

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO  
GRANDE DO SUL  
CAMPUS RIO GRANDE  
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE  
SISTEMAS

EVERTON LUIS DE ALMEIDA

**BlindBus: Tecnologia assistiva para  
deficientes visuais no uso de transporte  
público**

Trabalho de Conclusão de Curso  
apresentado como requisito parcial  
para a obtenção do grau de Tecnólogo em  
Análise e Desenvolvimento de Sistemas

Profa. Dra. Raquel de Miranda Barbosa  
Orientador

Rio Grande, dezembro de 2018

## CIP – CATALOGAÇÃO NA PUBLICAÇÃO

de Almeida, Everton Luis

BlindBus: Tecnologia assistiva para deficientes visuais no uso de transporte público / Everton Luis de Almeida. – Rio Grande: Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, 2018.

53 f.: il.

Trabalho de Conclusão de Curso (tecnólogo) – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Rio Grande, BR-RS, 2018. Orientador: Raquel de Miranda Barbosa.

1. Beacon. 2. Tecnologias Assistivas. 3. Deficiente Visual.  
4. Transporte Público. I. de Miranda Barbosa, Raquel. II. Título.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL

Reitor: Prof. Julio Xandro Heck

Pró-Reitor de Graduação: Prof. Lucas Coradini

Diretor Geral do Campus Rio Grande: Prof. Alexandre Jesus da Silva Machado

Coordenador do curso: Prof. Luciano Vargas Gonçalves

## **FOLHA DE APROVAÇÃO**

Monografia sob o título "*BlindBus: Tecnologia assistiva para deficientes visuais no uso de transporte público*", defendida por Everton Luis de Almeida e aprovada em 12 de dezembro de 2018, em Rio Grande, RS, pela banca examinadora constituída pelos professores:

---

Profa. Dra. Raquel de Miranda Barbosa  
Orientador

---

Prof. Dr. Eduardo Wenzel Brião  
IFRS - Campus Rio Grande

---

Prof. MSc. Tiago Guimarães Moraes  
IFRS - Campus Rio Grande

## **AGRADECIMENTOS**

Primeiramente agradeço a minha esposa Larielle de Oliveira, que me deu apoio e incentivo nas horas difíceis, de desânimo e cansaço.

A minha orientadora Profa Dra. Raquel de Miranda Barbosa pela oportunidade e apoio na elaboração deste projeto e pelo paciente trabalho de revisão do mesmo.

Aos meus pais, pelo amor, incentivo e apoio na minha vida inteira.

Ao Prof. Dr. Eduardo Wenzel Brião, pelas boas dicas dadas ao longo de todo curso.

Aos demais professores, não somente por terem me ensinado, mas por terem me feito aprender.

Aos meus amigos e companheiros de trabalho que fizeram parte da minha formação, especialmente ao Felipe Santos por estar sempre ao meu lado tanto nas horas de diversão quanto naqueles minutinhos antes de uma prova, e ao Jair Soares pelas ideias dadas a este trabalho.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

## RESUMO

Nos dias atuais percebemos um grande avanço na área de tecnologia com uma crescente utilização de equipamentos *smart* tais como: *smart tv* e *smartphone*, porém ainda existem pessoas que não podem usufruir destes benefícios por possuírem algum tipo de deficiência. A área de tecnologias assistivas procura tratar deste problema apresentando soluções que sejam capazes de promover uma maior autonomia para essas pessoas. Este projeto foi desenvolvido levando em consideração as dificuldades que os deficientes visuais enfrentam no dia-a-dia para locomover-se em suas cidades.

A solução desenvolvida apresenta um aplicativo para ajudar as pessoas com deficiência visual a se locomoverem utilizando o transporte público. Através de um hardware, chamado *beacon*, que fica instalado dentro do ônibus, é transmitido um sinal, dentro de uma área de abrangência, para o aparelho celular do usuário, que faz uma busca no banco de dados através do sinal recebido e informa o nome da linha do ônibus que está se aproximando e o sentido que ele está indo (através de mensagens sonoras).

O diferencial desta aplicação, é a utilização do *beacon*, pois possibilita dar ao usuário uma precisão bem maior que outras aplicações similares que utilizam *CHIP* para fazer essa comunicação. A solução desenvolvida é adaptável para qualquer tipo de *beacon*. Existem *beacons* que guardam informações no próprio hardware, porém o utilizado para essa solução, não dispõe desse benefício. Devido a esse fato, foi utilizada a comunicação com banco de dados externo para que fosse possível chegar o mais próximo da solução ideal.

**Palavras-chave:** Beacon. Tecnologias Assistivas. Deficiente Visual. Transporte Pú-  
blico.

## **LISTA DE SIGLAS E ABREVIATURAS**

ADA	<i>American with Disabilities Act</i> (Ato dos Americanos com Deficiências)
APK	<i>Android Package</i>
API	<i>Application Programming Interface</i>
BLE	<i>Bluetooth Low Energy</i>
CAA	Comunicação Aumentativa e Alternativa
EID	<i>Ephemeral ID</i>
FM	Frequência Modulada
GPRS	<i>General Packet Radio Services</i> (Serviços Gerais de Pacote por Rádio)
GPS	<i>Global Positioning System</i> (Sistema de Posicionamento Global)
GTFS	Especificação Geral para Feeds de Transporte Público
HTTP	<i>HyperText Transfer Protocol</i> (Protocolo de Transferência de Hipertexto)
IDE	<i>Integrated Development Environment</i> (Ambiente de Desenvolvimento Integrado)
JAWS	<i>Job Access With Speech</i>
JSON	<i>JavaScript Object Notation</i> (Notação de Objetos JavaScript)
NVDA	<i>Non Visual Desktop Access</i>
PCS	<i>Picture Communication Symbols</i> (Símbolos de Comunicação Pictórica)
SGBD	Sistema de Gerenciamento de Banco de Dados
TA	Tecnologia Assistiva
TLM	Telemetria
UID	<i>User ID</i> (identificador de usuário)
URL	<i>Uniform Resource Locator</i> (Localizador Padrão de Recursos)
UUID	<i>Universal Unique Identifier</i> (Identificador único universal)

## **LISTA DE FIGURAS**

Figura 2.1:	Materias adaptados. Fonte: (BERSCH, 2013) . . . . .	14
Figura 2.2:	Prancha com PCS. Fonte: (BERSCH, 2013) . . . . .	15
Figura 2.3:	Recursos de acessibilidade. Fonte: (BERSCH, 2013) . . . . .	15
Figura 2.4:	Controle de ambientes. Fonte: (BERSCH, 2013) . . . . .	16
Figura 2.5:	Projeto de acessibilidade. Fonte: (BERSCH, 2013) . . . . .	16
Figura 2.6:	Prótese e Órtese. Fonte: (BERSCH, 2013) . . . . .	17
Figura 2.7:	Adequação Postural. Fonte: (BERSCH, 2013) . . . . .	17
Figura 2.8:	Auxílios de Mobilidade. Fonte: (BERSCH, 2013) . . . . .	18
Figura 2.9:	Auxílios para deficientes visuais. Fonte: (BERSCH, 2013) . . . . .	18
Figura 2.10:	Auxílios para deficientes auditivos. Fonte: (BERSCH, 2013) . . . . .	18
Figura 2.11:	Elevador para cadeirantes. Fonte: (BERSCH, 2013) . . . . .	19
Figura 2.12:	Tela inicial do Dosvox. . . . .	21
Figura 2.13:	Compatibilidade <i>beacon</i> . . . . .	23
Figura 2.14:	Funcionamento Beacon. Fonte: CARNEIRO (2015) (adaptado pelo autor) . . . . .	24
Figura 2.15:	Sistema de gerenciamento de banco de dados (SGBD) . . . . .	27
Figura 3.1:	Ciclo do CittaMobi. Fonte: (CITTAMOBI, 2015) . . . . .	28
Figura 3.2:	Menu inicial do aplicativo CittaMobi . . . . .	30
Figura 3.3:	Telas de opções do CittaMobi (1) . . . . .	30
Figura 3.4:	Telas de opções do CittaMobi (2) . . . . .	30
Figura 3.5:	Ferramentas Moovit. Fonte: (INC, 2017) . . . . .	31
Figura 3.6:	Telas de opções do Moovit (1) . . . . .	32
Figura 3.7:	Telas de opções do Moovit (2) . . . . .	32
Figura 3.8:	Telas de opções do Wayfinder . . . . .	32
Figura 3.9:	Estátua do Buda . . . . .	33
Figura 4.1:	Funcionamento do sistema . . . . .	35
Figura 4.2:	Diagrama Caso de Uso BlindBus Admin . . . . .	36
Figura 4.3:	Diagrama Caso de Uso BlindBus User . . . . .	36
Figura 4.4:	Diagrama de Classes . . . . .	36
Figura 4.5:	Fluxo da aplicação do motorista . . . . .	37
Figura 4.6:	Tela da aplicação do deficiente visual . . . . .	38
Figura 5.1:	Estrutura do aplicativo “BlindBus Admin” . . . . .	39
Figura 5.2:	Carregar e Inserir Usuários . . . . .	40
Figura 5.3:	Tela de <i>Login</i> e Tela de Configurações . . . . .	40
Figura 5.4:	Configurações de Linhas . . . . .	41

Figura 5.5:	Cadastro e edição de Linhas	42
Figura 5.6:	Envia dados ao Servidor	42
Figura 5.7:	Aguadando retorno do banco	43
Figura 5.8:	Entrando no Aplicativo	43
Figura 5.9:	Aplicativo para deficiente.	44
Figura 5.10:	Comunicação entre ônibus e <i>smartphone</i>	44
Figura 5.11:	Função que verifica se possui internet	45
Figura 5.12:	Classe de conexão à internet	46
Figura 5.13:	Funcionamento do serviço externo	46
Figura 5.14:	Arquivo JSON de configuração do <i>Firebase</i>	47
Figura 5.15:	Variáveis de inicialização para utilização do <i>beacon</i>	48
Figura 5.16:	Funções para utilização do <i>beacon</i>	48

## **LISTA DE TABELAS**

Tabela 3.1: Comparando os aplicativos . . . . .	34
Tabela 5.1: Comparaçao entre <i>beacons</i> com diferentes abrangências . . . . .	49

# SUMÁRIO

<b>1 INTRODUÇÃO . . . . .</b>	12
<b>1.1 Objetivos . . . . .</b>	13
<b>1.2 Organização do texto . . . . .</b>	13
<b>2 FUNDAMENTAÇÃO TEÓRICA . . . . .</b>	14
<b>2.1 Tecnologias Assistivas . . . . .</b>	14
<b>2.2 Tecnologias Assistivas para deficientes visuais . . . . .</b>	19
<b>2.2.1 Virtual Vision (VV) . . . . .</b>	20
<b>2.2.2 Job Access With Speech (JAWS) . . . . .</b>	20
<b>2.2.3 Talkback . . . . .</b>	21
<b>2.2.4 Dosvox . . . . .</b>	21
<b>2.2.5 NVDA (<i>NonVisual Desktop Access</i>) . . . . .</b>	22
<b>2.3 Beacons . . . . .</b>	22
<b>2.4 Ferramentas de desenvolvimento . . . . .</b>	24
<b>2.4.1 Android Studio . . . . .</b>	25
<b>2.4.2 SQLite . . . . .</b>	25
<b>2.4.3 Firebase . . . . .</b>	26
<b>3 TRABALHOS RELACIONADOS . . . . .</b>	28
<b>3.1 CittaMobi . . . . .</b>	28
<b>3.2 Moovit . . . . .</b>	30
<b>3.3 Wayfinder . . . . .</b>	32
<b>3.4 TagPoint . . . . .</b>	33
<b>3.5 Comparação entre os aplicativos . . . . .</b>	33
<b>4 DESCRIÇÃO E MODELAGEM . . . . .</b>	35
<b>4.1 Diagrama de Casos de Uso . . . . .</b>	35
<b>4.2 Diagrama de Classes . . . . .</b>	36
<b>4.3 Telas dos aplicativos . . . . .</b>	37
<b>5 BLINDBUS: IMPLEMENTAÇÃO E TESTES . . . . .</b>	39
<b>5.1 BlindBus Admin . . . . .</b>	39
<b>5.2 BlindBus User . . . . .</b>	43
<b>5.3 Comunicação . . . . .</b>	44
<b>5.3.1 Internet . . . . .</b>	45
<b>5.3.2 Serviço externo . . . . .</b>	45
<b>5.3.3 Beacon . . . . .</b>	47
<b>5.4 Requisitos das aplicações . . . . .</b>	49

<b>5.5</b>	<b>Testes e avaliação</b>	49
<b>6</b>	<b>CONCLUSÕES</b>	50
<b>6.1</b>	<b>Trabalhos futuros</b>	51
<b>REFERÊNCIAS</b>		52

## 1 INTRODUÇÃO

O diálogo entre os seres vivos que constituem uma sociedade não se limita à leitura e à escrita tradicional, pois há diversos meios de um ser se comunicar com outro. Esses meios ocorrem devido ao fato de haver indivíduos com limitações físicas, cuja forma pela qual eles se expressam é restrita. É de muita importância que as pessoas tenham o conhecimento das formas de comunicação, para que a compreensão entre todos se torne normal, pois para que ocorra um verdadeiro diálogo, é necessário que haja conhecimento delas entre os dois ou mais lados.

Uma pessoa com deficiência visual pode comunicar-se através da fala, gestos e sons, porém não é só destes meios que se necessita para ter uma boa interação na sociedade. Um dos elementos para que haja essa interação é a visão e, para suprir esta falta nos deficientes visuais, foi criada uma linguagem escrita chamada braille.

O Braille é um sistema de leitura tátil e escrita para pessoas com deficiência visual. Através deste sistema, o deficiente visual vê suas possibilidades ampliadas, de acesso à informação e comunicação. O mundo passa a ter sentido e construir significados por meio daquilo que, até então, estava oculto pela falta de acesso. Com ele surgem múltiplas possibilidades de atribuir sentidos e construir imagens para representar o mundo que se amplia e se desvenda na ponta dos dedos (PRATICA, 2011).

A inclusão de pessoas com deficiência visual no meio digital se faz importante para minimizar as desigualdades sociais que ainda existem no mundo em que vivemos. Um mecanismo que facilita a inclusão digital de deficientes visuais são as tecnologias assistivas que surgiram com o objetivo de ampliar habilidades funcionais de pessoas com deficiência de forma a promover maior autonomia e independência dessas pessoas (BERSCH, 2013).

A Constituição Federal de 1988 em seu Capítulo VII, art. 227 parágrafo segundo prevê “a facilitação do acesso aos bens e serviços coletivos, com a eliminação de preconceitos e obstáculos arquitetônicos”. Determina ainda, em seu parágrafo segundo que “a lei disporá sobre normas de construção dos logradouros e dos edifícios de uso público e de fabricação de veículos de transporte coletivo, a fim de garantir acesso adequado às pessoas com deficiência” (SENADO, 2017). O Capítulo IX do art. 244, dispõe sobre a “adaptação dos logradouros, dos edifícios de uso público e dos veículos de transporte coletivo atualmente existentes a fim de garantir acesso adequado às pessoas com deficiência” (SENADO, 2016).

Neste contexto, este trabalho visa facilitar o deslocamento de pessoas com deficiência visual pela cidade, utilizando transporte público, principalmente por intermédio de ônibus, sem que se sintam privadas por suas limitações. Para isso será utilizado um dispositivo *beacon* que é um pequeno aparelho de proximidade que emite informações através de *bluetooth*, diretamente aos *smartphones* cadastrados. A ideia é que se tenha um aplica-

tivo que seja utilizado pelo motorista para cadastrar as linhas no ônibus, vinculado a um dispositivo *beacon* que, ao se aproximar de uma parada de ônibus envia informações para um aplicativo que ficará com o usuário deficiente visual. Este aplicativo, por sua vez, vai emitir as informações do ônibus que se aproxima através mensagens sonoras.

## 1.1 Objetivos

O trabalho tem como objetivo geral, desenvolver uma tecnologia assistiva que facilite o uso de transporte coletivo para deficientes visuais através do uso do dispositivo *beacon* associado a um aplicativo acessível a deficientes visuais.

Como objetivos específicos, tem-se:

- Desenvolver um aplicativo para utilização de *beacon*.
- Desenvolver um aplicativo acessível para deficientes visuais.
- Fazer a integração das informações geradas.
- Facilitar a acessibilidade do deficiente visual no uso do transporte coletivo.
- Executar testes e avaliar seu funcionamento.

## 1.2 Organização do texto

O texto está organizado conforme o descrito a seguir. No capítulo 2 é apresentada a fundamentação teórica deste trabalho, destacando tecnologias assistivas e sua aplicação a deficientes visuais, bem como a tecnologia *beacon* utilizada como recurso para transmitir as informações. O capítulo 3 descreve alguns trabalhos relacionados comparando-os com o trabalho proposto. O capítulo 4 descreve a modelagem do projeto. No capítulo 5 é apresentado o “BlindBus” com a descrição dos aplicativos desenvolvidos, seus requisitos e a comunicação entre eles. O capítulo 6 mostra os resultados e considerações finais, destacando algumas possibilidades de trabalhos futuros. Após, são encontras as referências e anexos.

## 2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo descreve alguns conceitos importantes utilizados para a compreensão e desenvolvimento desse trabalho.

### 2.1 Tecnologias Assistivas

Tecnologia Assistiva (TA) é um termo ainda novo, utilizado para identificar todo o arsenal de recursos e serviços que contribuem para proporcionar ou ampliar habilidades funcionais de pessoas com deficiência e, consequentemente promover vida independente e inclusão (BERSCH, 2013).

A evolução tecnológica está, cada vez mais, facilitando nossas vidas e ao mesmo tempo nos tornamos dependentes delas. Utilizamos diariamente ferramentas desenvolvidas para simplificar nossas atividades, como controle remoto, automóveis, computadores, celulares, entre tantos outros.

A TA tem algumas classificações que são construídas com base nas diretrizes gerais do *American with Disabilities Act* (ADA), porém estas não são definitivas, podendo variar segundo alguns autores. As categorias apresentadas pelo ADA (BERSCH, 2013) são:

- Auxílios para a vida diária e vida prática

Para conseguir alcançar independência ou diminuir a dependência de outras pessoas nas atividades diárias, existe uma série de adaptações que variam desde engrossar um cabo de colher, colocar velcro no lugar de cadarços de um calçado, até um sistema de controle ambiental.

Na Figura 2.1 são mostrados alguns materiais escolares que foram adaptados favorecendo o recorte, a leitura e a escrita.



Figura 2.1: Materiais adaptados. Fonte: (BERSCH, 2013)

- Comunicação Aumentativa e Alternativa (CAA)

Ferramentas e estratégias utilizadas para resolver os desafios de comunicação do cotidiano. A CAA é usada para descrever métodos de comunicação que podem ajudar as pessoas com algum tipo de deficiência ou limitação. Esses métodos podem beneficiar muitas pessoas desde aqueles que estão começando a se comunicar até aqueles que já tem uma certa prática.

Na Figura 2.2 observa-se uma prancha de símbolos de comunicação pictórica (PCS - *Picture Communication Symbols*) que são extremamente úteis para criação de atividades educacionais.



Figura 2.2: Prancha com PCS. Fonte: (BERSCH, 2013)

- Recursos de acessibilidade ao computador

Conjunto de *hardware* e *software* que torna o computador acessível a pessoas com algum tipo de deficiência visual e/ou auditiva. Inclui equipamentos de entrada como *mouses*, teclados modificados ou alternativos e acionadores e equipamentos de saída como sons, imagens e informações táteis.

Na Figura 2.3 são mostrados o teclado *Intellikeys*, acionadores com mouse adaptado, mouse com movimento da cabeça, monitor com tela de toque e órtese para digitação.



Figura 2.3: Recursos de acessibilidade. Fonte: (BERSCH, 2013)

- Sistemas de controle de ambiente

Sistemas eletrônicos que permitem às pessoas com limitações moto-locomotoras, controlar remotamente aparelhos eletro-eletrônicos como a luz, o som, televisores, ventiladores, sistemas de segurança, entre outros, localizados em seu quarto, sala, escritório, casa e arredores.

Na Figura 2.4 observa-se uma representação de como funciona um controle de ambiente, onde existe um dispositivo em cima da porta que é acionado pelo deficiente para abri-la quando ele escuta a campainha.



Figura 2.4: Controle de ambientes. Fonte: (BERSCH, 2013)

- Projetos arquitetônicos para acessibilidade

Adaptações estruturais e reformas na casa e/ou ambiente de trabalho, através de rampas, elevadores, adaptações em banheiros entre outras, que retiram ou reduzem as barreiras físicas, facilitando a locomoção da pessoa com deficiência.

A Figura 2.5 mostra projetos arquitetônicos para acessibilidade em elevadores, calçadas e banheiro, respectivamente.



Figura 2.5: Projeto de acessibilidade. Fonte: (BERSCH, 2013)

- Órteses e próteses

Troca ou ajuste de partes do corpo, faltantes ou de funcionamento comprometido, por membros artificiais ou outros recursos ortopédicos. São normalmente confeccionadas sob medida e servem no auxílio de mobilidade e funções manuais (escrita, digitação, utilização de talheres, manejo de objetos para higiene pessoal), correção postural, entre outros.



Figura 2.6: Prótese e Órtese. Fonte: (BERSCH, 2013)

Na Figura 2.6 são apresentadas prótese do membro inferior e órtese da mão.

- Adequação Postural

Adaptações para cadeira de rodas ou outro sistema de sentar visando o conforto e distribuição adequada da pressão na superfície da pele (almofadas especiais, assentos e encostos anatômicos), bem como posicionadores e contentores que propiciam maior estabilidade e postura adequada do corpo através do suporte e posicionamento de tronco/cabeça/membros.

Na Figura 2.7 tem-se um desenho representativo da adequação postural, poltrona postural na escola e no carrinho para transporte.



Figura 2.7: Adequação Postural. Fonte: (BERSCH, 2013)

- Auxílios de mobilidade

Cadeiras de rodas manuais e motorizadas, bengalas, muletas, andadores, carrinhos e qualquer outro veículo utilizado na melhoria da mobilidade pessoal.

Na Figura 2.8 observa-se cadeiras de rodas motorizadas, equipamento para cadeiras de rodas subirem e desceram escadas, cadeira de rodas especial para praia e andador com freio.

- Auxílios para cegos ou para pessoas com visão subnormal

Auxílios para grupos específicos que incluem lupas e lentes, Braille para equipamentos com síntese de voz, grandes telas de impressão, sistema de televisão com aumento para leitura de documentos, publicações etc.

A Figura 2.9 mostra termômetro, relógio e teclado falado.



Figura 2.8: Auxílios de Mobilidade. Fonte: (BERSCH, 2013)



Figura 2.9: Auxílios para deficientes visuais. Fonte: (BERSCH, 2013)

- Auxílios para pessoas com surdez ou com déficit auditivo

Auxílios que incluem vários equipamentos (infravermelho, FM), aparelhos para surdez, telefones com teclado-teletipo (TTY), sistemas com alerta táctil-visual, celular com mensagens escritas e chamadas por vibração, *software* que favorece a comunicação ao telefone celular transformando em voz o texto digitado no celular e em texto a mensagem falada. Livros, textos e dicionários digitais em língua de sinais.

Na Figura 2.10 tem-se o teclado-teletipo (TTY), e o celular com mensagens escritas e chamada por vibração.



Figura 2.10: Auxílios para deficientes auditivos. Fonte: (BERSCH, 2013)

- Adaptações em veículos

Acessórios e adaptações que possibilitam a condução do veículo, elevadores para cadeiras de rodas, camionetas modificadas e outros veículos automotores usados no transporte pessoal.

A Figura 2.11 mostra um elevador de cadeiras de rodas.



Figura 2.11: Elevador para cadeirantes. Fonte: (BERSCH, 2013)

## 2.2 Tecnologias Assistivas para deficientes visuais

Vivemos em um mundo totalmente dependente da informação. Para que haja sucesso nesta sociedade é preciso que o acesso e utilização das tecnologias de informação e comunicação esteja disponível a todos, para evitar a exclusão social. Neste ponto a internet tem um papel importante a desempenhar. Com ela quebram-se barreiras físicas e espaciais, servindo de suporte a inúmeras atividades possíveis de serem desempenhadas por portadores de deficiência.

A Tecnologia Assistiva exerce um papel fundamental para os deficientes visuais, incentivando-os a executar tarefas que seriam praticamente impossíveis sem o auxílio apropriado. O afastamento deste apoio impõe a diminuição ao acesso na inclusão digital, além de dificultar a inclusão destes cidadãos na sociedade. Algumas dicas básicas para ajudar com a acessibilidade para deficientes visuais são mostradas em (ONLINE, 2017). São elas:

1. Utilizar faixas no piso, nas áreas de circulação, com textura e cor diferenciadas, para facilitar a identificação do percurso para deficientes visuais;
2. Verificar os obstáculos existentes nas áreas de circulação e principalmente se tais obstáculos sofrem mudança de localização periódica ou eventualmente;
3. Nos elevadores, as botoeiras e comandos devem ser acompanhados dos símbolos em Braille;
4. Para um número de parada superior a dois andares, deve também haver comunicação auditiva dentro da cabine do elevador, indicando o andar onde o elevador se encontra parado;
5. Identificar os sinais luminosos que existem no ambiente de trabalho, para que sejam acompanhados por sinais sonoros;
6. Implantar software com sintetizadores de voz nos computadores;

7. Realizar adaptações na construção do textitsite da empresa, permitindo o acesso dos seus colaboradores e clientes externos.

Na área da tecnologia existem algumas ferramentas que auxiliam tanto no aprendizado quanto na vida de pessoas com deficiência visual. A seguir são citadas algumas delas:

### **2.2.1 Virtual Vision (VV)**

Essa ferramenta foi desenvolvida pela empresa MicroPower, auxiliando pessoas com deficiência visual a utilizar com autonomia o Windows e outros aplicativos. Algumas de suas principais funcionalidades, conforme especificado no textitsite da empresa (MICRO-POWER, 2016) são:

- Leitura de informações mostradas dentro de blocos no menu iniciar;
- Sistema de navegação, mais rápido e responsável, incluindo navegação letra a letra ou palavra a palavra em cada objeto navegador;
- Navegação por cabeçalhos (usando teclas 1 até 6);
- Aviso sonoro ao entrar em campos de formulário;
- Detecção de campos de edição não convencionais em *browsers*;
- Suporte a *links* para a mesma página;
- Possibilidade de uso da tecla *Insert* para compor teclas de atalho de comandos do Virtual Vision e mapeamentos;
- Leitura automática de e-mail no Outlook ao abrir uma mensagem;
- Compatibilidade com Skype no Windows;
- *Layout* de teclado pré configurado para *notebook*;
- Suporte a sistemas operacionais Windows de 64 bits;
- Suporte a vozes SAPI 5.4 (vozes de alta qualidade da Microsoft em vários idiomas, incluindo português);
- Comando para informar a duração de bateria em *notebooks*;
- Opção para manter uma única característica de voz (velocidade e tonalidade) para qualquer tipo de texto.

### **2.2.2 Job Access With Speech (JAWS)**

Conforme (SITEINA, 2015) o JAWS é o mais conhecido *software* para acessibilidade de pessoas com deficiência visual do mundo. Ele é composto por um sistema que lê as informações dispostas na tela e por um sintetizador de voz para reconhecer comandos executados pelo usuário. Com o auxílio deste *software*, pessoas com deficiência visual podem usufruir dos recursos de informática e navegação na internet com mais facilidade e sem depender da ajuda de terceiros. O *software* conta com diversos atalhos de teclado para facilitar a navegação do usuário. Também pode ser usado com um *hardware* de linhas de braille, para substituir a leitura de tela. Ele conta com 17 idiomas de saída, inclusive o Português (Brasil). O comando de voz do usuário, no entanto, só é suportado no idioma inglês.

### 2.2.3 Talkback

É um recurso padrão na maioria dos aparelhos com sistema operacional Android que visa ajudar deficientes visuais. Este recurso funciona de modo a ajudar pessoas com perda parcial ou total da visão, fazendo o aparelho falar qual opção foi tocada antes de abrir a opção. Ao ativar essa opção o aparelho não funciona mais da forma normal, isto por que o *Talkback* exige um modo diferente para deslizar a tela e para abrir aplicativos, tornando o uso mais fácil para deficientes visuais.

No modo *Talkback*, para controlar o aparelho, utiliza-se dois dedos para deslizar a tela e, para abrir uma opção, é necessário tocar uma vez para selecionar e assim o aparelho diz o nome do aplicativo ou opção escolhida. Depois é necessário dar mais dois toques na opção selecionada para abri-la. O modo *Talkback* funciona dessa forma, pois quando uma pessoa com problema de perda de visão usa o aparelho tem que ter certeza de onde tocou, e se estiver correta a opção escolhida será confirmada com os dois toques (SELETRONIC, 2017).

### 2.2.4 Dosvox

É um sistema para computadores que se comunica com o usuário através de síntese de voz, viabilizando, deste modo, o uso de computadores por deficientes visuais, que adquirem um alto grau de independência no estudo e no trabalho. O sistema realiza a comunicação com o deficiente visual através de síntese de voz em português, sendo que a síntese de textos pode ser configurada para outros idiomas. Seu diferencial é que a comunicação homem-máquina é muito mais simples, e leva em conta as especificidades e limitações dessas pessoas. Ao invés de simplesmente ler o que está escrito na tela, o DOSVOX estabelece um diálogo amigável, através de programas específicos e interfaces adaptativas (JAVAVOX, 2015). Com o “DOSVOX”, o computador é ligado normalmente. Os sons característicos da entrada do *Windows* são importantes para que o deficiente visual saiba quando é possível começar a executar o DOSVOX. O acionamento é feito apertando-se as teclas “ctrl + alt + d”, sendo então sintetizada a frase “DOSVOX - O que você deseja?”, que será ouvida sempre que o sistema necessitar de uma nova informação assim como mostra a Figura 2.12.

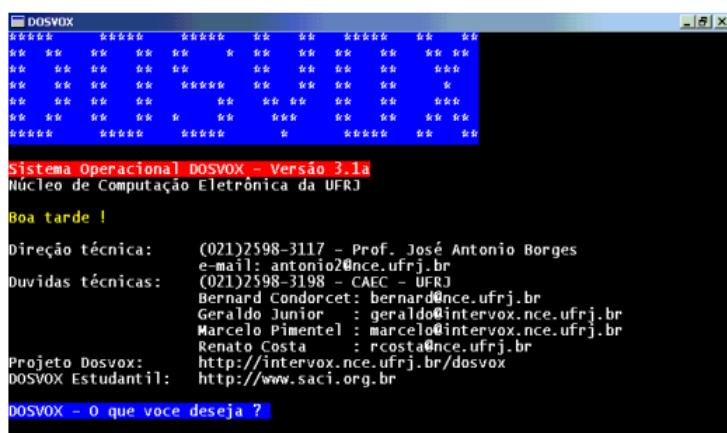


Figura 2.12: Tela inicial do Dosvox.

Pressionando-se F1 o “menu principal” será apresentado ao mesmo tempo que sonorizado. Na tela inicial aparecerão também informações sobre como adquirir, ou obter ajuda sobre o DOSVOX (BORGES, 2000).

### 2.2.5 NVDA (*NonVisual Desktop Access*)

É um leitor de tela gratuito que permite que as pessoas com deficiência visual usem computadores. Ele lê o texto na tela em uma voz computadorizada. Pode-se controlar o que é lido, movendo o cursor para a área relevante de texto com um mouse ou as setas em seu teclado. Também é possível converter o texto em braille se o usuário do computador possui um dispositivo chamado “exibição braille”. O NVDA funciona com o Microsoft Windows (ACCESS, 2017).

- Teclas de Comandos do NVDA.

A maioria dos comandos de teclado específicos do NVDA consiste em pressionar a tecla modificadora do NVDA (*insert*) juntamente com uma ou mais outras teclas. Uma exceção são os comandos de revisão de texto que utilizam somente as teclas do *numpad* (teclado numérico). Por padrão, ambas as teclas *numpad Insert* e *extended Insert* podem ser usadas como teclas modificadoras no NVDA.

## 2.3 Beacons

São pequenos aparelhos de proximidade que emitem informações, por meio da tecnologia *bluetooth*, diretamente aos *smartphones* cadastrados. A ideia é permitir a interação mais rápida de possíveis clientes com seus interesses. A grande novidade dos aparelhos *beacons*, além do custo acessível, é que eles podem ser instalados em paredes, produtos ou vitrines, permitindo a comunicação entre empresa e público por meio da localização e sem a necessidade de acesso à internet, já que utiliza o *bluetooth* do *smartphone* para enviar as mensagens (IMPACTA, 2017).

Os *beacons* usam *Bluetooth Low Energy* (BLE) para detectarem a proximidade de outros dispositivos e transmitirem um número identificador único, que é então recebido pelo sistema operacional do dispositivo com o qual ele está se comunicando. Segundo (BRASIL, 2017), quando a comunicação é estabelecida existem duas ações que podem ser tomadas pelo sistema: passiva e uma ativa.

- A ação passiva trata simplesmente de armazenar, na memória local ou em algum banco de dados, que aquela conexão aconteceu. Para exemplificar, podemos considerar a frase “Eu sei que o *iPhone* de código 12345 se aproximou do sensor de código 67890”. Na prática, isso significa: “Um cliente do supermercado portando um *iPhone* acabou de passar pelo corredor dos cereais”.
- A ação ativa acontece quando essa comunicação inicia alguma atividade no dispositivo do usuário. Enviar notificações, mudar o estado do sistema, iniciar alguma ação dentro de um aplicativo específico, fazer *check-in* em alguma rede social, etc. Por exemplo, o usuário recebe uma notificação dizendo: “Semana *black friday*! Venha aproveitar”.

A Figura 2.13 mostra algumas plataformas compatíveis com o *beacon*.

Para integrar a plataforma Google Chrome com o *beacon* é utilizada o *Eddystone*, que é um protocolo próprio para *beacon*, criado pelo Google, que permite aos dispositivos BLE, transmitirem dados a curta distância para um receptor. A principal vantagem deste protocolo em relação aos existentes, é que ele permite que sejam definidos e enviados quatro diferentes pacotes de informações (RFID, 2017).

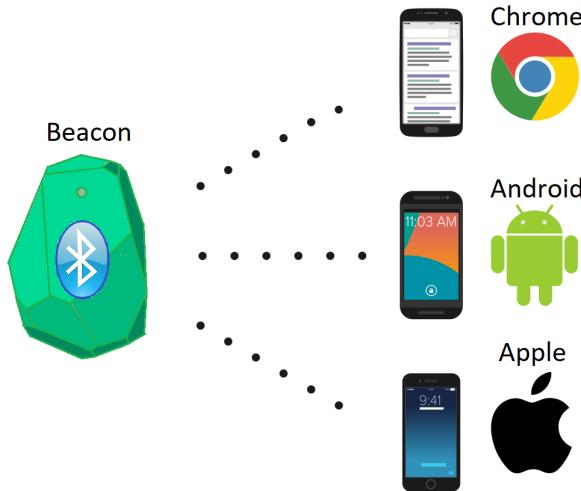


Figura 2.13: Compatibilidade *beacon*.

- Eddystone - *Uniform Resource Locator* (URL)

É o formato de *beacon* para a *web* física, onde é colocado o conteúdo que todos possam acessá-lo. Eddystone-URL não requer um aplicativo personalizado. Ele funciona com *beacon scanners*, às vezes chamados de *beacon browsers*. Ele também funcionará com aplicativos desenvolvidos para plataformas de *marketing*. O Eddystone-URL transmite uma URL, que é simplesmente uma página da *web*. É possível vinculá-lo a qualquer *site* na internet.

- Eddystone - *User ID* (UID)

Requer um aplicativo customizado para interpretar esse UID específico e interagir com ele. Muito semelhante ao *beacon*, porém ele só transmite uma sequência de 16 caracteres, divididos em: 10 caracteres para o "*Namespace*" e 6 caracteres para o "*InstanceID*". Normalmente o "*Namespace*" é utilizado para identificar uma entidade e o "*InstanceID*" um *beacon* específico.

- Eddystone - Telemetria (TLM)

É utilizado tanto para aplicativos, quanto para *browser*. A sua principal função é transmitir dados de sensores e dados administrativos do próprio *beacon*.

- Eddystone - *Ephemeral ID* (EID)

Ele funciona como Eddystone-UID, exceto que ele é projetado para possuir mais segurança. Ao invés de um "*Namespace*" e um "*InstanceID*" fixos, o EID é um identificador aleatório e é interpretado pelo *Google Beacon Registry*, onde ele pode retornar anexos atribuídos a ele, a partir do *Google Proximity API* ou do *Google Developer's Console*. O ID aleatório do EID o protege contra fraudes e impede que outros clonem os *beacons*, utilizando uma identificação fixa igual.

Conforme CARNEIRO (2015) o *beacon* tem quatro identificadores que são responsáveis pelo seu funcionamento. São eles:

- *Universal Unique Identifier* (UUID): É uma sequência de 16 bytes usada para diferenciar um grande grupo de *beacons* relacionados. Por exemplo, se um supermercado utilizar um grande conjunto de *beacons* relacionados, todos terão o mesmo

UUID. Isso permite que o aplicativo específico do supermercado, possa identificar que o *beacon* que está transmitindo pertence ao supermercado.

- Mor: É uma sequência de 2 *bytes* usada para distinguir um subconjunto menor de *beacons* dentro de um grupo maior. Se o supermercado em questão for uma rede, com várias lojas, o mor será utilizado para definir em qual das lojas o *beacon* está.
- Minor: É uma sequência de 2 *bytes* destinada a identificar *beacons* individuais. É utilizada para identificar o local exato onde o cliente está na loja, por exemplo.
- Tx Potência: É usada para determinar a distância do emissor (*Beacon*), também conhecido como farol. A potência de transmissão é definida como a força do sinal. Isso deverá ser calibrado e codificado com antecedência. Os dispositivos podem então usar isto como uma base para estimar a proximidade do *smartphone* do farol.

Na Figura 2.14, observa-se o funcionamento do *beacon* simplificado. Em uma loja específica, o *beacon* emite um pacote de informação e este é capturado por um *smartphone*. Nesse pacote, um ID de um produto é informado, o aplicativo da loja captura a informação transmitida, recupera esse ID e em seguida consulta um servidor, buscando o produto relacionado a este ID (CARNEIRO, 2015).



Figura 2.14: Funcionamento Beacon. Fonte: CARNEIRO (2015) (adaptado pelo autor)

O UUID possui 32 dígitos hexadecimais, divididos em 5 grupos, separados por hífens. Este pode ser considerado o *ID* do *beacon*. Ele é o número que o identifica. Por exemplo: para um *beacon* com o UUID f1234da5-6fa7-8e90-1234-bc5b67e8901e tem-se cinco grupos e os grupos devem seguir o seguinte número de caracteres:

- Primeira sequência: 8
- Segunda sequência: 4
- Terceira sequência: 4
- Quarta sequência: 4
- Quinta sequência: 12

Os caracteres devem ser números de 0-9, ou letras de A a F. Um grupo pode ser feito inteiramente de apenas números ou letras ou uma combinação de ambos. O objetivo do ID é distinguir os *beacons* em sua rede. Por padrão cada *beacon* terá apenas um UUID.

## 2.4 Ferramentas de desenvolvimento

Nesta seção são apresentadas as ferramentas utilizadas para o desenvolvimento desse trabalho.

#### 2.4.1 Android Studio

É um ambiente de desenvolvimento integrado (IDE) que oferece as ferramentas mais rápidas para a criação de aplicativos em todos os tipos de dispositivos Android. Além do editor de código e das ferramentas de desenvolvedor avançados, segundo STUDIO (2017) o *Android Studio* oferece mais recursos para aumentar a produtividade na criação de aplicativos *Android*, como:

1. Um emulador com diversos recursos;
2. Um ambiente unificado para poder desenvolver para todos os dispositivos Android;
3. *Instant Run* para aplicar alterações a aplicativos em execução sem precisar compilar um novo Android Package (APK);
4. Modelos de códigos e integração com GitHub para ajudar a criar recursos comuns dos aplicativos e importar exemplos de código;
5. Ferramentas de verificação de código suspeito para detectar problemas de desempenho, usabilidade, compatibilidade com versões e outros.

A estrutura do projeto do *Android Studio* funciona com um ou mais módulos, com arquivos de código-fonte e recursos. Alguns deles são: Módulos de aplicativo *Android*, Módulos de biblioteca e Módulos do *Google App Engine*.

Referente aos módulos de biblioteca tem-se a biblioteca *Beacon* do *Android* que permite que os dispositivos *Android* usem balizas como os dispositivos iOS. Um aplicativo pode solicitar uma notificação quando um ou mais *beacons* aparecem ou desaparecem. Um aplicativo também pode solicitar uma atualização de um ou mais beacons a uma frequência de aproximadamente 1Hz (NETWORKS, 2015).

A ferramenta conta com a ajuda de um emulador de aplicativos, desenvolvido também pela equipe do *Android Studio* chamada *Android Emulator* que se encarrega de simular um dispositivo móvel e exibir no seu computador de desenvolvimento. Com ele, é possível criar protótipos, desenvolver e testar aplicativos do *Android* sem usar o dispositivo de *hardware*.

#### 2.4.2 SQLite

O SQLite é um mecanismo de banco de dados SQL incorporado, ele não possui um processo de servidor separado. SQLite lê e grava diretamente em arquivos de disco comuns. Um banco de dados SQL completo com várias tabelas, índices, disparadores e visualizações está contido em um único arquivo de disco. O formato do arquivo do banco de dados é de plataforma cruzada - é possível copiar livremente um banco de dados entre sistemas de 32 bits e 64 bits ou entre arquiteturas *big-endian* e *little-endian*. Esses recursos tornam o SQLite uma escolha popular como um formato de arquivo de aplicativo (SQLITE, 2017).

SQLite é uma biblioteca compacta. Com todos os recursos ativados. O tamanho da biblioteca pode ser inferior a 500KiB, dependendo da plataforma de destino e configurações de otimização do compilador. (O código de 64 bits é maior. E algumas otimizações do compilador, como a ativação agressiva da função e o desdobramento do *loop*, podem

fazer com que o código do objeto seja muito maior). Se os recursos opcionais forem omitidos, o tamanho da biblioteca SQLite pode ser reduzido abaixo de 300KiB. O SQLite também pode ser feito para rodar em espaço de pilha mínimo (4KiB) e muito pouco *heap* (100KiB), tornando o SQLite uma escolha de mecanismo de banco de dados popular em dispositivos com restrições de memória. Há uma compensação entre o uso da memória e a velocidade. O SQLite geralmente corre mais rápido que a memória que foi fornecida. No entanto, o desempenho geralmente é bastante bom mesmo em ambientes de baixa memória. Dependendo de como ele é usado, pode ser mais rápido do que o sistema direto de arquivos (SQLITE, 2017).

O SQLite é uma escolha popular para o mecanismo de banco de dados em celulares, Beacons, MP3 players e outros aparelhos eletrônicos. Ele também tem uma pequena pegada de código, faz um uso eficiente da memória, do espaço em disco e da largura de banda do disco. É altamente confiável e não requer manutenção de um Administrador de banco de dados. Os formatos suportados são XML, JSON, CSV (SQLITE, 2017).

#### 2.4.3 Firebase

O Firebase foi desenvolvido por James Tamplin e Andrew Lee em 2011 para ser uma plataforma de desenvolvimento de aplicativos para dispositivos móveis e Web. Em 2014 a plataforma foi adquirida pela *Google*, onde foi sendo aperfeiçoada. A partir de outubro de 2018, a plataforma *firebase* já possui 18 produtos usados por 1.5 milhões de aplicativos. Um dos serviços oferecidos pela *Google* através do *firebase* é o banco de dados em tempo real e um *back-end* como um serviço. É fornecida uma API aos desenvolvedores que permite que os dados do aplicativo sejam sincronizados entre os clientes e armazenados na nuvem do *firebase*. A *Google* fornece bibliotecas de cliente que permitem a integração com *Android* , *iOS* , *JavaScript* , *Java* , *Objective-C* , *Swift* e *Node.js*. Além da biblioteca o banco pode ser acessado por meio de uma API REST que utiliza o protocolo *Server-Sent Events*, que é uma API para criar conexões HTTP para receber notificações *push* de um servidor. O banco de dados conta com regras de segurança impostas pelo lado do servidor da empresa (FIREBASE, 2018).

A Figura 2.15 mostra o sistema de gerenciamento de banco de dados (SGBD) oferecido gratuitamente aos desenvolvedores. Com esse sistema pode ser feito todo gerenciamento do banco de dados assim como cuidados com segurança, autenticações, análises de uso e análise de finanças, caso seja usada a versão paga.

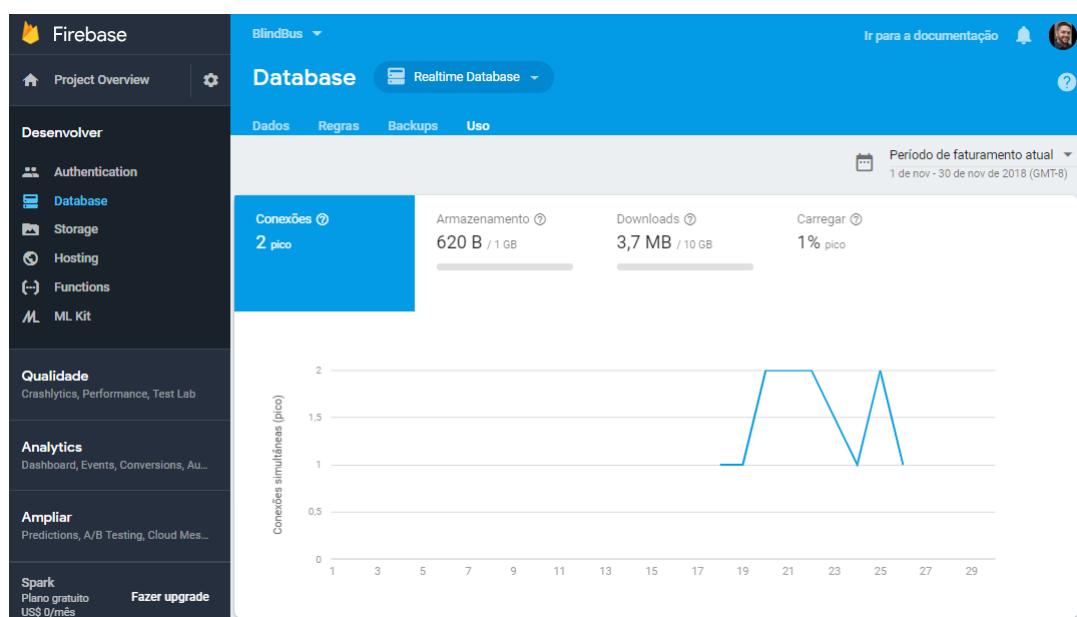


Figura 2.15: Sistema de gerenciamento de banco de dados (SGBD)

### 3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma análise de interfaces dos aplicativos mais relevantes ao projeto. A partir dessas análises foram identificadas características importantes e semelhanças à proposta.

#### 3.1 CittaMobi

CittaMobi é um aplicativo que foi desenvolvido para fornecer as previsões de chegada de ônibus em tempo real, nos pontos específicos, utilizando a sua localização atual, através de um *CHIP* de uma operadora que cubra a maior parte da cidade que se encontra.

A Figura 3.1 apresenta o ciclo do CittaMobi, cujo funcionamento ocorre da seguinte forma (CITTAMOBI, 2015):

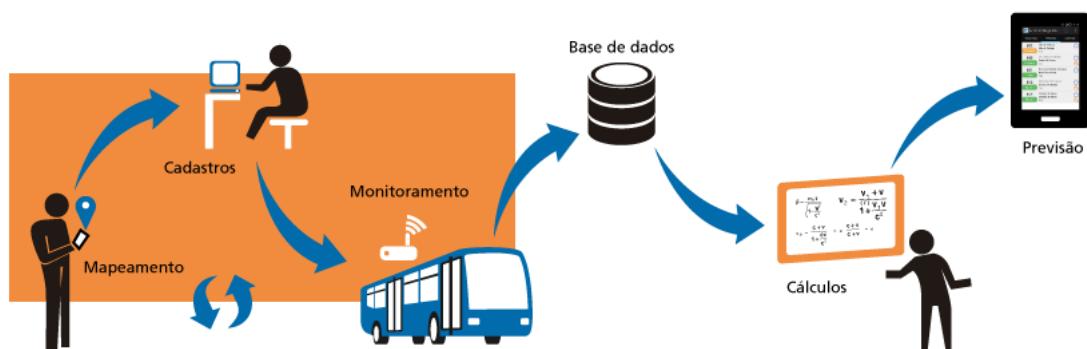


Figura 3.1: Ciclo do CittaMobi. Fonte: (CITTAMOBI, 2015)

1. Mapeamento - Tudo começa com o processo de mapeamento dos pontos de parada de ônibus de uma cidade, porém a disposição de pontos de parada é extremamente dinâmica, e estes frequentemente mudam de lugar. Por essa razão alguns pontos podem aparecer fora de lugar, se repetir, ou não aparecer.
2. Cadastros - Utilizando como base os pontos de parada mapeados, é necessário criar cadastros das linhas da cidade, quais empresas operam estas linhas, quais veículos estão em cada linha, e quais os horários programados de passagem dos veículos.
3. Monitoramento - Depois de tudo estar devidamente cadastrado, são instalados nos ônibus pequenos transmissores, de modo que sua posição captada por GPS possa

ser enviada constantemente por sinal de celular - utilizando a tecnologia GPRS - para os servidores.

4. Base de Dados - As informações dos veículos são recebidas pelos servidores e armazenadas por uma grande base de dados, e a partir desse momento ficam disponíveis para serem posteriormente utilizadas.
5. Cálculos - Com base nas informações disponíveis, são executados inúmeros cálculos considerando trânsito, posição dos veículos, linhas cadastradas, e outras variáveis, para gerar as previsões de chegada/saída dos veículos em cada ponto mapeado.
6. Previsões - Finalmente as previsões ficam disponíveis para serem visualizadas no CittaMobi.
7. Alterações - Quando ocorrem alterações de pontos de parada, linhas, veículos e horários, a base de dados é atualizada e as previsões são recalculadas. Devido à frequência em que essas alterações ocorrem no dia-a-dia, essas informações podem ficar desatualizadas por algum período se por algum motivo elas não forem enviadas para o CittaMobi, com isso gerando previsões imprecisas.

Outras funcionalidades que são oferecidas pelo aplicativo são:

- Pontos mais próximos – Mostra todos os pontos que estão próximos a uma determinada localização.
- Busca por ônibus adaptados para cadeirantes – filtra as previsões para que apareçam apenas as de ônibus adaptados.
- Pontos Favoritos – Permite identificar como favoritos pontos e linhas, para ter acesso rápido à previsão de chegada do ônibus esperado.
- Linhas Selecionadas - Seleciona previsões apenas das linhas que interessam ao usuário no ponto selecionado.

A partir deste aplicativo foi gerado uma versão voltada a deficientes visuais. Por sua vez, para se tornar um aplicativo audível é preciso que o usuário ative a função *Talback* do celular. Porém algumas funcionalidades do aplicativo não são executadas por intermédio da voz, se tornando difícil a interação do deficiente visual com a aplicação (LOPES, 2015).

A Figura 3.2 apresenta o menu inicial do aplicativo, onde observa-se as seguintes opções: Pontos próximos, Favoritos, Buscar por endereço, Ir para, Consultar Linhas, Configurar e Ajuda.

A Figura 3.3 apresenta as três primeiras opções do programa.

Na tela da esquerda são mostrados os pontos mais próximos e os ônibus que irão passar por ele. A tela do meio mostra a lista de favoritos e a tela à direita mostra a tela onde pode-se fazer uma busca por endereço.

As três próximas opções do programa são apresentadas na Figura 3.4.

A tela da esquerda mostra a opção de escolher para onde quer ir. Na tela do meio observa-se a seleção por linha desejada e, na tela à direita, encontram-se as opções do programa.

O aplicativo é disponível para Android 2.3.3 ou superior e requer acesso à internet.



Figura 3.2: Menu inicial do aplicativo CittaMobi



Figura 3.3: Telas de opções do CittaMobi (1)

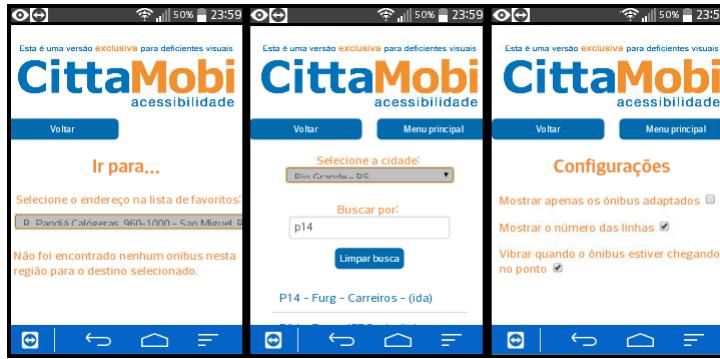


Figura 3.4: Telas de opções do CittaMobi (2)

### 3.2 Moovit

O aplicativo foi desenvolvido com o intuito de planejar viagens de transporte público, fornecendo a localização do ônibus desejado. Constantemente seu banco de dados é atualizado pelos horários cedidos pelas empresas de transporte público, informando também se o ponto está ativo e se o ônibus não alterou sua rota.

A Figura 3.5 apresenta todas as funcionalidades disponíveis pelo aplicativo, conforme descrito em (INC, 2017):

1. A ferramenta combina múltiplas fontes de dados com algoritmos avançados, analisa movimentos populacionais em torno da cidade e fornece *insights* visuais, todos a partir de dados agregados anônimos.



Figura 3.5: Ferramentas Moovit. Fonte: (INC, 2017)

2. Um sistema on-line que cria, distribui e analisa pesquisas relacionadas ao transporte público, que são respondidas por usuários não-incentivados e segmentados do aplicativo Moovit. Estas pesquisas são geradas em tempo real e fornecem visualização de resultados, bem como as respostas detalhadas e anonimizadas.
3. Uma ferramenta de gerenciamento de dados de transporte público com uma interface web fácil de usar permite criar, editar e gerenciar informações de trânsito público. Com base no padrão global GTFS, os dados das paradas, linhas, viagens, horários e mais, podem ser acessados e usados imediatamente por qualquer site externo ou aplicativo.
4. Sistema leve e fácil de integrar baseado na nuvem que monitora, exibe e analisa o local em tempo real de ônibus urbanos. Também permite a distribuição de informações de chegada de ônibus em tempo real para os passageiros através de qualquer site, aplicativo móvel ou sinais digitais conectados à API do Moovit.
5. Um serviço web acessado através de uma interface HTTP que permite solicitar direções de transporte público entre locais.

É possível também utilizar o *talback* para pessoas com deficiência visual, mas devido ao grande número de informações, o usuário tem dificuldades com a interação. Além de não transmitir com precisão as suas informações (LOPES, 2015).

A Figura 3.6 apresenta as três primeiras opções do aplicativo.

Na tela da esquerda mostra-se a opção de escolher o local que se deseja ir. A tela do meio mostra o ponto mais próximo e o tempo que irá levar para chegar até ele a pé e, à direita, encontra-se a tela de favoritos.

A Figura 3.7 apresenta as três próximas opções do aplicativo.

Na tela da esquerda observa-se a possibilidade de escolha da linha que se deseja pegar no ponto mais próximo além de escolher onde se deseja descer. Na tela do meio observa-se a tabela de todos os horários da linha selecionada e, na tela à direita, observa-se todas as linhas cadastradas no programa.

O aplicativo é disponível para Android, iPhone(iOS) e Windows Phone e requer acesso à internet.

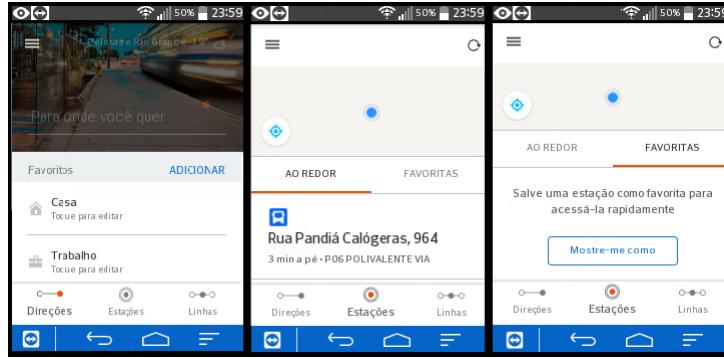


Figura 3.6: Telas de opções do Moovit (1)

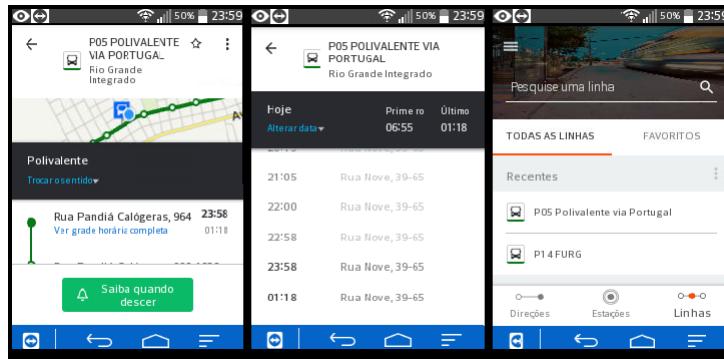


Figura 3.7: Telas de opções do Moovit (2)

### 3.3 Wayfinder

O aplicativo foi desenvolvido e usado em terminais de transportes ferroviários já equipados com a tecnologia *beacons*. Ele foi criado especialmente para deficientes visuais para facilitar a locomoção dos mesmos pelos metrôs em Londres. Para permitir que o usuário continue ouvindo os sons ao redor, e também as instruções do aplicativo, foram projetados fones de ouvidos especiais que emitem vibração mecânica através dos ossos intracranianos. Além disso o aplicativo usa os dados de localização passados pelos *beacons* via *Bluetooth* para localizar onde está a pessoa e ajudá-la a ir ao seu destino final (LOPES, 2015).

A Figura 3.8 apresenta as três principais telas do aplicativo.

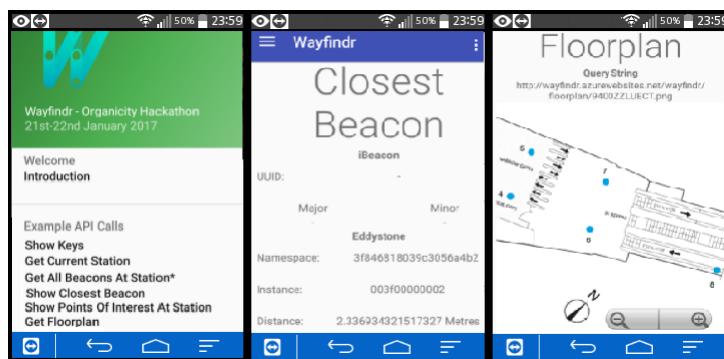


Figura 3.8: Telas de opções do Wayfinder

Na tela da esquerda encontra-se o menu inicial, onde o usuário pode escolher saber a

estação atual, atualizar todos pontos que tem *beacon* na estação, bem como mostrar qual é o mais próximo e todos os pontos de interesse na estação. Na tela do meio observa-se qual é o *beacon* mais próximo e suas configurações, e na tela à direita observa-se a planta da estação e os locais onde estão os *beacons*.

### 3.4 TagPoint

É um aplicativo que envia mensagens para *smartphones* e *tablets* que passam próximos aos *Beacons*, chegando a ter um raio de aproximadamente 70m de alcance. Os *beacons* são capazes de interagir com certos objetos físicos específicos, como pontos de ônibus, cancelas de estacionamento, máquinas de vendas, entre outras, para obter informações diretamente na tela do *smartphone* ou *tablet*, sem a necessidade de um APP para funcionar. O usuário vai receber informações direto no dispositivo móvel, dados como preços, horários, localizações e muito mais. Os aplicativos têm uma opção chamada "Acessibilidade", que faz a leitura de todas as mensagens que o *smartphone* recebe, porém nem se compara com o *TalkBack* nativo dos sistemas operacionais, entretanto os aplicativos estão adequados para o *TalkBack* (SMART, 2017).

A Figura 3.9 apresenta o funcionamento do aplicativo TagPoint para deficientes visuais na estátua do buda instalada no Parque Farroupilha em Porto Alegre. O aplicativo passa as características da estátua assim como sua localização.



Figura 3.9: Estátua do Buda

O Aplicativo está disponível para iOS superior ao 8.0 e Android apartir da versão 4.3 e é compatível com smartphones e Tablets, além de ser totalmente gratuito.

### 3.5 Comparação entre os aplicativos

A tabela 3.1 apresenta um comparativo entre as principais características dos aplicativos estudados e o proposto neste trabalho.

Tabela 3.1: Comparando os aplicativos

Projeto	CittaMobi	Moovit	Wayfinder	TagPoint
Natureza	Aplicativo	Aplicativo	Aplicativo	Aplicativo
Objetivo	Fornecer as previsões de chegada de ônibus em tempo real, nos pontos específicos, utilizando a sua localização atual, através de um chip de uma operadora que cubra a maior parte da cidade que se encontra.	Planejar viagens de transporte público, fornecendo a localização do ônibus desejado.	Foi criado especialmente para deficientes visuais para facilitar a locomoção dos mesmos pelos metrôs em Londres. Além disso o aplicativo usa os dados de localização passados pelos <i>beacons</i> via <i>Bluetooth</i> para localizar onde está a pessoa e ajudá-la a ir ao seu destino final.	Plataforma que envia mensagens para smartphones e tablets que passam próximos aos <i>Beacons</i> . Os <i>beacons</i> são capazes de interagir com certos objetos físicos específicos, como pontos de ônibus, cancelas de estacionamento, máquinas de vendas, entre outras.
Aplicação	São Paulo; Espírito Santo; Minas Gerais; Alagoas; Pernambuco; Acre; Rio Grande do Sul; Bahia; Paraíba; Rio de Janeiro.	Estados Unidos; Inglaterra; Espanha; Itália; França; Polônia; Suécia; Rússia; Holanda; Israel; Chile; Colômbia; Cidade do México; Peru; Nova Zelândia; Filipinas; Grécia; Coréia do Sul; Noruega; Brasil.	Metrô de Londres.	(Não disponível no site da empresa)
Tecnologias Utilizadas	smartphone, Chip SIM	smartphone	smartphone, Beacon	smartphone, Beacon e Tablets
Sistemas Utilizados	Android 4.0 ou superior e iOS	Android, Windows Fone e iOS	Android e iOS	Android 4.3 ou superior e iOS 8.0 ou superior

## 4 DESCRIÇÃO E MODELAGEM

O presente trabalho propõe o BlindBus, uma tecnologia assistiva para auxiliar a mobilidade dos deficientes visuais, auxiliando-os no uso do transporte público. Ele é composto por dois aplicativos e um *Beacon* que transmite informações. A Figura 4.1 mostra a ideia geral do sistema, onde o dispositivo *Beacon* fica alocado na parte interna do ônibus e o aplicativo “BlindBus Admin” é utilizado pelo motorista e o aplicativo “BlindBus User” é utilizado pelo deficiente visual que se encontra na parada de ônibus.

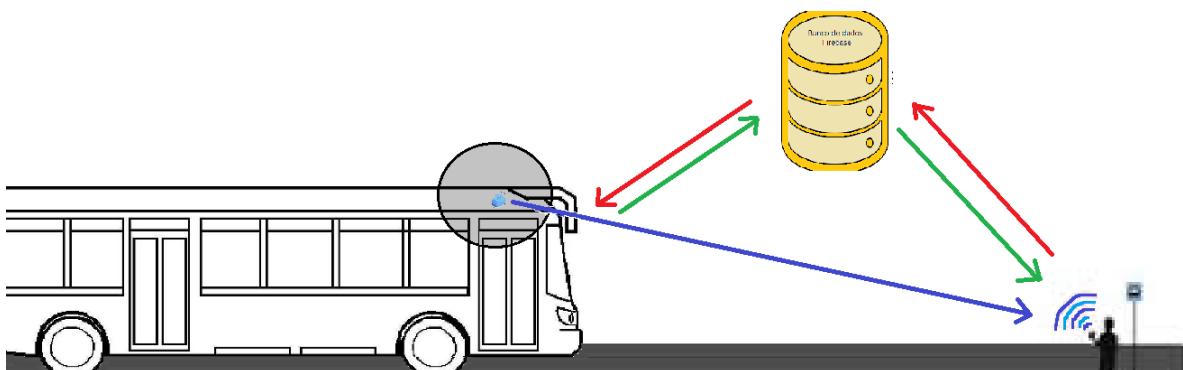


Figura 4.1: Funcionamento do sistema

O “BlindBus Admin” é responsável pela entrada e armazenamento de dados. Por intermédio deste aplicativo, também é feita a transmissão da linha desejada para o banco de dados externo. O segundo aplicativo, “BlindBus User”, quando ativado, fica aguardando até que algum ônibus se aproxime para receber as informações do *beacon* e consultar no banco externo, recebendo informações da viagem e transmitindo-as para o usuário.

Em relação aos aplicativos apresentados na Tabela 3.1, o “BlindBus” é por natureza aplicativo para celular, utiliza as tecnologias *smartphone* e *beacon* e sua aplicação foi projetada para a cidade de Rio Grande. Em relação aos demais, ele apresenta uma maior precisão na comunicação entre o ônibus e o usuário, pois utiliza a tecnologia *BLE* enquanto as demais utilizam *CHIP*, com exceção da *Wayfinder* que, apesar de também utilizar *BLE*, foi desenvolvida para metrôs com aplicação em Londres.

### 4.1 Diagrama de Casos de Uso

A Figura 4.2 mostra o diagrama de caso de uso do “BlindBus Admin”, onde tem-se um ator chamado Motorista. Ele possui dois casos de uso. A ação gerenciar linhas e destino do ônibus referem-se ao gerenciamento da aplicação do motorista.

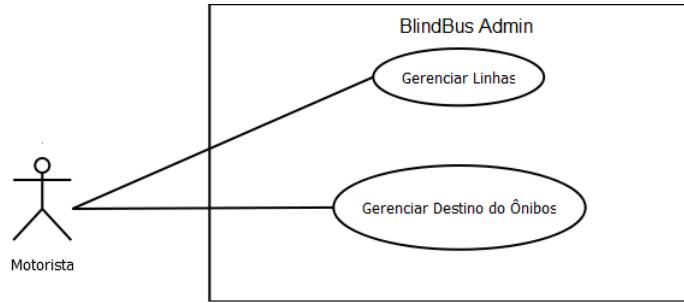


Figura 4.2: Diagrama Caso de Uso BlindBus Admin

A Figura 4.3 mostra o diagrama de caso de uso do “BlindBus User”, onde tem-se apenas um ator, e seu respectivo caso de uso. Neste caso indica que o sistema irá emitir um alerta para o usuário.

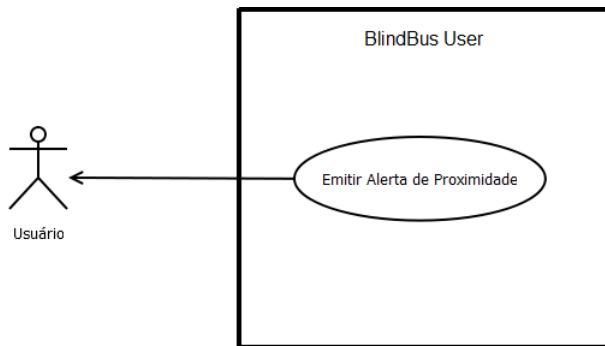


Figura 4.3: Diagrama Caso de Uso BlindBus User

## 4.2 Diagrama de Classes

A Figura 4.4 apresenta o diagrama de classes do “BlindBus Admin”.

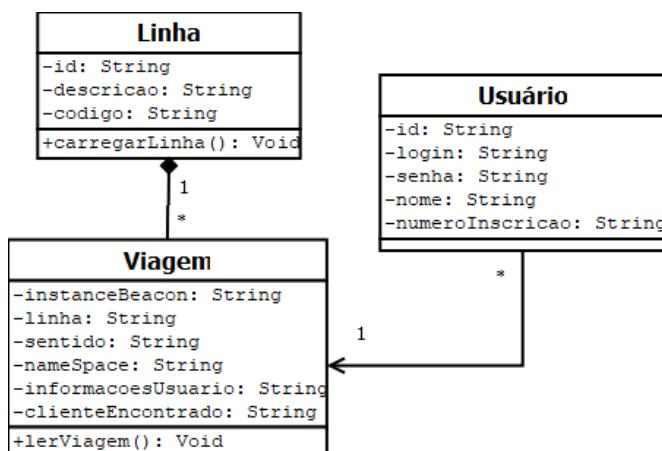


Figura 4.4: Diagrama de Classes

Nesse diagrama observam-se as três classes que compõem o modelo: Usuário, Linha e Viagem. A classe Linha que é responsável por carregar todas as linhas; Viagem, que está associada a uma linha e lê as informações quando houver alteração no servidor.

## 4.3 Telas dos aplicativos

A Figura 4.5 mostra que, ao entrar na aplicação do motorista, é preciso fazer a autenticação do mesmo. Se os dados de *login* e senha estiverem corretos ele passa para a tela de configuração, onde tem-se duas escolhas: “CONFIGURAR LINHA” e “CONFIGURAR VIAGEM”. Ao entrar na tela de configuração de linha o motorista tem opção de clicar na linha para editá-la ou clicar em “CADASTRAR” para inserir uma nova linha. Ambas levam para a tela de editar linha com comportamentos diferentes dependendo da escolha do usuário na tela anterior.

Após esse cadastramento o motorista pode voltar à tela de configuração e escolher a “CONFIGURAR VIAGEM”. Ao entrar na tela de configurar viagem ele seleciona a linha e o sentido que o ônibus vai cumprir e confirma. Após a confirmação ele vai para a tela de transmissão de viagem, onde transmite para o banco de dados as informações escolhidas pelo motorista e fica aguardando a identificação do deficiente visual.

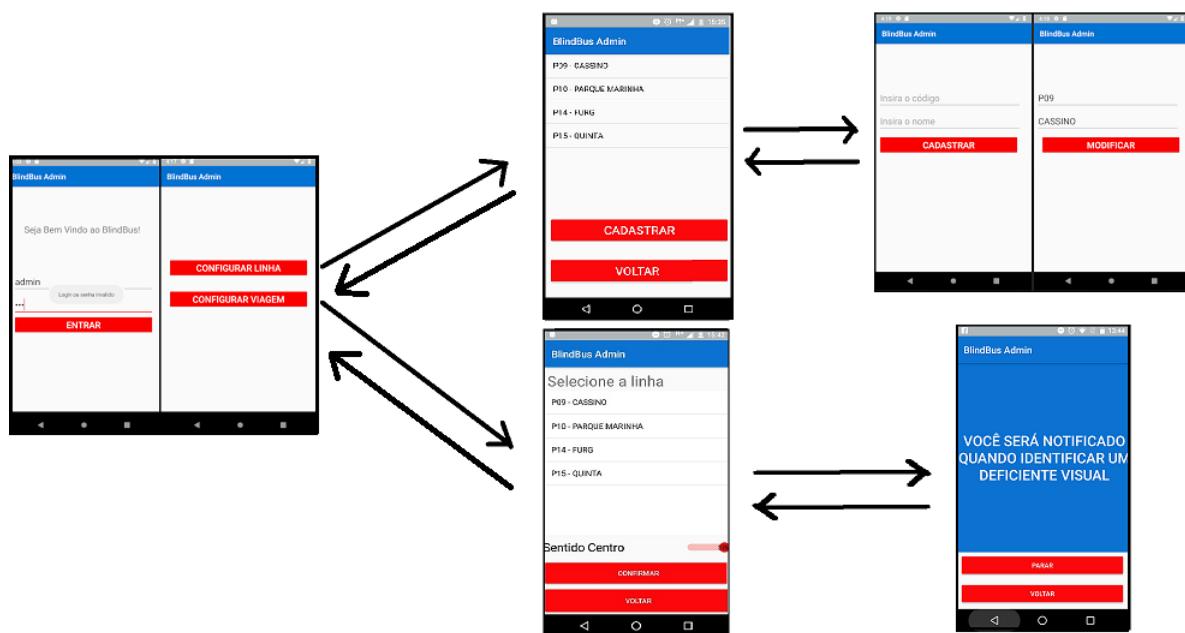


Figura 4.5: Fluxo da aplicação do motorista

A Figura 4.6 mostra a única tela do aplicativo que é utilizado pelos deficientes visuais. Após clicar o botão “Procurar”, a mesma fica aguardando a aproximação de um ônibus, e informa ao usuário (através de mensagem sonora), quando o mesmo se aproximar.



Figura 4.6: Tela da aplicação do deficiente visual

## 5 BLINDBUS: IMPLEMENTAÇÃO E TESTES

Este capítulo destina-se a apresentar as aplicações móveis desenvolvidas para o sistema *Android*, organizando-se da seguinte forma: Descrição do aplicativo “BlindBus Admin” destinado ao motorista do ônibus; Descrição do aplicativo “BlindBus User”, destinado ao usuário final; Comunicação com o banco de dados e Comunicação com *beacon*; Requisitos das aplicações. Ao final são descritos os testes realizados e avaliação.

### 5.1 BlindBus Admin

Foi desenvolvida uma aplicação móvel, para o sistema *Android* responsável por atribuir junto ao banco de dados uma linha e um sentido ao *beacon*. A Figura 5.1 apresenta a estrutura do projeto para o aplicativo “BlindBus Admin”, contendo três pacotes (activity, modelo e persistencia ).

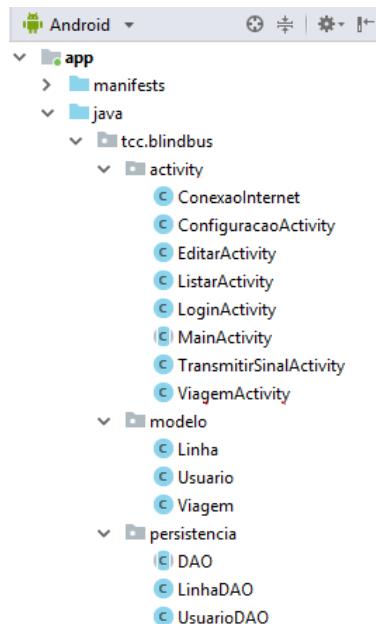


Figura 5.1: Estrutura do aplicativo “BlindBus Admin”

Ao abrir a aplicação, a *activity LoginActivity* é chamada. Ela inicializa a comunicação com o *Firebase* e busca todos os Usuários que existem no banco. A Figura 5.2 mostra que é feita uma repetição para pegar cada filho da chave “Usuario” e passá-lo para um objeto *usuario*. Dentro desse *for* é chamada a função *inserirUsuario* que recebe por parâmetros os valores vindos da chave “Usuario” e é responsável por inserir

um novo usuário com esses parâmetros na tabela “usuario” no banco de dados *SQLLite*.

```

    Classe LoginActivity
    package tcc.blindbus.activity;
    databaseReference.child("Usuario").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            for (DataSnapshot objSnapshot: dataSnapshot.getChildren()){
                usuario = objSnapshot.getValue(Usuario.class);
                daoUsuario.inserirUsuario(new Usuario(usuario.getNome(),usuario.getLogin(),
                    usuario.getSenha(),usuario.getNumeroInscricao()));
                usuario = new Usuario();
            }
        }
    });

    Classe UsuarioDAO
    package tcc.blindbus.persistencia;
    public void inserirUsuario(Usuario usuario)
    {
        ContentValues values = new ContentValues();
        values.put("nome",usuario.getNome());
        values.put("login",usuario.getLogin());
        values.put("senha",usuario.getSenha());
        values.put("numeroInscricao",usuario.getNumeroInscricao());
        inserir(values);
    }

    Classe DAO
    protected void inserir(ContentValues values) {
        getWritableDatabase().insert(getTableName(), nullColumnHack: null,values);
    }

```

Figura 5.2: Carregar e Inserir Usuários

No lado esquerdo da Figura 5.3 é mostrado que, para entrar no menu inicial, primeiro deve ser feito o *login*. Após o motorista digitar o *login* e senha e apertar no botão “ENTRAR” é feita uma consulta na tabela “usuario” no banco local para verificar se esse usuário existe, caso o retorno seja verdadeiro o usuário é encaminhado para a tela de configurações. Caso o retorno seja falso, ele receberá uma mensagem informando que o usuário ou senha estão inválidos, permitindo que ele faça a alteração dos campos para que seja feita a verificação novamente.

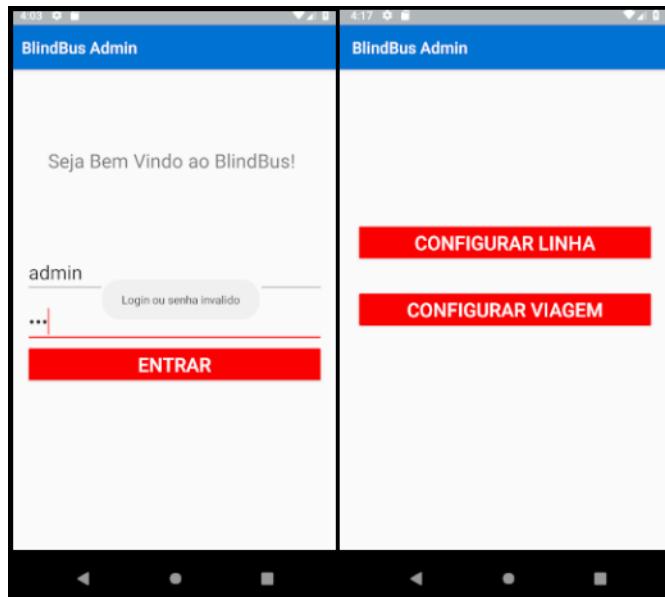


Figura 5.3: Tela de *Login* e Tela de Configurações

No lado direito da Figura 5.3 é mostrada a tela de configuração, onde existem duas opções: “Configurar linha” e “configurar viagem”. Ao clicar no botão “CONFIGURAR

LINHA”, é feito o direcionamento para `textitactivity Listaractivity`, que por sua vez quando acessada chama a função `onResume()` que é responsável por chamar a função `carregarLinha()` antes que a tela seja montada.

A Figura 5.4 mostra a tela configuração de linha com cada evento que pode ser gerado através dela. Clicando diretamente em uma linha é disparado o evento `onItemClick()` que recebe por parâmetros um `AdapterView` que intermedia a criação da lista, a `View`, a posição que a linha se encontra e o `id` que é o identificador de cada linha. Esse evento é responsável por enviar para a `activity Editaractivity` a linha que o motorista deseja editar. Outra maneira de ir para a `activity Editaractivity` é clicando no botão de “CADASTRAR”, que vai passar somente a `View` por parâmetro. E ao clicar no botão “VOLTAR”, o usuário será redirecionado a `activity ConfiguraçãoActivity`.

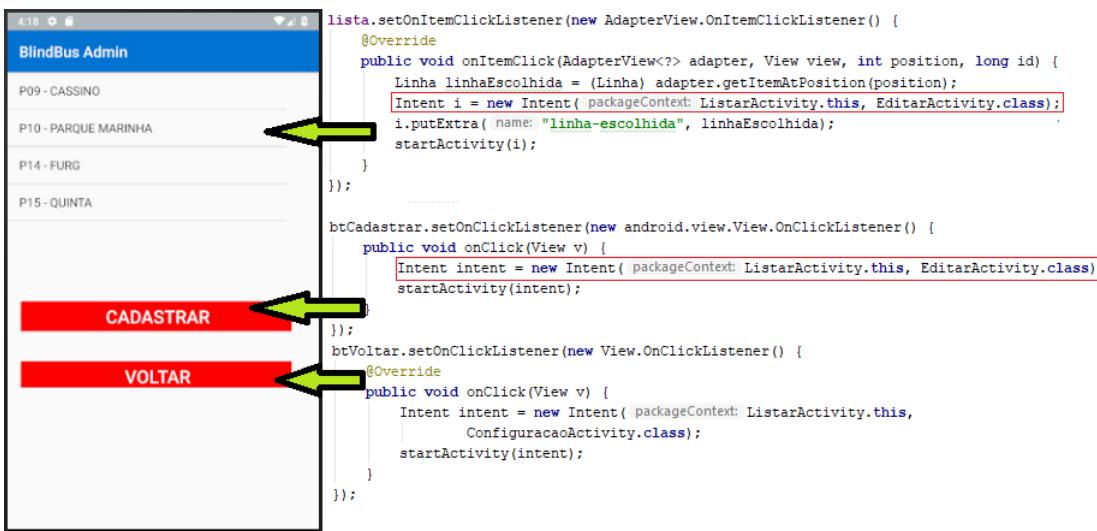


Figura 5.4: Configurações de Linhas

A Figura 5.5 mostra a tela de editar. Essa tela tem dois comportamentos diferentes: caso o `id` que foi passado para a `activity EditarActivity` seja igual a zero, os campos vêm vazios e o texto do botão como “CADASTRAR”, caso seja diferente de zero os campos vêm populados com suas respectivas informações e o texto do botão vem como “MODIFICAR”. Os dois campos aceitam `String` e os valores cadastrados ou editados, são alterados apenas no banco local, para que não haja perigo do motorista cadastrar dados não adequados no servidor. O banco local é apagado toda vez que o motorista sai do aplicativo. Com isso essa linhas se tornam temporárias.

Ao clicar no botão “CONFIGURAR VIAGEM” na tela de configuração, é feito o direcionamento para `activity ViagemActivity`, que por sua vez quando acessada chama a função `onResume()` responsável por chamar a função `carregarLinha()` antes que a tela seja montada.

A Figura 5.6 mostra como é adicionada a viagem no servidor. Ao selecionar uma linha, e informar no `Switch` o sentido que o ônibus vai ir, o motorista clica no botão “CONFIRMAR” onde vai acionar o evento `onClick()` passando por parâmetro a `View`. Dentro dele é verificado se o `Switch` está como verdadeiro ou falso. Se estiver verdadeiro o atributo “sentido” do objeto “viagemSelecionada” que foi inicializado no início dessa `activity` receberá “CENTRO”, se for falso receberá “BAIRRO”. O atributo “InformacoesUsuario” vai receber o número de inscrição e o nome do motorista logado. Após é

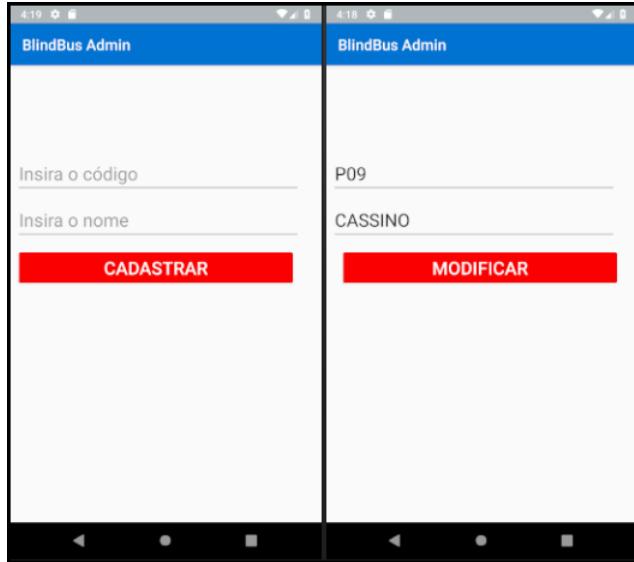


Figura 5.5: Cadastro e edição de Linhas

criada uma nova viagem onde o *Instance* e o *NameSpace* do *beacon* são fixos, pois cada *smartphone* e *beacon* são fixos do carro. Além desses atributos é passado a linha selecionada, o sentido escolhido, as informações do usuário logado e que essa viagem não encontrou nenhum deficiente visual. Após a nova viagem ser populada é verificado no servidor se tem alguma outra viagem com essa chave, e se houver é excluída, após é adicionada uma nova viagem no servidor e vai para a *activity* *TransmitirSinalActivity*.



Figura 5.6: Envia dados ao Servidor

A Figura 5.7 mostra a tela onde será informado ao motorista quando houver um deficiente visual próximo. Nessa *activity* tem a função *onDataChange()* que fica aguardando alguma alteração no banco *firebase* na chave *instanceBeacon* utilizado. Quando detectada a alteração do “filho” *clienteEncontrado* de “NAO” para “SIM” são chamadas as funções *alert()* que dá a mensagem textual de “Deficiente identificado” na tela, a função *editarViagem()* que é responsável de atualizar o filho *clienteEncontrado* no banco *firebase* para “NAO”, a função *playSound()* que lê um arquivo

*mp3* colocado na pasta *raw/res* com a seguinte frase: “Deficiente visual identificado” (informado oralmente ao motorista) e a função *piscar()*, que faz a tela piscar até o botão parar ser pressionado. Após pressionado, ele pára a execução do código por trinta segundos. Se dentro desse tempo o *smartphone* do passageiro não estiver mais transmitindo para o servidor ou não estiver mais no alcance, a função *onDataChange()* fica aguardando o próximo sinal. Caso o sinal ainda esteja ativo é feita toda operação até que não tenha mais nenhum deficiente com aplicativo próximo.



Figura 5.7: Aguadando retorno do banco

O aplicativo é encerrado voltando no próprio botão de voltar do *smartphone*.

## 5.2 BlindBus User

A aplicação direcionada ao deficiente visual, foi desenvolvida com o intuito de facilitar a comunicação entre ele e o *smartphone*

Utilizando a função *talkback*, o usuário dá um clique simples no celular que emite uma mensagem sonora informando o nome do aplicativo selecionado e, com clique duplo, ele abre o aplicativo, como está ilustrado na Figura 5.8.



Figura 5.8: Entrando no Aplicativo

A Figura 5.9 mostra a única tela do aplicativo que é utilizado pelos usuários. Logo que entra no aplicativo uma mensagem sonora é lançada “Aperte lado esquerdo para procurar, ou lado direito para parar”.



Figura 5.9: Aplicativo para deficiente.

Após o usuário apertar o botão de procurar são chamadas as funções de permissões de: localização, internet e *bluetooth*. Assim que essas permissões são concedidas é chamada a função que fica escutando o sinal do **beacon**. Quando o ônibus chega no raio de abrangência do **beacon**, a uma determinada distância do usuário que está utilizando o aplicativo, ele envia o *instance* e o *namespace* via *bluetooth* para o *smartphone*, conforme mostra a Figura 5.10.

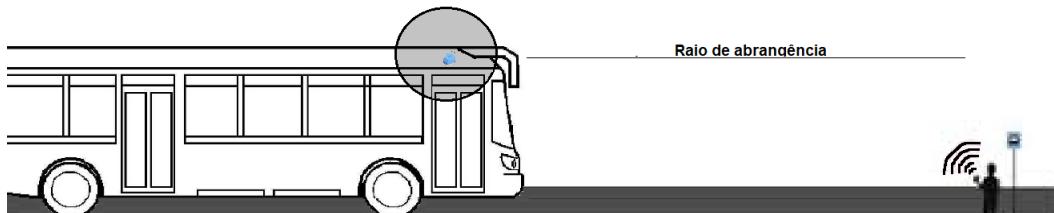


Figura 5.10: Comunicação entre ônibus e *smartphone*

O aparelho, ao receber essas informações, inicializa uma conexão com o servidor e faz uma consulta, verificando se tem alguma chave com *instance* igual ao recebido ele busca as informações da linha e sentido e é informado para o usuário através do *talkback*. Se houver aproximação de mais de um ônibus as pesquisas são feitas conforme a fila. O usuário tem a opção de continuar com a procura ou parar. Se ele continuar com a procura o ciclo vai se repetir até que o ônibus saia do raio de abrangência.

### 5.3 Comunicação

A comunicação é o fator mais importante para esse trabalho, visando que é preciso ter uma boa comunicação tanto de banco de dados quanto de internet e *bluetooth*, pois

o tempo de resposta que o usuário terá para tomar uma decisão depende exclusivamente desses três tipos de comunicação.

### 5.3.1 Internet

Devido ao fato de as aplicações terem acesso a um banco de dados externo, as duas necessitam de comunicação com a internet. Para que isso ocorra é necessário adicionar `<uses-permission android:name="android.permission.INTERNET"/>` `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>` no arquivo “AndroidManifest.XML” para que os projetos reconheçam que vão demandar de acesso externo.

A Figura 5.11 mostra que na *activity* MainActivity tem uma função (`verificaConexao()`) que verifica se o *smartphone* está com conexão a internet. Essa função é chamada ao clique do botão “LOGAR” na tela de login e do botão “CONFIRMAR” na tela de configuração de viagem. Além desses lugares ela é chamada dentro de uma função de monitoramento do status de internet que fica na *activity* TransmitirViagemActivity. Essa função chama a classe ConexaoInternet (Figura 5.12) e, caso deixe de identificar o acesso à internet, o usuário é notificado.

```
protected boolean verificaConexao() {
    boolean conectado;
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    if (connectivityManager.getActiveNetworkInfo() != null
        && connectivityManager.getActiveNetworkInfo().isAvailable()
        && connectivityManager.getActiveNetworkInfo().isConnected()) {
        conectado = true;
    } else {
        conectado = false;
    }
    return conectado;
}
```

Figura 5.11: Função que verifica se possui internet

### 5.3.2 Serviço externo

A Figura 5.13 mostra como é utilizado o banco de dados *firebase*. Para vincular a linha ao *beacon* tem-se uma chave no banco chamada “Viagem”. Através dessa chave é possível adicionar sub-chaves. Cada sub-chave é identificada pelo *instance* de cada *beacon*, para que seja possível fazer pesquisas, alterações e exclusões através destes. Quando o motorista adiciona uma nova viagem pela sua aplicação, ele envia os dados da mesma para a chave “Viagem” no *firebase*.

Cada ônibus terá um *smartphone* e um *beacon*. Na aplicação desse *smartphone* já vai ter definido o *instance* do seu respectivo *beacon*. Sempre que uma viagem é iniciada pela aplicação do motorista, a mesma exclui a última sub-chave com aquele identificador e adiciona outra com os parâmetros novos. Um dos parâmetros é chamado de “clienteEncontrado” que é setado com o valor “NAO”.

Quando o *smartphone* do deficiente visual localizar um *beacon* se aproximando ele faz uma pesquisa do *intance* vindo do *beacon* no banco de dados. Se localizar, o valor da variável “clienteEncontrado” passa a ser “SIM”. Para que haja conexão são instalados nos dois aplicativos os pacotes do *firebase* nativos do android studio. Foi necessário criar uma conta no *firebase*, onde foi montada a estrutura e baixado o JSON de configuração que encontra-se na Figura 5.14. Essa estrutura foi colocada nas duas aplicações para que elas apontem para o mesmo banco e com as mesmas permissões. Sobre as permissões, podem ser alteradas no SGBD na aba regras.

```

public class ConexaoInternet extends BroadcastReceiver {
    public enum TipoConexao {TIPO_NAO_CONECTADO, TIPO_WIFI, TIPO_MOBILE};
    private TipoConexao tipoConexaoATUAL = TipoConexao.TIPO_NAO_CONECTADO;
    private TipoConexao getStatusConexao(Context context) {
        synchronized (tipoConexaoATUAL) {
            ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
            tipoConexaoATUAL = TipoConexao.TIPO_NAO_CONECTADO;
            NetworkInfo activeNetwork = cm==null ? null : cm.getActiveNetworkInfo();
            if (null != activeNetwork) {
                if (activeNetwork.getType() == ConnectivityManager.TYPE_WIFI) tipoConexaoATUAL = TipoConexao.TIPO_WIFI;
                if (activeNetwork.getType() == ConnectivityManager.TYPE_MOBILE) tipoConexaoATUAL = TipoConexao.TIPO_MOBILE;
            }
        }
        return tipoConexaoATUAL;
    }
}
public interface IOnMudarEstadoConexao{ void onMudar(TipoConexao tipoConexao);}
private ArrayList<IOnMudarEstadoConexao> onMudarEstadoConexoesListeners = new ArrayList<>();
public void addOnMudarEstadoConexao(IOnMudarEstadoConexao t){onMudarEstadoConexoesListeners.add(t);}
@Override
public void onReceive(Context context, Intent intent) {
    TipoConexao tipo = getStatusConexao(context);
    for(IOnMudarEstadoConexao o : onMudarEstadoConexoesListeners){
        o.onMudar(tipo);
    }
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
        ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
        if(connectivityManager!=null){
            for (Network net : connectivityManager.getAllNetworks()) {
                NetworkInfo networkInfo = connectivityManager.getNetworkInfo(net);
                if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI) {
                    connectivityManager.bindProcessToNetwork(net);
                    break;
                }
            }
        }
    }
}
}

```

Figura 5.12: Classe de conexão à internet

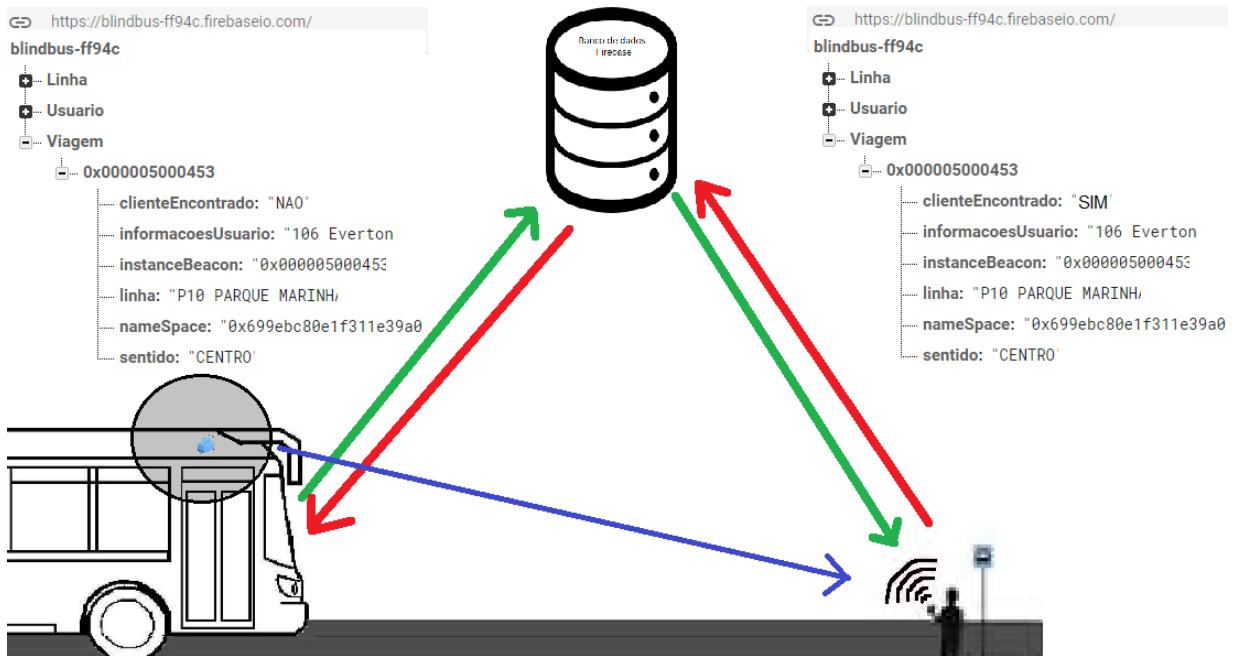


Figura 5.13: Funcionamento do serviço externo

```
{
  "project_info": {
    "project_number": "269319451268",
    "firebase_url": "https://blindbus-ff94c.firebaseio.com",
    "project_id": "blindbus-ff94c",
    "storage_bucket": "blindbus-ff94c.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:269319451268:android:5b4b1d3534dab624",
        "android_client_info": {
          "package_name": "tcc.blindbus"
        }
      },
      "oauth_client": [
        {
          "client_id": "269319451268-bhci2dq5sjqvobc3177g30giv3omnspd.apps.googleusercontent.com",
          "client_type": 1,
          "android_info": {
            "package_name": "tcc.blindbus",
            "certificate_hash": "a48382407c4162ff6bec8d9dd1c5a817fe40d23"
          }
        },
        {
          "client_id": "269319451268-9b5c23tufc0i2qajv86e1nck1pkkv8kq.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyCLO941G2e5SKkczDnCfIsCe6IbPQpM2FM"
        }
      ],
      "services": {
        "analytics_service": {
          "status": 1
        },
        "appinvite_service": {
          "status": 2,
          "other_platform_oauth_client": [
            {
              "client_id": "269319451268-9b5c23tufc0i2qajv86e1nck1pkkv8kq.apps.googleusercontent.com",
              "client_type": 3
            }
          ]
        },
        "ads_service": {
          "status": 2
        }
      }
    ],
    "configuration_version": "1"
  }
}
```

Figura 5.14: Arquivo JSON de configuração do *Firebase*

### 5.3.3 Beacon

Para a utilização do *beacon*, é sincronizado o pacote `org.altbeacon.beacon.*` na *activity* `MainActivity`. Após implementada a biblioteca `BeaconConsumer` à *activity*. Cinco variáveis para utilização do *beacon* foram inicializadas no começo da classe conforme mostra a Figura 5.15.

```

import org.altbeacon.beacon.Beacon;
import org.altbeacon.beacon.BeaconConsumer;
import org.altbeacon.beacon.BeaconManager;
import org.altbeacon.beacon.BeaconParser;
import org.altbeacon.beacon.Identifier;
import org.altbeacon.beacon.RangeNotifier;
import org.altbeacon.beacon.Region;
public class MainActivity extends AppCompatActivity implements View.OnClickListener, BeaconConsumer, RangeNotifier {
    private static final int REQUEST_ENABLE_BLUETOOTH = 1;
    private static final long DEFAULT_SCAN_PERIOD_MS = 1000;
    private static final String ALL_BEACONS_REGION = "AllBeaconsRegion";
    BeaconManager mBeaconManager;
    private Region mRegion;
}

```

Figura 5.15: Variáveis de inicialização para utilização do *beacon*

A primeira variável é responsável por habilitar a solicitação *bluetooth*. A segunda por determinar o tempo de cada escuta em milissegundos. A terceira por habilitar todos *beacons* que passarem na região. A quarta para interagir com *beacons* de uma entidade e a quinta representa os critérios dos campos com os quais procurar *beacons*.

Para que seja possível utilizar o protocolo “EDDYSTONE” foi utilizado o comando `mBeaconManager.getBeaconParsers().add( new BeaconParser() .setBeaconLayout(BeaconParser.EDDYSTONE_UID_LAYOUT) )` na função `onCreate()` da `MainActivity`. Esse comando permite que seja utilizado o protocolo “EDDYSTONE” para que possa ser feita a leitura de todos os tipos de *beacons*.

A Figura 5.16 mostra as funções que são utilizadas para utilização dos *beacons*, sendo a primeira responsável por iniciar a busca de *beacons* na região, a segunda por pegar todos *beacons* enfileirados e capturar seus identificadores e a terceira por finalizar a busca por *beacons*.

```

@Override
public void onBeaconServiceConnect() {
    try {
        mBeaconManager.startRangingBeaconsInRegion(mRegion);
        alert(getString(R.string.ligando_busca_de_beacons));
    } catch (RemoteException e) {
        Log.d(TAG, msg: "Ocorreu um erro ao começar a procurar ônibus " + e.getMessage());
    }
    mBeaconManager.addRangeNotifier(this);
}

@Override
public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region region) {
    for (Beacon beacon : beacons) {
        instanceBeacon = beacon.getId2().toString();
    }
}

private void stopDetectingBeacons() {
    try {
        mBeaconManager.stopMonitoringBeaconsInRegion(mRegion);
        alert(getString(R.string.desligando_busca_de_beacons));
    } catch (RemoteException e) {
        Log.d(TAG, msg: "Ocorreu um erro ao parar de procurar ônibus " + e.getMessage());
    }
    mBeaconManager.removeAllRangeNotifiers();
    mBeaconManager.unbind(consumer: this);
}

```

Figura 5.16: Funções para utilização do *beacon*

## 5.4 Requisitos das aplicações

Para utilização dessas aplicações é necessário que se utilizem *smartphones* com o sistema operacional *Android* 4.0 ou superior com conexão à internet e o dispositivo utilizado pelo deficiente deve ter *bluetooth* 4.0 ou superior.

Em relação ao *Beacon* pode ser utilizado qualquer modelo, independente do fabricante, com diferentes faixas de abrangência.

## 5.5 Testes e avaliação

Para testar essa solução foi utilizado um *smartphone* Motorola Moto G 2 com Android 4.4.4 e *bluetooth* 4.0, um *smartphone* Alcatel A3 XL com Android 7.0 e *bluetooth* 4.2 e um *beacon* BLE Bluetooth Rfid Ativa com protocolo *Eddystone - UID* e alcance máximo de 40 metros. Os dois *smartphones* utilizaram internet 4G.

As duas aplicações foram testadas nos dois *smartphones* e ambas apresentaram os mesmos resultados. Com o *beacon* ligado foram feitos testes de distância, e o mesmo respondeu à especificação do mesmo de responder dentro de 40 metros em campo aberto, porém o raio diminui quando utilizado em lugares fechados.

Ambas aplicações responderam ao que foi proposto nesse projeto, porém se fosse utilizado um *beacon* com protocolo *Eddystone - URL* poderia enviar uma mensagem de até 16 bytes para o usuário, isentando assim o uso de internet, pois os *beacons* com esse protocolo aceitam o armazenamento de mensagens.

Foram feitos cálculos considerando a velocidade máxima permitida para o transporte público na cidade de Rio Grande e a abrangência de dois tipos de *beacons*: o utilizado para este trabalho, que possui uma abrangência de 40 metros, e o *beacon* estimote, com abrangência de 100 metros. Considerando a velocidade de 80km/h para vias de trânsito rápido e 50Km/h para as demais vias, foi realizada a multiplicação da abrangência por 3600 (quantidade de segundos por hora) e dividida pela velocidade permitida na via (multiplicada por 1000 para passar para metros). Com isso foi obtido o resultado de resposta em segundos para que o deficiente sinalize se deseja embarcar no ônibus que se aproxima e o tempo de ação do motorista para parar na parada.

Conforme a Tabela 5.1 percebe-se que foi obtida uma melhora de 60% no tempo de resposta com o *beacon* estimote.

Tabela 5.1: Comparação entre *beacons* com diferentes abrangências

<b>Abrangência</b>	<b>Velocidade permitida em km/h</b>	
	<b>50km/h</b>	<b>80km/h</b>
<b>40m</b>	2,9 segundos	1,8 segundos
<b>100m</b>	7,2 segundos	4,5 segundos

A utilização do *talkback* é bem complexa para quem nunca utilizou, porém cumpre com o que foi proposto visto que o público alvo (deficientes visuais) está familiarizado com esta ferramenta.

A partir dos testes, conclui-se que a solução desenvolvida pode ajudar pessoas com deficiência visual no uso do transporte público sem depender da ajuda de outras pessoas, contribuindo para uma maior autonomia para esses usuários.

## 6 CONCLUSÕES

Com o aumento exponencial da internet das coisas, o grande crescimento no uso de equipamentos *smart*, e a criação da tecnologia *beacon*, permitiram abrir um novo ponto no que diz respeito à interação entre pessoas e equipamentos.

Sabemos que a falta de acessibilidade dificulta o convívio. E é a partir desses problemas e da própria conscientização de diversas pessoas que a área de Tecnologias Assistivas está crescendo junto com as novas tecnologias, para facilitar e derrubar as barreiras que as pessoas com deficiência possuem e promover maior autonomia para essas pessoas. Hoje, não só o deficiente visual, mas qualquer pessoa que tenha alguma deficiência, possui mais chances de ter uma vida normal, concorrer a uma vaga no mercado de trabalho ou em escolas e tudo isto pelos recursos que o mundo e suas novas tecnologias oferecem.

Este trabalho apresentou uma solução para o facilitar o deslocamento de pessoas com deficiência visual pela cidade através de transporte público. Esta proposta consiste no uso de dois aplicativos (Um para o motorista do ônibus e outro para o usuário) e um dispositivo *beacon* que fica localizado no interior do ônibus. A solução apresentada distingue-se de outras semelhantes devido à precisão que se tem na comunicação entre o *beacon* e o *smartphone*. Diferencia-se, ainda, por ser compatível com qualquer tipo de *beacon*, independentemente do fabricante, podendo assim se adequar aos melhores preços no mercado tendo diferentes tipos de marcas utilizando as mesmas aplicações.

Esta solução destina-se a ser utilizada de forma simultânea tanto da parte dos ônibus e motoristas quanto na parte do consumidor final, usuários.

As duas aplicações foram planejadas e desenvolvidas para que mesmo os usuários com pouca experiência com *smartphone* tenham facilidade de utilizá-las. A aplicação destinada ao deficiente visual foi desenvolvida levando-se em consideração aspectos de acessibilidade para deficientes visuais, apresentando apenas uma tela e dois botões para que facilite o uso da aplicação e ele se sinta motivado a utilizá-la. Em relação aos motoristas, com apenas algumas interações eles já podem fazer o cadastro das suas viagens e assim atender esses usuários com necessidades especiais de uma melhor maneira possível.

O projeto foi desenvolvido com o protocolo *Eddystone - UID* e por essa razão ele necessita de acesso a internet, diferentemente se fosse feito com *Eddystone - URL* que poderia enviar uma mensagem de até 16 bytes para o usuário isentando assim o uso de internet, possibilitando o aviso mais rápido ao usuário.

Acredita-se que essa solução possa ajudar novos programadores a entenderem e se interessarem pela área de TA e uso de dispositivos *beacons*, e juntamente com as novas tecnologias que estão surgindo, sejam capazes de desenvolver outras soluções capazes de facilitar a vida das pessoas.

## 6.1 Trabalhos futuros

A partir do presente trabalho, vislumbra-se a possibilidade de diversos outros trabalhos tanto no que diz respeito à melhoria e modificações deste, quanto a diferentes aplicações da ideia apresentada com o uso de *beacons*.

Para a melhoria e alterações nesta proposta, seria possível:

- Utilização de *beacons* com maior abrangência;
- Utilização de *beacon* com capacidade de armazenar mensagens de até 16 bytes, para que possa enviar essa mensagem para o usuário, isentando assim o uso de internet;
- Criação de sistema de ponto para os motorista através do aplicativo;
- Comunicação da empresa com motoristas via, aplicação;
- Desenvolvimento de um método para que o usuário possa escolher e cadastrar as linhas desejadas.

Diferentes aplicações dessa ideia podem estar associadas a:

- Identificação de lugares públicos e privados;
- Utilização em sistemas na área do transporte privado urbano (Táxis, Uber);
- Uso em visitas guiadas.
- Utilização em comandas de festas, bares e restaurantes.

## REFERÊNCIAS

ACCESS, N. **What is NVDA.** acessado em 15/11/2017, <https://www.nvaccess.org/>.

BERSCH, R. Introdução à tecnologia assistiva. **CEDI**, Porto Alegre, 2013.

BORGES, A. **Ferramentas do sistema DOSVOX.** acessado em 15/11/2017, <http://intervox.nce.ufrj.br/dosvox/ferramentas.htm>.

BRASIL, U. **Tudo o que você precisa saber para começar a brincar com iBeacons.** acessado em 28/10/2017, [https://brasil.uxdesign.cc/tudo-o-que-vocé-precisa-saber-para-começar-a-brincar-com-ibeacons-fdf5847e640b](https://brasil.uxdesign.cc/tudo-o-que-voc%C3%A9-precisa-saber-para-come%C3%A7ar-a-brincar-com-ibeacons-fdf5847e640b).

CARNEIRO, C. **IBeacon:** tudo que você precisa saber. acessado em 02/11/2017, <http://www.decom.ufop.br/imobilis/ibeacon-tudo-que-voce-precisa-saber/>.

CITTAMOBI. **CittaMobi.** acessado em 21/09/2017, <https://www.cittamobi.com.br/sobre?1>.

FIREBASE. **Firebase helps mobile app teams succeed.** acessado em 27/11/2018, <https://firebase.google.com/>.

IMPACTA. **O que são Beacons e como eles mudarão a sua rotina?** acessado em 28/10/2017, <http://www.impacta.com.br/blog/2014/12/09/o-que-sao-beacons-como-mudarao-rotina/>.

INC, M. **Moovit is the world's 1 transit app.** acessado em 08/10/2017, <https://www.solutions.moovitapp.com/portugues>.

JAVAVOX. **Projeto Dosvox.** acessado em 15/11/2017, <http://www.jogavox.nce.ufrj.br/conheca-o-projeto-dosvox>.

LOPES, B. G. A tecnologia como meio de inclusão dos deficientes visuais no transporte públicos. **Revista de Iniciação Científica, Tecnológica e Artística**, São Paulo, v.5, n.4, p.51, dezembro 2015.

MICROPOWER. **Virtual Vision.** acessado em 26/10/2017, <http://www.virtualvision.com.br>.

NETWORKS, R. **Android Beacon Library.** acessado em 27/11/2017, <https://altbeacon.github.io/android-beacon-library/>.

ONLINE, D. **Acessibilidade.** acessado em 29/10/2017, [http://www.deficienteonline.com.br/principais-adaptacoes-para-pessoas-com-deficiencia-visual\\_\\_10.html](http://www.deficienteonline.com.br/principais-adaptacoes-para-pessoas-com-deficiencia-visual__10.html).

PRATICA, A. na. **Deficiência visual:** formas de leitura e acessibilidade à informação. acessado em 30/11/2017, <http://www.acessibilidadenapratica.com.br/textos/deficiencia-visual-formas-de-leitura-e-acessibilidade-a-informacao/>.

RFID, T. S. **Qual a diferença entre Eddystone e iBeacon?** acessado em 29/10/2017, <https://taggen.zendesk.com/hc/pt-br/articles/115000402724-Qual-a-diferença-entre-Eddystone-e-iBeacon>.

SELETRONIC. **O que é Talkback, para que serve?** acessado em 26/10/2017, <https://seletronic.com.br/noticias/android/aplicativos-android/o-que-e-talkback-para-que-serv/>.

SENADO. **Constituição da República Federativa do Brasil - Art. 244.** acessado em 30/11/2017, [http://www.senado.gov.br/atividade/const/con1988/con1988\\_12.07.2016/art\\_244\\_.asp](http://www.senado.gov.br/atividade/const/con1988/con1988_12.07.2016/art_244_.asp).

SENADO. **Constituição da República Federativa do Brasil - Art. 227.** acessado em 30/11/2017, [http://www.senado.gov.br/atividade/const/con1988/con1988\\_06.06.2017/art\\_227\\_.asp](http://www.senado.gov.br/atividade/const/con1988/con1988_06.06.2017/art_227_.asp).

SITEINA. **JAWS Para Windows.** acessado em 26/10/2017, <http://licenciamentodesoftware.com.br/jaws-para-windows-software-para-acessibilidade-de-deficientes-visuais/>.

SMART, T. B. **TagPoint e Web Física:** ficou ainda mais fácil se conectar. acessado em 12/11/2017, <http://tagcity.com.br/tagpoint-e-e-web-fisica-ficou-ainda-mais-facil-se-conectar/>.

SQLITE. **Sobre o SQLite.** acessado em 27/11/2017, <http://www.sqlite.org/index.html>.

STUDIO, A. **Conheça o Android Studio.** acessado em 27/11/2017, <https://developer.android.com/studio/intro/index.html?hl=pt-br>.