UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

SUL

Sistema Útil de Localização

Matheus Miranda Ferreira da Costa

Orientador

Geiza Maria Hamazaki da Silva

RIO DE JANEIRO, RJ – BRASIL

JULHO DE 2017

SUL

Matheus Miranda Ferreira da Costa

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Geiza Maria Hamazaki da Silva

_____

Adriana Cesário de Faria Alvim

_____

Morganna Carmem Diniz

Catalogação informatizada pelo(a) autor(a)

MM672s
Miranda Ferreira da Costa, Matheus
Sistema Útil de Localização / Matheus Miranda Ferreira da Costa. -- Rio de Janeiro, 2017.
66

Orientadora: Geiza Maria Hamazaki da Silva.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2017.

1. Geolocalização. I. Maria Hamazaki da Silva, Geiza , orient. II. Título.

Rio de Janeiro, RJ – Brasil.

Julho de 2017

Agradecimentos

Aos meus pais, pelo amor, incentivo e apoio incondicional.

A Universidade UNIRIO, pela oportunidade de fazer o curso e pelo ambiente criativo e amigável que proporciona.

A minha orientadora Geiza Maria Hamazaki da Silva, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos.

Aos meus colegas pelo companheirismo e disponibilidade para me auxiliar em vários momentos, com destaque ao Natália Costa, Igor Balteiro e Thais Mester que participaram ativamente no projeto.

RESUMO

Com a expansão da internet e o uso de dispositivos móveis, surgiram softwares para apoiar o deslocamento de pessoas. Desse contexto surgiu o estímulo de desenvolver o SUL, um sistema de gestão de rotas para parte interna de estruturas como prédios, shoppings e estádios. Atualmente as propostas para solucionar este problema são ineficientes na criação de rotas ou são custosas. Este projeto propõe desenvolver um aplicativo para a plataforma Android, a fim de facilitar o deslocamento de pessoas dentro de estruturas fechadas.

Palavras-chave: Geolocalização, Caminho mínimo, Rotas, Android, Dispositivo móvel.

ABSTRACT

With the expansion of the internet and the use of mobile devices, softwares had emerged to assist the orientation of people indoors. From this context came the motivation to develop SUL project, a route management system for the interior of structures such as buildings, malls and stadiums. Projects that try to solve this issue are currently inefficient in creating routes or are costly. This project proposes to develop an application for the Android platform in order to facilitate the movement of people inside closed structures.

Keywords: Geolocation, Minimal path, Routes, Android, Mobile.

Índice

Índice de Tabelas

Índice de Figuras

1 Introdução

1.1 Contextualização

O mercado de celulares está crescendo a cada ano, estudos mostram que atualmente no Brasil existem cerca de 168 milhões de *Smartphones* em uso com expectativa de serem 236 milhões nos próximos anos (Bruno,2016). Na esfera global, se calcula que 3 bilhões de pessoas no mundo possuem um aparelho celular, e isto corresponde a mais ou menos metade da população mundial atualmente (Lecheta, 2009).

Há tempos, o celular, por exemplo, deixou de ser apenas um dispositivo para receber e realizar chamadas, se tornando cada vez mais uma fonte de informação e entretenimento. Desta forma há um amplo mercado a ser explorado, com oportunidades para todas as áreas, que podem se apossar da tecnologia, capacidade e conveniência dos *Smartphones* para gerar ferramentas que sejam de interesse dos usuários. É comum o interesse por celulares com o grande variedade de recursos, criando uma demanda para o desenvolvimento de aplicativos que os satisfaçam e atendam as expectativas e necessidades de seus usuários.

O estudo feito neste projeto usando tecnologias de geolocalização, visa demonstrar o uso de ferramentas para desenvolvimento de aplicativos que facilitem a vida do usuário quando o mesmo precisa encontrar um determinado lugar (sala, auditório, prédio) numa determinada Instituição, de forma simples através de seu *Smartphone*. Segundo Favretto (Favretto. N,2012) cerca de 65% dos donos de *Smartphones* utilizam diariamente aplicativos de localização onde 15% aceitariam pagar por um serviço de boa qualidade.

A geolocalização é definida pela determinação do local em que uma entidade, pessoa ou estrutura é localizada. Esta ferramenta é utilizada por várias aplicações e sistemas no dia a dia. Uma área que se aproveitou

bastante desse conceito foram as redes sociais, na qual abrange mais de 2 bilhões de pessoas (Silva,2015) e que criou ferramentas intuitivas e interessantes.

Os aparelhos telefônicos inteligentes utilizam algumas ferramentas para o funcionamento da geolocalização; a mais conhecida e utilizada é o GPS. O GPS que a sigla significa para *Global Positioning System*, que na língua portuguesa significa “Sistema de Posicionamento Global”, e consiste numa tecnologia de localização por satélite. Os sinais dos celulares são enviados e recebidos por múltiplos satélites, assim criando a capacidade de triangular e definir a posição do equipamento portátil com uma certa precisão (Martins,2009).

A precisão do GPS é um grande problema para qualquer aplicação que deseja saber a exata posição do usuário, pois a tecnologia embutida em *SmartPhones* oferecem uma precisão de até 3 metros de circunferência de erro. Para agravar o problema os GPS’s não são capazes de informar profundidade, tornando impossível a definição em qual andar que se localiza o aparelho, tornando assim inviável o uso somente desta tecnologia para localização dentro de estruturas.

Há mais de uma década, mapas digitais e o aumento inexorável de *Smartphones* mudaram radicalmente a forma como localizamos, navegamos e planejamos nossas jornadas. Celulares inteligentes e computadores atuam hoje em dia como os antigos mapas de papel e os rústicos atlas, agora os celulares vêm com a tecnologia de GPS embutidas e funcionam como guias para tudo.

Aplicativos de geolocalização é um das interfaces entre o mundo físico e o mundo digital, o que significa que, as pessoas não precisam se perder novamente mesmo que não tenham sentido de direção. Este projeto vem como mais uma solução para tornar essa parte da vida das pessoas mais fácil, com o diferencial de guiar as pessoas dentro de localidades fechadas.

Neste trabalho será desenvolvido um aplicativo para a plataforma *Android*, que tem como objetivo primário orientar, por meio de mapas e rotas, seus usuários a se locomoverem por localidades fechadas. O aplicativo será

disponibilizado gratuitamente à todos aqueles que possuírem um dispositivo móvel *Android*. A UNIRIO (Campus 458) é o primeiro local a ser aplicado o projeto mas não será limitado para ser usado somente neste ambientes, sendo possível ser implementado em outros, aumentando o alcance do aplicativo para outros estabelecimentos como por exemplo: o Campi, shoppings, estádios e hospitais.

1.2 Organização do texto

Capítulo 2: Evidencia a análise dos serviços e produtos com o escopo similar a este projeto.

Capítulo 3: Ilustra todos os fluxos possíveis para o utilizar a aplicação pelo usuário.

Capítulo 4: Descreve as principais tecnologias envolvidas com o projeto e suas funcionalidades.

Capítulo 5: Mostra como como foi criado a estrutura do projeto para o suporte do desenvolvimento do algoritmo de caminho mínimo.

Capítulo 6: Explica detalhadamente o passo a passo do funcionamento do algoritmo de rota mínima.

Capítulo 7: Exemplifica o funcionamento da aplicação por meio de um teste controlado.

Capítulo 8: Mostra as conclusões deste trabalho junto com algumas propostas de trabalhos futuros.

2 Estado da Arte

Neste capítulo é apresentado a análise de projetos de similares com objetivo de verificar as características das aplicações que se propõe a solucionar o problema em questão ou parte deste. Os projetos/produtos analisados foram Google Indoors[1], Infsoft[2], Meridum Aruba[3], Mally[4], Point Inside[5] e VisioGlobe[6].

Como as empresas citadas são concorrentes e algumas com produtos similares, nem todas (**Point Inside** e **VisioGlobe**) foram descritas nesta seção, mas as características de seus produtos foram detalhados na tabela 1.

Google Indoors - Este Projeto está em desenvolvimento, é um produto da empresa Google que está integrado com o google maps, seu funcionamento é ativado quando um usuário do google maps dá um zoom em alguma estrutura, como um prédio, que foi mapeada pela empresa e contém o diagrama da estrutura. Depois que a aplicação é ativada o usuário pode visualizar toda a estrutura do complexo dividido por andares, é possível selecionar o andar desejado e o mapa é ampliado para melhor visualização. Nessa aplicação o usuário pode se localizar, se estiver dentro da estrutura, pela posição que o GPS disponibiliza. Porém, é exigida a seleção manual de qual andar ele se localiza. Google Indoors não oferece um sistema de geração de rotas, quer dizer, não é possível selecionar um local de partida e destino com intuito de gerar uma rota entre os dois pontos.

[1] Acessado em https://www.google.com/maps/about/partners/indoormaps/
[2] Acessado em https://www.infsoft.com/
[3] Acessado em http://www.arubanetworks.com/products/mobile-engagement/app-platform/
[4] Acessado em http://nocamels.com/2013/12/mally-wants-to-be-the-waze-of-indoor-navigation/
[5] Acessado em http://www.pointinside.com/
[6] Acessado em https://visioglobe.com/

Insoft - Esta empresa fornece como seu principal produto um sistema de navegação em locais fechados, a aplicação pode ser implementada em estruturas como shoppings, estádios e estacionamentos. Diferente do Google Indoors, este produto disponibiliza para o usuário o sistema de rotas, possibilitando a visualização do caminho que deve seguir para ir de um ponto A para o B, através da aplicação.

A navegação em qualquer local fechado é impossível de ser feita com precisão somente utilizando GPS, então a empresa criou um sistema de sinalizador (*beacon*) que utiliza a tecnologia *BlueTooth* e de aparelhos de *Wifi*, que estes são distribuídos dentro da edificação. Informação de posição são divididas entre smartphone e dispositivos, assim é possível fazer a triangularização bem precisa da posição dos usuários. Pela necessidade de existir aparelhos distribuídos no local desejado para o funcionamento do aplicativo, cria-se um custo para produção e distribuição dos dispositivos. Uma limitação de seu funcionamento é que o usuário esteja entre 5 a 15 metros dos dispositivos da Infosoft e estar conectado a wifi.

Meridium Aruba - Este produto dispõe de uma aplicação de navegação em locais fechados, porém o foco são para organização de eventos. O aplicativo disponibiliza localização e gerenciamento de rotas parecido com o Infosoft, utilizando *beacons bluetooth* para conhecer a localização do aparelho móvel.

Os pontos baixos de infraestrutura e limitação de distância dos *beacons* são os mesmos encontrados no produto passado, porém esta empresa dispõe de um sistema de gerência da aplicação que traz um dinamismo interessante, pois o contratante do produto pode trocar informações (como horário de palestras ou mudança de local de apresentação) com usuário final em tempo real.

Mally - O produto desta empresa tem o objetivo de atender usuários de shoppings, diferentemente dos outros aplicativos, este não disponibiliza nenhum sistema de localização ou rotas, porém fornece ferramentas para achar produtos, lojas ou atividades. A proposta também fornece mapas 3D do mapa do shopping, fotos e informações específicas de cada loja.

Mally é o único aplicativo analisado que disponibiliza uma ferramenta de interação entre os usuários, já que disponibiliza ferramenta de comentários para as lojas, atividades do shopping e compartilhamento de informações na rede social sobre produtos. Produtos podem ser pesquisados e comprados pela aplicação e retirados na loja.

Todos os produtos citados são desenvolvidos por empresas privadas nas quais criam suas aplicações com um foco específico para seu público alvo, desta forma pode ser observado uma divergência nos produtos mesmo com o mesmo escopo. Diferente das aplicações listadas, o **SUL** tem o objetivo de ser uma aplicação com objetivo de fornecer a ferramenta de geração de rota para o usuário, visando a fácil utilização e aplicação em novas localizações.

	Google Indoors	Infsoft	Meridum Aruba	Point Inside	VisioGlobe
Produto é a empresa		x	x	x	x
Sistema de Gerenciamento		x	x		
Multiplataforma	x				
Possível programação pelo contratante			x		
Troca de informação em tempo real			x		
Possibilidade de implementação em múltiplas localizações	x	x			x
Utiliza GPS	x			x	
Utiliza *beacon*		x	x		
Utiliza Wifi	x	x		x	
Localização do usuário	x	x	x	x	
Gerenciamento de rota		x	x		
Busca de destino para criar rota				x	
Informações de lojas				x	
Troca de informação entre Usuário				x	
Carrinho de compra				x	
Mapa 3D				x	

Tabela 1: Análise de similares

Com os dados das características existentes nos produtos similares do mercado e com a visão do objetivo deste projeto foi realizada uma análise da melhor relação entre produto e consumidor da aplicação, estas características serão apresentadas no capítulo quatro que trata sobre quais as ferramentas foram utilizadas

e no capítulo 5 que explica como foram aplicadas. Para um melhor entendimento das tecnologias, será detalhado os fluxos e funcionamento do aplicativo no capítulo a seguir.

3 O SUL

Este capítulo descreve a aplicação, apontando os detalhes do ciclo de funcionamento e interação com o usuário. Serão detalhados quais os meios de comunicação a aplicação disponibiliza para os usuários e as ações que devem ser realizadas por ele. O capítulo está dividido em: passo a passo da aplicação e limitações.

3.1 Passo a Passo da Aplicação

Nesta seção serão descritos e ilustrados os fluxos possíveis de utilização do aplicativo, ilustrando as telas existentes e as opções de escolha do usuário.

Tela Inicial

A tela inicial tem como função expor a Logo do projeto e disponibilizar a primeira função do aplicativo: a escolha do ponto de partida para a geração da rota. O usuário tem a opção de inserir este ponto de forma manual ou com a leitura de um QrCode. Esta escolha é apresentada em forma de dois botões localizados na parte inferior da tela (Figura 1).

Figura 1: Tela inicial do SUL

Escolha do ponto de origem

Se o usuário selecionar a opção de inserir o ponto de origem via QrCode, a câmera do aparelho é aberta para que, após o posicionamento correto da câmera, a leitura deste código seja feita.

Se a opção selecionada for para definir o ponto de partida manualmente, uma tela é aberta apresentando as localidades já cadastradas na aplicação (figura 2).

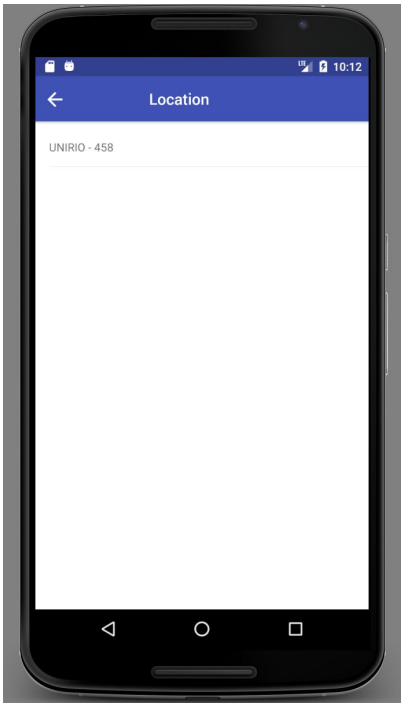

Figura 2: Tela de escolha de localidade

Depois da seleção da localidade a escolha do ponto de destino é necessária. Para isso, é disponibilizada para o usuário uma lista dividida por seções que representam cada estrutura, com todos os pontos, ver figura 3.

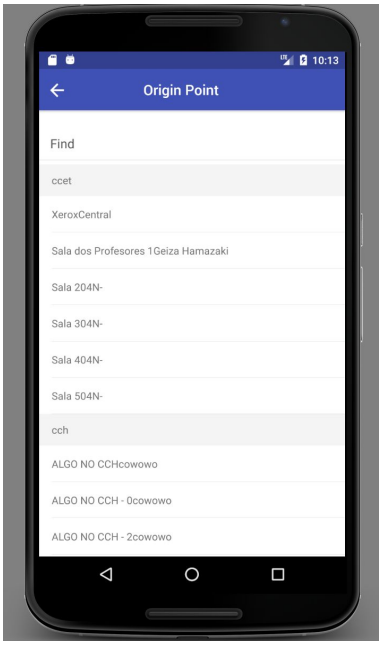

Figura 3: Tela de escolha de ponto de origem

Escolha do ponto de destino

Para o cálculo da rota, o usuário precisa fornecer, além do ponto de origem, o ponto de destino desejado. Estes pontos, que foram cadastrados previamente pelo administrador que cadastrou os mapas, são apresentados para o usuário em forma de uma lista. Tal lista segue o mesmo modelo da lista da escolha do ponto de origem, com os pontos divididos por estrutura. A tela para escolha do ponto de destino está representada na figura 4.

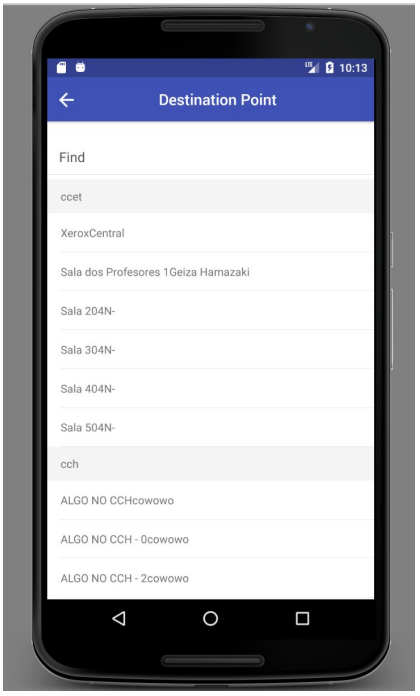

Figura 4: Tela de escolha de ponto de destino

Visualização do mapa

Após a escolha do destino, a rota de caminho mínimo é gerada pelo aplicativo, em seguida deve ser apresentado qual o caminho o usuário pode seguir. Para isso, sobre o mapa é feita uma sobreposição de imagens com dois elementos: os pinos que representam a origem e o destino e o caminho a ser percorrido, ilustrado por retas.

O ponto de saída é representado por um círculo com borda verde, e o ponto de destino é representado por um pino também verde, mostrado a seguir

na figura 5.

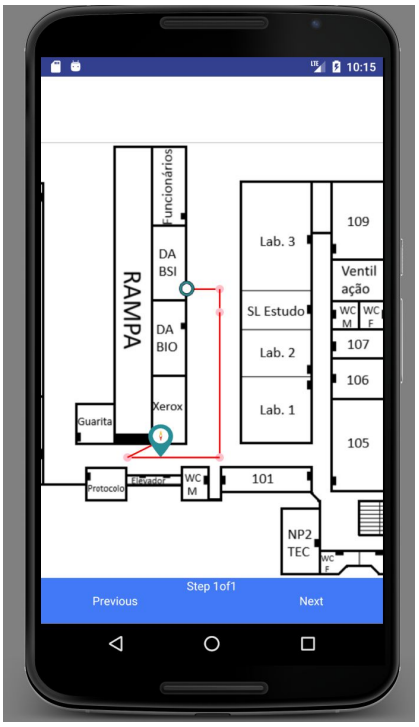

Figura 5: Tela de rota de mesmo andar

Se o ponto de origem e o ponto de destino não se encontram no mesmo andar do mesmo edifício, é necessário a subdivisão do caminho por andar. Neste caso, é disponibilizado na parte inferior da tela dois botões de deslocamento, que mostram o próximo mapa ou o mapa anterior. Também é criado um terceiro pino vermelho representando o ponto destino daquele andar. Este ponto está representado na figura 6.

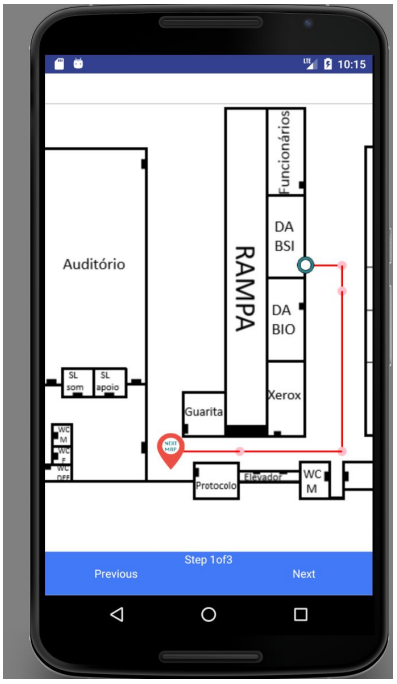

Figura 6: Tela de rota de andares diferentes

3.2 Limitações

Este projeto enfrentou problemas que moldaram o produto final. Tais empecilhos foram tomados como premissas para o projeto. No cenário ideal, as premissas devem ser identificadas na fase de levantamento do projeto, pois caso contrário o custo de modificar as especificações do projeto podem geram um custo maior ou tempo de trabalhado não calculado anteriormente. As principais premissas que moldaram este projeto foram:

- é impossível a utilização da tecnologia GPS para localização em edifícios;
- é impossível criar uma aplicação onde todos os dados da aplicação se encontrem no dispositivo móvel.
- a complexidade de se criar uma solução de caminho mínimo eficiente.

Com as definições dos fluxos das funcionalidades da aplicação junto com as limitações encontradas, é mais clara a escolha de qual tipo de tecnologias deve se buscar para cumprir os requisitos e obedecer às premissas do projeto. Por

esse motivo é possível introduzir as tecnologias utilizadas neste projeto, e o próximo capítulo irá detalhar quais são e o que propiciam para o SUL.

4 Tecnologias Utilizadas

As tecnologias e soluções utilizadas no desenvolvimento do SUL são relativamente novas e permitem grande modelagem de seus dados. Um dos requisitos do projeto é a utilização de ferramentas *Open Source* ou de uso livre, assim trazendo o custo financeiros para o desenvolvimento para zero. Outra característica relevante é o emprego de variadas tecnologias, abrangendo múltiplos ecossistemas como o mobile, desenvolvimento e servidores web. A figura 4.1 é uma ilustração de como estes ecossistemas se relacionam dentro do projeto.

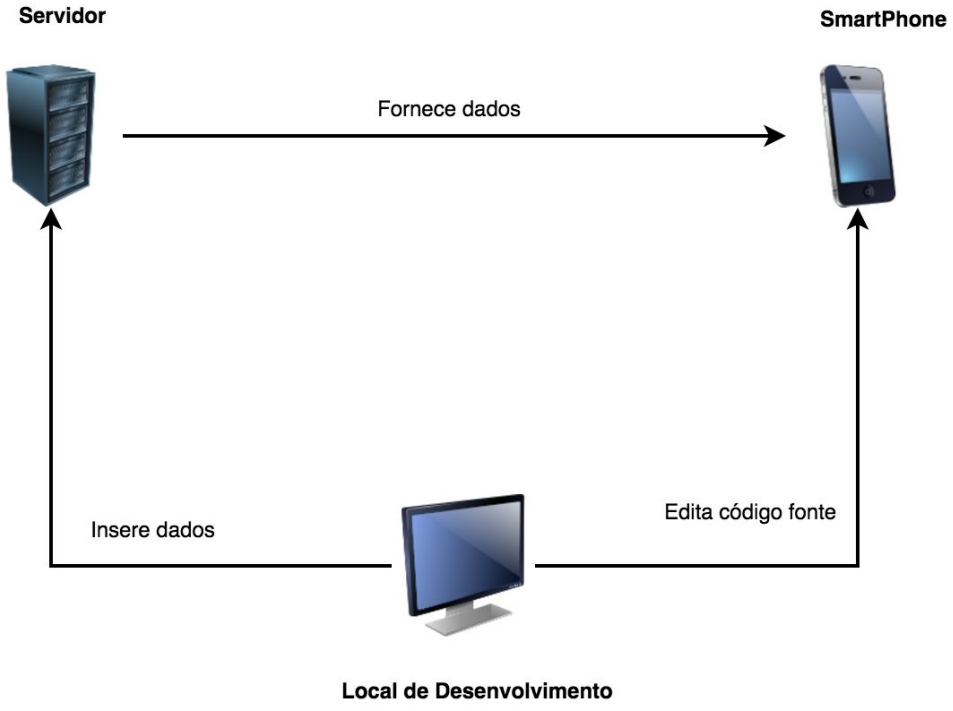

Figura 4.1: Relacionamento de ecossistemas

A criação do QrCode data mais de vinte anos, em 1994 (Brown, 2008), entretanto é pouco o conhecimento da sua capacidade de utilização. Ao mesmo tempo tornado cada vez mais presente na rotina das pessoas, os QRCodes ainda não são bem entendidos, sendo sua função principal ser um meio de transmitir rapidamente informações a dispositivos que são capazes de o ler. A leitura do código pode ser realizada por basicamente qualquer *Smartphone*, sendo somente necessário uma câmera e um software de leitura. O QRCode é um código de barras disposto em 2D que pode ser escaneado. Esse código, após a decodificação, passa a ser um trecho de texto.

A aplicação utilizará de imagens para dispor seus mapas para os usuários, e estes serão carregados somente quando necessário a fim de não existir uma sobrecarga do espaço. Compactação se define pelo processamento de um arquivo e na saída de um documento com um tamanho menor e que seja possível o reprocessamento para ter o arquivo original sem nenhum dano aos dados. (Harris,2011).

Muitas tecnologias foram utilizadas para o desenvolvimento deste projeto e, visando facilitar o entendimento das mesmas, suas definições estão descritas na seção a seguir.

4.1 Descrição das Tecnologias

As tecnologias deste tópico são referentes ao desenvolvimento de software, tanto para o *Smartphone* quanto para o servidor. As figuras 8, 9 e 10 definem as dependências internas dos três ecossistemas: mobile, desenvolvimento e servidores web. A figura a seguir (Figura 8) ilustra as interações entre funções, tecnologias e pacotes dentro do aparelho do usuário.

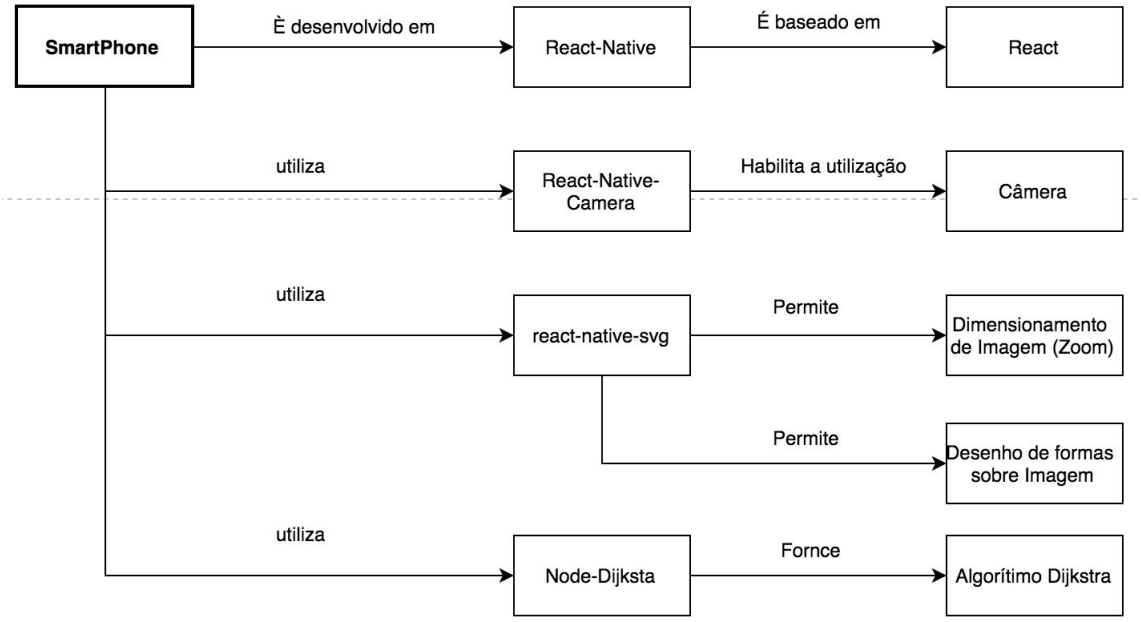

Figura 8: Diagrama de dependência do ecossistema do *SmartPhone*

A figura 9 especifica as tecnologias utilizadas para o funcionamento do servidor.

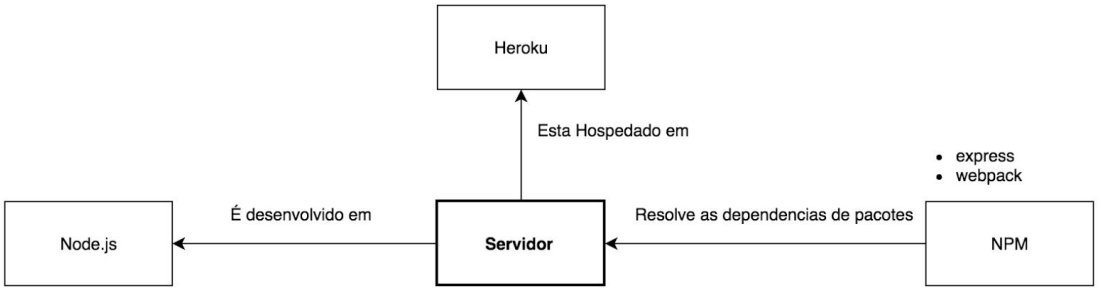

Figura 9: Diagrama de dependência do ecossistema do Servidor

Foi exemplificado na figura 10 as principais tecnologias utilizadas para possibilitar o desenvolvimento do projeto do SUL tanto para mobile, quanto

para o servidor.

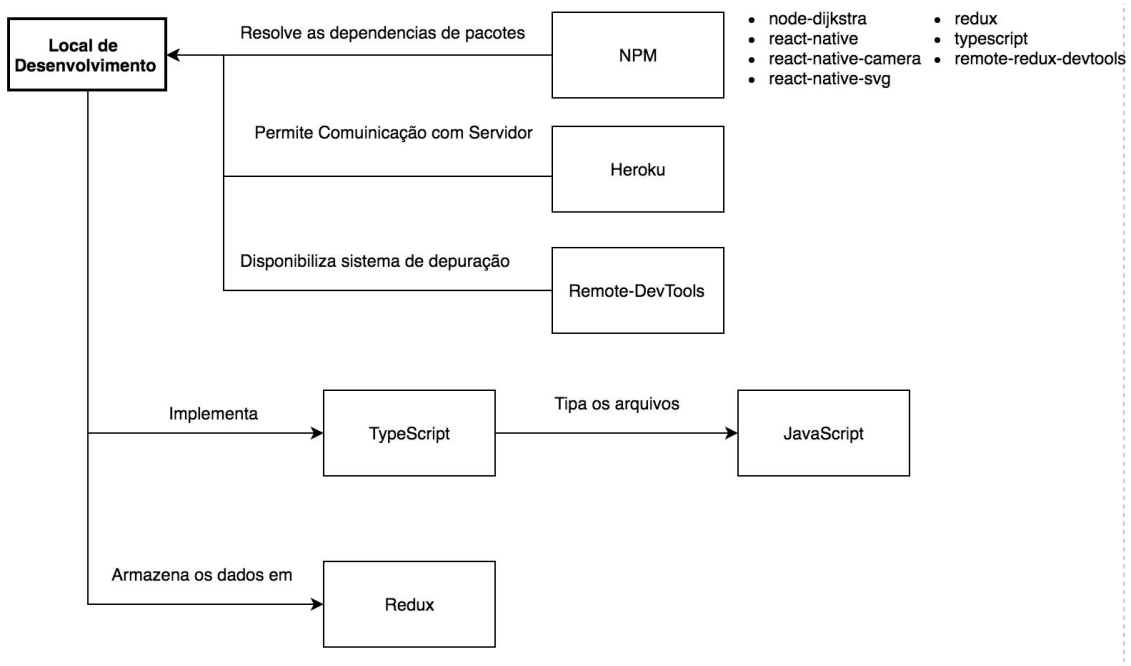

Figura 10: Diagrama de dependência do ecossistema do Local de Desenvolvimento

As dependências apresentadas nas três imagens acima estão descritas abaixo e foram divididas em dois tópicos: ferramentas e pacotes de terceiros.

4.1.1 Ferramentas

React:

O React disponibiliza componentes para renderizar apenas os dados necessários para as telas da aplicação, o que o diferencia das outras tecnologias do mercado. Além disso é possível com a biblioteca modularizar a interface com o usuário da aplicação Web com grande eficiência. Esta ferramenta facilita a argumentação, construção e manutenção de interfaces de usuário porque cria a divisão de seus dados em componentes e containers que facilitam a construção do projeto.

React-Native:

React Native é uma tecnologia construída em JavaScript para o desenvolvimento de aplicativos móveis com total funcionamento nativo para iOS e Android. É baseado no React, mas direcionado para aplicativos de dispositivos móveis. Desta forma os desenvolvedores web podem escrever aplicativos móveis que parecem e dão o sentimento de verdadeiramente aplicativos "nativos", utilizando uma biblioteca de JavaScript que é uma linguagem comum no mundo de desenvolvimento web.

Semelhante a aplicações para a *Browsers*, os aplicativos React Native são escritos usando uma mistura de JavaScript e XML *markup*, conhecido como JSX. Desta forma o React Native invoca as APIs de renderização nativas em Objective-C (para iOS) ou Java (para Android). Assim, o desenvolvimento é semelhante ao de qualquer página web, porém utilizando componente nativos do sistema operacional. React Native também expõe interfaces JavaScript para APIs de plataforma, de forma que seus aplicativos React Native possam acessar recursos desta, como a câmera do telefone ou a localização do usuário.

Redux

Redux tem o objetivo de organizar e direcionar todos os dados e estados da aplicação, a sua função pode ser considerada como um banco de dados de um *back-end* no lado do cliente, onde é armazenado todas as informações necessárias para gerar a exibição e funcionamento da aplicação. Este disponibiliza todas as funcionalidades que um banco de dados, como leitura, escrita e edição dos dados.

TypeScript

TypeScript é uma linguagem de programação livre, de código aberto e mantida pela Microsoft. É uma super classe de JavaScript, e adiciona opcionalmente a digitação estática e a programação orientada a objetos

baseada em classe para o idioma.

Este produto é projetado para o desenvolvimento de grandes aplicações pré-compiladas para JavaScript. Quaisquer programas JavaScript existentes também são programas TypeScript válidos. A tecnologia suporta arquivos de definição que podem conter informações de tipo de bibliotecas JavaScript existentes.

Devido à natureza dinâmica das variáveis JavaScript, pode ser difícil identificar *bugs* até que o código seja executado em tempo de execução. O TypeScript ajuda a resolver isso fornecendo a digitação estática opcional. A diferença entre a digitação dinâmica e estática pode ser enorme quando se trata de depurar um aplicativo. Também permite tornar o código mais legível para outras pessoas. Outro bônus de uma linguagem fortemente tipada é que os IDEs modernos podem tirar vantagem desse fato e fornecer opções de conclusão de código.

Node.js

Node.js é um ambiente do lado do servidor que permite aos desenvolvedores desenvolverem servidores e aplicativos com o JavaScript. Isso significa que sites inteiros podem ser executados em uma pilha de JavaScript unificada - tanto o software do lado do cliente quanto o software do lado do servidor.

Tecnicamente, é uma plataforma de desenvolvimento, não uma estrutura - com inúmeras estruturas que funcionam em cima dela. O Node.js executa uma construção de programação chamada "loop de eventos", que aguarda pedidos do cliente e os envia para o servidor ou banco de dados, no qual pode ser adicionado bibliotecas e conectores como HTTP, SSL e TCP para configurar rapidamente uma dinâmica de trabalho Servidor web, com apenas algumas linhas de JavaScript.

NPM

NPM facilita o compartilhamento de código entre os desenvolvedores de JavaScript, de forma que estes possam ser reutilizados. Uma vez que existe uma dependência de algum pacote no projeto, o NPM facilita a verificação e inserção de atualizações do código nas aplicações.

Esses blocos de código reutilizáveis são chamados de pacotes, ou às vezes módulos. Um pacote é apenas um diretório contendo um ou mais arquivos, que também possui um arquivo chamado "package.json" com alguns metadados sobre este pacote. Um aplicativo típico, como um site, dependerá de dezenas ou centenas de pacotes. Esses pacotes geralmente são pequenos. A idéia geral é que seja criado um pequeno bloco de construção que resolva bem um determinado problema. Isso permite a compilação de soluções maiores e personalizadas desses blocos de construção pequenos e compartilhados.

HEROKU

A Heroku é uma plataforma na nuvem que permite uma empresa criar, entregar, monitorar e dimensionar aplicativos dentro do seu ecossistema, basta somente fornecer metadados e um código fonte o qual será o ponto central no servidor, assim possibilitando a modelagem de qualquer função para um servidor.

DevTools

As ferramentas de desenvolvimento do Chrome (DevTools para abreviar) são um conjunto de ferramentas de criação e depuração da Web incorporadas no Google Chrome. O DevTools fornece aos desenvolvedores web um acesso profundo aos internos do navegador e à sua aplicação web. Este deve ser utilizado para rastrear problemas de *layout* de forma eficiente, definir pontos de interrupção de JavaScript e obter informações sobre otimização de código.

4.1.2 Pacote de terceiros

React-Native-Camera

O React-Native-Camera é um pacote do NPM, que permite a aplicação React Native tirar fotos e vídeos utilizando a câmera do celular. A instalação deste pacote é simples bem como sua utilização. Não sendo necessário então a programação nativa da linguagem (Java).

A única responsabilidade do desenvolvedor ao utilizar esta tecnologia é chamar as funções referentes às ações desejadas e o próprio pacote se responsabiliza em fazer a tradução e conversar com o sistema operacional do dispositivo para fazer as funções necessárias para a aplicação

Node-Dijksta

Este é uma implementação rápida de JavaScript do problema de caminho mais curto do Dijkstra para o NodeJS. Este pacote fornece para o desenvolvedor o código já adaptado para JavaScript do algoritmo de caminho mínimo do Dijkstra. Esta adaptação utiliza a estrutura de lista encadeada dos nós para implementar o algoritmo. A lista deve ser instanciada pelo próprio desenvolvedor.

react-native-svg

O react-native-svg é criado para fornecer uma interface para ser possível a visualização e interação de imagens em um aplicativo nativo tanto no iOS quanto no Android, isso quer dizer que é possível a manipulação das dimensões da imagem pelo usuário na tela do *SmartPhone*. Este pacote permite que a aplicação aplique alguns efeitos como zoom, redirecionamento e desenhos simples sobre a imagem.

Remote-DevTools

Redux DevTools deve ser usado com uma forma remota para React Native, híbrido, desktop e servidor Redux aplicativos. Este pacote permite que o desenvolvedor possa visualizar os dados dos estados da aplicação que estão sendo geridos pelo Redux, assim sendo uma interação entre aplicação e browser ou simulador local do celular (Android Studio).

Para o melhor aproveitamento de todas tecnologias acima citadas foi utilizada um editor de texto capaz de identificar e disponibilizar ferramentas para arquivos tipados. Como o Sistema operacional da equipe do SUL foi o MAC Os, foi escolhido então a IDE da MicroSoft, o Visual Code. Outra ferramenta fundamental para o projeto foi o simulador de Android do Android Studio, que é capaz de simular o sistema operacional e todas as funcionalidade de um *SmartPhone* dentro de um desktop.

5 Aplicação das Tecnologias

O software SUL foi desenvolvido para disponibilizar para o usuários de smartphones uma aplicação geradora de rota mínima entre um ponto A para o B dentro de uma estrutura, assim foram usados várias tecnologias voltadas para mobile e algoritmos para auxiliar na busca dos resultados desejados. Este capítulo irá descrever a solução do problema.

A utilização do GPS para a localização do aparelho celular é impossível por sua baixa precisão e sua incapacidade de reconhecimento de profundidade, não possibilitando assim o reconhecimento do andar que o dispositivo se encontra dentro de um edifício Por esse motivo foi escolhida a tecnologia de QrCode para ser um meio de localização. Desta forma é necessário a utilização da câmera do celular para a leitura dos QrCodes, na solução proposta são criado blocos de código reutilizáveis, selecionado o *React-Native-Câmera como integrador da aplicação com a câmera do celular.*

Em um mundo ideal, todos os projetos de software devem passar pelo fase de levantamento de requisitos para auxiliar o melhor entendimento de como vai ser construído todo o sistema da aplicação, nesta etapa do projeto a equipe de desenvolvimento do SUL percebeu a necessidade da utilização de um servidor para o funcionamento das aplicações localizadas nos *Smartphones* de seus usuários, dado que não é possível cada dispositivo arcar com responsabilidade de armazenar os dados da aplicação.

Deve-se ter em mente que o aplicativo foi criado para ser implantado em vários estabelecimentos. Dados como: planta de cada prédio, informações sobre salas e fotos, devem ser armazenados, porém com o aumento do volume de dados é necessário uma grande utilização da memória dos dispositivos dos usuários se tais informações fossem armazenadas nestes. Desta maneira seria observado um grande prejuízo na eficiência da aplicação e então foi tomada a decisão de guardar essas informações em um servidor.

Como este projeto não teve nenhum investimento externo, foi feito a pesquisa para a escolha do hospedeiro que estaria mais apto para suportar o servidor, o resultado foi a escolha do serviço chamado *HEROKU* que permite rodar projetos em Node.js sem nenhum custo no caso de se limitar a um número de chamadas por tempo determinado.

O princípio da aplicação é achar o caminho que o usuário deve tomar para chegar de um ponto A para o B. Para alcançar o resultado desejado deve-se abstrair o problema do mundo real em alguma estrutura computacional, a representação escolhida é a estrutura de grafos, que neste problema será conexo. O conjunto de nós e arestas deste grafo representam os possíveis caminhos dentro de um mapa, dessa forma pode-se obter o caminho mínimo entre a origem e o destino..

Existem no mundo vários algoritmos que solucionam este problema, alguns com uma performance melhor que outros. Como exemplo o Algoritmo de Bellman-Ford (Thomas, 2007) que resolve o problema para grafos com um vértice-fonte e arestas que podem ter pesos negativos ou o Algoritmo A* (Russel, 1995) que é um algoritmo heurístico que calcula o caminho mínimo com um vértice-fonte. Na solução proposta foi escolhido o Algoritmo de Dijkstra que resolve o problema com um vértice-fonte em grafos cujas arestas tenham peso maior ou igual a zero. Sem reduzir o desempenho, este algoritmo é capaz de determinar o caminho mínimo, saindo de um vértice de partida para *todos* os outros vértices do grafo.

Com o algoritmo escolhido, foi selecionado o pacote do NPM Node-Dijksta, que fornece o algoritmo desejado com baixo custo de implementação. Para o funcionamento do pacote é preciso os nós, quais são

adjacentes e qual o peso para acessar o nó adjacente, assim criando internamente o grafo desejado. Para obter o caminho entre 2 pontos é preciso somente colocar o nome de cada nó em uma função e esta retorna o resultado.

Para o funcionamento do algoritmo escolhido é necessária a construção de de um ecossistema robusto capaz de fornecer os dados necessários e modelá-los corretamente. As características deste ecossistema é detalhado na seção 5.1.

5.1 O ecossistema

Nesta seção são apresentados a implementação do ecossistema do projeto e como as duas grandes partes do software da aplicação, o servidor e o aplicativo, se comunicam para solucionar o problema proposto.

5.1.1 O servidor

O servidor tem a função de fornecer as informações e imagens de todos os estabelecimentos que o projeto foi implantado, e precisa fornecê-los para todos os dispositivos conectados a ele. Além disso este deve estar disponível para qualquer requisição de seus usuários, por este motivo foi escolhido o serviço da empresa Heroku.

Projetos que usam o Heroku devem criar uma conta no site da empresa, e fazer o processo de liberação do serviço. É possível a criação de um projeto totalmente gratuito se for respeitado uma lista de condições e esta foi a escolha feita para a aplicação do SUL. Após a criação do projeto deve ser gerado um primeiro arquivo com metadados do novo projeto para fazer o primeiro *upload* de arquivos. Em seguida, qualquer utilizador do serviço deve criar uma forma de fazer o *upload* de seu código fonte. Existem 3 formas de realizar este processo: *upload* manual, utilização de webhook do github ou a utilização de CLI do Heroku.

Como a está na própria descrição, o *upload* manual é feito pelo *input* do

código no site do serviço manualmente, neste método não é necessário nenhum recurso ou tempo para configurar o sistema, porém existe o custo rotineiro de abrir o site e fazer um *upload* mais lento pelo browser. O webhook do github é realizado por uma interação entre o GitHub com o Heroku. Quando um novo código é inserido no GitHub é disparada uma ação avisando o Heroku que é necessário a atualização de seu serviço. A vantagem deste modo é que é possível utilizar um sistema de controle de versão de arquivos para o projeto, o lado negativo é o grande esforço e conhecimento do sistema do GitHub que é necessário para fazer tudo funcionar perfeitamente. A última forma, que foi a empregada na implementação, é a do CLI do Heroku, a vantagem deste método é que o próprio Heroku cria um projeto do GitHub para utilização, deste modo mantendo o serviço de controle de versionamento, possibilita também aplicar comandos no terminal para verificar a saúde e funcionamento do servidor remotamente e fazer *upload* do código via terminal com uma velocidade bem maior, o problema é a necessidade de um grande conhecimento de utilização de terminal, GitHub e do sistema do Heroku.

Dado a solução de hospedagem do serviço, a próxima decisão é qual tecnologia deve ser utilizada para desenvolver o código fonte do servidor. Considerando que o servidor deve ser capaz de responder com uma velocidade de boa qualidade e com baixo custo, além do conhecimento que o Heroku suporta a linguagem JavaScript, foi escolhido o Node.js como tecnologia para ser utilizada. Leve e eficiente, o Node.js é ideal para aplicações que necessitam de escalabilidade e grande quantidade de troca de dados.

Os princípios adotados na construção do servidor é que este deve ser de fácil utilização e de grande escalabilidade. É fundamental que outras pessoas, fora a equipe de de desenvolvimento do projeto, possam fazer a atualização e manutenção dos dados dentro do servidor com facilidade, adicionando o fato que o processo de adicionar uma nova localidade fosse rápido e intuitivo. Por estes motivos foi escolhido uma estrutura de arquivos para o armazenamento dos dados.

5.1.2 O Aplicativo

Um software instalado no celular é a vitrine de todo o projeto do aplicativo móvel. Por esse motivo, uma atenção especial deve ser tomada com a sua aparência e fluidez. Uma das escolhas de todo projeto é em que tipo de plataforma ou sistema operacional o serviço deve funcionar, no caso do SUL foi escolhida a plataforma Android como pioneira pois é *Open Source* o que quer dizer para o projeto que pode ser utilizada livremente sem nenhum custo.

Com o objetivo de criar uma aplicação escalável foi escolhida a utilização do React Native uma plataforma que permite seu desenvolvimento na linguagem JavaScript e que tem a capacidade de se acoplar no sistema Operacional IOS e Android. Desta maneira, não é preciso desenvolver um aplicativo para cada plataforma e sim somente um que utiliza ferramentas nativas de ambas as plataformas.

5.2 Arquitetura do Sistema

Softwares tem como objetivo obter dados do mundo real, fazer operações com estes e transformá-los em informações, que são o produto final da aplicação. Os dados estão diretamente relacionados com o objetivo de cada projeto, assim deve-se criar ações e processos que facilitem todos os passos de funcionamento da aplicação. Nesta seção é discutida a estrutura do software que foi implementado para armazenar os dados e como estes foram modelados para o SUL.

5.2.1 Armazenamento de Dados

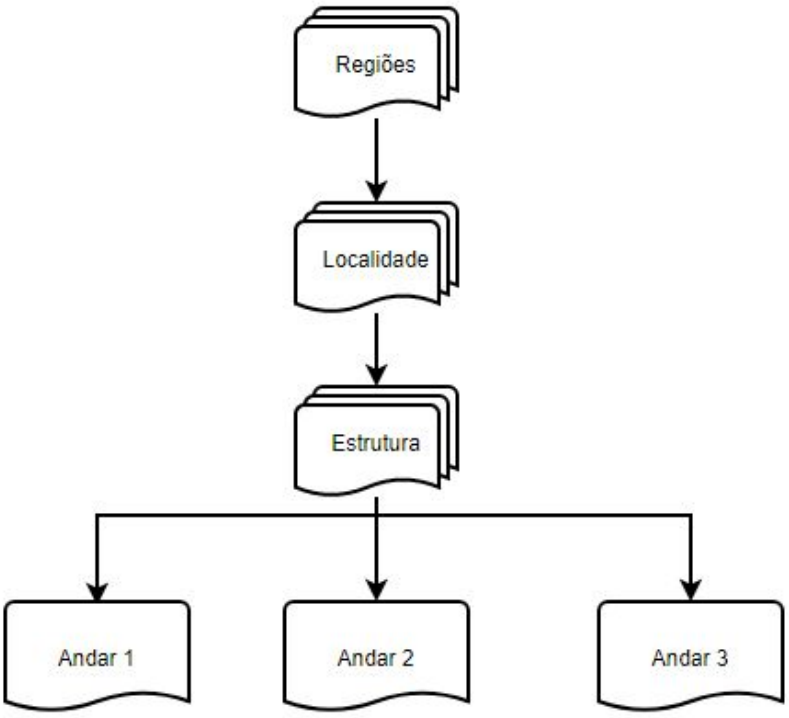

Figura 11: Diagrama de hierarquia de arquivos

Foi escolhido o sistema de arquivos como modelo de armazenamento por se assemelhar com a disposição dos mapas dentro da maioria dos atlas, assim trazendo uma melhor familiaridade com o sistema. Isto quer dizer que os dados são divididos hierarquicamente, as informações que representam o menor nível da hierarquia ou que abrangem menos terreno (no exemplo da figura 11 os andares) estão localizadas dentro dos arquivos de alto nível ou que abrangem mais território (Regiões). Esta estrutura pode ser exemplificada na forma que as informações são dispostas em um atlas, são mostrados primeiramente os mapas com maior área, como por exemplo os países, e depois são mostrados os mapas com menor representatividade, como por exemplo os bairros de uma cidade.

Os campos da figura 11 podem ser multivaloradas, isto é, podem existir casos com um ou mais elementos dentro de cada pasta. Arquivos de mesmo nível devem sempre estar referenciando ao arquivo de mais alto nível, por exemplo, um andar deve sempre referenciar em qual prédio está localizado.

Por definição do sistema o último elemento da hierarquia será sempre um andar e o conjunto de regiões é o primeiro nó da árvore da figura 11.

5.2.2 Modelagem dos Dados

Nesta seção serão apresentados como foram configurados os arquivos dentro das pastas e os dados dentro dos arquivos.

Dado que as pastas dispostas em sistema de arquivos, onde cada arquivo possui um significado para os demais arquivos dos outros níveis. Existem três tipos de arquivos dentro do servidor: Imagem, Arquivo de definição de possíveis rotas (AQR) e Arquivo de definição de mapas existentes (AQM).

Como a aplicação tem como objetivo guiar o usuário por dentro de um mapa, tem-se que a imagem de todos os locais devem ser armazenadas. Estas são definidas pelos arquivos de imagem e tem a extensão .PNG . As imagens representam os mapas detalhados dos locais, com as limitações (paredes ou muros), assim quando calculada, a rota será desenhada sobre ela. A ilustração a seguir (figura 12) é um exemplo desse tipo de arquivo.

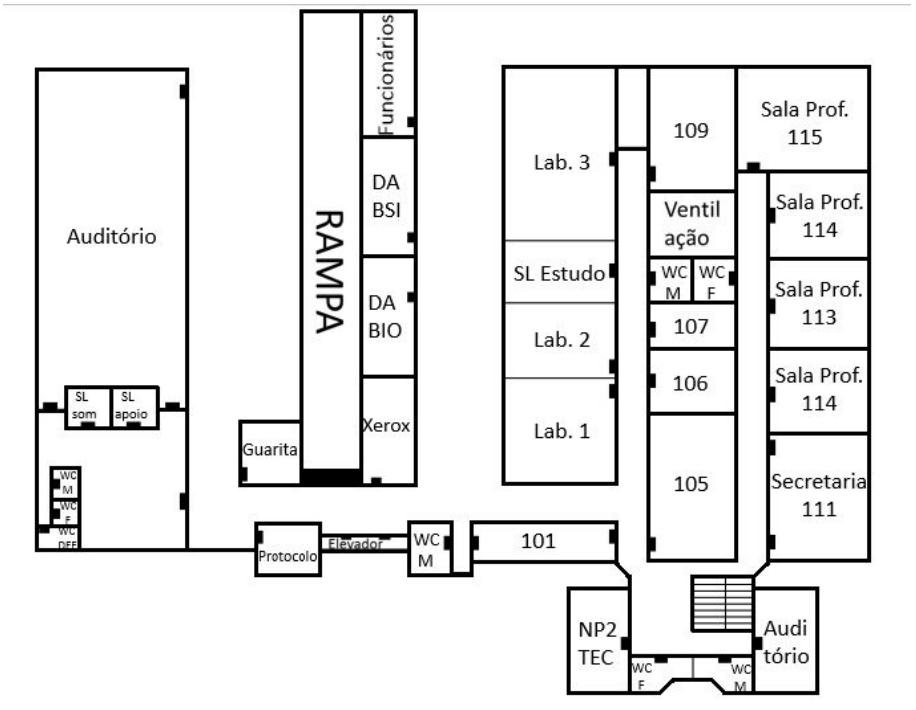

Figura 12: imagem do primeiro andar do CCET

O pilar do funcionamento do aplicativo são os arquivos de definição de possíveis rotas, visto que são usados para o cálculo da rota que o usuário deseja. Dessa maneira, quando alguém utiliza a funcionalidade de achar uma rota o que é modelado são as informação contidas nestes arquivos.

Os AQR são constituidos por um conjunto de objetos de formato Json, estes definem os múltiplos pontos espalhados pelos mapas e suas possíveis conexões entre si e tem a funcionalidade de guardar as possíveis rodas de um mapa. A estrutura dos arquivos AQR foi definida com as seguintes características:

- É necessário indicar em cada ponto qual mapa em que se localiza (*MapReference*),
- qual a sua posição dentro do mapa por coordenadas (*X* e *Y*),
- os nós adjacentes deste ponto (*adjacentes*), e
- cada ponto deve ter uma identificação única (id).

A figura 13 ilustra um exemplo do objeto de um arquivo de definição de possíveis rotas.

Figura 13: Objeto do arquivo de definição de possíveis rotas

O campo *BuildingReference* representa qual estrutura o ponto pertence, por exemplo, se tal ponto estiver dentro de algum prédio esse campo indicaria o nome deste prédio. O campo *Type* pode assumir 2 estados *Transition* e *Path* que definem se são pontos de conexão entre mapas diferentes ou não,

respectivamente. O campo *TrasitionAcces* existe somente se o nó representa um ponto de conexão entre dois mapas. Pode-se observar que este conjunto de objetos são sobrepostos na imagem de um mapa, e estes pontos e suas conexões são utilizados para o cálculo de caminho mínimo. Este processo de calcular um caminho entre pontos será melhor detalhado mais à frente no capítulo 6.

O ACM, é um arquivo de controle e comunicação entre o servidor e a aplicação. Estes também são definidos como arquivos .Json e foram projetados para minimizar ao máximo o custo de armazenamento e processamento do celular, dado que contém o mínimo possível de informação. As informações consistem na URL de onde encontrar o mapa e os metadados, então a aplicação irá fazer o *download* das imagens dos mapas somente quando necessário, minimizando o custo de armazenamento no dispositivo. A imagem 14 a estrutura dos arquivos ACM

Figura 14: Objeto de controle e comunicação I

O objeto *PointsOfInterest* tem um campo chamado *Description* que representa para o aplicativo o texto que será exibido para o usuário na descrição dos pontos de destino, para exemplificar, se um *PointsOfInterest*

representar um auditório, pode ser definido neste campo a descrição de uma apresentação do dia. O seu *Id* e o campo *Description* são utilizados para filtro na seleção de ponto de origem e ponto de destino.

O objeto *Maps* disponibiliza todos os dados necessários para a representação dos mapas da aplicação. Todo mapa precisa ter a identificação de qual prédio ele se encontra e o andar que está localizado. O campo *Path* representa a URL no qual o servidor responde a imagem .PNG, assim se necessário a utilização de um certo mapa esta URL representa o endereço que o servidor disponibiliza a imagem. Todos os mapas devem ter um tamanho de altura e de largura, representados por *Height* e *Width,* respectivamente, para assim possibilitar o desenho sobreposto de rotas.

Uma outra versão do arquivo ACM é o arquivo de definição de quais os ID's das estruturas dentro de uma localidade. Esse é o primeiro arquivo que deve ser obtido pelo aplicativo para seu funcionamento, pois é necessário estes ID`s para obter os seus arquivos de configuração. A figura 15 representa tal arquivo.

Figura 15: Objeto de controle e comunicação II

5.3 Implementação

O servidor tem a função de armazenar as informações necessárias para a execução dos softwares que compõem o SUL. Já o aplicativo tem a função de ser o centro de processamento de tais dados com o intuito de atingir o objetivo desejado. Para alcançar o propósito do projeto várias pequenas soluções tiveram que ser desenvolvidas:

- a comunicação entre aplicação e servidor;
- desenho da rota dentro de um mapa;

- saber reconhecer de onde ir e pra onde ir;
- como adaptar o algoritmo de Dijkstra para aplicação.

Na seção anterior (5.2) foi explicado onde e como os dados da aplicação são armazenados, porém não foi detalhada como é feita a troca de informações entre servidor e aplicativo. Este processo pode ser dividido em dois passos: o de obtenção de arquivos de definição da localização e o de obtenção das imagens do mapa. Esta divisão tem o objetivo de utilizar espaço da memória do *Smartphone* com eficiência, fazendo as requisições para o servidor de acordo com necessidade.

Pode-se exemplificar esta afirmação no momento que é selecionado uma localização e nesse momento são baixado os arquivos de configuração contendo todos os metadados de rota. Somente no momento que algum mapa é apresentado na tela do usuária é feito o *download* da imagem correta, não carregando dados não necessários para o dispositivo. O fato que a estrutura dos arquivos .PNG ocupam muito mais espaço de memória que arquivos .Json, explica o motivo de não trazer junto com os dados AQR as imagens.

Geralmente busca-se criar aplicativos simples e intuitivo para o público alvo, e esta é uma das premissas do SUL. Por este motivo buscou-se uma forma simples e limpa de apresentar o caminho que o usuário deve tomar para chegar no ponto desejado. Foi escolhido um *design* dos mapas parecido com aplicativo Google Maps (detalhado na seção de Análise de Similares) que se resume em um mapa responsivo: sensível ao zoom, que possui um ponto de origem, um ponto de destino e uma linha representando o caminho que deve tomar. Para atingir este objetivo foi utilizado o react-native-svg que disponibiliza a ferramenta de zoom de uma imagem e ao mesmo tempo permite o desenho de formas simples como círculos e retas sobre a mesma.

```
<ImageZoom cropWidth={Dimensions.get('window').width}
    cropHeight={Dimensions.get('window').height}
    imageWidth={this.props.mapMetadata.width}
    imageHeight={this.props.mapMetadata.height}>
    <View>
        <Image style={{width:this.props.mapMetadata.width, height:this.props.mapMetadata.height,position:'absolute'}}
            source={{uri: this.props.mapMetadata.path}}/>
        <View style={{position:'absolute'}}>
            <DrawPath
                destinationPoint={this.props.destinationPoint}
                currentMapName={this.props.currentMapName}
                mapMetadata={this.props.mapMetadata}
                pathOriginToDestinationCurrentMap={this.props.pathOriginToDestinationCurrentMap}
                getPointCordenates={this.props.getPointCordenates}
                totalMapIndex={this.props.totalMapIndex}
                />
        </View >
```

Figura 16: Objeto do ImageZoom

A *Tag ImageZoom* representa este pacote, onde é necessário fornecer as dimensões da imagem e da tela do celular, dado que o componente precisa destes dados para encaixar na tela as imagens, pois existem vários modelos de *Smartphone* e cada um tem um formato de tela diferente. A *Tag* Image representa a imagem que será desenhada na tela do *Smartphone* e a propriedade *Source* recebe a URL do servidor do atual mapa a ser mostrado. A *Tag DrawPath* é responsável pelo desenho dos pontos de origem, os pontos de destino e o caminho a ser percorrido. O componente *DrawPath* obtém uma certa lógica computacional, pois este deve ser capaz de identificar se os pontos que devem ser colocados são pontos finais ou não. Outras tarefas como: onde devem ser colocados e como serão desenhado toda a rota entre eles também são de sua responsabilidade.

O responsável por decidir qual vai ser o ponto de início e o ponto de destino é o usuário da aplicação. Existem duas formas de seleção de ponto de origem: a leitura de um QrCode e a seleção manual do local do usuário, o ponto de destino sempre será fornecido manualmente.

Para definir qual o ponto de origem para *input* manual, deve-se primeiramente definir qual é a localização desejada, neste passo do processo todos os dados desta localidade são entregues para o aplicativo. A informação desejada para definir o ponto são os *interestPoints* que são possíveis origens e destinos. Na tela do usuário é possível fazer uma busca de qual o ponto de origem desejado, utilizando o nome do campo ou sua descrição. Com a origem

selecionada a mesma etapa deve ser selecionada para o destino.

Para definir qual o ponto de origem com a leitura de QrCode, deve-se abrir a aplicação, ir em uma localidade que dispõe dos QrCodes da aplicação e escaneá-lo. Depois da leitura a aplicação salva todos os dados obtidos do QrCode e os processa com o intuito de criar um ponto de origem válido. Como já dito, é possível criar rotas de um ponto de interesse para outro ponto de interesse. Pelo motivo que o QrCode pode referenciar qualquer ponto dentro do mapa, então o QrCode tem a capacidade de simular dados de um *PointOfInterest*, desta forma podendo ser considerado como um ponto de origem válido. O QrCode depois de decodificado é uma representação de um objeto Json, um exemplo deste objeto pode ser visto na figura 17.

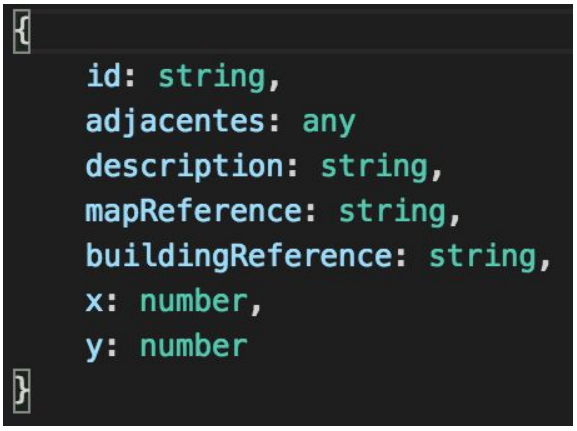

Figura 17 : Objeto de *PointOfInterest*

A figura 18 representa um QrCode válido que tem uma posição dentro de algum mapa.

Figura 18 : Imagem de QrCode válido

No desenvolvimento desta solução o intuito foi utilizar algoritmos já existentes. Assim tem-se que este projeto fez a escolha da representação do possíveis caminhos mínimos de seus mapas por grafos e subgrafos.

Um subgrafo pode ser definido como um grafo cujo conjunto de vértices é um subconjunto do conjunto de vértices de outro grafo. Para exemplificar no caso do SUL, o grafo que um andar de um prédio representa é um subgrafo deste mesmo prédio. Pode-se concluir então que no sistema de hierarquia tem-se que o um arquivo de mais baixo nível representa um subgrafo de um arquivo de mais alto nível.

No caso do aplicativo do SUL, tem-se a representação de uma localidade por um grafo conexo configurado por todos os pontos, sendo pontos de rotas e de interesse. A definição de acessibilidade de um nó ao outro se dá pelo campo *Adjacentes* já mencionados anteriormente.

Existe uma inconsistência de pensamento se for analisado que os arquivos no servidor são armazenados em pastas e subpastas e que o grafo da localidade é um objeto único. Este problema teve que ser abordado de uma forma que fosse possível transformar os nós representados nos arquivos da hierarquia em um grafo conexo, assim foi utilizado o campo *TransitionAccess* que tem a funcionalidade de fazer a conexão entre nós de arquivos distintos.

Este campo então, tem a funcionalidade de fazer a conexão entre dois pontos em mapas diferentes, podendo representar por exemplo em um prédio uma escada, um elevador, ou uma porta para o lado de fora deste mesmo edifício. A figura 19 ilustra a união de dois pontos de andares diferentes.

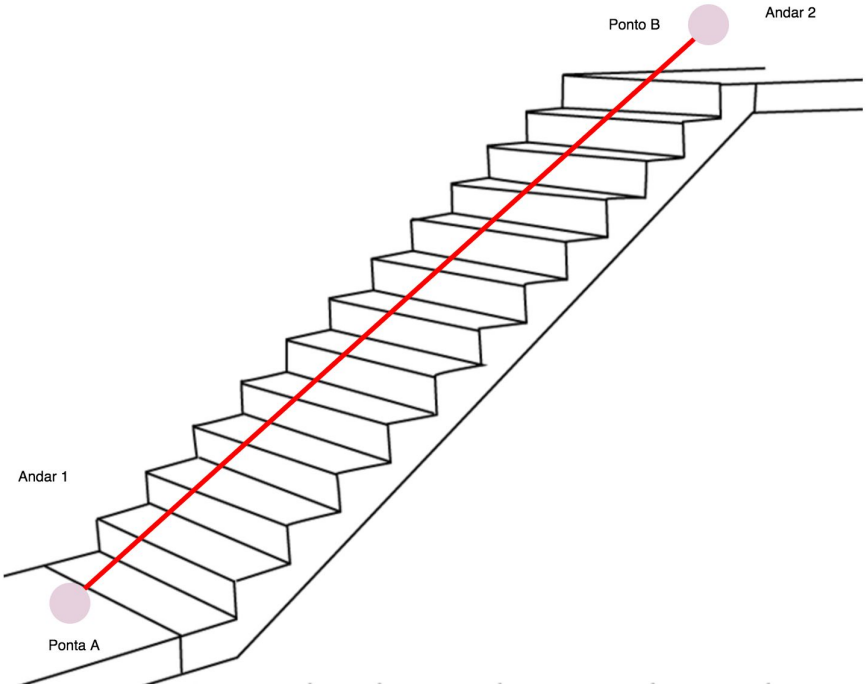

Figura 19: Ilustração de *transitionAccess*

Com o detalhamento de como foi modelado os dados do servidor para que se encaixem em uma estrutura de grafos, é possível entender as dificuldade encontradas para o funcionamento do projeto que são apresentados no próximo capítulo.

5.4 Dificuldades Encontradas

Este seção visa resumir as dificuldades encontradas no desenvolvimento do presente Trabalho.

Nos primeiros passos do desenvolvimento foi decidido a utilização do serviço do HEROKU como hospedeiro do servidor, foi escolhido o sistema de *upload* do código fonte da aplicação pelo CLI deste servico, porém essa não foi

a primeira escolha. O desejo da equipe de desenvolvimento do projeto foi a utilização do webhook do GitHub, porém a inconsistência de sucesso e falha no envio de tal webhook para o HEROKU pelo GitHub desestimulou o seu uso.

Para uma evolução com mais velocidade, o desenvolvimento de um software deve ser auxiliado por uma boa ferramenta de depuração, para que erros lógicos no código sejam identificados com velocidade e a tomada de decisão seja rápida e precisa. Por este motivo o DevTools foi escolhido como tecnologia de depuração, porém no desenvolvimento do código o projeto se deparou com um problema de versões de pacotes que impossibilitou a utilização desta ferramenta, ditando assim a necessidade de implantação do Remote DevTools que é uma versão mais simplificada da tecnologia.

A instalação do app no celular foi uma etapa árdua, pois o nível de conhecimento do React-Native e do Sistema Operacional do Android deve ser elevado para a implantação. Dentro do código fonte do aplicativo vários arquivos devem ser definidos para que a aplicação possa rodar em um *SmartPhone* e tais definições são complexas e poucas informações podem ser encontradas na internet, assim dificultando a migração do projeto do sistema de desenvolvimento para produção.

A etapa que foi responsável pelo maior uso de recursos do projeto foi a definição do algoritmo que calcula a rota mínima. Para chegar no resultado final algumas dificuldades foram superadas, principalmente para adaptar o algoritmo de Dijkstra. Para o cenário deste trabalho a única implementação na linguagem desejada que foi encontrada foi a do pacote Node-Dijkstra, mas mesmo para a utilização deste pacote algumas adaptações foram feitas, pois os tipos de objetos que são recebidos pelas funções do pacote são muito diferentes dos que são necessários para o funcionamento da aplicação, assim demandou um custo muito grande de transformação de objetos e cria uma complexidade robusta no projeto.

6 Algoritmo de caminho mínimo

Neste capítulo será explicado o algoritmo desenvolvido para o cálculo da rota mínima. Este cálculo poderia ser realizado sobre o grafo que representa toda uma localidade, porém um dos desafios deste trabalho foi criar uma forma adaptada e eficaz para utilizar o algoritmo. Para a possibilidade de tal adaptação foi necessário uma subdivisão do problema em três cenários. Estes devem ser identificados pelo programa e tem como objetivo gerar um menor trabalho de processamento, oferecendo para o usuário uma sensação de leveza para o aplicativo.

6.1 Do Algoritmo

Como prérequisito para tomada de decisão sobre esses cenários, as informações de ponto de origem, ponto de destino e os arquivos de configuração da localidade devem ser baixadas previamente no dispositivo do usuário. Primeiramente é observado dois dados dentro dos pontos de origem e ponto de destino, o *MapReference* e o *BuildingReference*, com esta informação procura-se saber em qual dos casos a seguir se encontra a busca: Se o ponto de origem e destino estão no mesmo mapa ou andar, se se encontram na

mesma estrutura mas em andares diferentes ou se localizam em estruturas diferentes.

Para o caso que o ponto de origem e destino se localizam no mesmo mapa, tem-se que não é necessário o cálculo de caminho mínimo no grafo completo e sim no subgrafo que representa aquele andar. Já que os dados de um andar são armazenado somente em um arquivo, não existe a necessidade de utilizar múltiplos arquivos de definição e nem a junção de subgrafos.

Para o caso do ponto de origem e destino se encontrarem na mesma estrutura mas em andares diferentes, tem-se que é preciso fazer a união dos subgrafos de todos os andares e gerar um novo grafo que representa o edifício. Dado o grafo é aplicado o algoritmo de Dijkstra obtendo a rota desejada. Um novo problema surge no momento de exibir a rota, o fato que só é possível visualizar um mapa por vez na tela do celular, assim deve ser feito nessa parte do algoritmo uma divisão sequencial de cada mapa, possibilitando o usuário observar o passo a passo para chegar no destino.

O caso em que o ponto de origem se encontre em um prédio diferente do ponto de destino tem grau de complexidade maior, pois não seria conveniente aplicar a mesma lógica utilizada para os dois casos anteriores de somente calcular a rota mínima, porque agora é preciso fazer a união de todos os subgrafos daquela localidade. Foi feito um estudo de como seria a melhor abordagem deste tema e chegou-se à um modelo que subdivide o problema em subtarefas, o que significa para o contexto que deve-se achar o caminho mínimo dentro dos subgrafos e depois com vários caminhos mínimos gerar um novo grafo mais simplificado e sobre ele fazer o cálculo final.

A ideia da solução consiste em, resolver vários problemas de tamanho menor ao invés de tentar resolver somente um problema maior. Assim para solucionar o problema o algoritmo monta vários subgrafos que unindo representam as estruturas. Desta forma pode-se exemplificar que na figura 20 podem ser observados três subgrafos dentro desta localidade, um para cada prédio e um para a Área comum.

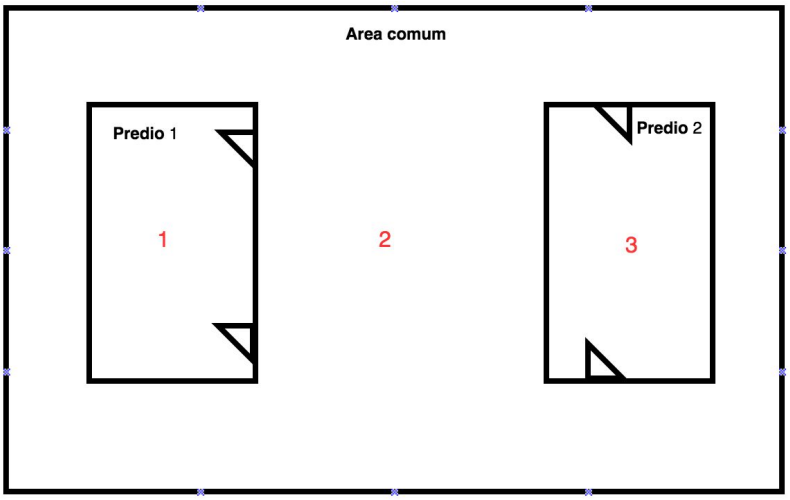

Figura 20: representação de subgrafos em uma localidade

A imagem a seguir (figura 21) é a representação do Prédio 1 da figura 20 com um maior detalhamento, o que inclui as suas limitações internas e os possíveis caminhos representados pelas linhas vermelhas. Cada andar é uma subparte de um prédio, desta forma é possível chegar na conclusão que um andar é um subgrafo de um prédio.

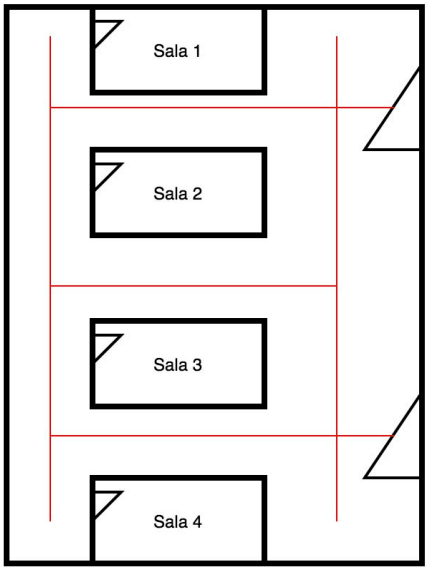

Figura 21: *PathPoints* desenhados em um andar

Com o conhecimento que os pontos de origem e destino estão em prédios diferentes tem-se as seguintes afirmações:

- existe uma estrutura que o usuário precisa sair (prédio de origem),
- uma estrutura onde precisa entrar (prédio de destino), e
- talvez alguma estrutura que ele precisa passar por dentro para chegar no destino.

Para análise de caminho mínimo, todos os prédios daquela localidade são avaliados, é gerado um subgrafo com somente os nós relevantes e neste ponto o sistema deve decidir qual dos três cenários citados se encaixa para cada prédio.

Quando a estrutura é analisada e o ponto de origem se localiza nesta estrutura é calculado o caminho mínimo para as saídas do prédio. Representado na figura 22.

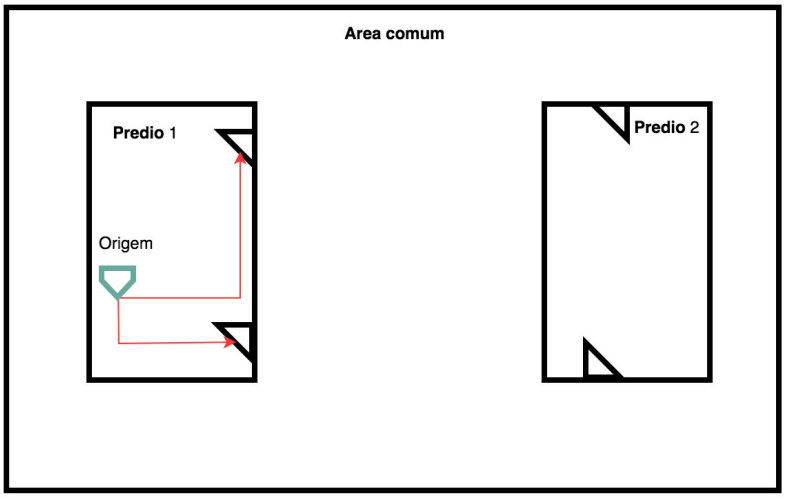

Figura 22: Menores rotas de ponto de origem para saída do Edifício

Quando o ponto de destino se encontra no prédio o cálculo é feito inversamente, o que quer dizer que é calculado os caminhos mínimos das saídas do edifício para o ponto destino. Representado na figura 23.

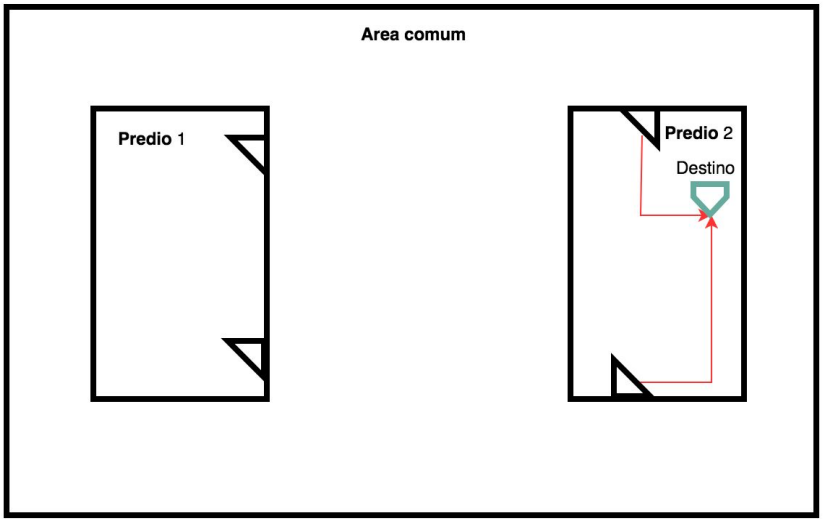

Figura 23: Menores rotas de saída do edifício para ponto destino

Em qualquer outra estrutura é feito o cálculo de todas as saídas para todas as saídas. Representado no estado na imagem 24.

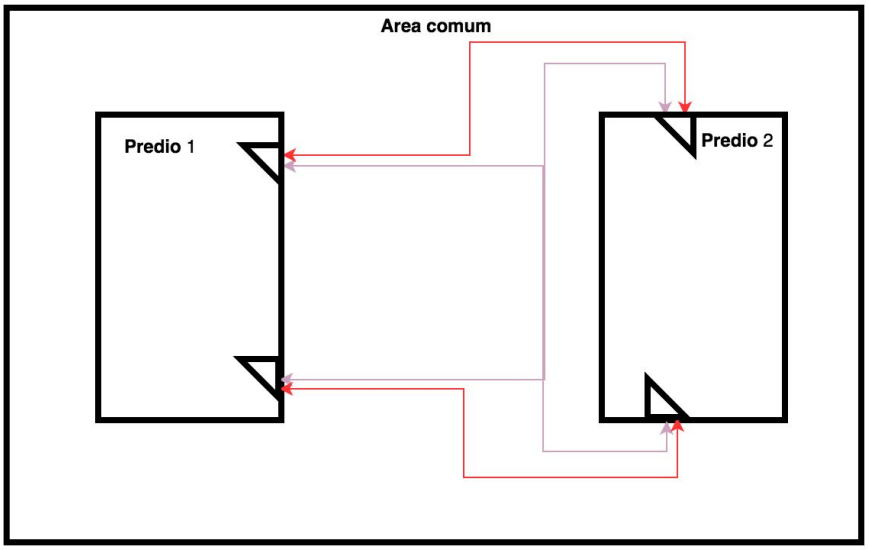

Figura 24: Menores rotas de todas as saídas de todos os edifícios

Depois de gerar os novos subgrafos todos são unidos gerando um novo subgrafo, este contendo somente o conjunto de nós relevantes para o cálculo da roda. Então é aplicado o algoritmo de Dijkstra para obter o resultado final que representa o caminho que deve ser tomado pelo usuário. Tal processo pode ser observado na imagem 25. A tarefa de subdivisão do caminho também é feita pois não é possível mostrar todas as imagens ao mesmo tempo.

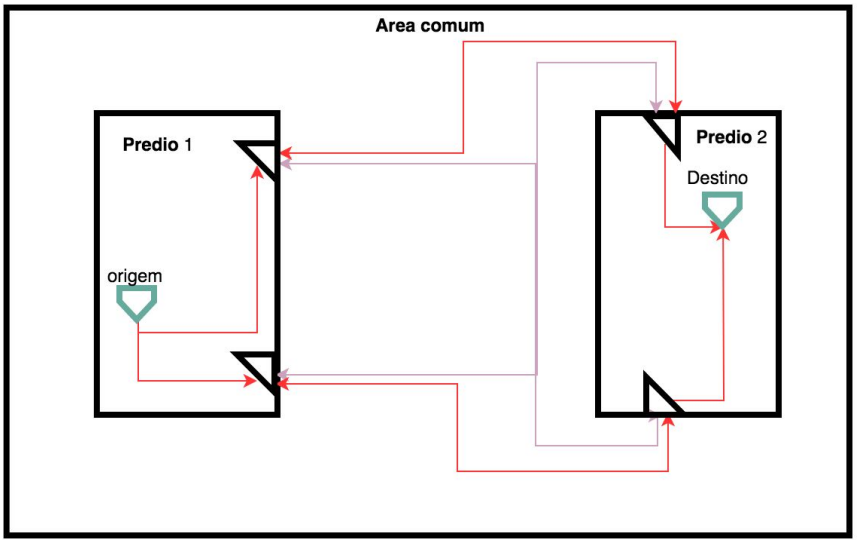

Figura 25: Subgrafo resultante do menores caminhos

O algoritmo é capaz de saber qual é o caminho mínimo dentro de um grafo pois cada aresta do grafo tem um peso. Em outras palavras, é possível armazenar a distância entre pontos. No final do processo tem-se a lista de nós que criam a rota que o usuário deve percorrer, assim possibilitando o desenho na tela do *SmartPhone*. A imagem 26 mostrar o caminho de menor custo entre o ponto Origem e Destino do exemplo.

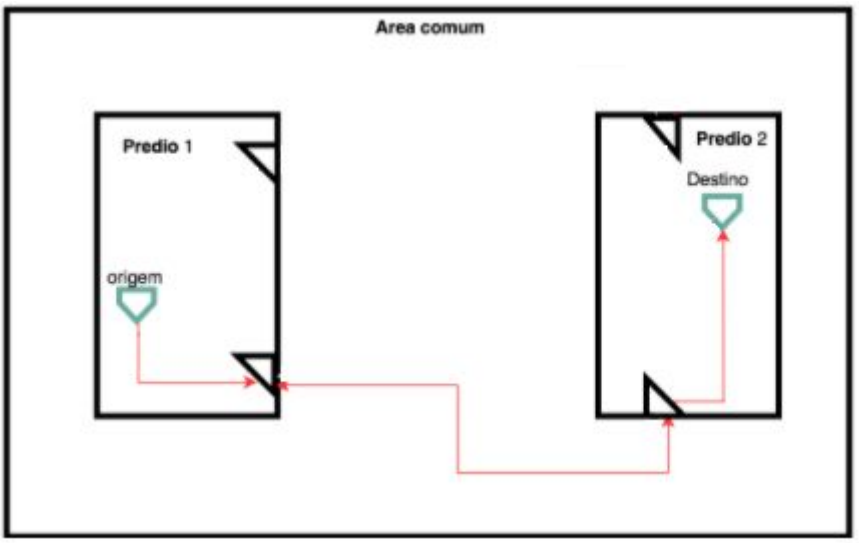

Figura 26: Caminho mínimo entre ponto de origem e destino

6.2 Pontos de Interesse

Como já mencionados anteriormente os pontos de interesse tem a finalidade de simbolizar certos espaços dentro de um mapa, como por exemplo uma sala ou um auditório. Eles são representados como um ponto dentro do grafo de uma localização, porém estes não são incluídos no cálculo de rota pois são considerados pontos iniciais ou finais de qualquer rota. A decisão de não incluí-los como um ponto para cálculo de um caminho deve-se ao fato de que se existirem menos nós no grafo, diminuem a quantidade de operações, assim melhorando a performance do programa..

Foi feita a diferenciação entre pontos de rota e pontos de interesse. Onde se dá toda responsabilidade de desenho de caminhos para os *PathPoints*, assim restando para o *InterestPoints* a função de se ligar o subgrafo e representar pontos de origem e destino. A figura 27 representa essa característica.

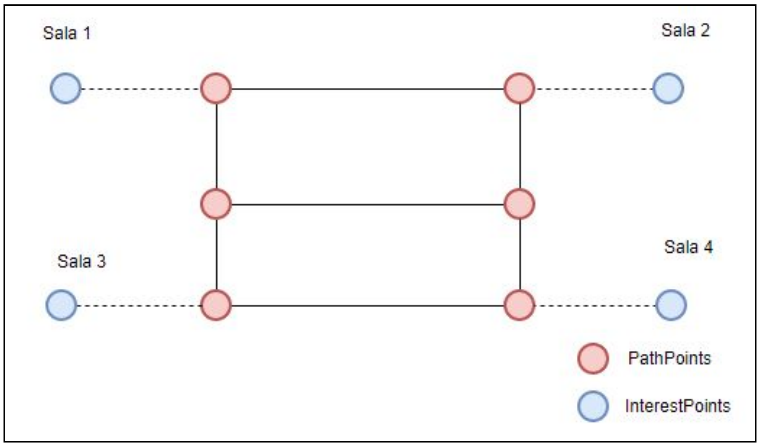

Figura 27: Representação de *PathPoints* e *InterestPoints*

O outro objetivo de diferenciar estes pontos de pontos de rotas é de possibilitar uma fácil inclusão no programa, desta forma quando se deseja adicionar um ponto novo no sistema é necessário fornecer somente seus metadados para o servidor. Essa tal funcionalidade é a que possibilita a leitura e utilização do QrCode, pois é definido um novo ponto de interesse no aplicativo na leitura do QrCode e em seguida é colocado como ponto de origem, assim possibilitando um novo cálculo de rota.

7 Prova de Conceito

Com o propósito de medir se o grau de proximidade que o projeto chegou de seus objetivos, foi conduzida uma prova de conceito. O objetivo da prova de conceito não é realizar um experimento formal (TONELLA et al. 2007), mas sim evidenciar a produção de conhecimento feita em algum processo, demonstrado como a utilização da aplicação e verificar pode a sua colaboração

A prova de conceito foi conduzida no dia 29/06/2017 no Campus da Unirio numero 458 e envolveu um grupo de cinco pessoas, com faixa etária entre 22 e 24 anos, com conhecimento no uso de dispositivos móveis e internet. Para a realização da prova, foi disponibilizado um aparelho de *Smartphone* com a aplicação do SUL instalada. Para a condução da prova de conceito teste foi criada dada a instrução que o usuário deveria escolher livremente uma rota dentro do campus da faculdade. Os passos esperados pelos os usuários ao utilizar a aplicação são: a inicialização do app, a escolha do ponto inicial e final e a chegada do usuário ao seu destino com sucesso.

Os participantes do teste tiveram uma breve explicação do objetivo da aplicação, pois o teste também tem como intuito captar o entendimento do propósito do aplicativo, pois este pode ser distribuído pelo Google Play onde as informações da aplicação são limitadas. Junto com a explicação foi disponibilizado um QrCode válido com uma posição definida para o teste, assim os sujeitos do testes poderiam utilizar as duas formas de definição do ponto de origem, via QrCode ou input manual.

Como o teste deixava livre a escolha de qual destino o usuário deveria ir os tempos de utilização foi variado, porém os envolvidos no teste utilizaram o aplicativo por cerca de 5 minutos, sendo que todos chegaram no seu destino com sucesso. Depois do teste foi disponibilizado dois arquivos o anexo 1 que é um resumo das funções do aplicativo para o caso teste e do anexo 2 que é o formulário respondido pelos sujeitos do teste, o formulário tem como objetivo medir o grau de dificuldade de entendimento e utilização do aplicativo.

A tabela a seguir apresenta os resultados obtidos com a prova de conceito e representa os dados inseridos no formulário do anexo 2 pelos sujeitos do teste.

	Tipo Ponto Origem	**Ponto de Origem**	**Ponto de Destino**	**Informações Necessárias**
Usuário 1	QrCode	DA CCET	Sala 204N	Sim

Usuário 2	Input Manual	DA CCET	Sala 204N	Sim
Usuário 3	Input Manual	Xerox CCET	Sala 504N	Sim
Usuário 4	Input Manual	Direção E.P.	Sala Prof. Mat	Sim
Usuário 5	QrCode	DA CCET	Sala 204N	Sim

Tabela 2 Prova de conceito I

	Chegou no Destino	**Avaliação**
Usuário 1	Sim	Satisfatório
Usuário 2	Sim	Satisfatório
Usuário 3	Sim	Satisfatório
Usuário 4	Sim	Satisfatório
Usuário 5	Sim	Satisfatório

Tabela 3: Prova de conceito II

8 Conclusão

8.1 Considerações Finais

Este projeto propôs elaborar um sistema que fosse capaz de fornecer a usuários de *Smartphone*, uma aplicação capaz de gerar rotas de um ponto ao outro dentro de uma localidade. Para que fosse fornecido um produto que atendesse os requisitos necessários para cumprir o objetivo do projeto, foi feita uma pesquisa com produtos e empresas que fornecem soluções similares, para que as funcionalidades essenciais do aplicativo fossem levantadas.

A abordagem feita pela equipe do SUL para desenvolver a estrutura do algoritmo de caminho mínimo foi fundamental para o resultado obtido. Mesmo com a utilização do pacote Node-Dijkstra, que forneceu o algoritmo de Dijkstra para JavaScript, foi necessário o desenvolvimento de uma solução visando diminuir o tempo de cálculo do algoritmo.

Com a construção da aplicação foi possível observar que este projeto pode auxiliar de forma expressiva no deslocamento de pessoas, mostrando que o aplicativo atende os requisitos iniciais, fornecendo uma nova forma de gerenciar a tomada de decisão quando se é tratado o assunto de geolocalização.

A integração entre diferentes ecossistemas traz uma complexidade para projetos, entretanto apesar dos obstáculos tecnológicos o ganho de produtividade no desenvolvimento é considerável, mesmo com a utilização de dezenas de tecnologias e plataformas.

O mercado de tecnologia está aberta para novos produtos. Uma aplicação confiável para ser aplicada em qualquer tipo de estrutura e que cumpre seu objetivo, serve de subsídio para acreditar que o SUL tenha a oportunidade se inserir no mercado amplo como o de celulares inteligentes.

8.2 Trabalhos Futuros

Entre os trabalhos futuros indicados nesta seção pode-se observar novas ferramentas, novos sistemas operacionais e testes e eficiência.

Um dos objetivos deste projeto é propor uma solução eficiente, porém não foi feito um estudo detalhado sobre o verdadeiro impacto das alterações na forma que o algoritmo foi implementado. Um teste válido seria propor uma análise para ver o resultado de qual o ganho real quando se calcula a rota mínima quando é utilizado todos os pontos de um mapa e a forma desenvolvida no SUL, que seria com a subdivisão em subgrafos.

O projeto propôs desenvolver um aplicativo na plataforma Android, porém seria possível, com uma certa facilidade, migra-lo para a plataforma

IOS, pois o React-Native que permite o funcionamento do mesmo código em ambas as plataformas. Para a utilização do aplicativo no sistema operacional da Apple alguns protocolos devem ser seguidos, como por exemplo o de permissão de uso de ferramentas e configurações como a câmera e o 3G.

Outra ferramenta que pode ser desenvolvida é permitir que o usuário faça uma busca por rota personalizada, pode-se imaginar uma busca de rota que seja acessível para um cadeirante. Para este procedimento deve ser feito uma alteração nos pontos de caminho adicionando um campo que define se a transição entre os pontos é acessível ou não, sendo assim possível o cálculo de uma rota mínima acessível para um cadeirante.

Referências Bibliográficas

Capelas, Bruno. Brasil chega a 168 milhões de smartphones em uso. *O Estado de S. Paulo,* São Paulo, 15 abri. 2016

Harris,Tom. Como funciona compressão? *How stuff works?* São Paulo. 15 jun. 2009

Martins, Elaine.Como funciona o GPS. *TEC MUNDO. São Paulo. 10 Ago. 2009*

Brown, Peter. History of QrCode. QrCode.com. 21 Set 2008

Bassi,Silva. O planeta já tem 2 bilhões de de pessoas utilizando redes sociais. *IDGNOW. Sao Paulo. 06 marc. 2015*

Favretto. N,Renata. 2012. Estudo avalia fatores que influenciam a instalação de aplicativos no celular.

TONELLA, P., TORCHIANO, M., DU BOIS, B., SYSTA, T., Empirical studies in reverse engineering: state of the art and future trends, Empirical Softw. Engg, Springer- Link, v. 12, n. 5, p. 551-571, mar. 2007,

Thomas H. Cormen, Charles Leiserson, Ronald L. Rivest, Clifford Stein: *Algorithmen – Eine Einführung.* 2. Auflage. Oldenbourg Wissenschaftsverlag, München 2007.

RUSSEL, S. and NORVING, P. Artificial Intelligence. A Modern Approach. Prentice Hall, 1995.

Anexo 1

A lógica do aplicativo

Uma das maneiras para a inicialização do aplicativo é a ação da primeira leitura de um QRCode no local desejado, esses dados carregarão o mapa específico do local e a posição deste QRCode no referente mapa. Este, já lido, será o ponto inicial para determinar a projeção da rota. Entretanto, é possível também ao usuário obter essas informações pelo sistema do aplicativo sem ser preciso a leitura do QRCode presencialmente. Selecionar qualquer QRCode como ponto inicial para a criação da rota pelo aplicativo também é viável.

Disposição dos QRCode

A distribuição dos QRCode será feita em pontos específicos espalhados pela dependências do local onde o aplicativo for implementado. Como por exemplo: entrada da do estabelecimento, espalhados em cada prédio (no final de cada escada/elevador, implantados em pontos estratégicos em cada andar).

Para isso, será feito um estudo com o objetivo de determinar os locais padrões e o design (cor, tamanho e formato) dos adesivos QRCode para que os usuários os encontrem mais facilmente.

A seguir é mostrado a estruturação e detalhamento da hierarquia do campus 458 da Unirio.

Campus

Sendo este o nível mais externo, ele mostrará o mapa mais abrangente de todos. Nele podemos visualizar as estruturas do perímetro, como: os

centros, estacionamento e refeitório. Ele será o primeiro mapa a aparecer no aplicativo se o usuário utilizar um QRCode de fora de um dos centros. Porém, também é possível ser visualizado se um QRCode de dentro de um centro for ativado.

Centros

Este fornece a foto das fachadas dos prédios, número de andares e nome dos centros (siglas e nomes por extenso). Neste nível será perguntado ao usuário se este é realmente o centro desejado para ser visualizado. Se sim, dados sobre ele serão carregadas como mapas e informações úteis.

Andares

Neste nível, cada andar tem seu mapa com suas características , como salas de aula e de professores, secretarias, bebedouros e banheiros. Ao selecionar um desses itens, como por exemplo a sala dos professores, será oferecido, se ouver, informações relevantes, como por exemplo, quais são os professores pertencentes a tal sala e suas respectivas atividades.

Movimentação interna

Um exemplo de movimentação seria: usuário quer se deslocar de um andar a outro dentro do mesmo prédio. Assim foi desenvolvido um sistema de páginas para resolver o problema de visualização para o usuário e conexão entre andares. Neste sistema, uma página corresponde ao mapa de um andar. O usuário pode fazer a leitura de sua trajetória do andar seguinte ou anterior somente mudando de página, deslizando o dedo sobre a tela do dispositivo móvel (sistema equivalente ao tablet). Assim que o usuário traçar uma rota, ele visualiza a trajetória por um traço colorido dentro do próprio andar em que se localiza até o próximo andar. Para conectar a trajetória entre andares o aplicativo traça uma rota entre, por exemplo, a localização do QRCode lido e

de ponto de chegada. No andar de destino, o usuário visualiza novamente a trajetória dentro do próprio andar até chegar no ponto de destino. Outra maneira de criação de rota é a de posicionamento inicial em algum ponto fixo , como uma sala ou auditório para um QRCode ou para outro ponto. Uma rota fixa que também é oferecida é a do último ponto referenciado como objetivo para a saída do prédio. Esta estratégia foi elaborada devido pela quantidade limitada de QRCodes por andar e por ser desvantajoso ao usuário sempre ter de procurar um QRCode para iniciar sua busca.

(

Anexo 2

1. Para a busca de seu ponto de origem, Você utilizou qual funcionalidade?

   ()Input Manual () Utilização de QrCode

2. Qual foi seu ponto de origem ?

   _____

3. Qual foi seu ponto de destino?

   _____

4. O aplicativo forneceu as informações desejadas?

   _____

5. Foi possível chegar no seu ponto de destino?

   _____

6. Como você considera a precisão da localização geográfica identificada pelo aplicativo?

   () Satisfatória ()Aproximada ()Insatisfatória

7. Comente a experiência de utilização do aplicativo. Sugira melhorias/critique, se for o caso.

   _____

   _____

   _____