Protótipo de aplicativo para georreferenciamento em interiores de edificações para portadores de deficiência visual

Fabrício Pedroso Nunes [1]
Orientador: Victor Billy da Silva [2]

[1]Ciência da Computação – Universidade de Passo Fundo (UPF) – Campus 1
Av. Brasil Leste, 285 - São José, Passo Fundo - RS, 99052-900

139834@upf.br[1]

victorbilly@upf.br[2]

Abstract. *Blind people encounter challenges in their daily lives, move from one environment to another without the help of a guide is one of them. Using mobile indoor localization technologies that uses the strength of WiFi routers signal receptions and classifiers as KNN algorithm, and classic algorithms like Dijkstra to create indoor routes we propose to create a solution for this problem and ease people with visual impairment lives'. As result, was created a WebService and application prototype do consume it that will do the job of guiding a person from one point of a building to another point of the same building. Despite some problems and improvements that still need do be done, the results were very satisfactory, obtaining the location, route and direction that the user should follow, demonstrating that this is a viable solution.*

Resumo. *Deficientes visuais encontram diversos desafios no seu dia-a-dia, um desses é conseguir locomover-se de um ambiente para outro de forma autônoma, sem o auxílio de um guia. Utilizando tecnologias como a de localização de dispositivos móveis em ambientes internos, através de potência de recepção dos sinais de roteadores WiFi e classificadores como algoritmo KNN, além de algoritmos para traçar rotas, propomos criar uma alternativa para esse problema buscando dar maior autonomia a pessoas com deficiência visual. Como resultado, foi criado um WebService e um protótipo de aplicativo para consumi-lo, gerando assim uma aplicação que é capaz de guiar uma pessoa de um ponto de um prédio até outro ponto especificado do mesmo prédio. Apesar de alguns problemas e melhorias que precisam ser ajustadas, os resultados obtidos foram satisfatórios, conseguindo obter a localização, a rota e a direção que o usuário deve seguir, demonstrando que tal solução é viável.*

1. Introdução

Estima-se que no Brasil, segundo o IBGE (2010)[1], mais de 6 milhões de pessoas declararam ter grande dificuldade permanente de enxergar. Destes, cerca de 530 mil declaram-se completamente cegos. Ressalta-se que apesar da deficiência, boa parte dessas pessoas executam atividades diárias como trabalhar, estudar e praticar esportes, embora muitas delas ainda relatam ter grande dificuldades no quesito mobilidade nos centros urbanos.

[1]Último censo que consta essa informação

Uma pessoa cega utiliza-se da bengala e do piso tátil como auxílio para se locomover. Contudo caso ela não conheça a direção que deve seguir, acaba por ficar dependente de alguma outra pessoa vidente, como um guia que à oriente para que lado deve seguir até chegar ao seu destino.

Atualmente existem as tecnologias assistivas que são recursos e serviços que contribuem para proporcionar ou ampliar as habilidades funcionais de pessoas com alguma deficiência, e, tais ferramentas, têm promovido uma maior autonomia e inclusão destas. Para pessoas que utilizam aparelhos eletrônicos, como por exemplo smartphones, destaca-se aplicativos de acessibilidade como o TalkBack do Android, o qual possibilita que um deficiente visual consiga fazer uso de um smartphone de forma autônoma (BERSCH, 2017).

Dentro da Universidade de Passo Fundo existe o Setor de Atenção ao Estudante (SAES) que auxilia os alunos com deficiência visual, esse setor busca contribuir para a inclusão dos alunos auxiliando com acessibilidade e permanência do acadêmico na vida universitária (SAES, 2019).

A universidade conta com piso tátil e placas em Braile para identificar cada uma das salas em todos os prédios. No entanto, os alunos com deficiência ainda encontram dificuldades para se dirigir de um prédio ao outro e principalmente entre salas, necessitando geralmente de um terceiro que os guie até o seu destino. Na instituição, o SAES disponibiliza uma profissional que serve de guia para as pessoas com essa deficiência, porém, esse serviço necessita de pessoas o que pode acabar gerando um alto custo para a instituição.

Analisando esta situação, neste artigo, aborda-se a elaboração de uma arquitetura de aplicação que visa auxiliar usuários cegos ou deficientes visuais a se orientar dentro de edificações, utilizando como piloto o prédio B5 do campus 1 da Universidade de Passo Fundo. Tal aplicação baseia-se em estudos de critérios de acessibilidade, tecnologias e teorias necessárias para o desenvolvimento desse projeto e será descrito nas seções seguintes.

2. Pesquisa Bibliográfica

Nos últimos anos têm-se notado o aumento em massa do uso de dispositivos móveis no Brasil, entre esses dispositivos o smartphone é o principal. Para as pessoas cegas esse dado não é diferente, ainda que, para estas, têm aumentado a quantidade de tecnologias assistivas e políticas de inclusão social (MONTEIRO, 2011).

Nos smartphones uma das principais tecnologias para a localização ou posicionamento, são os receptores GPS (*Global Positioning System*), que está disponível na maior parte destes dispositivos. Porém, o GPS tem baixa performance em ambientes internos, causado pela falta de sinal diretamente com os satélites, gerando assim erros de localização (MOURÃO e OLIVEIRA, 2013).

Segundo BAHL e PADMANABHAN (2000), um dos primeiros sistemas de localização, o RADAR, utiliza frequências de sinais chamadas RSSI para obter a localização. Para este modelo, várias leituras de RSSI são realizadas em diversos pontos do local em que se quer realizar a localização. As leituras contém os valores de potência de sinal recebido dos diversos roteadores. Com essas informações na base de dados é

utilizado um classificador para retornar a posição em que foi enviada a informação.

No trabalho de MOURÃO e OLIVEIRA (2013) foi feito um sistema de localização dentro de uma área interna utilizando a força do sinal de Access Points Wireless para, utilizando as tecnologias apresentadas no sistema de RADAR, desenvolver uma forma de obter uma localização mais precisa do que o GPS em ambientes internos. Os resultados deste trabalho, segundo os criadores, foram bastante promissores, alcançando uma precisão menor do que um metro em 96% dos casos em um ambiente interno de 7 x 7 metros com quatro roteadores WiFi.

Seguindo a ideia dos trabalhos citados acima, o objetivo deste trabalho é utilizar a força do sinal dos roteadores WiFi dentro de um prédio para realizar o mapeamento das salas e, através do treinamento da rede e com a localização do usuário, utilizar algoritmos de roteamento para guiar o usuário de um ambiente a outro dentro do prédio. Para treinamento da rede utilizamos o algoritmo KNN pois demonstrou uma boa eficiência em trabalhos passados.

Devido às diferenças de programação entre dispositivos Android e IOs, foi optado por criar um WebService com o qual qualquer dispositivo possa se comunicar, e, passando os dados corretos, conseguir obter sua posição e rota.

Nas seções a seguir vamos detalhar o trabalho que foi feito no servidor na parte da interface de cadastro dos dados e na API, o que foi feito no protótipo do aplicativo e como foi feito o mapeamento das salas, corredores e pontos de interesse dentro do prédio B5 do Campus I da Universidade de Passo Fundo que foi o local piloto para estudo.

3. Aplicação Desenvolvida

A partir das tecnologias e fontes buscadas, desenvolveu-se uma aplicação em uma arquitetura do tipo *Cliente X Servidor*, sendo utilizado um serviço WEB (WebService) no lado do servidor, o qual irá comunicar-se, neste caso, com um cliente que será um aplicativo, considerando as recomendações necessárias para tal ser acessível a pessoas com deficiência visual, como auxílio na orientação de cegos em ambientes internos.

Neste sentido, o usuário irá informar através de uma lista de destinos disponíveis para a localização atual, qual sala deseja visitar e a aplicação buscará a sua posição dentro do prédio e, a partir dessa posição e do destino informado, irá calcular uma rota e guiá-la até seu objetivo. A seguir vamos detalhar a arquitetura utilizada para desenvolvimento desta aplicação.

3.1. Arquitetura

A arquitetura criada para este projeto inclui tecnologias de georreferenciamento, acessibilidade, além de tecnologias de redes sem fio utilizadas para obter melhor precisão em ambientes internos em relação ao GPS tradicional.

Para facilitar o reaproveitamento do processo em diversos sistemas operacionais de dispositivos móveis foi desenvolvido um WebService (WS) com o qual o aplicativo deve se comunicar. Esse WS foi feito em estrutura REST que utiliza o protocolo HTTP na comunicação e troca de mensagens, as quais serão enviadas em formato JSON (*JavaScript Object Notation*) que é um formato leve para intercâmbio de dados (MOTA e FERREIRA, 2014).

O protótipo de aplicativo, que irá consumir tais serviços, vai compreender somente a estrutura para guiar o deficiente dentro das dependências do prédio em questão, neste trabalho será utilizado como piloto o prédio B5 do Campus 1 da Universidade de Passo Fundo, devido a necessidade de posicionamento e armazenamento de informações sobre localização das dependência. Contudo, a arquitetura é genérica e pode ser ajustada para os demais edificações, conforme será explanado na Seção 3.3.

A estrutura geral da aplicação desenvolvida pode ser dividida em três partes: (1) O Servidor WebService REST que é uma API com a qual o protótipo de aplicativo irá se comunicar para obter a localização e rota que o usuário deve seguir; (2) o protótipo de aplicativo que irá guiar o usuário; e, (3) o mapeamento interno que consiste nos dados que o WebService deve consumir para poder obter a localização do usuário e guiá-lo.

Destaca-se que utilizou-se para a implementação do servidor a linguagem PHP por ser livre e possuir uma grande quantidade de bibliotecas gratuitas e livres para utilização. Para o WebService, foi escolhido o tipo REST pois segundo PLANSKY (2014) este estilo de WS simples e performático, não enviando uma grande quantidade desnecessária de dados como no SOAP.

3.2. Protótipo de Aplicativo

Para desenvolvimento do protótipo do aplicativo foi utilizado o framework React Native. Esse Framework foi desenvolvido pelos engenheiros do Facebook e, segundo CABRAL (2016) consiste por uma série de ferramentas que viabilizam a criação de aplicações móveis nativas, tanto para o sistema operacional iOS quanto para Android.

É importante ressaltar que esse framework utiliza a linguagem JavaScript que é livre, amplamente utilizada e documentada. O React Native utiliza uma ferramenta chamada Expo, que permite que o usuário rode um servidor local na sua máquina e acesse o aplicativo de seu celular ou emulador a partir da URL deste servidor, o que facilita o processo de testes (PROBST, 2017).

Para o desenvolvimento e testes foi utilizado o sistema operacional Android 9, e utilizado as bibliotecas do REACT para comunicação com o WebService acessar as informações de rede e bússola do smartphone.

O protótipo de aplicativo foi desenvolvido utilizando as normas de recomendação da W3C para pessoas com baixa visão. Para as pessoas totalmente cegas o recurso de acessibilidade mobile foi utilizado o TalkBack que segundo SANTOS (2013) é uma ferramenta nativa do Android e que já é utilizada de forma abrangente por deficientes visuais.

Destaca-se que os testes foram realizados para Android, porém com a adoção de um framework como o React Native, o aplicativo pode ser facilmente ajustado, com pequenas modificações, para rodar em IOS.

3.3. Mapeamento Interno

Para que seja possível integrar os algoritmos de busca e localização utilizados neste trabalho, fez-se necessário a utilização de uma estrutura para mapeamento de salas. Esse mapeamento é o conjunto de dados armazenados que identificam onde fica cada sala, corredor e escadaria dentro do prédio e as ligações entre esses espaços.

Este mapeamento pode ser modelado como um Grafo, no qual as salas, pontos nos corredores e escadarias serão representados por nodos, enquanto as informações de ligação entre esses nodos formam as arestas. Tal grafo chamaremos de **malha interna**.

Para gerar essa malha as seguintes tecnologias foram utilizadas:

- **Access Points Wireless**: Para a utilização da aplicação em ambientes internos é necessário ter vários Access Points (AP) no prédio. Esses APs são utilizados para fazer a triangulação da posição do usuário em ambientes fechados. Esses Access Points não necessariamente precisam estar mapeados, nem a sua posição geográfica pois a aplicação vai usar somente uma posição relativa e os dados de nodos da malha interna para conseguir localizar a posição do usuário.
- **WiFi Analyzer**: Está é uma ferramenta nativa do Android que será utilizada para obter o endereço MAC e o nível de sinal de todos os Access Points ao redor do aplicativo, para posteriormente enviar ao servidor para o cálculo da posição do usuário.
- **Bússola**: A bússola será utilizada para saber em quantos graus o usuário precisará estar direcionado para conseguir mudar de um nodo para o outro.

Cada nodo da malha interna deverá possuir as informações dos Access Points que estão nas proximidades e sua respectiva força de sinal, para as arestas basta saber quais os dois nodos que está ligando e quantos graus os usuários devem estar direcionados para conseguir passar de um nodo para o outro.

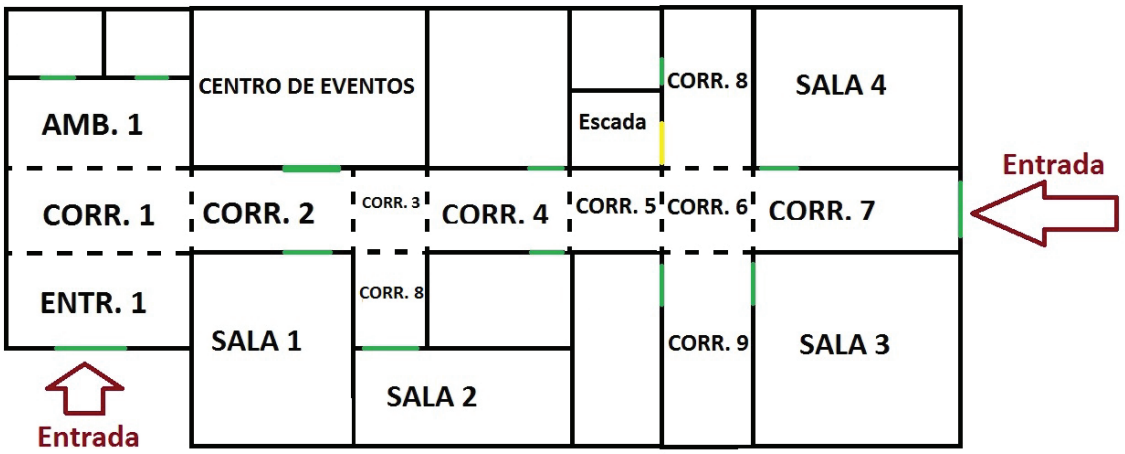


Figura 1. Exemplo do mapeamento interno

Na Figura 1 temos o exemplo do mapeamento interno do prédio B5. Cada ambiente está mapeado com a força de sinal em relação à 4 Access Points que possuem a maior força de sinal em relação à sua posição. Quando o usuário envia os dados com os endereços MACs dos Access Points e a força de sinal que tem em relação a eles, o servidor analisa esses dados para verificar quão próximo ele estaria de cada um dos ambientes e retorna a posição mais provável dele dentro desse prédio.

Digamos que o algoritmo fez o cálculo e o usuário realmente está na Entr. 1 e quer ir para a Sala 2. Neste momento entram as informações das Arestas, seguindo o exemplo da Figura 1, a aresta que liga a Entr. 1 com o Corr. 1 deve conter as informações de

origem (Entr. 1), destino (Corr. 1) e a direção seria em graus para trocar de nodo. Desta forma o algoritmo de direcionamento sabe para qual lado o usuário deve estar direcionado para mudar do nodo Entr. 1 para Corr 1 e desta forma ir seguindo até chegar à Sala 2 que seria seu objetivo.

Para a malha interna é importante ressaltar que a entrada de algumas salas é muito próxima a outras, fazendo com que o aplicativo possa perder-se em certos pontos. A princípio não foi localizada uma solução computacional para resolver esse problema, mas cabe destacar que o aplicativo ainda não substitui a utilização das placas em Braile na entrada das salas e nem o piso tátil que são duas tecnologias de acessibilidade já utilizadas, e que resolveria tal situação.

Outra questão, da malha interna, é que ela deve informar o ponto onde se inicia uma escadaria, para que o aplicativo saiba informar ao usuário que ali há uma escada e que ele deve ter cuidado ao se locomover por este local.

4. Implementação

Nesta seção será descrito a implementação do servidor e protótipo de aplicativo. O servidor foi implementado na linguagem PHP utilizando o ambiente de desenvolvimento PHPStorm (na versão de estudante). Foi criado uma interface web administrativa, onde pode-se informar os dados de localização dos nodos e arestas que o WebService REST utilizará para a comunicação com o aplicativo.

4.1. API WebService

O WebService se baseia em três algoritmos para seu funcionamento: (1) o algoritmo de localização, que calcula a posição do usuário do aplicativo a partir da força dos sinais WiFi; (2) o algoritmo de roteirização, que se baseia na ligação entre os nodos e arestas para obter o caminho mais curto até o destino desejado; e, (3) o algoritmo de direcionamento, que serve para calcular a direção que o usuário deve seguir. Tais algoritmos serão detalhados a seguir.

4.1.1. Algoritmo de Localização

O algoritmo de localização é responsável por obter a localização do usuário dentro do prédio, utilizando os dados de nodos da malha interna recebidos (previamente cadastrados pelo administrador), bem como os dados de força de sinal, e os identificadores dos Access Points ao redor do usuário. A partir desses sinais este algoritmo, utiliza-se do algoritmo de *machine learning* chamado K-Nearest-Neighbor (KNN) para obter a localização relativa do usuário, dentro do mapeamento gerado pela malha.

O K-Nearest-Neighbor, segundo MOURÃO e OLIVEIRA (2013) é muito utilizado para o objetivo de classificar os sinais e estimar a posição do usuário em ambientes internos. Para funcionar, o KNN deve ser treinado com os dados dos nodos obtidos da rede e que estão armazenados na malha interna, após feito esse treinamento o algoritmo executa um cálculo matemático para medir a distância entre os dados (nodos) e realizar a sua classificação. Depois de classificado o KNN está pronto para receber um dado não classificado (dados da identificação dos access points e a força de sinal desses dispositivos em relação ao smartphone do usuário) e a partir do cálculo definir o nodo que ele está

mais próximo, esse nodo, dentro da malha interna, vai ser a posição do usuário em um ambiente (JOSÉ, 2018).

Algoritmo 1 Algoritmo de localização

```
1: função OBTERLOCALIZACAO(arrPontos)
2:     arrNodos ← Banco :: obterNodos()
3:     qtNodos ← quantidade(arrNodos)
4:     para i ← 0 até qtNodos faça
5:         KNN::adicionarAmostra(arrNodos[i])
6:     fim para
7:     devolve KNN::localizar(arrPontos)
8: fim função
```

No Algoritmo 1, os nodos são obtidos da base de dados com o método *Banco::obterNodos*, depois disso são adicionados como amostras ao KNN que, com esses dados, consegue a posição mais provável do usuário pelo método *KNN::localizar*.

4.1.2. Algoritmo de Roteirização

O algoritmo de roteirização precisa obter a rota entre o ponto em que o usuário está, que foi retornada pelo Algoritmo de Localização, para o ponto em que ele deseja ir. Para isso é utilizado a malha interna que deve possuir os nodos e arestas que mapeiam o ambiente. Então, como a malha interna é um Grafo, para gerar a rota foi utilizado o algoritmo de Dijkstra que consegue calcular a menor rota dentro de um Grafo (SANTOS e RANGEL, 2010).

Algoritmo 2 Algoritmo de Roteirização

```
1: função OBTERROTA(origem, destino)
2:     arrArestas ← Banco :: obterArestas()
3:     qtArestas ← quantidade(arrArestas)
4:     para i ← 0 até qtArestas faça
5:         Dijkstra::adicionarAresta(arrArestas[i])
6:     fim para
7:     devolve Dijkstra::obterRota(origem, destino)
8: fim função
```

No Algoritmo 2, as arestas nodos são obtidas da base de dados com o método *Banco::obterArestas*, depois disso são adicionadas à um *Grafo*. Então o *Dijkstra* calcula a rota mais curta e retorna ela pelo método *Dijkstra::obterRota(origem, destino)*.

4.1.3. Algoritmo de Direcionamento

O algoritmo de Direcionamento se baseia na direção em que o usuário está apontando, essa direção é obtida pela bússola do smartphone e repassada juntamente com as informações dos access points na requisição com o servidor. A partir da direção obtida o

algoritmo analisa quantos graus o celular precisa estar direcionado para conseguir mudar de nodo e retorna essa mensagem para o usuário.

No algoritmo de direcionamento foi utilizado como base 60 graus para identificar que o usuário está virado para frente, ou seja, para onde precisaria ir. Então se no servidor tiver a informação que o próximo nodo fica ao Norte, assumindo que o Norte ficaria a 0 graus, caso o usuário esteja virado a 270 graus (esquerda do ponto onde ele desejaria ir) o aplicativo retornará a mensagem "Vire à direita". Caso o usuário estiver entre 330 e 360 graus, ou 0 graus e 30 graus (intervalo de 60 graus) a aplicação reconhecerá que ele está virado de frente para o seu objetivo.

Algoritmo 3 Algoritmo de Direcionamento

```
1:  função OBTERDIRECAO(direcaoatual, direcaodesejada)
2:      direcao ← direcaodesejada − direcaoatual
3:      se direcao < 0 então
4:          direcao ← direcao + 360
5:      senão
6:          se direcao > 360 então
7:              direcao ← direcao − 360
8:          fim se
9:      fim se
10:     se direcao >= (360 − 30) ou direcao <= (30) então
11:         devolve SIGA EM FRENTE
12:     fim se
13:     se direcao >= (180 − 30) e direcao <= (180 + 30) então
14:         devolve DE MEIA VOLTA
15:     fim se
16:     se direcao < 180 então
17:         devolve VIRE A DIREITA
18:     fim se
19:     devolve VIRE A ESQUERDA
20: fim função
```

O Algoritmo 3 calcula a direção em que o usuário deve estar virado para conseguir seguir de um nodo para outro. Esse direcionamento é obtido pelo cálculo dos graus em que ele precisa ir menos os graus que ele está direcionado. Por exemplo, se o usuário está direcionado a 90 graus e precisa estar em 270 graus, então a volta que ele precisa fazer é de 180 graus, ou seja, dar meia volta.

4.2. Protótipo da Interface Web

Para o cadastro das posições dos nodos e das arestas foi desenvolvido uma interface onde o responsável pela aplicação poderá realizar esses cadastros. Esses dados deverão ser armazenados em formato de JSON no servidor para que a aplicação do WebService consiga carregar os dados para utilização.

Basicamente serão necessários dois tipos de JSON base, os dados de Nodos que vão identificar um nodo e o sinal dos Access Points ao seu redor, e os dados de arestas que vão ligar esses nodos entre si.

Para o JSON com os dados de NODOS[2] será necessário as seguintes informações:

Nome do item	Tipo de dado	Obr.	Descrição
predio_id	String	Sim	Identificação do prédio em que se encontra o Nodo
area_id	String	Sim	Identificação do Nodo, deve ser único
sala	String	Não	Nome da sala em que se localiza um Nodo.
arr_wifi	Array<InfoWifi>	Sim	Array de dados dos Access Points que estão próximos ao aparelho. O tipo de dado necessário foi chamado de InfoWifi, a Tabela 2 explana a estrutura desse dado.

Tabela 1. Informações existentes no JSON dos NODOS

Identificação do tipo de dado **InfoWifi**:

Nome do item	Tipo de dado	Obr.	Descrição
wifi_id	String	Sim	Identificação do MAC do Access Point, deve ser único.
forca_sinal	Float	Sim	Numérico com a força do sinal do Access Point em relação ao Nodo.

Tabela 2. Estrutura do tipo de dados *InfoWifi*

Segue as informações necessárias para montar o JSON das ARESTAS[3]:

Nome do item	Tipo de dado	Obr.	Descrição
predio_id	String	Sim	Identificação do Prédio da Aresta.
de	String	Sim	Identificação do Nodo de Origem.
para	String	Sim	Identificação do Nodo de Destino.
graus_direcao	Float	Sim	Grau em que o usuário deve estar direcionado para conseguir fazer a troca de nodo.

Tabela 3. Informações existentes no JSON das ARESTAS

Dos dados que deverão ser cadastrados no servidor, as informações de prédio, identificação dos nodos e identificação das salas devem ser informados pelo usuário, os dados de força dos WiFi e graus podem ser facilmente obtidos utilizando as tecnologias de sinal Wireless e bússola existentes na maioria dos smartphones atuais.

Para a interface de entrada dos dados foi feito uma tela de login para autenticação dos usuários que vão realizar o cadastro dos nodos e arestas, e foi feito uma tela de cadastro para os nodos e uma tela para cadastro das arestas.

[2]No apêndice 1 tem um exemplo de JSON com os dados de NODOS.
[3]No apêndice 2 tem um exemplo de JSON com os dados de ARESTAS.

O conjunto de dados que for cadastrado pela Interface Web deverá formar a malha interna.

Figura 2. Tela de cadastro dos Nodos

4.3. Protótipo do Aplicativo

O protótipo de aplicativo foi desenvolvido com utilizando a linguagem de programação JavaScript e o FrameWork React Native, o aplicativo deve obter o sinal dos Access Points que estão próximos ao usuário e obter a direção da bússola em graus e enviar esses dados para o servidor. Uma vez enviado esses dados o servidor vai processá-los e retornar para o usuário a informação do próximo passo para seguir em direção ao seu objetivo.

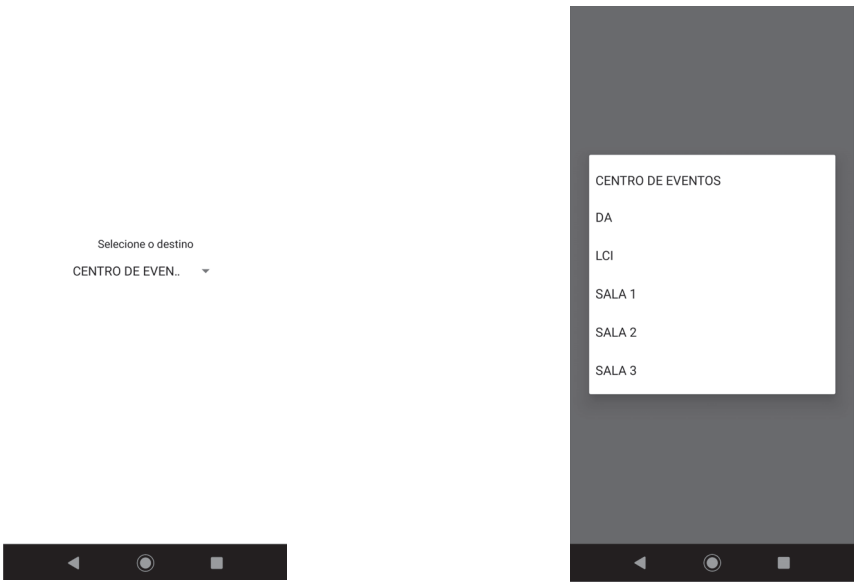


Figura 3. (A) Tela inicial do protótipo do aplicativo, onde o usuário deve informar o destino.
Figura 3. (B) Tela com a listagem dos destinos possíveis para o usuário.

Para obter o sinal dos Access Points foi utilizada a biblioteca *WifiManager*[4] que trabalha com os dados de rede do smartphone. Para obter a direção da Bússola foi utilizado as biblioteca *Magnetometer*[5], que é responsável por obter os graus da bússola e *ScreenOrientation*[6] que define a posição que o celular deve estar para permitir a orientação da bússola. As bibliotecas *Magnetometer* e *ScreenOrientation* são nativas do React Native, a biblioteca *WifiManager* é um pacote que pode ser instalado. O envio dos dados foi feito com a função fetch, que tem o objetivo de fazer comunicação com WebServices do tipo REST.

O aplicativo deve fazer uma requisição no WebService utilizando o método POST enviando os seguintes dados em um pacote JSON[7]:

Nome do item	**Tipo de dado**	**Obr.**	**Descrição**
token	String	Sim	Informação de acesso do usuário no servidor, deve ser um base64 de "usuário:senha".
direction	Float	Sim	Grau em que o usuário está direcionado, essa informação deve ser obtida da bússola do smartphone.
destination	String	Sim	Destino do usuário conforme informado por ele na primeira tela do Protótipo do Aplicativo.
position	Array<InfoWifi>	Sim	Array de informações de InfoWifi, deve ser a mesma estrutura detalhada na Tabela 2.

Tabela 4. Informações para serem enviadas como requisição ao servidor

A partir disso o servidor irá calcular a posição do usuário e a rota que ele deve seguir para chegar ao seu objetivo retornando o seguinte pacote JSON para ser usado pelo aplicativo.

Nome do item	**Tipo de dado**	**Descrição**
local	String	Local em que o usuário está.
rota	Array<String>	O array com uma listagem dos locais em que o usuário deve passar para chegar ao seu objetivo.
direcao	String	A direção que o usuário deve seguir em seu próximo passo.

Tabela 5. Estrutura de retorno do servidor

[4]"react-native-android-wifi-manager - npm."8 out. 2018, https://www.npmjs.com/package/react-native-android-wifi-manager. Acessado em 19 jun. 2019.

[5]"Magnetometer - Expo Documentation."https://docs.expo.io/versions/latest/sdk/magnetometer. Acessado em 19 jun. 2019.

[6]"ScreenOrientation - Expo Documentation."https://docs.expo.io/versions/latest/sdk/screen-orientation. Acessado em 19 jun. 2019.

[7]No apêndice 3 tem um exemplo do JSON com as informações que devem ser enviadas para o servidor.

Com o retorno desses dados, o protótipo do aplicativo utiliza a informação que retorna no campo “direcao” e mostrar na tela para o usuário. Essa informação é o próximo passo que deve ser seguido para direcionar a pessoa ao seu objetivo. O retorno desse campo é, por exemplo, “Vire à direita” ou “Vire à esquerda”.

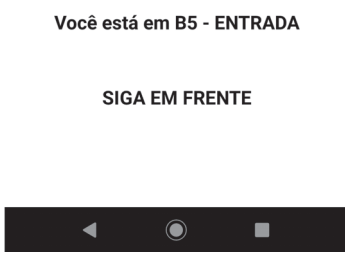


Figura 4. Tela do sistema que guia o usuário

O protótipo foi feito com um layout simples, apenas para validar que o servidor está funcionando corretamente. Foi utilizado fonte grande, como é a recomendação para aplicativos com acessibilidade e, para o usuário cego, esse protótipo está pronto para funcionar juntamente com o TalkBack no Android. Com o TalkBack o usuário cego consegue fazer com que o que estiver na tela do aplicativo seja lido para ele, e com isso utilizá-lo de forma eficiente.

5. Resultados Obtidos e Conclusão

Através de testes realizados no prédio B5 da Universidade de Passo Fundo onde um usuário definia um ponto de destino a partir de qualquer ponto mapeado dentro do prédio e seguia as instruções da aplicação até chegar ao seu destino. Conseguimos verificar que o algoritmo KNN utilizando a força de sinais de Access Points Wireless demonstrou um resultado satisfatório no processo de obter a localização do usuário desde que os dados da malha interna estejam corretamente cadastrados e com várias amostras. O algoritmo de Dijkstra serviu perfeitamente para o processo de geração da rota para o usuário.

Foi observado que o SmartPhone tem uma certa inconstância com a ferramenta da Bússola, não é possível precisar corretamente os graus em que o usuário está direcionado pois esse valor fica variando. Como solução para isso seria necessário implementar no aplicativo um sistema de calibragem e correção da bússola similar ao utilizado pelo aplicativo Google Maps.

Analisando a execução do algoritmo de localização, foi verificado que o servidor precisa calcular toda a malha interna com o algoritmo KNN toda vez que é enviado uma requisição para o WebService, o que poderia ocasionar problemas de lentidão no processamento. Vale ressaltar que este problema não foi percebido nos testes com o protótipo.

A forma de cadastro da malha interna não é eficiente. A tela de cadastro serve ao seu propósito, porém é muito trabalhoso para o administrador cadastrar ponto por ponto dos nodos e das arestas.

O WebService serviu bem ao seu propósito, com ele é possível que aplicações criadas em qualquer sistema operacional consigam consumir seus serviços e utilizar o Guia. A arquitetura em si está bem estruturada e organizada, acreditamos que pode ser

facilmente implementada em outros prédios da universidade simplesmente realizando o cadastro da malha interna desses prédios.

Apesar do problema da bússola e das melhorias necessárias para as próximas versões do aplicativo e das interfaces Web, concluímos que os algoritmos funcionam bem e serviram ao seu propósito.

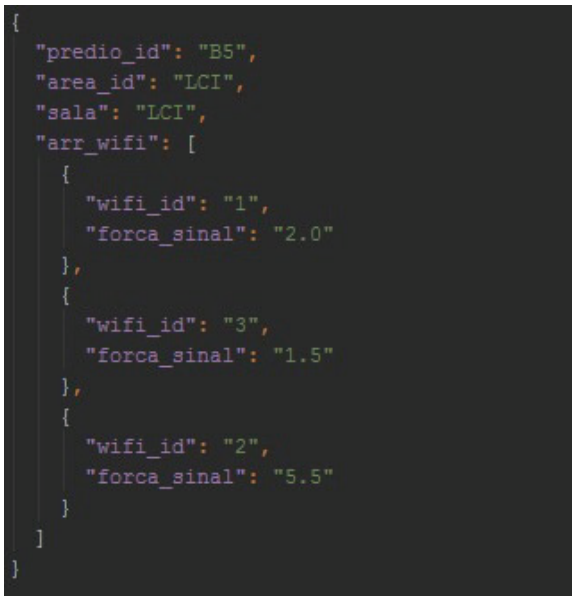
6. Trabalhos Futuros

Para trabalhos futuros, sugere-se corrigir o problema do cálculo da malha interna, para que não seja necessário calcular a malha toda vez que é feita a requisição para o servidor. Para isso, a malha poderia ser armazenada calculada, assim, toda vez que houvesse a requisição, não precisaria recalculá-la, evitando assim, possíveis problemas no desempenho.

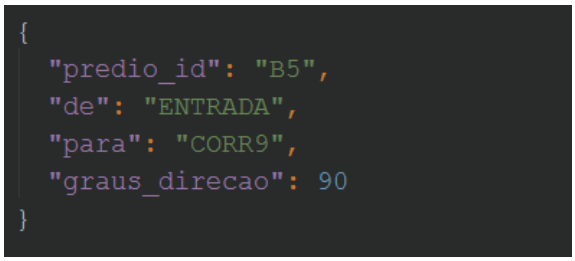
Como a forma de cadastro dos nodos e das arestas não é eficiente, foi planejado ajustar para ser feito pelo aplicativo, em um modo administrador. Assim o aplicativo poderia obter todos os sinais para os nodos e cadastrar automaticamente no servidor, facilitando o serviço do administrador.

Apêndice

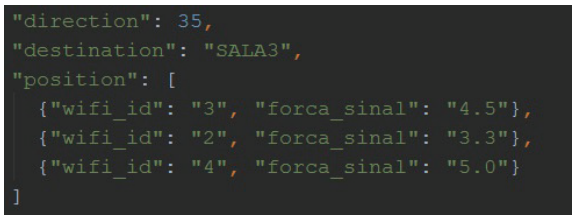
1. Exemplo de JSON dos NODOS



2. Exemplo de JSON das ARESTAS



3. Exemplo de requisição



Referências

BERSCH, Rita. INTRODUÇÃO À TECNOLOGIA ASSISTIVA. 2017. Disponível em: <http://www.assistiva.com.br/Introducao_Tecnologia_Assistiva.pdf>. Acesso em: 21 maio 2018.

MONTEIRO, Janete Lopes. OS DESAFIOS DOS CEGOS NOS ESPAÇOS SOCIAS: UM OLHAR SOBRE A ACESSIBILIDADE. 2011. Disponível em: <http://www.ucs.br/etc/conferencias/index.php/anpedsul/9anpedsul/paper/viewFile/1081/649>. Acesso em: 25 maio 2019.

MOURÃO, Helmer A. S.; OLIVEIRA, Horácio A. B. F. de. SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS – SBRC 2013, 31., 2013, Brasília. Localização de Dispositivos Móveis em Redes WiFi usando Variação da Potência de Transmissão e kNN. Brasília: Sociedade Brasileira de Computação, 2013. 12 p. Disponível em: <http://sbrc2013.unb.br/files/anais/trilha-principal/artigos/artigo-27.pdf>. Acesso em: 10 out. 2017.

Bahl, P. and Padmanabhan, V. (2000). Radar: an in-building rf-based user location and tracking system. In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, pages 775– 784 vol.2.

FERREIRA, Cleber de F.; MOTA, Roberto Dias. COMPARANDO APLICAÇÃO WEB SERVICE REST E SOAP. 2014. Disponível em: <http://web.unipar.br/~seinpar/2014/artigos/pos/Cleber_de_F_Ferreira_Roberto_Dias_Mota%20(1).pdf>. Acesso em: 25 maio 2019.

PLANSKY, Ricardo. Definição, restrições e benefícios do modelo de arquitetura REST. 2014. Disponível em: <https://imasters.com.br/desenvolvimento/definicao-restricoes-e-beneficios-modelo-de-arquitetura-rest/?trace=1519021197&source=single>. Acesso em: 15 nov. 2017.

CABRAL, Carlos. React Native: Construa aplicações móveis nativas com JavaScript. 2016. Disponível em: <https://tableless.com.br/react-native-construa-aplicacoes-moveis-nativas-com-javascript/>. Acesso em: 02 abr. 2019.

PROBST, Renato. Desenvolvimento de Apps em React Native. 2017. Disponível em: <https://www.eng.com.br/artigo.cfm?id=6169>. Acesso em: 02 abr. 2019.

SANTOS, Daniel dos. Aplicativos ajudam cegos e surdos em atividades do dia a dia; veja opções. 2013. Disponível em: <https://noticias.uol.com.br/tecnologia/noticias/redacao/2013/10/21/

aplicativos-ajudam-cegos-e-surdos-em-atividades-do-dia-a-dia-veja-opcoes.htm>. Acesso em: 12 nov. 2017.

JOSÉ, Italo. KNN (K-Nearest Neighbors) #1: Como Funciona. 2018. Disponível em: <https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e>. Acesso em: 01 abr. 2019.

SANTOS, Heverton Ramos dos; RANGEL JUNIOR, Alamir Rodrigues. Aplicação do algoritmo de Dijkstra para o problema de roteamento da frota de táxis partindo de um ponto fixo. 2010. 7 f. Monografia (Especialização) - Curso de Produção e Sistemas, Instituto Federal Farroupilha, Farroupilha, 2010. Disponível em: <http://bd.centro.iff.edu.br/xmlui/bitstream/handle/123456789/21/Artigo_Producao_Sistemas_Versao_Final.pdf?sequence=1&isAllowed=y>. Acesso em: 06 maio 2019.