos nos últimos anos ainda utilizam ideias de implementação similares aos dois algoritmos citados anteriormente, porém com adaptações para melhorar o desempenho em grafos com um número de vértices e arestas grande.

Alguns desses algoritmos serão apresentados a seguir.

O algoritmo *Contraction Hierarchies* desenvolvido por Geisberger et al. (2008) funciona em rotas hierárquicas, em os vértices do grafo são ordenados por uma medida de importância, e utiliza o método de contração de nós que consiste em encontrar atalhos entre vértices que contenham um único melhor caminho. Essa etapa de contração diminui a quantidade de nós visitados na etapa de busca e consequentemente reduz o tempo de execução do algoritmo, porém é necessário realizar um pré-processamento do grafo e a medida utilizada para definir a importância dos vértices pode influenciar no resultado. O projeto *Open Source Routing Machine* (OSRM) é uma implementação do algoritmo *Contraction Hierarchies* que funciona com o modelo de dados do projeto OpenStreetMap (LUXEN; VETTER, 2011).

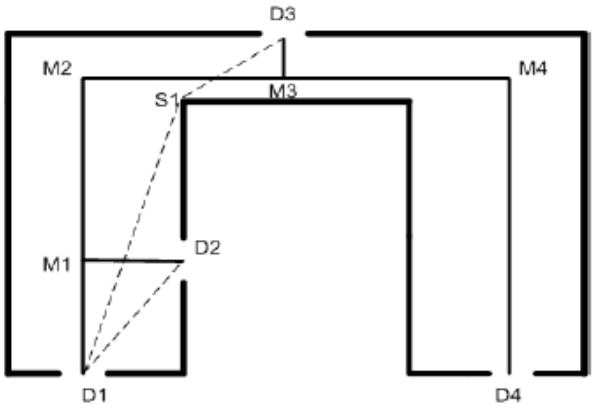
Goldberg e Harrelson (2004) desenvolveram uma modificação do algoritmo A* que realiza um pré-processamento calculando e armazenando a distância do menor caminho entre todos os nós do grafo e "nós marcos" (*landmarks*), que são vértices escolhidos de acordo com um métodos de seleção, a busca então utiliza as distâncias entre vértice de origem e marco e vértice de destino e marco para, a partir da desigualdade de triângulos, aproximar a distância entre os vértices de origem e destino.

Os algoritmos utilizados em mapas *outdoor* também podem ser utilizados em mapas *indoor*, porém dadas as restrições (não direcionado e menor quantidade de vértices e arestas) dos mapas *indoor* é possível utilizar algoritmos que, devido às suas complexidades de tempo ou de espaço, são proibitivos em mapas muito grandes. Esses algoritmos utilizam principalmente métodos de pré-processamento e alguns serão apresentados nos parágrafos a seguir.

Liu e Zlatanova (2011) definem um algoritmo que utiliza uma abordagem chamada de "porta-a-porta" (*door-to-door*), onde as rotas apresentam formatos mais naturais e que realmente são o caminho mais curto, diferentemente das abordagens que utilizam grafos duais que deixam as rotas com formato de *grid*. Nessa abordagem os vértices são as portas e as arestas uma reta entre elas, quando uma porta não tem uma reta sem obstruções até outra porta, são criados vértices auxiliares nos cantos das paredes para permitir desviar dos obstáculos. A figura 2 apresenta uma comparação das duas abordagens.

Dudas, Ghafourian e Karimi (2009) desenvolveram uma ontologia para auxiliar algoritmos de busca de menor caminho para levar em conta as necessidades dos usuários, como dificuldades motoras com uso de cadeiras de rodas ou muletas, deficiências visuais e pessoas idosas com lapsos de memória. Os autores definem dois tipos de rotas, as rotas factíveis, que são as rotas entre um ponto de origem e destino que o usuário pode cruzar dado suas limitações, por exemplo um cadeirante não poder utilizar rotas com escadas, e as rotas confortáveis que são um subconjunto das rotas factíveis que levam em conta a preferência do usuário, por exemplo utilizar escada no lugar do elevador. O algoritmo utilizado é baseado no algoritmo de Dijkstra.

Figura 2 – Comparação das abordagens *door-to-door* (linha tracejada) e grafo dual (linha sólida).



Fonte: Liu e Zlatanova (2011)

Shao et al. (2016) desenvolveram um algoritmo que particiona e combina partes do ambiente *indoor* para formar uma árvore de busca que agrupa os nós caso eles sejam adjacentes, que os autores definiram como sendo áreas com "portas de acesso", ou seja, portas que ligam uma partição do ambiente à outra. Os autores compararam seu resultado à uma matriz de distâncias e conseguiram resultados alguns graus de magnitude menores em relação a complexidade de espaço e de tempo.

2.6 Localização e Navegação Interna

A maioria das técnicas de navegação externa utilizam sistemas de navegação baseado em GPS para localizar um objeto em qualquer área externa, essas técnicas funcionam bem em espaços abertos devido a ausência de obstáculos entre o objeto e o satélite, logo não desempenham tão bem em espaços internos (VERMA et al., 2016).

Diversas técnicas como triangulação de sinais Wi-Fi e Bluetooth, beacons, RFID (Identificação por Radiofrequência) e navegação estimada são utilizados para navegação interna. A fim de exemplificar, serão apresentados alguns estudos que utilizam essas técnicas.

Hoffmann, Werner e Schauer (2016) oferecem suporte para detecção de proximidade via Wi-Fi e apresentam um conceito chamado *Virtual Anchor Point* que consiste em detectar locais que apresentam padrões inequívocos de sinais Wi-Fi para uma dada área para definir a localização do dispositivo. Os autores também criticam a utilização de beacons BLE, pois são inviáveis em grandes espaços devido à grande quantidade de beacons necessários para cobrir todo o espaço.

Yedavalli et al. (2005) apresentam um algoritmo de localização chamado de *Eco-location* que examina uma sequência ordenada de recebimento de medições de intensidade de sinais de Radiofrequência (RF) para definir a localização atual. Nesse estudo, o autor

afirma, por meio de estudos comparativos com outras quatro técnicas de localização baseadas em RF e que utilizam simulações ou implementação de sistemas do mundo real, que o *Ecolocation* apresenta melhor desempenho nas diversas condições testadas.

Verma et al. (2016) apresenta um projeto de sistema de navegação interna para smartphones. Em seu trabalho, propõe um método que utiliza sensores do dispositivo, como o acelerômetro e magnetômetro, para realizar a técnica *Dead-reckoning* que consiste em estimar a posição atual de acordo com o posicionamento prévio. O método é suportado por um sistema baseado em arquitetura web que permite a criação de mapas internos e o fornecimento de informações de localização interna para navegação.

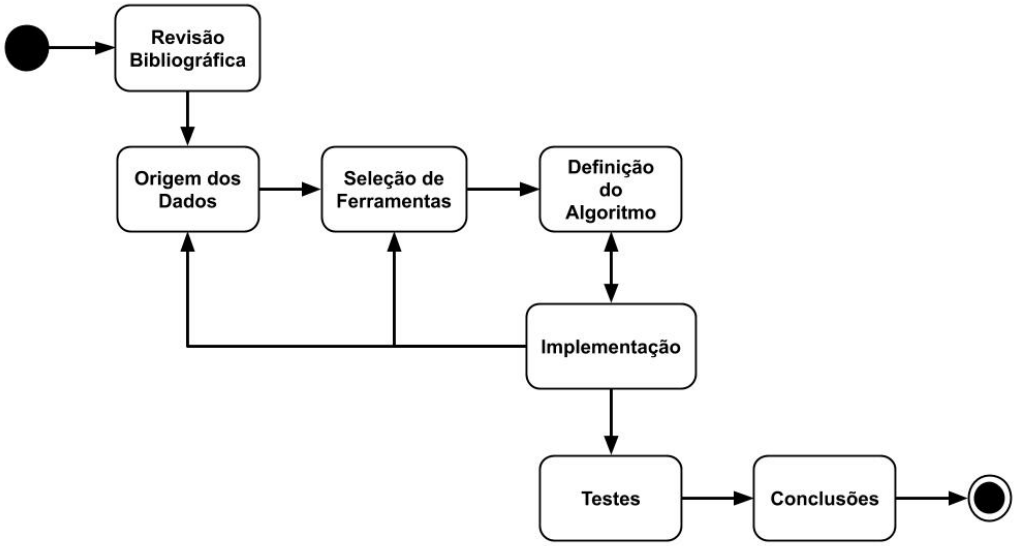
3 METODOLOGIA

Este capítulo destina-se a apresentar a metodologia que foi seguida para atingir os objetivos propostos por este trabalho. Cada etapa da metodologia será brevemente explicada. Também serão apresentados os recursos e ferramentas de software e hardware necessários para o desenvolvimento do protótipo.

3.1 Etapas da Metodologia

Nesta seção serão apresentadas e descritas as sete etapas da metodologia. A figura 3 apresenta a ordem em que as etapas foram executadas.

Figura 3 – Fluxograma da metodologia.



Fonte: Autoria própria

- Revisão Bibliográfica: Nesta etapa foi realizada uma pesquisa de publicações científicas, trabalhos correlatos, produtos comerciais e tecnologias existentes. Os trabalhos e tecnologias mais relevantes encontrados foram apresentados no Capítulo 2.
- Origem dos Dados: Nesta etapa foi definido qual seria a origem dos dados de mapeamento. Foi escolhido o projeto OpenStreetMap por sua licença permissiva e facilidade de acesso e edição dos dados.
- Seleção de Ferramentas: Para a realização do desenvolvimento do protótipo foi necessário definir quais ferramentas, bibliotecas e frameworks seriam usadas. As ferramentas escolhidas são apresentadas na subseção “Recursos de Hardware e Software” deste mesmo capítulo.

- Definição do Algoritmo: Nesta etapa foram realizados testes nos dados utilizados no protótipo com o algoritmo A* apresentado no Capítulo 2.
- Implementação: Nesta etapa foi feito o mapeamento da área utilizando o editor JOSM e o desenvolvimento do protótipo, além disso foi analisada a viabilidade de utilizar alguma das técnicas de navegação e localização em tempo real.
- Testes: Nesta etapa foram realizados testes no protótipo desenvolvido na etapa anterior.
- Conclusões: Nesta etapa foram analisados, apresentados e discutidos os resultados obtidos. Também foram discutidos os objetivos que não puderam ser atingidos e foram feitas recomendações para trabalhos futuros que envolvam o desenvolvimento de mapas interativos.

3.2 Recursos de Hardware e Software

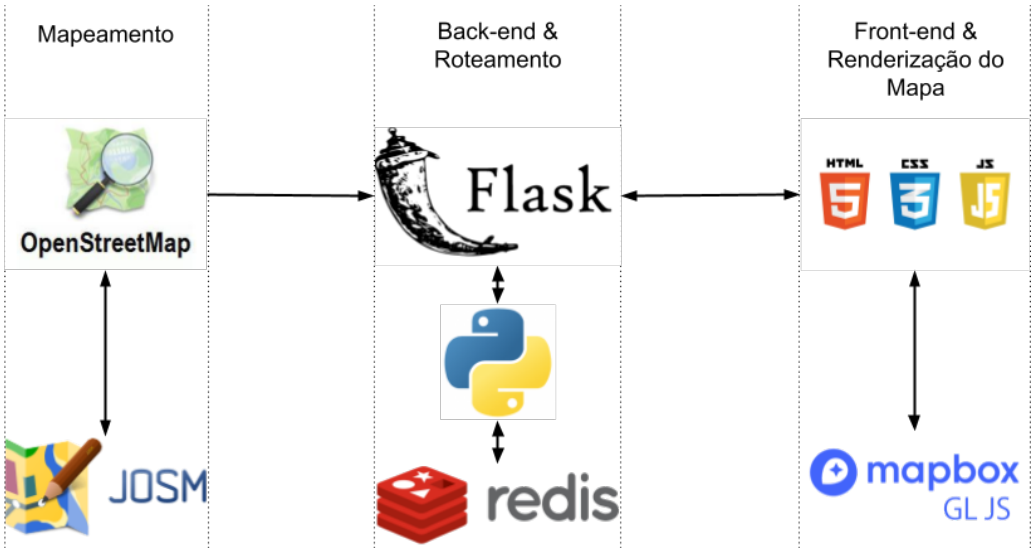
Para realizar os objetivos deste trabalho é necessário definir os recursos de hardware e software que serão utilizados em seu desenvolvimento, nesta seção estes recursos são apresentados. Primeiro são apresentados os recursos para obtenção e edição dos dados de mapeamento, em seguida os recursos de software para o desenvolvimento do protótipo e finalmente os recursos de hardware que foram utilizados para a implementação e testes do protótipo.

Os dados foram obtidos do projeto OpenStreetMap, como a área trabalhada é pequena foi possível fazer a extração diretamente pelo site do projeto. Para edição dos dados foi utilizado o editor JOSM que conta com diversos plugins para facilitar na edição dos mapas. Os dados extraídos são do formato OSM e foram convertidos para *vector tiles* utilizando a ferramenta Tippecanoe. Devido ao pequeno tamanho da área que foi trabalhada os *tiles* foram gerados em arquivos do formato PBF e armazenados em disco, isso possibilita a criação de cache dos *tiles* que aumentam o desempenho do servidor.

O protótipo do mapa interativo é uma aplicação web com o front-end desenvolvido utilizando a linguagem de programação JavaScript e a biblioteca Mapbox GL JS[1]. Existem outras bibliotecas para esta finalidade como LeafLet[2] e OpenLayers[3], a escolha pela Mapbox GL JS se deu pela qualidade da documentação e grande quantidade de exemplos de utilização da biblioteca, além de funcionar nativamente com *vector tiles* e existirem diversos projetos de diferentes escopos que utilizam a biblioteca. O back-end foi desenvolvido utilizando a linguagem de programação Python com a biblioteca Flask[4] e o banco de dados em memória Redis[5]. A imagem 4 apresenta as ferramentas utilizadas no projeto.

[1] <https://www.mapbox.com/mapbox-gljs>
[2] <https://leafletjs.com/>
[3] <https://openlayers.org/>
[4] <https://flask.palletsprojects.com/>
[5] <https://redis.io/>

Figura 4 – Recursos de Software Utilizados.



Fonte: Autoria própria

Os recursos de hardware utilizados foram computadores e smartphones de origem pessoal. Os computadores utilizados tem as seguintes configurações: processador Intel i5 3330 e 8GB de memória RAM; processador Intel i3 5005u e 12GB de memória RAM. As duas configurações foram testadas nos sistemas operacionais Windows e GNU/Linux. Os smartphones são do modelo Samsung Galaxy S20 e Samsung Galaxy A10, ambos do sistema operacional Android.

4 DESENVOLVIMENTO

Este capítulo apresenta os detalhes do processo de desenvolvimento realizado no projeto de implementação do protótipo do mapa interno interativo do campus UTFPR Curitiba Sede Centro.

4.1 Protótipo

Nesta sessão serão descritos os requisitos funcionais e não funcionais do protótipo e serão apresentados os diagramas elaborados, entre eles diagramas UML como o diagrama de classes, caso de uso e um diagrama da visão geral da arquitetura do protótipo.

Considerando o objetivo de desenvolver um mapa interno interativo do campus UTFPR-CT, em que os principais usuários seriam discentes, incluindo alunos de intercâmbio, e visitantes externos que, por exemplo, estariam participando de um evento realizado nas dependências da universidade, foram definidos os seguintes requisitos funcionais e não funcionais para o protótipo:

RF01 - O software deverá permitir movimentar, rotacionar e transladar o mapa.

RF02 - O software deverá permitir interações com o mapa por meio de cliques do usuário.

RF03 - O software deverá permitir a busca de áreas.

RF04 - O software deverá apresentar sugestões de busca de acordo com a entrada do usuário.

RF05 - O software deverá apresentar uma rota entre duas áreas definidas pela entrada do usuário.

RF06 - O software deverá permitir escolher a opção de exibir uma rota que evite escadas.

RF07 - O software deverá permitir que o usuário selecione o andar que está sendo exibido no mapa.

RF08 - O software deverá permitir que o usuário remova a rota que está sendo exibida.

RF09 - O software deverá permitir a troca do idioma do mapa.

RF10 - O software deverá permitir a troca do tema do mapa.

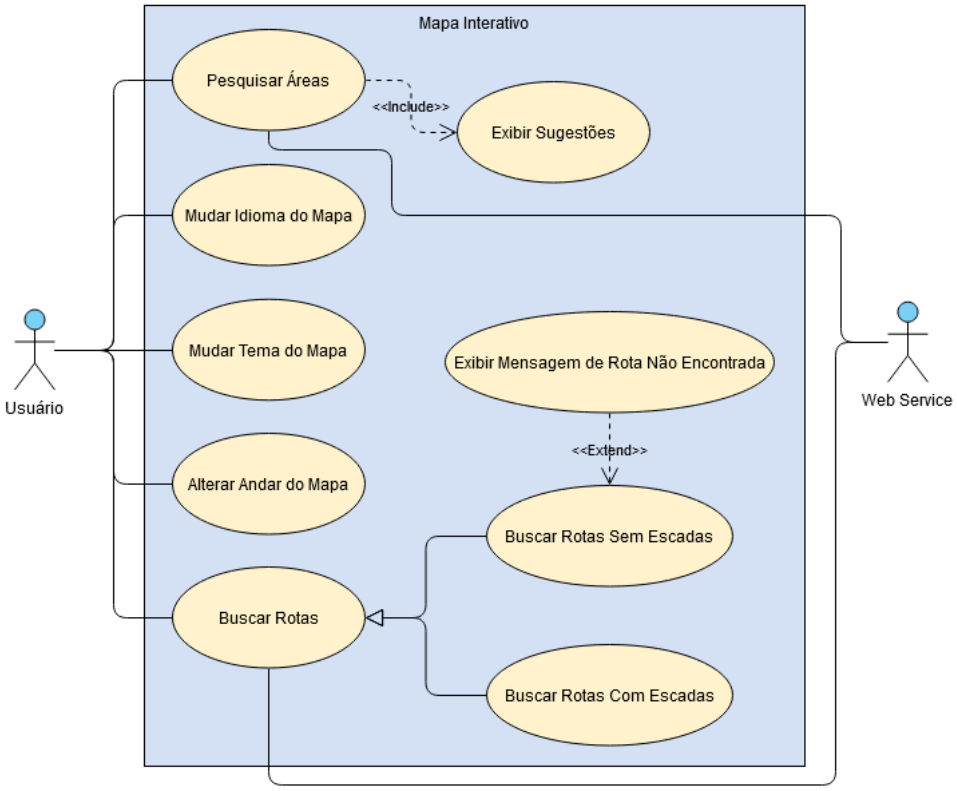
RNF01 - O software deverá ser desenvolvido em Python (back-end) e JavaScript (front-end).

RNF02 - O software deverá funcionar nas últimas versões dos navegadores Microsoft Edge, Mozilla Firefox e Google Chrome.

RNF03 - O software deverá funcionar em dispositivos móveis e computadores pessoais.

A figura 5 apresenta o diagrama de caso de uso do protótipo o qual apresenta as diversas ações que o usuário pode tomar, dentre elas temos: pesquisar áreas, mudar idioma do mapa, mudar o tema do mapa, alterar o andar do mapa e buscar rotas com ou sem escada.

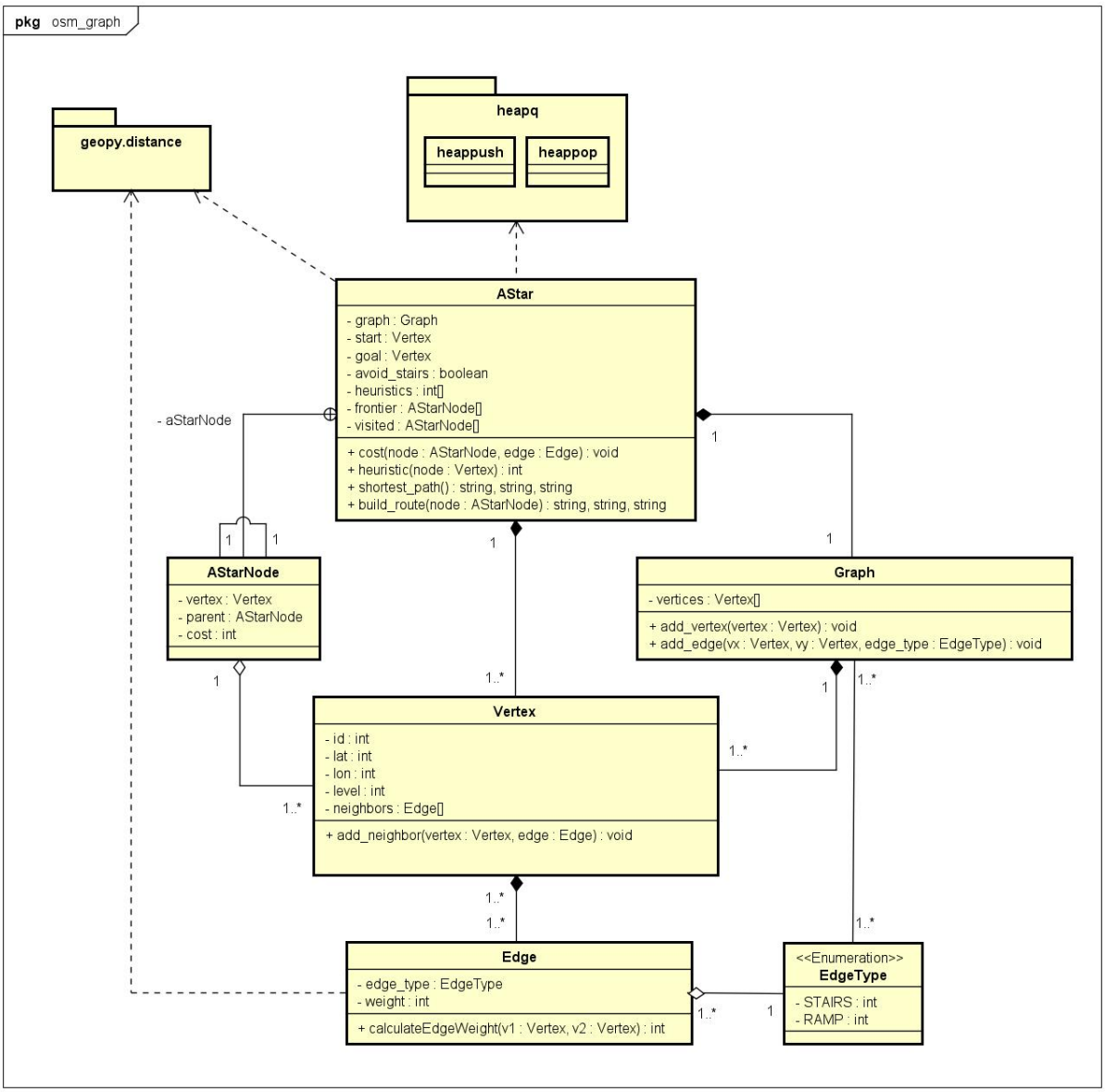
Figura 5 – Diagrama de Caso de Uso.



Fonte: Autoria própria

O diagrama de classes representado pela imagem 6 apresenta as classes reponsáveis pelo roteamento do protótipo. A classe *AStar* implementa o algoritmo A* para buscar a rota mais curta. *AStarNode* é uma classe aninhada do próprio *AStar* responsável por armazenar em memória os vértices da rota. Já a classe *Graph* implementa uma representação de grafo, *Vertex* corresponde aos vértices do grafo e *Edge* as arestas.

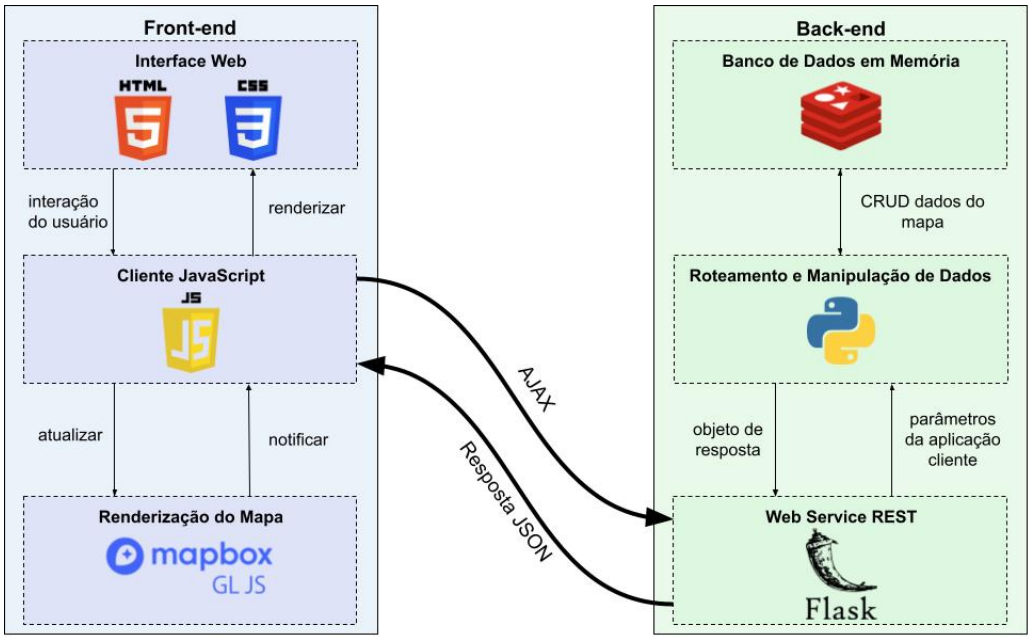
Figura 6 – Diagrama de Classes.



Fonte: Autoria própria

A figura 7 apresenta uma visão de alto nível da arquitetura do protótipo, com as aplicações front-end e back-end e seus respectivos componentes. Esses componentes são explicados nas sessões seguintes.

Figura 7 – Diagrama da Arquitetura do Protótipo.



Fonte: Autoria própria

4.2 Mapeamento

O mapeamento foi realizado a partir do formato de dados OpenStreetMap. As áreas internas foram mapeadas utilizando um esquema de marcação similar ao *Simple Indoor Tagging*[1] que é um esquema que define alguns rótulos para identificar espaços internos. Alguns exemplos dos rótulos utilizados são apresentados no quadro 1.

Quadro 1 – Rótulos utilizados no mapeamento.

Rótulo	**Descrição**
indoor	O valor *room* indica que a área é um ambiente interno.
level	Contém o valor numérico que indica o andar da área interna, caso a área esteja em mais de um andar os valores são separados por vírgula.
highway	define caminhos de diversos tipos, como calçadas com o valor *pedestrian*, escadas ou rampas com o valor *steps* e até ruas com os valores *primary, secondary*, entre outros.
ramp	Criado especificamente para o protótipo, utilizado para diferenciar rampas (valor *yes*) de escadas (sem um valor para este rótulo).

Fonte: Autoria própria

O editor escolhido para realizar o mapeamento foi o JOSM, que conta com um plugin para filtrar a exibição de áreas por andar, o que facilita o mapeamento interno

[1] <https://wiki.openstreetmap.org/wiki/Simple_Indoor_Tagging>

devido às salas que se sobrepõem nos andares, além de diversas outras ferramentas e plugins que ajudam no processo de mapeamento. Como o foco da implementação é somente o campus da universidade, todos os outros objetos do mapa, exceto as ruas ao redor, foram removidos.

O formato de dados OSM define retas e áreas a partir de um conjunto de pontos no mapa, por causa desta característica os caminhos do mapa seguem um formato de malha, com ângulos majoritariamente retos. Isso faz com que as rotas também sigam este formato e não pareçam tão naturais como seriam caso utilizassem toda a área disponível para deslocamento como a abordagem "porta-a-porta" apresentada anteriormente, seção 2.5.

4.3 Criação dos *Vector Tiles*

Com o mapeamento pronto, a próxima etapa consiste em gerar os *tiles* a partir dos dados do mapeamento. Para esta tarefa em um primeiro momento foi utilizado o *TileMaker*[2] que é uma ferramenta gratuita e open-source para a geração de *vector tiles* a partir de arquivos OSM, porém o software apresentou alguns problemas devido aos altos níveis de ampliação que precisaram ser utilizados para exibir em detalhe as áreas internas. Para um dos problemas foi feita uma sugestão de correção[3], porém outros problemas ainda persistiram e por isso o *TileMaker* foi substituído pelo *Tippecannoe*[4] que é uma ferramenta desenvolvida pela Mapbox, também gratuita e de código aberto que gera os tiles a partir de arquivos do formato GeoJSON, formato que é exportado nativamente pelo JOSM.

Uma das etapas do processo de criação dos *vector tiles* consiste em definir uma série de camadas que contenham objetos específicos, como ruas, construções, salas, áreas verdes, legendas, entre outros que posteriormente serão utilizados para exibir esses diferentes objetos de maneiras distintas no mapa. Para o desenvolvimento do protótipo foram criadas nove camadas, que são descritas no quadro 2.

Com as camadas definidas a próxima etapa é criar os *vector tiles*, para isso foi utilizado o *Tippecanoe*. A primeira etapa da criação dos tiles é definir os centros das áreas para o posterior posicionamento das legendas, isso foi feito utilizando um script Python que calcula uma aproximação de um ponto central de um polígono. A camada *road* não precisa deste tratamento especial, pois os nomes das ruas são posicionados diretamente sobre a linha da rua, não sendo necessário encontrar um ponto central.

A segunda etapa é separar do arquivo GeoJSON principal os objetos referentes a cada uma das camadas descritas anteriormente. Para isso são usados arquivos JSON que contém filtros para rótulos e valores de cada objeto OSM, assim como descrito no quadro 2, ao final desta etapa é gerado um novo arquivo para cada camada, incluindo as camadas

[2] <https://github.com/systemed/tilemaker>
[3] <https://github.com/systemed/tilemaker/issues/169>
[4] <https://github.com/mapbox/tippecanoe>

Quadro 2 – Camadas utilizadas no protótipo.

Nome	**Descrição**
road	Camada que contém ruas e legendas com os nomes das ruas. A filtragem foi feita por objetos que contém o rótulo “highway” com os valores "primary", "secondary", "tertiary", "primary_link", "service", "residential". Esses valores definem diferentes tipos de ruas no formato de dados do projeto OSM.
natural	Contém as áreas verdes do mapa (gramados, áreas com árvores, etc.). Filtragem realizada por objetos que contém o rótulo “natural”.
leisure	Contém as duas quadras externas do campus. Filtro feito pelo rótulo “leisure”.
barrier	Contém muros e cercas do campus. Filtro feito pelo rótulo “barrier”.
building	Contém todas as construções (blocos) do campus. Filtro feito pelo rótulo “building”.
building_label	Contém os nomes dos objetos da camada anterior e um ponto central deste objeto que posteriormente será utilizado para o posicionamento da legenda com o nome dos blocos.
indoor	Contém as áreas internas que representam as salas, departamentos, auditórios, etc.
indoor_label	Contém os nomes dos objetos da camada anterior e uma coordenada que representa um ponto centralizado dentro deste objeto que posteriormente será utilizado para o posicionamento da legenda com o nome das áreas internas.
entrance	Contém as coordenadas e nomes das entradas do campus. A filtragem foi realizada por objetos que contém o rótulo "entrance".

Fonte: Autoria própria

de legenda, com os dados filtrados.

A terceira etapa é a criação dos *tiles* a partir dos arquivos gerados na etapa anterior, nesta etapa o *Tippecanoe* junta todas as camadas em um único conjunto de *vector tiles* no formato PBF que são separados em uma hierarquia de diretórios organizados por nível de zoom e posicionamento no mapa. Para o protótipo foram criados tiles somente no zoom 17, níveis maiores de zoom serão utilizados durante a renderização do mapa, porém como estão sendo utilizados *vector tiles* é possível ampliar os objetos do nível 17 diretamente na aplicação cliente sem sofrer grandes alterações na qualidade do mapa.

As fontes e ícones utilizados no mapa também precisam ser gerados para serem renderizados corretamente. Isso se deve ao fato de que a grande maioria das bibliotecas usadas para renderizar mapas utilizam a tecnologia WebGL que possui limitações e pode

renderizar somente pontos, linhas e triângulos. Portanto, as fontes tradicionais e ícones não podem ser renderizadas no WebGL e precisam passar por uma etapa de processamento antes de serem usadas. Para gerar as fontes em um formato que o WebGL consiga renderizar foi utilizado o *fontnik*[5] e para os sprites foi utilizado o *spritezero*[6].

4.4 Front-end

O protótipo é uma aplicação web e portanto foram utilizados HTML, CSS e JavaScript para a criação da página que contém o mapa interativo. A comunicação entre front-end e back-end é feita por meio de requisições AJAX. Para a exibição dos *vector tiles* foi utilizada a biblioteca Javascript Mapbox GL JS que é uma biblioteca de código aberto e livre que utiliza WebGL para renderizar os tiles. Para exibir os tiles é necessário criar um arquivo de configuração JSON que define o estilo que será utilizado por cada uma das camadas que foram definidas na seção 4.3. Este arquivo se assemelha muito com um arquivo CSS e nele é possível definir o estilo de cada camada, por exemplo é possível escolher as cores do fundo e da borda dos objetos, a grossura da borda, o ícone que uma determinada camada deve usar, o tamanho, cor e tipo de fonte, também é possível definir filtros de visibilidade que exibem ou ocultam objetos de uma camada dependendo do valor de uma variável como o nível de ampliação que está sendo aplicado ou o andar que está selecionado, entre outras. A ordem de exibição das camadas também é definida neste arquivo, as camadas declaradas primeiro ficam no fundo e as declaradas por último ficam na frente do mapa.

O trecho de código a seguir apresenta a requisição AJAX feita para encontrar a rota entre dois pontos (*source* e *target*). Na linha 5 é criado o filtro *highlighted-area* para destacar as áreas no mapa, a filtragem é feita em qualquer objeto que contenha o atributo *id* com valor igual ao atributo *data-room-id* de um dos pontos. Caso uma rota já esteja sendo exibida no mapa, as camadas que contém as rotas e os ícones de escadas e rampas são removidas (linhas 7 até 11). Na sequência é realizada a requisição para o back-end passando como parâmetros os *ids* dos ponto de origem e destino e um booleano que identifica se a opção *Evitar escadas* está marcada. A resposta do back-end é um GeoJSON que é transformado em uma camada do mapa que renderiza a rota e os ícones de escadas e rampas.

```
1 function getRoute(source, target) {
2     const sourceId = source.getAttribute('data-room-id');
3     const targetId = target.getAttribute('data-room-id');
4
5     map.setFilter('highlighted-area', ['in', 'id', sourceId, targetId
          ]);
```

[5] https://github.com/mapbox/fontnik
[6] https://github.com/mapbox/spritezero

```
6
7       if (map.getLayer('route') !== undefined) {
8           map.removeLayer('route');
9           map.removeSource('route');
10          map.removeLayer('stairs');
11          map.removeSource('stairs');
12      }
13
14      const avoidStairs = document.getElementById('avoidStairs').
            checked;
15
16      const xhr = new XMLHttpRequest();
17      xhr.open('GET', `http://localhost:8080/route?source=${
            encodeURIComponent(sourceId)}&target=${encodeURIComponent(
            targetId)}&avoidStairs=${avoidStairs}`, true);
18      xhr.onload = () => {
19          if (xhr.status === 200) {
20              addRouteToMap(JSON.parse(xhr.response));
21          }
22      }
23      xhr.send();
24  }
```

A biblioteca Mapbox GL JS sofreu uma alteração de licença durante o desenvolvimento deste trabalho. A nova licença define que a partir da versão 2 a biblioteca passa a ser paga por utilização e novos recursos serão implementados somente nesta versão, a versão 1 receberá somente atualizações para problemas críticos, mas manterá a mesma licença. A comunidade que acompanhava o desenvolvimento da biblioteca protestou, porém a Mapbox não alterou sua posição e manteve a nova licença. Diante disso, a comunidade criou um *fork* do Mapbox GL JS versão 1 denominado de *MapLibre GL* que mantém a licença livre.

4.5 Back-end e roteamento

Para servir os *tiles*, fazer o cálculo de rota e a sugestão de busca de áreas foi desenvolvida uma API REST na linguagem de programação Python utilizando o framework Flask. Para o armazenamento de dados foi utilizado o Redis, que é um banco de dados em memória que armazena pares chave-valor. Para criar o grafo necessário para calcular as rotas foi utilizado o arquivo OSM exportado a partir do JOSM que é lido como um arquivo XML em um script Python. O script utiliza a sintaxe XPath para filtrar o arquivo

e encontrar os nós que fazem parte de corredores ou que são portas e com isso criar os vértices do grafo. Em uma segunda iteração pelo arquivo, são encontrados os vizinhos de cada vértice do grafo para criar as arestas com seus respectivos pesos.

Os pesos das arestas foram calculados usando a distância euclidiana entre cada nó, ignorando a curvatura da Terra como já apresentado anteriormente, dado que a escala cartográfica é muito grande e a curvatura da Terra não interfere de forma significativa nos cálculos de distância nessa escala. Caso a aresta seja uma rampa ou uma escada esse valor é acrescido de 5 ou 10 metros, respectivamente. Esses valores foram utilizados para refletir a distância vertical percorrida na mudança de andares e também o grau de esforço necessário para esse deslocamento, considerando que escadas exigem maior esforço para serem utilizadas do que rampas.

O algoritmo de menor caminho implementado foi o A*. Dado que não foi possível realizar o mapeamento completo do campus, o grafo final ficou consideravelmente menor do que o grafo previsto inicialmente, que seria uma representação de todo o campus. Com isso, os resultados obtidos utilizando o A* já foram altamente satisfatórios e qualquer diferença de performance que outros algoritmos ou estratégias poderiam apresentar em grafos maiores não seriam perceptíveis neste caso. O trecho de código abaixo coresponde a implementação em Python do algoritmo A*. Na implementação foi utilizada a estrutura de dados heap para armazenar os nós visitados. A classe *AStarNode* armazena os nós do grafo com os atributos de nó pai, custo e vértice. Ao final do algoritmo (linha 14) é chamado um método que constrói um objeto GeoJSON com a lista de coordenadas da rota separadas por andar que então serão enviadas ao front-end para renderização.

```
1  def shortest_path(self):
2      self.heuristics[self.start.id_] = self.heuristic(self.start)
3      start_node = self.AStarNode(self.start, None, 0)
4
5      heappush(self.frontier, (start_node.cost + self.heuristics[self.
           start.id_], start_node))
6
7      self.visited[self.start.id_] = start_node
8
9      while self.frontier:
10         node_tuple = heappop(self.frontier)
11         node = node_tuple[1]
12
13         if node.vertex == self.goal:
14             return self.build_route(node)
15
16         vertex = node.vertex
17         for neighbor_id, edge in vertex.neighbors.items():
```

```
18              neighbor_vertex = self.graph.vertices[neighbor_id]
19
20              if neighbor_id not in self.heuristics:
21                  self.heuristics[neighbor_id] = self.heuristic(
                        neighbor_vertex)
22
23              new_cost = self.cost(node, edge)
24
25              if neighbor_vertex.id_ not in self.visited or new_cost <
                    self.visited[neighbor_vertex.id_].cost:
26                  neighbor_node = self.AStarNode(neighbor_vertex, node,
                        new_cost)
27                  self.visited[neighbor_vertex.id_] = neighbor_node
28
29                  heappush(self.frontier, (new_cost + self.heuristics[
                        neighbor_id], neighbor_node))
30      return None
```

A heurística utilizada é a distância euclidiana que é uma heurística admissível e com isso garante a otimalidade do algoritmo. Para atender a opção de rota que evita escadas foi feita uma modificação na função de custo do algoritmo para adicionar uma constante de valor 100 ao custo caso a aresta seja uma escada e a opção esteja selecionada. Com essa modificação, caso uma rota sem escadas não seja encontrada será exibida a rota com o menor número de escadas possível, também é exibido um aviso na interface de que não foi encontrada uma rota sem escadas.

4.6 Localização e Navegação

As estratégias de localização e navegação interna não puderam ser testadas devido às restrições de acesso ao campus por causa da pandemia de COVID-19. Considerando os artigos que foram analisados, a implementação mais provável seria a do GPS dado que os navegadores já possuem APIs desenvolvidas para utilizar a geolocalização por meio do sinal de GPS do dispositivo, porém existiriam problemas devido a degradação do sinal em ambientes internos e o mapeamento precisaria ser muito preciso para os resultados serem satisfatórios, o que dificultaria ainda mais o processo. Existem APIs JavaScript para sinal Bluetooth, porém ainda estão em fase experimental e podem não funcionar em todos os navegadores, portanto triangulação de sinais Bluetooth ou beacons BLE também não seriam uma alternativa viável.

4.7 Considerações sobre o desenvolvimento

O desenvolvimento do protótipo demorou mais que o planejado, especialmente por causa da mudança de ferramentas por problemas com sua utilização, a falta de documentação de algumas ferramentas também dificultou o processo, porém foi possível desenvolver um protótipo totalmente funcional utilizando somente ferramentas de código aberto e livre e dados igualmente livres por meio do projeto OSM. Mudanças de licenças como a que aconteceu com a biblioteca Mapbox GL JS devem ser consideradas durante o planejamento de um projeto, especialmente os de longo prazo, para definir o impacto dessas mudanças e possíveis alternativas caso a ferramenta precise ser substituída durante o desenvolvimento. Software livre e de código aberto ainda é extremamente importante para o desenvolvimento de software e no caso de mapas interativos isso não é diferente, existe um grande número de ferramentas livres que possibilitaram o desenvolvimento deste trabalho.

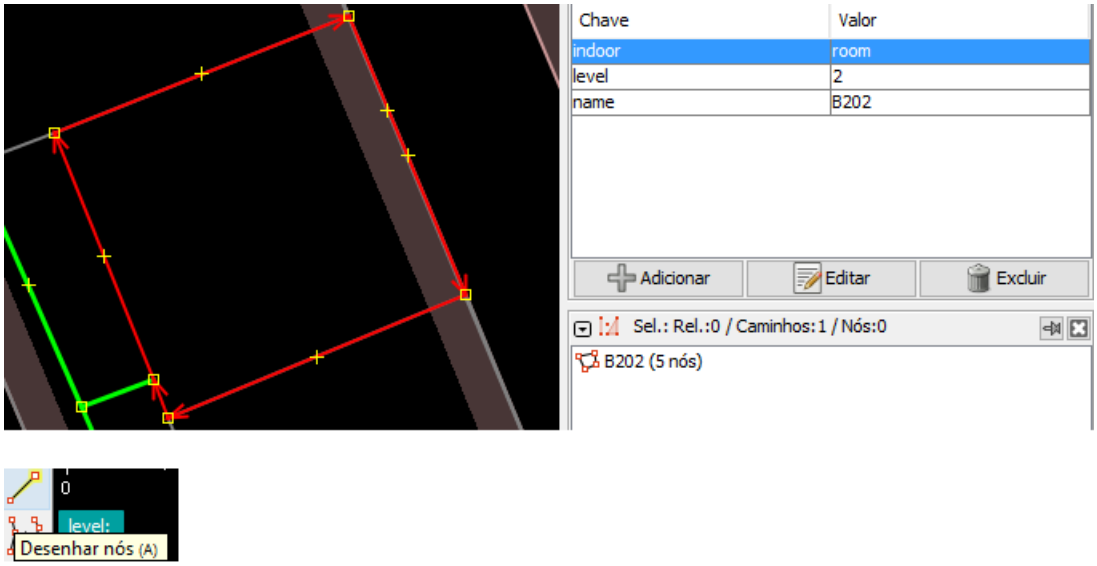
5 RESULTADOS

Este capítulo apresenta os resultados obtidos a partir do projeto de implementação do protótipo do mapa interno interativo do campus UTFPR CT.

5.1 Protótipo

A ferramenta utilizada para o mapeamento deste projeto foi o editor JOSM que gera arquivos OSM (com conteúdo xml e de extensão .osm) que fornecem os dados de cada elemento (rotas, salas, etc) mapeado. A partir dessa ferramenta é possível traçar rotas, representar salas, paredes, ruas, entre outros e rotular cada um desses elementos presentes no mapa, um exemplo disso é apresentado na figura 8.

Figura 8 – Editor JOSM.

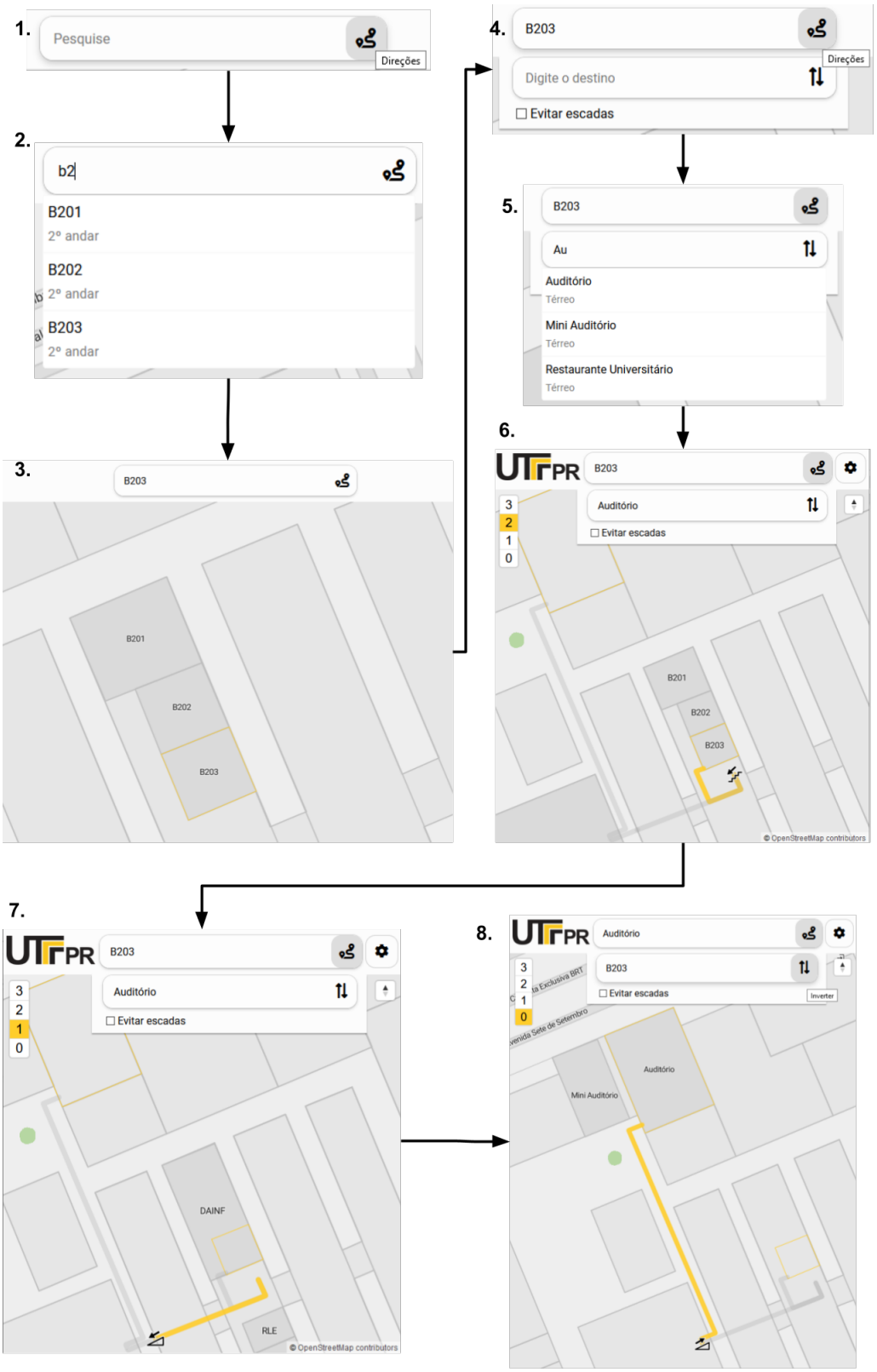


Fonte: Autoria própria

A sala B202, representada na figura 8, foi traçada por meio de 5 nós, para inserir os nós basta selecionar a opção “Desenhar nós (A)”, e possui rótulos(tags) chave-valor indicando que é uma sala de ambiente interno (indoor: room), está localizada no segundo andar (level: 2) e possui o nome B202 (name: B202). Após realizado o mapeamento, é gerado o arquivo xml, através do próprio editor, com todas as informações do mapa a serem consumidas pela aplicação.

A figura 9 apresenta o funcionamento padrão da aplicação, nesse caso, considerando o cenário de uma pessoa que está localizada na sala B203 e deseja ir para o auditório. Para que a pessoa tenha conhecimento sobre qual caminho deve seguir para chegar ao seu destino é necessário realizar as etapas, de um a sete, que estão ilustradas pela imagem 9.

Figura 9 – Fluxograma de usabilidade da aplicação.



Fonte: Autoria própria

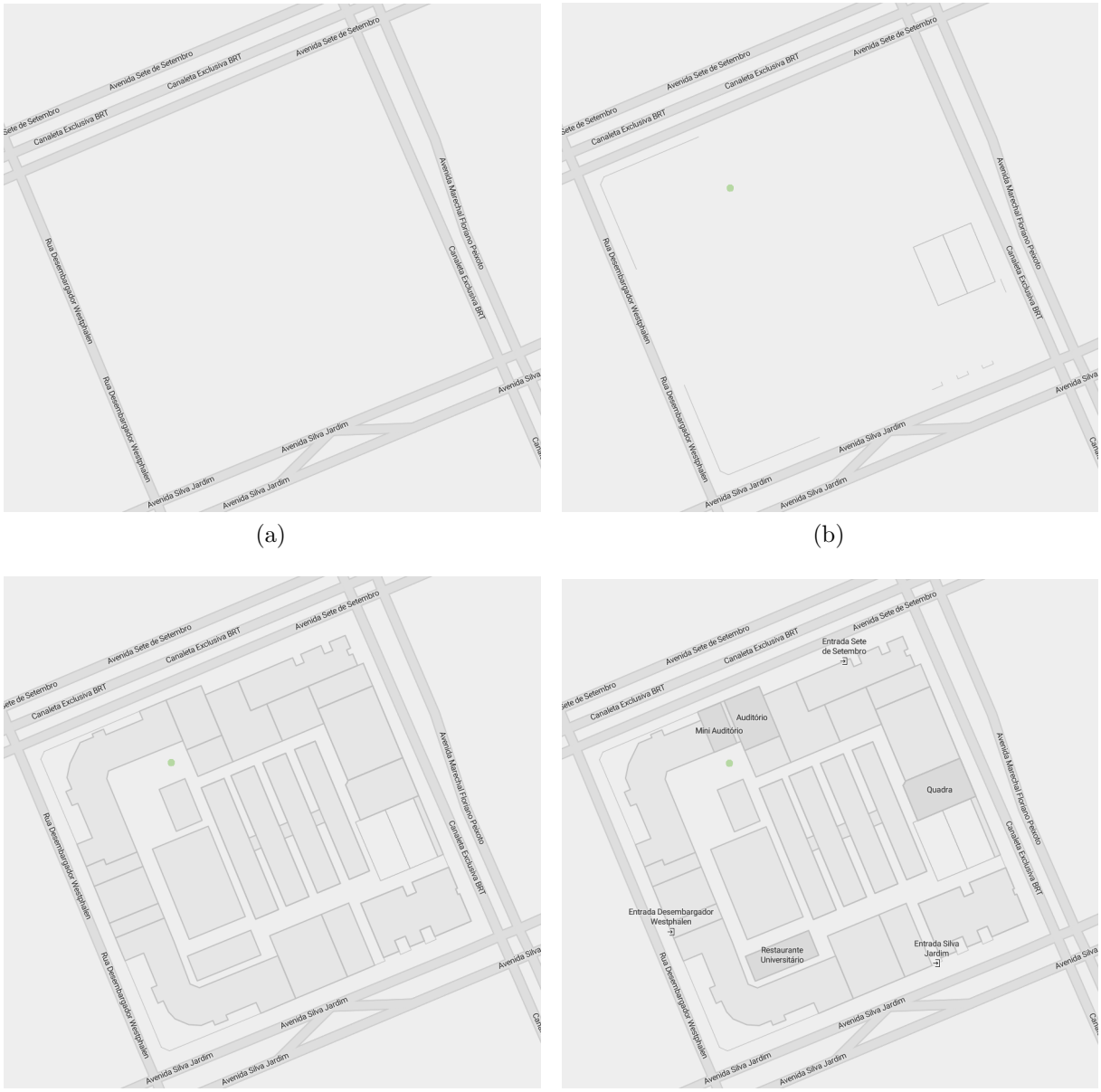
O item 1 se refere a barra de pesquisa, na qual é possível inserir o ponto de partida

para que, a partir dele, a aplicação trace toda a rota. As sugestões de busca são exibidas em um menu dropdown, também conhecido como autocompletar (autocomplete), que completa o termo inserido com a finalidade de facilitar a pesquisa pelo local de origem do usuário, representado pelo item 2. Após selecionar um dos itens dos resultados da busca, a aplicação amplia e destaca o local selecionado (item 3). Também é possível selecionar um local clicando diretamente sobre ele no mapa. Ao pressionar o botão direções, é expandida a barra de pesquisa do local de destino, também é disponibilizada uma opção (checkbox) para que seja traçada uma rota alternativa sem escadas, representados pelo item 4.

Assim como a barra de pesquisa do ponto de partida, a do local de destino também possui o autocompletar, com os itens a serem selecionados de acordo com o termo inserido (item 5). Ao selecionar um desses itens, a aplicação traça a rota mais curta entre o local de partida do usuário e a região onde ele deseja estar (item 6). É possível expandir a visualização da rota até a próxima escada ou rampa ao clicar no ícone, representado pelo item 7. O ícone exibido é diferente para escadas e rampas e apresenta uma seta que indica se o usuário deve descer ou subir. A aplicação também possui a opção de traçar a rota inversa da que foi anteriormente apresentada, para isso basta apertar o botão inverter bem como ilustrado pelo item 8 da figura 9.

Caso a opção de rota sem escadas seja selecionada e um caminho sem escadas não seja encontrado, será exibido um aviso em tela para o usuário com a mensagem “Nenhum caminho sem escadas foi encontrado, exibindo a rota com a menor número de escadas possível.” e, onde for possível, rotas que utilizam rampas terão preferência sobre rotas com escadas mesmo que essa rota seja mais longa.

Figura 10 – Camadas de representação do mapa.



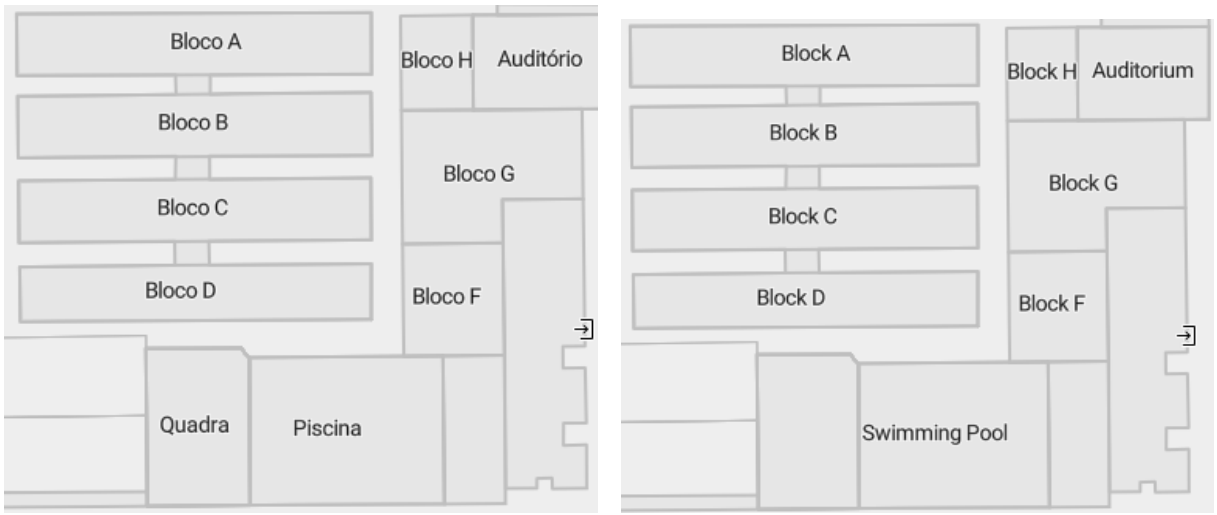
(a) (b)

(c) (d)

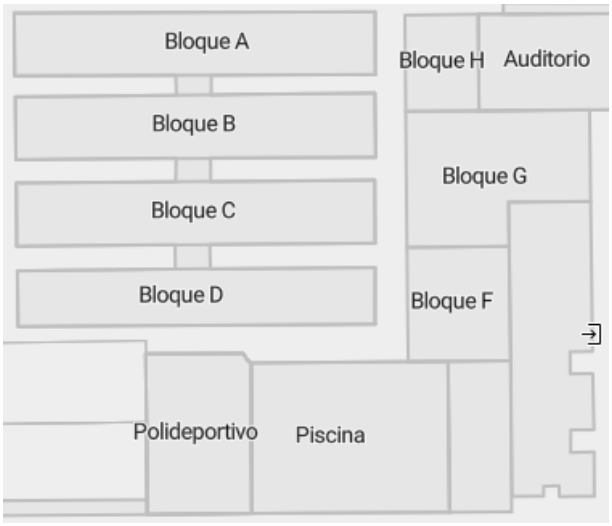
Fonte: Autoria própria

A figura 10 ilustra as diversas camadas (*layers*) que são renderizadas para compor o mapa, explanado na seção 4.3. A camada de fundo (*background*) que é uma camada padrão e camada que contém as ruas (*roads*) são demonstradas em (a). Já a imagem (b) apresenta as camadas anteriormente mencionadas e, também, as camadas *natural*, *barrier* e *leisure*. Em (c) foi adicionada a *building*. Por fim, em (d) foram adicionadas as camadas *indoor* e *entrance*. A especificação de cada camada pode ser encontrada no quadro 2.

Figura 11 – Mapa com legendas em português (a), inglês (b) e espanhol (c).



(a) (b)

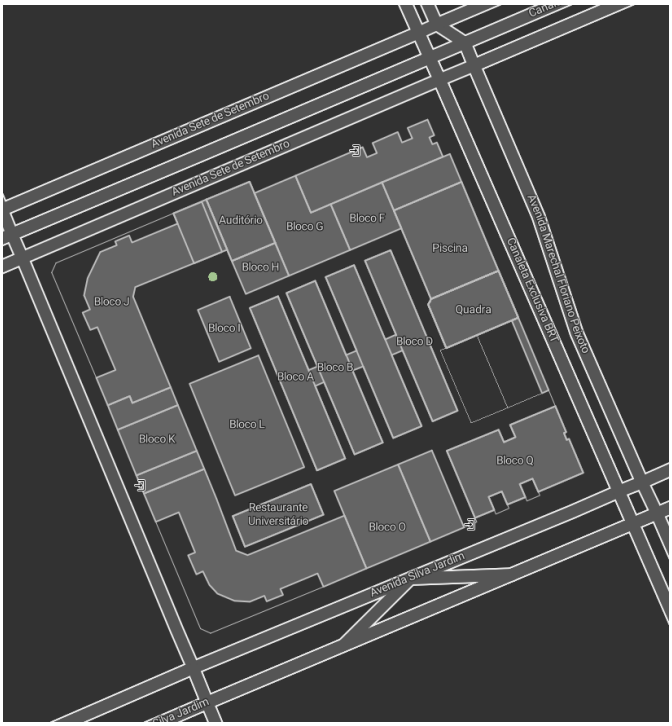


(c)

Fonte: Autoria própria

Um dos benefícios dos *vector tiles* é a possibilidade de interagir e mudar os atributos do mapa de forma dinâmica e sem precisar fazer a requisição de um novo *tile* ao servidor, diferentemente dos *raster tiles*. Um exemplo disto é a mudança do idioma das legendas do mapa que pode ser feita selecionando um idioma no menu de configurações. Estão disponíveis os idiomas português, inglês e espanhol como a figura 11 mostra.

Figura 12 – Mapa com tema escuro.



Fonte: Autoria própria

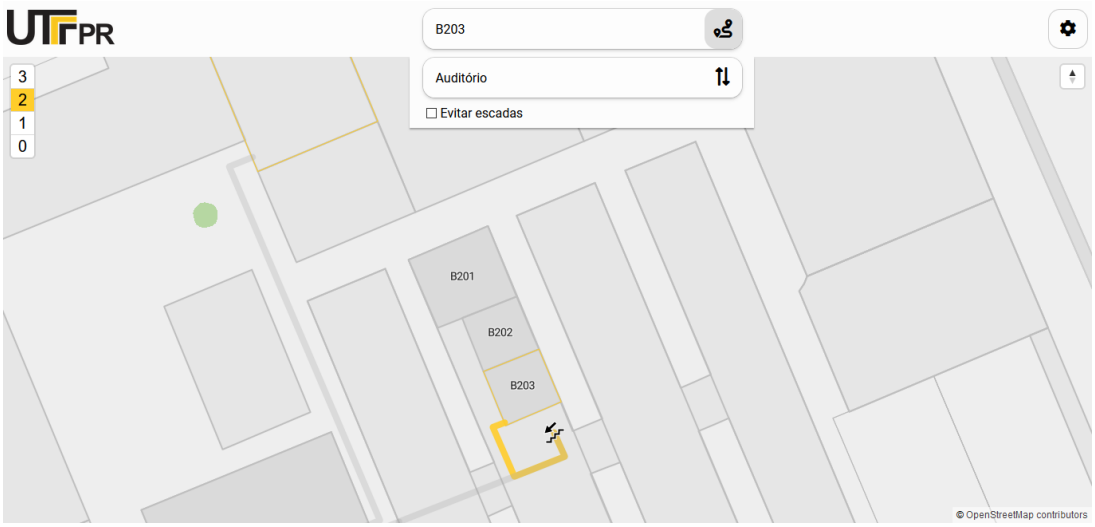
Também está disponível a opção de escolher entre os temas claro e escuro, como ilustrado na figura 12, que muda todo o estilo do mapa sem precisar fazer uma nova requisição para o servidor, todo o processamento da renderização é feito diretamente na aplicação cliente utilizando os dados que já tinham sido requisitados no carregamento inicial do mapa. As opções para trocar o tema e o idioma podem ser alteradas no menu de configurações como exibido na figura 13.

Figura 13 – Menu de configurações do mapa.

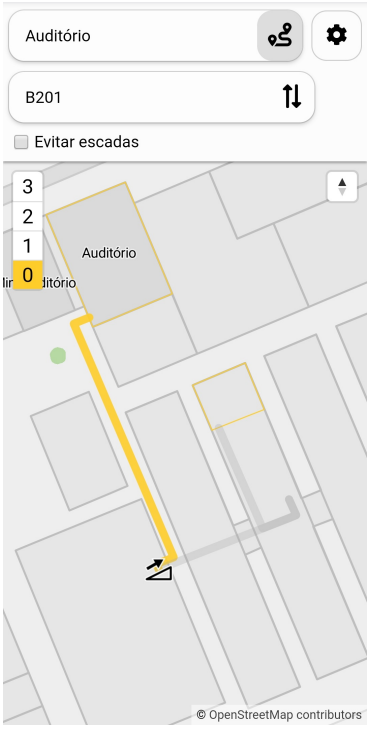
Fonte: Autoria própria

A aplicação pode ser utilizada tanto por computadores quanto por dispositivos móveis. Como apresentado na imagem 14, existem apenas algumas diferenças de layout entre uma plataforma e outra.

Figura 14 – Layouts desktop (a) e mobile (b).



(a)



(b)

Fonte: Autoria própria

Os resultados obtidos por este projeto se relacionam com o objetivo geral uma vez que a aplicação não somente proporciona a rota mais curta para o usuário, facilitando sua circulação pelo campus, como também informa o usuário sobre áreas, como salas e departamentos, quais o mesmo pode não ter conhecimento sobre e, além disso, a barra de pesquisa com o autocompletar acaba se tornando um facilitador de todo esse processo de informar o usuário. Também pode evitar problemas cotidianos como atrasos e imprevistos pelo simples fato de não conseguir se localizar no campus.

5.2 Testes

Para obter resultados qualitativos, seria necessário realizar uma pesquisa de campo a qual exigiria testes executados por um grupo de pessoas que, por sua vez, iriam avaliar o protótipo e levantar os aspectos positivos e negativos da aplicação. O principal fator que impediu esses testes de serem realizados foi a pandemia de COVID-19 a qual resultou na contramedida de quarentena que iniciou por volta do mês de março de 2020 e se estendeu além da data de finalização deste trabalho. Isso, além de dificultar a coleta dos resultados via pesquisa de usabilidade, também afetou o desenvolvimento do protótipo na etapa de mapeamento, também citado no terceiro parágrafo da seção 4.5. Realizar o mapeamento completo da universidade com todas as rotas e salas não foi possível devido à restrição de acesso ao campus e à falta de acesso a materiais com informações sobre o mapeamento da universidade.

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado todo o processo necessário para o desenvolvimento de um protótipo de um mapa interno interativo, desde o mapeamento até o seu funcionamento. O protótipo desenvolvido pode ser utilizado a partir de computadores ou dispositivos móveis e funciona em qualquer navegador moderno, podendo facilitar o deslocamento de visitantes, calouros e até mesmo veteranos e servidores pelo campus. O protótipo também expôs a falta de acessibilidade do campus, com diversas áreas sem acesso para pessoas que não podem subir ou descer escadas, o mapa pode encorajar eventuais melhorias na estrutura física do campus para tornar alguns locais mais acessíveis.

O desenvolvimento foi feito sem conhecimento prévio e utilizando ferramentas de código aberto e, preferencialmente, livres, porém as diferentes ferramentas necessárias para construir o mapa e o fato de que muitas dessas ferramentas não estão preparadas para funcionar com altos níveis de ampliação, como necessário em mapas internos, mostrou-se um desafio. As restrições devido a pandemia também dificultaram o desenvolvimento do projeto, principalmente a parte de mapeamento, dado que não foi possível visitar o campus para validar o mapeamento realizado.

Apesar das dificuldades, essas ferramentas e a comunidade que as desenvolve foram essenciais para o desenvolvimento bem sucedido do protótipo. Outro importante fator foi o mapeamento realizado pelos colaboradores do projeto OpenStreetMap que serviu como base para o mapeamento utilizado no protótipo.

Mapas internos ainda são pouco explorados e tem um processo de desenvolvimento que ainda precisa ser aprimorado para facilitar este processo. A etapa de mapeamento é especialmente difícil por causa da alta precisão necessária e os vários detalhes que são difíceis de serem mapeados pelos mesmos métodos utilizados para áreas externas, porém com a popularização e avanço de técnicas existentes e o desenvolvimento de novas técnicas o desenvolvimento de mapas internos deve evoluir e ficar mais fácil de ser realizado.

6.1 Trabalhos Futuros

Trabalhos futuros podem aprimorar e ampliar os recursos e funcionalidades desenvolvidos neste trabalho, algumas sugestões são:

- O processo de mapeamento pode ser aprimorado para ser feito de maneira mais rápida, por exemplo desenvolvendo um plugin com ferramentas específicas para o mapeamento de interiores para o JOSM ou outros editores OSM.
- Realizar um mapeamento completo do campus, incluindo casos como o do Bloco V que não é uma área contígua e fica em outra quadra, rotas com elevadores, andares intermediários que ficam entre os andares comuns como algumas áreas do Bloco E.

- Disponibilizar a tradução de toda a interface e não somente do mapa e também aumentar o número de idiomas disponíveis.
- Melhorar a acessibilidade da aplicação tornando-a compatível com leitores de tela e fornecendo instruções passo a passo da rota para que possam ser lidas pelo leitor.
- Utilizar uma solução mais robusta para a sugestão de busca como por exemplo o Apache Solr, que pode encontrar resultados mesmo com erros de digitação e permite a utilização de sinônimos entre outras funcionalidades que podem melhorar as sugestões de busca.
- Implementar uma técnica de navegação e localização em tempo real, utilizando GPS, BLE ou outra técnica apresentada anteriormente que possua uma alta precisão mesmo em ambientes internos.
- Implementar um painel administrativo para servidores da UTFPR adicionarem informações pertinentes tais como bloqueios, obras, eventos, entre outros com a finalidade da aplicação traçar rotas alternativas no caso de construções ou manutenções ou de indicar eventos.
- Realizar testes de usabilidade do protótipo com o público-alvo (docentes, discentes e servidores) para encontrar possíveis problemas e pontos de melhoria.
- Realizar um teste de campo do mapeamento do protótipo, visitando o câmpus presencialmente para validar o mapeamento realizado.

Referências

AGRAWAL, S.; GUPTA, R. Web gis and its architecture: a review. **Arabian Journal of Geosciences**, Springer, v. 10, n. 23, p. 518, 2017. Citado 2 vezes nas páginas 12 e 13.

ANTONIOU, V.; MORLEY, J.; HAKLAY, M. M. Tiled vectors: A method for vector transmission over the web. In: CARSWELL, J. D.; FOTHERINGHAM, A. S.; MCARDLE, G. (Ed.). **Web and Wireless Geographical Information Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 56–71. ISBN 978-3-642-10601-9. Citado na página 14.

ARCGIS. **ArcGIS**. 2021. Acessado em: 24/05/2021. Disponível em: <https://www.arcgis.com/index.html>. Citado na página 13.

AUTODESK. **AutoCAD Map 3D**. 2021. Acessado em: 24/05/2021. Disponível em: <https://www.autodesk.com.br/products/autocad/included-toolsets/autocad-map-3d>. Citado na página 13.

BATTERSBY, S. E. et al. Implications of web mercator and its use in online mapping. **Cartographica: The International Journal for Geographic Information and Geovisualization**, USGS Publications Warehouse, v. 49, n. 2, p. 85–101, 2014. Citado na página 12.

BOULOS, M. N. K.; GERAGHTY, E. M. Geographical tracking and mapping of coronavirus disease covid-19/severe acute respiratory syndrome coronavirus 2 (sars-cov-2) epidemic and associated events around the world: how 21st century gis technologies are supporting the global fight against outbreaks and epidemics. **International Journal of Health Geographics**, BioMed Central, 2020. Citado na página 9.

BURROUGH, P. Principles of geographical information systems for land resources assessment. **Geocarto International**, Taylor Francis, v. 1, n. 3, p. 54–54, 1986. Disponível em: <https://doi.org/10.1080/10106048609354060>. Citado na página 13.

COOPERATIVE, G.; COLLINS, F. The unique qualities of a geographic information system: a commentary. **Photogrammetric Engineering and Remote Sensing**, v. 54, n. 11, p. 1547–9, 1988. Citado na página 12.

COWEN, D. J. Gis versus cad versus dbms: What area the differences. **Photogrammetric Engineering and remote sensing**, v. 4, p. 1551–1555, 1988. Citado na página 13.

CRUZ, S. A. B. d.; SILVA, J.; MACÁRIO, C. d. N. Uma arquitetura de webgis para visualização de dados geoespaciais do pantanal. In: IN: SIMPÓSIO DE GEOTECNOLOGIAS NO PANTANAL, 5., 2014, CAMPO GRANDE, MS .... **Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)**. [S.l.], 2014. Citado na página 13.

DEVINE, H. A.; FIELD, R. C. The Gist of GIS. **Journal of Forestry**, v. 84, n. 8, p. 17–22, 08 1986. ISSN 0022-1201. Disponível em: <https://doi.org/10.1093/jof/84.8.17>. Citado na página 12.

DIJKSTRA, E. W. et al. A note on two problems in connexion with graphs. **Numerische mathematik**, v. 1, n. 1, p. 269–271, 1959. Citado na página 16.

DUDAS, P. M.; GHAFOURIAN, M.; KARIMI, H. A. Onalin: Ontology and algorithm for indoor routing. In: IEEE. **2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware**. [S.l.], 2009. p. 720–725. Citado na página 17.

GAFFURI, J. Toward web mapping with vector data. In: XIAO, N. et al. (Ed.). **Geographic Information Science**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 87–101. ISBN 978-3-642-33024-7. Citado na página 14.

GAO, S. et al. Online gis services for mapping and sharing disease information. **International Journal of Health Geographics**, Springer, v. 7, n. 1, p. 8, 2008. Citado na página 9.

GEISBERGER, R. et al. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In: SPRINGER. **International Workshop on Experimental and Efficient Algorithms**. [S.l.], 2008. p. 319–333. Citado na página 17.

GEOFABRIK. **OpenStreetMap Data Extracts**. 2020. Acessado em: 29/09/2020. Disponível em: <https://download.geofabrik.de/>. Citado na página 16.

GEOSERVER. **GeoServer**. 2021. Acessado em: 24/05/2021. Disponível em: <http://geoserver.org/>. Citado na página 13.

GOLDBERG, A.; HARRELSON, C. **Computing the Shortest Path: A* Search Meets Graph Theory**. [S.l.], 2004. 25 p. Disponível em: <https://www.microsoft.com/en-us/research/publication/computing-the-shortest-path-a-search-meets-graph-theory/>. Citado na página 17.

HAKLAY, M.; WEBER, P. Openstreetmap: User-generated street maps. **IEEE Pervasive Computing**, Ieee, v. 7, n. 4, p. 12–18, 2008. Citado na página 15.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE transactions on Systems Science and Cybernetics**, IEEE, v. 4, n. 2, p. 100–107, 1968. Citado na página 16.

HODGE, L. **Universities Navigating a Pandemic: How MazeMap's Indoor Wayfinding Can Help in the Midst of the COVID-19 Situation**. 2020. Disponível em: <https://blog.mazemap.com/how-mazemap-can-help-universities-covid-19-pandemic/>. Citado na página 9.

HOFFMANN, E. J.; WERNER, M.; SCHAUER, L. Indoor navigation using virtual anchor points. In: **2016 European Navigation Conference (ENC)**. [S.l.: s.n.], 2016. p. 1–8. Citado na página 18.

JENNY, B. Adaptive composite map projections. **IEEE Transactions on Visualization and Computer Graphics**, v. 18, n. 12, p. 2575–2582, 2012. Citado na página 12.

KARNEY, C. F. F. Algorithms for geodesics. **Journal of Geodesy**, Springer, v. 87, n. 1, p. 43–55, 2013. Citado na página 12.

KRAAK, M.-J. The role of the map in a web-gis environment. **Journal of Geographical Systems**, Springer, v. 6, n. 2, p. 83–93, 2004. Citado na página 12.

LIU, L.; ZLATANOVA, S. A"door-to-door"path-finding approach for indoor navigation. **Proceedings Gi4DM 2011: GeoInformation for Disaster Management, Antalya, Turkey, 3-8 May 2011**, International Society for Photogrammetry and Remote Sensing (ISPRS), 2011. Citado 2 vezes nas páginas 17 e 18.

LUXEN, D.; VETTER, C. Real-time routing with openstreetmap data. In: **Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems**. [S.l.: s.n.], 2011. p. 513–516. Citado na página 17.

MAPBOX. **Style Specification**. [S.l.], 2016. Disponível em: <https://docs.mapbox.com/mapbox-gl-js/style-spec/>. Citado na página 15.

MAPBOX. **Vector tile specification**. [S.l.], 2016. Disponível em: <https://docs.mapbox.com/vector-tiles/specification/>. Citado na página 15.

MAPILLARY. **Mapillary**. 2021. Acessado em: 24/05/2021. Disponível em: <https://www.mapillary.com/>. Citado na página 10.

NEIS, P.; ZIPF, A. Analyzing the contributor activity of a volunteered geographic information project — the case of openstreetmap. **ISPRS International Journal of Geo-Information**, MDPI AG, v. 1, n. 2, p. 146–165, Jul 2012. ISSN 2220-9964. Disponível em: <http://dx.doi.org/10.3390/ijgi1020146>. Citado na página 10.

OHRT, J.; TURAU, V. Simple indoor routing on svg maps. In: **International Conference on Indoor Positioning and Indoor Navigation**. [S.l.: s.n.], 2013. p. 1–6. Citado na página 10.

OPENAERIALMAP. **OpenAerialMap**. 2021. Acessado em: 24/05/2021. Disponível em: <https://openaerialmap.org/>. Citado na página 10.

OPENSTREETMAP. **OpenStreetMap**. 2020. Acessado em: 29/09/2020. Disponível em: <https://www.openstreetmap.org/>. Citado na página 10.

OPENSTREETMAP. **OpenStreetMap Planet**. 2020. Acessado em: 29/09/2020. Disponível em: <https://planet.openstreetmap.org/>. Citado na página 16.

OPENSTREETMAP. **OpenStreetMap Statistics**. 2020. Acessado em: 29/09/2020. Disponível em: <https://www.openstreetmap.org/stats/data_stats.html>. Citado na página 15.

POSTGIS. **PostGIS**. 2021. Acessado em: 24/05/2021. Disponível em: <https://postgis.net/>. Citado na página 13.

QGIS. **QGIS**. 2021. Acessado em: 24/05/2021. Disponível em: <https://qgis.org/>. Citado na página 13.

RIGAUX, P.; SCHOLL, M.; VOISARD, A. **Spatial databases with application to GIS**. [S.l.]: Morgan Kaufmann, 2002. ISBN 1-55860-558-6. Citado na página 12.

RUSTAGI, T.; YOO, K. Ar navigation solution using vector tiles. In: **Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology**. New York, NY, USA: Association for Computing Machinery, 2018. (VRST '18). ISBN 9781450360869. Disponível em: <https://doi.org/10.1145/3281505.3281616>. Citado na página 10.

RUSTAGI, T.; YOO, K. Indoor ar navigation using tilesets. In: **Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology**. New York, NY, USA: Association for Computing Machinery, 2018. (VRST '18). ISBN 9781450360869. Disponível em: <https://doi.org/10.1145/3281505.3281575>. Citado na página 10.

SHAO, Z. et al. Vip-tree: an effective index for indoor spatial queries. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 10, n. 4, p. 325–336, 2016. Citado na página 18.

VERMA, S. et al. A smartphone based indoor navigation system. In: **2016 28th International Conference on Microelectronics (ICM)**. [S.l.: s.n.], 2016. p. 345–348. Citado 3 vezes nas páginas 10, 18 e 19.

YEDAVALLI, K. et al. Ecolocation: a sequence based technique for rf localization in wireless sensor networks. In: **IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.** [S.l.: s.n.], 2005. p. 285–292. Citado na página 18.