CENTRO UNIVERSITÁRIO
Fundação Santo André

Arthur Cicuto Pires

Victor Vieira Paulino

Sistema de acompanhamento de transporte público para deficientes visuais

Santo André

2017

Arthur Cicuto Pires
Victor Vieira Paulino

Sistema de acompanhamento de transporte público para deficientes visuais

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Engenheiro Celso Daniel do Centro Universitário Fundação Santo André, como exigência parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador:
Prof. Dr. Marcos Forte

Santo André

2017

Agradecimentos

Ao nosso orientador Prof. Dr. Marcos Forte, pela paciência e suporte prestado durante o desenvolvimento deste projeto.

Ao Instituto de Cegos Padre Chico, pela hospitalidade demonstrada em nossa visita.

Ao Metractive, pelo incentivo e apoio ao uso de tecnologias.

Ao Raul Bertolo e à Caroline Grava pelo apoio proporcionado.

Às nossas famílias e amigos, pela compreensão e apoio durante esse ano.

Resumo

Segundo dados do censo do IBGE de 2010, 3,6% da população brasileira possui alguma deficiência visual. Deficiência esta que dificulta a execução de diversas atividades cotidianas, como, por exemplo, a locomoção no perímetro urbano usando transporte público. Com o objetivo de auxiliar a locomoção dessas pessoas com deficiência visual foi desenvolvido um sistema de escolha de linha e de previsão de chegada de ônibus, visto que o cenário atual não apresenta soluções práticas para esse público-alvo. Para compor o sistema proposto foram desenvolvidos: um aplicativo para smartphone baseados em Android; uma aplicação Web (API); além de módulos de hardware/software para as paradas e para os ônibus. A acessibilidade do aplicativo foi projetada detalhadamente, sendo implementado o feedback auditivo e por vibração, além da detecção automática do ponto onde o deficiente se localiza. A aplicação gera a lista de linhas de ônibus que passam pelo respectivo ponto, realizando os cálculos de distância e tempo, avisando ao usuário o tempo estimado de espera. O módulo do ônibus envia, em tempo real, a localização atual do ônibus e também avisa ao motorista que há um deficiente visual aguardando no próximo ponto. Por fim, após a implementação do sistema proposto, incluindo o desenvolvimento dos protótipos dos módulos do ponto e do ônibus, foram realizados testes para verificar se as funcionalidades definidas em projeto foram atendidas adequadamente.

Palavras-chave: Acessibilidade, deficientes visuais, smartphone, transporte público, aplicativo mobile.

Abstract

According to the IBGE, 3,6% of Brazil's population have some kind of visual impairment. This deficiency, hinder the execution of various everyday activities, like, for example, the usage of public transportation in the urban perimeter. With the objective to help the visual impaired to move around, it was developed a system to choose a bus line and give the bus arrival prediction, as the actual scene doesn't show practical solutions for the target audience. To compose the suggested system, it was developed: an application for Android based smartphones; a Web application (API); and hardware/software modules for the bus stops and the buses. The application accessibility was designed in detail, being implemented a feedback by audio and by vibration, besides the automatic detection of the bus stop where the visual impaired is. The application generates a list of the bus lines that go through that bus stop, doing the distance and arrival forecast calculations, notifying the user the estimated waiting time. The bus module sends, in real time, the bus actual location and warns the user that there is a visual impaired waiting in the next bus stop. Lastly, after the implementation of the system proposed, including the bus and bus stop modules prototypes development, were made some tests to see if the defined functionalities answer accordingly.

Keywords: Acessibility, visually impaired, smartphone, public transportation, mobile application.

Lista de ilustrações

Sumário

1 Introdução

Segundo o Censo Demográfico de 2010, realizado pelo IBGE - Instituto Brasileiro de Geografia e Estatística [IBGE, 2010], no Brasil, existe mais de 6,5 milhões de pessoas com alguma deficiência visual, representando cerca de 3,6% da população brasileira, sendo 500 mil pessoas declaradamente cegas (que não enxergam de modo algum) e seis milhões com grande dificuldade na visão.

A Fundação Dorina Nowill para cegos, [DORINA, 2017], descreve os níveis de deficiência visual da seguinte forma:

> A deficiência visual é definida como a perda total ou parcial, congênita ou adquirida, da visão. O nível de acuidade visual pode variar, o que determina dois grupos de deficiência:
>
> Cegueira: há perda total da visão ou pouquíssima capacidade de enxergar, o que leva a pessoa a necessitar do Sistema Braille como meio de leitura e escrita.
>
> Baixa visão ou visão subnormal: caracteriza-se pelo comprometimento do funcionamento visual dos olhos, mesmo após tratamento ou correção. As pessoas com baixa visão podem ler textos impressos ampliados ou com uso de recursos óticos especiais.

Como sugere Bersch [BERSCH, 2013], a tecnologia assistiva tem como objetivo dar maior independência para pessoas com alguma deficiência, aumentando a qualidade de vida através da realização de tarefas desejadas que antes tinham um grau de dificuldade ou eram impossibilitadas de fazer.

O desenvolvimento de recursos e outros elementos de tecnologia assistiva tem propiciado a valorização, integração e inclusão da pessoa com deficiência, promovendo seus direitos humanos. Por essa razão, o tema tem assumido um espaço importante nas ações desenvolvidas pela Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência e demais órgãos federais, como o Ministério da Saúde, Ministério da Educação e Ministério de Ciência e Tecnologia [SAÚDE, 2014].

Dentre os elementos da tecnologia assistiva pode-se utilizar smartphones, que tem se tornado cada vez mais presentes na vida das pessoas. Uma pesquisa realizada pelo FGV-SP em 2016, [MEIRELLES, 2016], demonstrou que o número de aparelhos chegou a 168 milhões só no

Brasil. Com sua facilidade de acesso, surgem inúmeras soluções que resolvem problemas do dia-a-dia dos usuários.

Dentre essas soluções, aplicações para smartphones que ajudam na mobilidade são cada vez mais comuns. Os aplicativos CittaMobi, [VIEIRA, 2015], e Moovit, [GOMES, 2015], vieram para mostrar que a tecnologia embarcada nos aparelhos podem ajudar a prever quanto tempo falta para o ônibus chegar em um ponto de parada, em tempo real. Eles capturam a geolocalização do usuário para saber qual ponto de ônibus eles estão próximos, possibilitando o usuário dizer de forma mais rápida qual seu ponto. Informando ao aplicativo qual seu ponto, eles podem selecionar um ônibus que passa no ponto selecionado, para saber quanto tempo resta para o veículo chegar.

Isso ajuda os usuários a se programar melhor, possibilitando a pessoa sair em um horário mais oportuno ou deixando ela mais tranquila sabendo que em breve seu ônibus chegará.

Para ajudar as pessoas com deficiência visual, a CittaMobi lançou um aplicativo, voltado para esse público, intitulado CittaMobi Acessibilidade, [Google Play, 2017]. No início de 2017, o aplicativo era baseado em listas, eliminando todas as interações com partes gráficas, como mapas e menus com animações. A nova versão do aplicativo, lançada durante o desenvolvimento deste trabalho, foi baseada em recursos de geolocalização, diminuindo o uso de listas e mostrando pontos próximos.

1.1 Descrição Geral

Com o objetivo de conhecer de perto as dificuldades que as pessoas com deficiência visual encontram para se locomover pelo perímetro urbano utilizando o transporte público, foi realizada uma visita técnica ao Instituto de Cegos Padre Chico [Instituto Padre Chico, 2017], localizado no bairro Ipiranga, na cidade de São Paulo. Durante a visita, foi notado que os principais pontos levantados por eles foram o medo de ficar com o smartphone em mãos e a vergonha de ter de parar todos os ônibus para saber qual a linha que o veículo atende.

Este trabalho visa facilitar a vida de pessoas com deficiência visual que utilizam ônibus como meio de transporte. O aplicativo proposto irá possibilitar ao deficiente visual saber quanto tempo falta para seu ônibus chegar, enquanto o sistema se encarrega de avisar o motorista do ônibus qual o próximo ponto onde terá um deficiente visual esperando por aquele ônibus.

Sistemas operacionais de smartphone, como Android e iOS, possuem ferramentas nativas que adaptam o uso de aplicativos para pessoas com deficiências, possibilitando a utilização do aparelho sem grandes dificuldades, mas, nem sempre, criam boas experiências de uso.

O Android possui a ferramenta *Talkback* [Google LLC, 2017], para auxiliar no uso de qualquer aplicativo. Ao desenvolver uma solução para o sistema, é possível colocar tags específicas em cada elemento da tela da sua aplicação. Isso possibilita ao Talkback ler a tela com mais detalhes para o deficiente visual ou utilizar a função de áudio dele para fazer audiodescrições mais detalhadas sobre o significado de uma tela. Já o iOS possui o *VoiceOver* [Apple Inc., 2017], que contém funcionalidades semelhantes aos do *Talkback*.

Em ambos os casos, o deficiente visual só necessita da ajuda de um terceiro para ativar a ferramenta de acessibilidade respectiva, dando total autonomia para o deficiente visual utilizar o aparelho.

Fazer aplicativos que funcionem em conjunto com essas tecnologias voltadas a pessoas com deficiência visual não é um trabalho difícil, mas criar boas experiências de uso que facilitem a vida de pessoas com deficiência visual é uma grande tarefa a ser cumprida.

Por isso é necessário adicionar outras tecnologias que facilitem o uso do app, neste caso, os *Beacons*. *Beacon* é um dispositivo que utiliza Bluetooth 4.0 (que tem baixo consumo de energia). Se existe um smartphone próximo a um *Beacon*, o aplicativo pode informar sua localização com maior precisão que um GPS. [TEIXEIRA, 2014].

Dessa forma quando um deficiente visual chegar no ponto, ele abre o aplicativo e, com uso do *Beacon* instalado no ponto, se detecta automaticamente em qual ponto o deficiente visual está. Sabendo isso, o app lista quais ônibus passam ali. Após o deficiente visual escolher um dos ônibus, o aplicativo vai notificar em intervalos pré-definidos quanto tempo falta para o ônibus chegar, em contrapartida o sistema irá alertar o motorista quando ele estiver próximo ao ponto em que existe um deficiente visual esperando por ele.

1.2 Restrições de projeto

- O smartphone deve ter o sistema operacional Android 4.1 (API Level 16) ou posterior instalado;

- O smartphone deve possuir Bluetooth 4.0 LE ou superior;
- O smartphone deve estar com a função Talkback ativada;
- O ônibus deve prover sinal de rede Wi-Fi para que o módulo do ônibus possa se comunicar.

2 Tecnologias Utilizadas

Para o desenvolvimento do projeto, foi elaborado um sistema dividido em quatro subsistemas, sendo eles o módulo do ônibus, o *beacon*, o *Web Service* e o aplicativo. Os métodos de comunicação e as tecnologias empregadas nelas estão ilustrados na Figura 1.

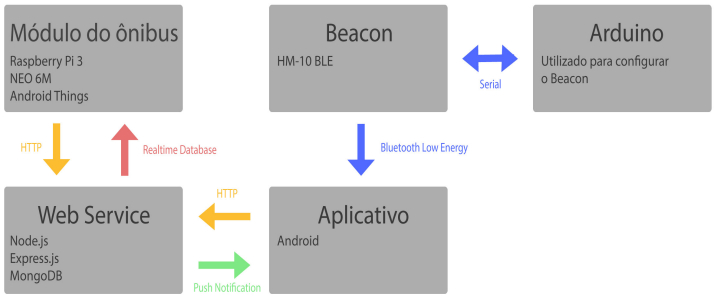

Figura 1 – Tecnologias utilizadas.

2.1 Comunicação

2.1.1 HTTP

O Protocolo de Transferência de Hipertexto (do inglês *Hypertext Transfer Protocol*), é um protocolo para tráfego de informações pela internet entre cliente e servidor. Sempre que o cliente deseja transmitir algo para o servidor, é enviada uma requisição *HTTP* contendo nela um cabeçalho, com informações sobre a requisição, e o corpo, contendo uma mensagem a ser entregue. Uma requisição *HTTP* pode ser do tipo GET, POST, PUT, DELETE, HEAD, TRACE, OPTIONS, ou CONNECT, e cada tipo defiine uma determinada ação solicitada pelo cliente ao servidor. O cabeçalho do protocolo contém o campo *Content-Type* com o tipo de *MIME* contido no corpo da requisição.

O tipo MIME (*Multipurpose Internet Mail Extensions* - Extensões de correio de Internet multifunções) é um padrão proposto pelos laboratórios *Bell Communications*, em 1991, para aumentar a possibilidade de inserir documentos (imagens, sons, texto etc.) em uma mensagem. Desde então, o tipo MIME é usado para formatar tanto documentos anexos em uma mensagem quanto os transferidos pelo protocolo HTTP [CCM, 2017].

2.1.2 Push Notification

Como explica [FINZI, 2016] *Push Notification* é um serviço de entrega de mensagens, parecido com *SMS* (*Shot Message Service*), que utiliza a internet para isso. Um caso comum do uso de *push notifications* é quando existe a necessidade de notificar um ou mais usuários de um aplicativo sobre algo, não sendo necessário que a aplicação esteja aberta. Alguns serviços são especializados na entrega de mensagens desse tipo, como o *Firebase Cloud Messaging*, que provê um *SDK* para que o desenvolvedor implemente o serviço de recebimento destas mensagens na sua aplicação para quando for preciso enviar uma mensagem, faça apenas requisições *HTTP* para o serviço e ele se encarregue de entregar a mensagem para os dispositivos disponíveis.

2.1.3 Beacons

O *Beacon* é um pequeno dispositivo que utiliza uma tecnologia chamada *Bluetooth Low Energy* (BLE), que emite um sinal intermitente de ondas de rádio que consegue localizar seu smartphone em um determinado raio de cobertura deste sinal. [CARNEIRO, 2016a]. A grande novidade desses aparelhos, 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 [SAES, 2014]. Como explica [CARNEIRO, 2016b], *iBeacon* é um protocolo de comunicação para dispositivos *Beacons*, lançado pela *Apple* em 2013.

De acordo com [CARNEIRO, 2016b] as informações do protocolo são formadas da seguinte forma:

1. Identificador Único Universal (do inglês *Universal Unique Identifier* ou UUID): É a sequência que identifica cada dispositivo.

2. Major: Utilizado para criar grupos de *Beacons*. Assim é possível fazer um filtro com todos dispositivos do mesmo grupo.

3. Minor: Utilizado para criar sub grupos, possibilitando o filtro dentro de grupos (*Major)*.

4. Frequência de Transmissão: Intervalo que o sinal é transmitido. Poder da transmissão: Alcance máximo do sinal.

2.1.4 Firebase Realtime Database

O *Firebase Realtime Database* é um banco de dados hospedado na nuvem. Os dados são armazenados no formato JSON e sincronizados em tempo real com todos os clientes conectados. Em vez de solicitações HTTP típicas, o *Firebase Realtime Database* usa a sincronização de dados. Sempre que os dados são alterados, todos os dispositivos conectados recebem essa atualização em milissegundos [DEVELOPERS, 2017].

2.2 Hardware

2.2.1 Raspberry Pi 3 (Módulo do ônibus)

Figura 2 – Raspberry Pi 3

Fonte: RASPBERRY PI

Segundo Upton [UPTON e Halfacree, 2017], "a *Raspberry Pi* é uma máquina completa, com considerável poder de processamento, em uma placa de circuito impresso menor do que um cartão de crédito. Com ela você pode ter resultados impressionantes". Esta pequena placa permite ter um computador em um pequeno espaço, contando com conectividades como *Bluetooth* e *WiFi*, é uma excelente opção para projetos que necessitam de mais poder de processamento em um pequeno espaço.

A *Fundação Raspberry Pi*, responsável pelo seu desenvolvimento, também mantém o foco da placa no meio educacional, apresentando uma plataforma acessível para que cada vez

mais pessoas se interessem por desenvolvimento de softwares.

Especificações:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 *wireless* LAN e *Bluetooth Low Energy* (BLE)
- 40-pin *extended* GPIO
- 4 USB 2 ports
- 4 *Pole stereo output* e *composite video port*
- Full size HDMI
- CSI *camera port* para conectar uma câmera *Raspberry Pi*
- DSI *display port* para conectar um display *touchscreen Raspberry Pi*
- Micro SD *port* para carregar o sistema operacional e armazenar dados

2.2.2 HM-10 (Beacon)

Figura 3 – Módulo Bluetooth HM-10

Fonte: GIANG

A empresa *Jinan Huamao* é responsável pelo desenvolvimento do HM-10 [Jinan Huamao, 2017], um pequeno módulo *Bluetooth* 4.0 BLE baseado no chip CC2540 que trabalha com alimentação de 3.3V. Com ele é possível realizar comunicações utilizando o protocolo *Bluetooth*.

2.2.3 Arduino Uno (Beacon)

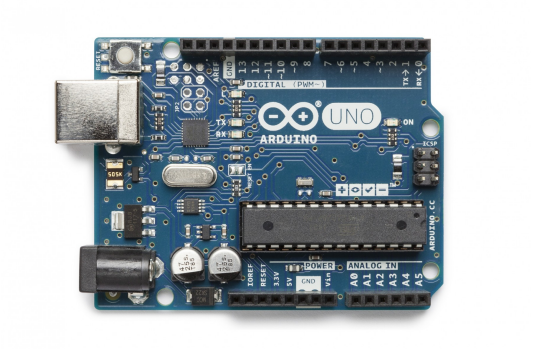

Figura 4 – Arduino Uno

Fonte: ARDUINO

Arduino é uma plataforma de prototipagem desenvolvida em 2005, na Itália. Conectando a placa via USB em um computador, é possível programar o microcontrolador Atmel que ela possui, tendo a possibilidade de controlar dispositivos externos por meio de suas portas *GPIO (General Purpose Input/Output)*.

O *Arduino* pode ser utilizado para desenvolver objetos interativos independentes, ou pode ser conectado a um computador, a uma rede, ou até mesmo à internet para recuperar e enviar dados do *Arduino* e atuar sobre eles [MCROBERTS, 2011].

Foi utilizado apenas para a configuração inicial do *Beacon*.

2.2.4 NEO 6M (Módulo do ônibus)

Figura 5 – Módulo NEO u-blox 6 GPS

Fonte: INSTRUCTABLES

O NEO U-BLOX 6 é um módulo que permite obter a geolocalização por meio do *GPS (Global Position System)*. De acordo com [Sousa, 2017] a obtenção da localização no globo é dada pela trilateração dos dados recebidos por pelo menos quatro satélites, onde cada um envia um sinal de sincronismo e em seguida sua posição.

Para obter os dados do módulo, realiza-se uma comunicação com protocolo UART (Universal Asynchronous Receiver/Transmitter). Tais informações obtidas seguem a especificação NMEA 0183 da (*National Marine Eletronic Association*). No trecho abaixo, [Sousa, 2017], fala do protocolo no Android:

> O protocolo NMEA é o mecanismo de comunicação padrão entre os receptores GPS e os drivers dos dispositivos receptores na arquitetura Android. Desta forma, os programas que fornecem soluções de posicionamento precisam decodificar as mensagens NMEA. A ideia do protocolo, no que diz respeito a coordenadas de posicionamento, consiste em enviar uma linha de dados, codificada em ASCII, que pode conter informações de posição, velocidade, tempo, dentre outros elementos processados pelo receptor GPS.

2.3 Software

2.3.1 Sistemas Operacionais

2.3.1.1 Android

Existem vários sistemas operacionais voltados para dispositivos móveis, sendo Android o mais popular. O sistema foi desenvolvido pela empresa Android Inc, posteriormente adquirida pela Google Inc. Foi lançado em 2008 e conta com o apoio da *Open Handset Alliance* (um conjunto de empresas que atuam juntas para o melhoramento dos padrões da telefonia e também do desenvolvimento do sistema Android).

Tal sistema é muito acessível e popular. Está disponível para uma enorme variedade de dispositivos. O Android tem como base o sistema operacional Linux, conhecido por ser um sistema flexível, adaptável a várias arquiteturas de processador, seguro e eficiente. Um dos grandes diferenciais do Android e que contribui para sua popularidade, é o fato de que o sistema é compatível com vários hardwares e pode estar disponível em *smartphones* de diversos fabricantes. [ANDROID, 2017]

Segundo uma pesquisa feita pela IDC (*International Data Corporation*) no primeiro quadrimestre de 2017 [IDC, 2017], o Android se mostra um sistema consolidado no meio que atua, tendo participação de 85% do mercado. O gráfico da Figura 6 mostra que desde 2014 é o sistema operacional mais utilizado em dispositivos móveis.

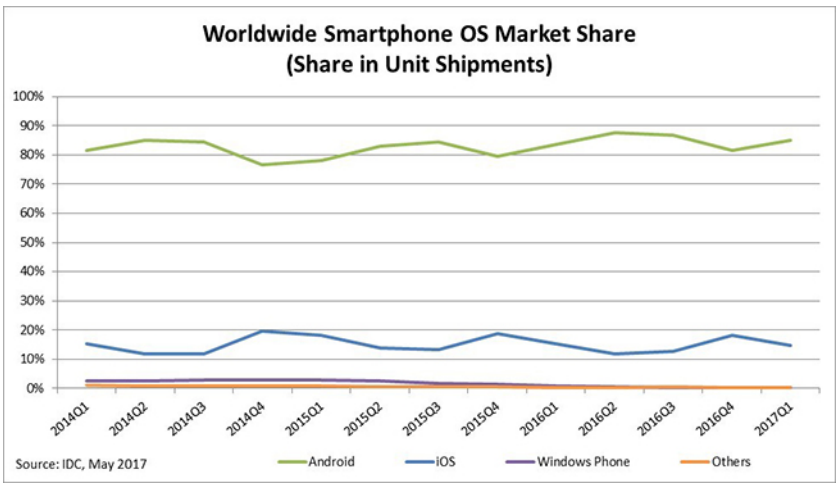

Figura 6 – Pesquisa realizada pela IDC

Fonte: IDC

2.3.1.2 Android Things

O Android Things foi criado voltado ao mercado *IoT (Internet of Things)* e a partir da versão do Android para *smartphones*, o que possibilitou o suporte a maioria das APIs disponíveis. Utilizando a mesma *IDE* para desenvolvimento, qualquer desenvolvedor Android pode desenvolver soluções *IoT* com os conhecimentos que já possui. [Android Things, 2015]

> *A comprehensive way to build IoT products with the power of Android, one of the world's most supported operating systems. Now any Android developer can quickly build a smart device using Android APIs and Google services, while staying highly secure with updates direct from Google.* [PIEKARSKI, 2016]

Atualmente o sistema tem suporte para rodar nas seguintes plataformas:

- NXP Pico i.MX7D
- NXP Pico i.MX6UL
- NXP Argon i.MX6UL
- NXP SprIoT i.MX6UL
- Raspberry Pi 3

Arquitetura utilizada

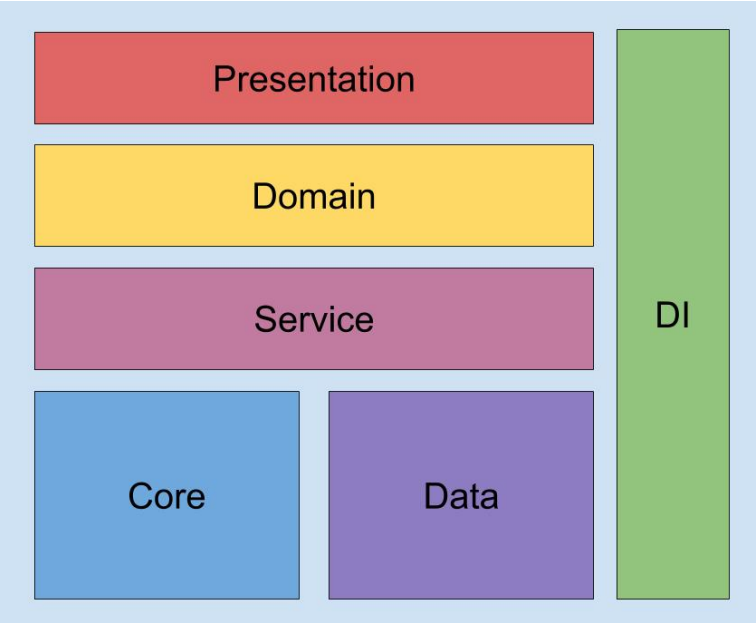

Figura 7 – Diagrama de blocos do aplicativo móvel.

Para desenvolvimento das aplicações que rodam nos sistemas operacionais Android, foi adotado o padrão *Clean Architecture*. Este padrão foi proposto por Robert Ceil Martin (conhecido como Uncle Bob) em [MARTIN, 2012], e foca no domínio da aplicação, fazendo *frameworks* e *drivers* serem detalhes de implementação. Seguindo seus princípios, a estrutura de camadas ficou da seguinte forma:

Core Não contém nenhuma lógica de negócio. Esta camada provê informações que são comuns para toda a aplicação, como informações estáticas e algumas classes e interfaces bases.

Data Responsável por prover dados para toda aplicação. Ela adota o padrão de arquitetura *Repository*, tendo uma interface de acesso aos dados. Uma grande vantagem em utilizar essa camada com esse padrão de arquitetura, é o respeito a responsabilidade única, um dos princípios do *SOLID* [CASTILHO, 2013]. Ela encapsula toda lógica de busca de dados, assim, caso uma classe precise de algum dado específico, ela solicita através da interface de comunicação e, a classe que implementa a interface, cuida de toda lógica de busca de dado, seja um dado armazenado localmente, em *cache* ou em um servidor remoto. Tudo fica transparente para a classe que solicitou o dado.

Service Provê serviços para qualquer camada. No caso do aplicativo para *smartphone*, a implementação do serviço de voz fica implementada neste pacote. Esta camada não existe no *Android Things*, por não possuir serviços característicos da mesma.

Domain Esta camada encapsula todas as regras de negócios da aplicação. Toda vez que é necessário realizar processamentos em dados para satisfazer funcionalidades, são feitos por esta camada.

Presentation Responsável por toda interface gráfica. Toda lógica de criação de telas e interceptação de interações do usuário com o aplicativo é feito aqui. Quando é necessário procurar dados para exibir ao usuário, são feitas solicitações deles para a camada *Domain* ou *Data* para que seja possível exibir os dados.

DI Esta camada utiliza o padrão de arquitetura *Injeção de Dependências*. Esta camada provê todas as dependências, fazendo a implementação mais limpa nas outras classes, já que não precisam saber como instânciar uma classe.

2.3.1.3 Linguagem de Programação

Segundo [AVRAM Abel traduzido por MALARA, 2017], a Google suportava as linguagens de programação Java e C++ para desenvolvimento de aplicativos Android. Em 2017, foi acrescentado suporte a linguagem Kotlin, por "ser concisa, expressiva e projetada para ser *type-safe* e *null safe*". Para este trabalho, dentre as três linguagens disponíveis, foi escolhido o Kotlin por demonstrar ser uma linguagem de sintaxe mais simplificada e trabalhar bem, caso necessário, com Java, como cita [KEWERSON, 2017]: "Como consequência das boas decisões dos designers da linguagem, temos uma linguagem concisa com a possibilidade de interoperar código Kotlin com Java e Java com Kotlin".

2.3.2 Backend

2.3.2.1 Linguagem

Para o desenvolvimento do *Web Service* foi escolhido um conjunto de tecnologias *server-side* baseadas em JavaScript.

2.3.2.1.1 JavaScript

JavaScript é uma linguagem de programação *client-side*, utilizada para manipular os comportamentos de uma página, controlando o HTML e o CSS. Outra característica é o fato de ser uma linguagem orientada à eventos.

Para explicar melhor o que são eventos, é importante citar que uma página HTML utiliza tags para representar seus elementos, podendo conter menus, botões e formulários, entre outros, em seu corpo. Cada elemento possui alguns atributos, sendo alguns desses chamados atributos de eventos, como por exemplo o *onClick* que realiza alguma função caso o elemento referente seja clicado pelo usuário.

Para especificar as ações que devem ser tomadas quando um evento é acionado, pode-se utilizar o JavaScript. Com o decorrer do tempo, foram desenvolvidas algumas modificações na linguagem JavaScript para possibilitar a utilização do mesmo no *server-side*, possibilitando o desenvolvimento de um *Web Service*.

Foram escolhidas, como ferramentas, o MongoDB, o Node.js e o Express.js, utilizando o [ALMEIDA, 2016] como referência teórica e prática para conhecer essas tecnologias.

2.3.2.2 MongoDB

É um banco de dados não relacional com uma escalabilidade muito boa. Ele utiliza conceitos de *collections* e *documents* em sua construção.

As *collections* são equivalentes aos bancos de um ambiente que utiliza o SQL. Já os *documents*, se equivalem aos registros de cada banco. Os dados são guardados em arquivos similares aos de formato JSON (*JavaScript Object Notation*).

Outro item importante sobre o MongoDB é o fato de ser *schemaless*, tornando-o bem flexível em relação a inclusão de dados diferentes em uma mesma *collection*, fazendo com que a validação de dados fique nas mãos dos desenvolvedores.

Apesar de *schemaless*, é possível criar *schemas* para auxiliar no desenvolvimento. Ao utilizar o mongoose, ferramenta desenvolvida em cima do MongoDB para trabalhar nele como se estivesse utilizando um banco relacional, é possível definir de antemão, quais os atributos que devem existir em cada *collection* necessária para a aplicação.

2.3.2.3 Node.js

Plataforma principal para o funcionamento da aplicação, construída sobre o motor Javascript V8 do Google Chrome, o Node.js foi desenvolvido para construir aplicações *web* escaláveis de forma assíncrona, cuidando de várias conexões de maneira concorrente. Ele consegue isso através da utilização de *callbacks* [NODEBR, 2016].

Sempre que ocorre um evento, é disparado um *callback* que dirá o que deve ser realizado pela aplicação. Esse processo é feito para todas as funções presentes na aplicação e sempre que houver um evento sendo disparado.

Como no lado do servidor não existe uma interface gráfica a ser visualizada, os eventos nesse caso são dados de I/O, como algum parâmetro usado em uma busca no banco de dados, a resposta para aquele parâmetro, entre outros casos.

O Node.js é uma plataforma extremamente modularizada, com módulos criados por desenvolvedores ativos no mundo todo. Para gerenciar esses módulos, ele utiliza o npm, responsável por instalar, atualizar ou remover suas dependências.

É o Node.js que realiza a conexão com servidores e diz quais bancos de dados serão utilizados pela aplicação, na configuração inicial.

2.3.2.4 Express.js

É um *framework* para Node.js que ajuda na organização de sua aplicação, caso use a arquitetura MVC (*Model View Control*), no lado do servidor. Uma de suas funções é a de facilitar a criação e manutenção de rotas, realizando uma configuração inicial com os caminhos para os *controllers*, *models* e *views* utilizados pela sua aplicação, além de informar os dados de inicialização do servidor.

2.3.2.5 REST

Segundo [ALMEIDA, 2016], o padrão REST utiliza um conjunto de operações aplicáveis a todos os recursos de informação utilizando o protocolo HTTP. Suas operações mais utilizadas são DELETE, POST, PUT e GET.

No padrão REST, cada recurso possui um identificador único, representado, por exemplo no HTTP, pela sua URL de acesso.

Dentre os princípios do padrão REST, segundo [GOMES André Faria apud TIKOV, 2009], é importante dar um foco em cinco deles.

São eles:

Dar um identificador a todas as coisas Considerando as coisas como sendo os recursos, o principal benefício desse item é a utilização de URIs que identifiquem o necessário, especificando se os recursos que sua aplicação oferece são itens individuais, conjuntos de itens, objetos virtuais e físicos, ou resultados de computação.

Vincular as coisas Esse princípio facilita a utilização de serviços externos em sua aplicação. A vinculação, nesse caso, é feita através de links para que o servidor saiba aonde estão alocados os recursos necessários. Esse princípio é extremamente útil para o desenvolvimento de aplicações dinâmicas.

Utilizar métodos padronizados Aqui entram os verbos HTTP, como GET, PUT, POST e DELETE. Esses verbos representam métodos padrões para que a aplicação *web* saiba o que deve ser feito com determinada URI. Essa padronização faz com que seu navegador saiba exatamente o que fazer.

Recursos com múltiplas representações Este princípio diz para oferecer formas diversas dos recursos para diferentes finalidades.

Comunicar sem estado Manter o estado da comunicação no cliente ou transformá-lo no estado do recurso. Isso faz com que a escalabilidade da aplicação seja um dos focos desse princípio. Outro aspecto desse tópico, é o isolamento do cliente em relação ao servidor, fazendo com que o cliente não fique preso ao mesmo servidor para realizar duas solicitações consecutivas, com isso, ele recebe as informações do servidor sem precisar se preocupar se está ocorrendo alguma troca de disco rígido ou atualização de software no servidor.

2.3.3 Controle de Versão Distribuído

Segundo Git SCM (Sistema de Controle de Versão do inglês *Source Control Management*):

> O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que possa recuperar versões específicas. [...] O método preferido de controle de versão por muitas pessoas é copiar arquivos em outro diretório (talvez um diretório com data e hora, se forem espertos). Esta abordagem é muito comum por ser tão simples, mas é também muito suscetível a erros. É fácil esquecer em qual diretório você está e gravar acidentalmente no arquivo errado ou sobrescrever arquivos sem querer. Para lidar com esse problema, alguns programadores desenvolveram há muito tempo VCSs locais que armazenavam todas as alterações dos arquivos sob controle de revisão.[...] Sistemas de Controle de Versão Distribuídos [...] os clientes não apenas fazem cópias das últimas versões dos arquivos: eles são cópias completas do repositório. Assim, se um servidor falha, qualquer um dos repositórios dos clientes pode ser copiado de volta para o servidor para restaurá-lo. [GIT, 2016]

Como explicado no site oficial do [GIT, 2016], o uso de um *software* controlador de versões trás para a equipe de desenvolvimento mais segurança ao longo do tempo, possibilitando voltar a estados anteriores do código. Isso facilita caso for necessário encontrar quais alterações foram incluídas que ocasionaram um problemas.

Git e GitHub

> Git é um sistema de controle de versão de arquivos. Através dele se pode desenvolver projetos onde diversas pessoas podem contribuir simultaneamente no mesmo, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.[...]

> O Github é um serviço *web* que oferece diversas funcionalidades extras aplicadas ao git. Resumindo, é possível utilizar gratuitamente o github para hospedar seus projetos pessoais. Além disso, quase todos os projetos/frameworks/bibliotecas sobre desenvolvimento *open source* estão no github, o que permite acompanhá-los através de novas versões, contribuir informando *bugs* ou até mesmo enviando código e correções. [SCHMITZ, 2015]

O uso do Git com GitHub permitiu que neste trabalho mais de um integrante trabalhasse no código sem a necessidade de estar próximo do outro. Ao realizar uma alteração ela é adicionada ao histórico de alterações do Git e enviada ao GitHub, onde ficava disponível para o outro integrante puxar e integrar com suas alterações.

3 Desenvolvimento

Esse capítulo visa apresentar os subsistemas, indicando quais as formas de comunicação entre eles e quais funções eles possuem para o correto funcionamento do sistema. Também serão descritos, em mais detalhes, as ferramentas, arquiteturas e hardwares utilizados para o desenvolvimento do projeto.

3.1 Descrição da Informação

Aqui os subsistemas são apresentados com uma breve descrição sobre eles e uma visão geral de como eles se comunicam.

3.1.1 Visão Geral

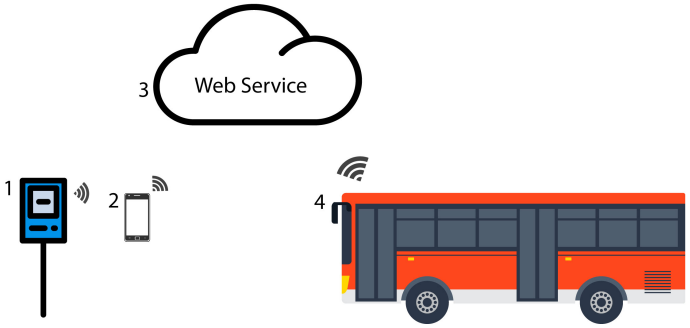

Figura 8 – Visão geral da comunicação dos componentes.

3.1.2 Descrição Funcional

1. Módulo do ponto de ônibus

Dispositivo localizado em um determinado ponto de parada de ônibus. Emite constantemente um sinal ID para a identificação do ponto que ele se refere.

2. Aplicativo para dispositivo móvel

Aplicativo que irá interagir com o deficiente visual. Sua função é verificar qual *Beacon* está mais próximo para que a API possa saber sua localização, podendo listar, via interface áudio-visual, para o usuário, quais linhas passam no ponto de parada que ele está.

3. Web Service

Sistema que recebe informações do aplicativo e do módulo do ônibus. Tem como objetivo acessar os dados gravados no banco de dados para que possa prover informações de previsão ao aplicativo. Também é responsável por verificar se o módulo do ônibus deve alertar a presença de um usuário no próximo ponto. Além de calcular a previsão de um ônibus até o ponto de parada selecionado.

4. Módulo do ônibus

Dispositivo instalado no ônibus. Mantém comunicação constante com a API para informar sua geolocalização. Verifica ao mesmo tempo a necessidade de alertar o motorista se existe um deficiente visual aguardando no próximo ponto de parada.

3.1.3 Representação do Fluxo da Informação

Aprofundando na parte da comunicação entre os subsistemas, a forma como eles se relacionam e os parâmetros necessários para o funcionamento, seguem a seguinte forma:

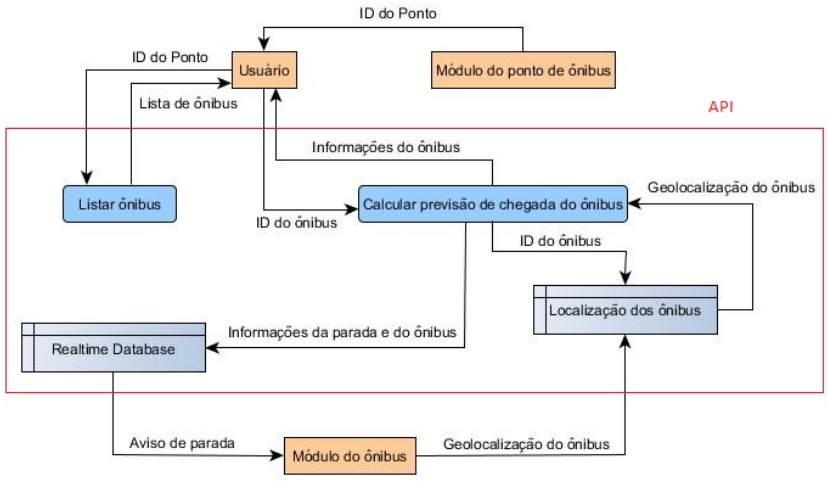

Figura 9 – Diagrama de fluxo de dados.

Para armazenar todas informações necessárias no sistema, foi utilizado o *schema* conforme a Figura 10:

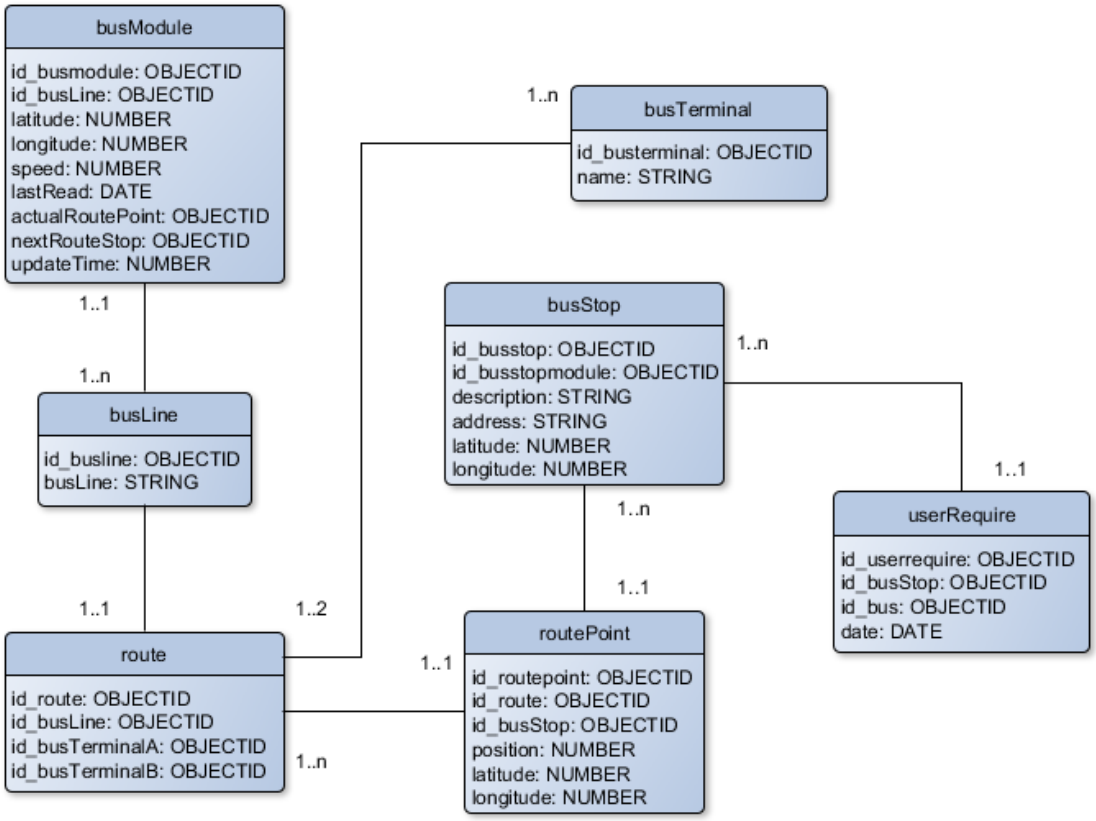

Figura 10 – Diagrama de banco de dados.

busModule: Contém as informações de todos módulos dos ônibus.

busLine: Contém os registros das linhas de ônibus disponíveis.

route: Contém o registro da rota que uma linha de ônibus faz. Nela é possível armazenar o terminal de origem e terminal de destino.

busTerminal: Contém os registros de todos terminais de ônibus.

routePoint: Cada registro em *route* é associado a N registros em *routePoint*, onde cada *routePoint* representa um ponto geográfico da rota.

busStop: Contém todos registros de pontos de parada que existem.

userRequire: Contém todos registros de requisições de parada. Quando um usuário solicitar um ônibus, sua solicitação é armazenada nesta *collection*.

3.1.4 Casos de Uso

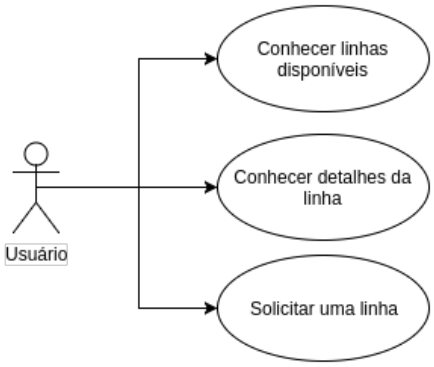

Figura 11 – Diagrama de caso de uso.

3.1.4.1 Narrativas: Casos de Uso

Solicitar horário do próximo ônibus da linha e sentido escolhido: este caso de uso acontece quando um usuário solicita qual será a previsão de horário do próximo ônibus, de uma linha e sentido que ele poderá escolher de acordo com o seu ponto de ônibus.

Solicitar parada do ônibus escolhido: este caso de uso é uma extensão do caso de uso Solicitar horário do próximo ônibus da linha e sentido escolhido, onde depois de escolher uma linha e sentido ele poderá solicitar a parada do próximo ônibus escolhido.

3.1.4.2 Diagramas de apoio para compreensão funcional

Tabela 1 – Tabela com caso de uso UC001.

Identificação: UC001	
Nome: Solicitar horário do próximo ônibus	
Atores: Usuário	
Pré-condições: O aplicativo precisa ter lido o ID do módulo do ponto de ônibus	
Pós-condições: Retorno do horário do próximo ônibus e da solicitação de parada	
Fluxo de eventos	
Ator	**Sistema**
1. Usuário chega ao ponto de ônibus	2. Sistema lê o ID do ponto de ônibus e retorna uma lista de linhas
3. Usuário escolhe uma linha	4. Informar constantemente o horário do ônibus
Fluxo alternativo	
Não possui fluxo alternativo	

Tabela 2 – Tabela com caso de uso UC002.

Identificação: UC002	
Nome: Solicitar parada do ônibus escolhido	
Atores: Usuário	
Pré-condições: O usuário precisa ter solicitado o ônibus de uma linha	
Pós-condições: Confirmação de parada	
Fluxo de eventos	
Ator	**Sistema**
1. Usuário confirma solicitação de parada no seu ponto	2. Sistema retorna tela de seleção de ponto de ônibus destino
Fluxo alternativo	
1.a 1. Usuário cancela solicitação de parada	2. Sistema retorna cancela operação
3.a 1. Usuário cancela escolha de ponto de ônibus	2. Sistema retorna cancela operação

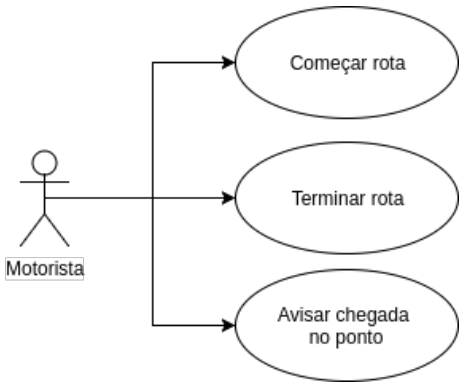

Figura 12 – Diagrama de caso de uso.

3.1.4.3 Narrativas: Casos de Uso

Iniciar uma rota: Este caso de uso acontece quando o motorista inicia sua rota e precisa informar isso ao módulo, para que o dispositivo comece a enviar seus dados de geolocalização para o *Web Service*.

Terminar uma rota: Este caso de uso acontece quando o motorista termina a rota do ônibus e precisa informar isso ao módulo, para que o dispositivo pare de enviar seus dados de geolocalização para o *Web Service*.

Avisar chegada no ponto: Este caso de uso acontece quando um deficiente visual solicitou o ônibus e o motorista está parado no ponto de ônibus. Quando motorista avisa que chegou, é disparado um alerta para o deficiente visual. Este aviso é ativado através de um botão no módulo do ônibus, para uma maior precisão e agilidade no processo, eliminando a necessidade de corrigir a posição do ônibus para disparar o aviso ao deficiente visual.

3.1.4.4 Diagramas de apoio para compreensão funcional

Tabela 3 – Tabela com caso de uso UC003.

Identificação: UC003	
Nome: Iniciar uma rota	
Atores: Motorista	
Pré-condições: O motorista precisa ter iniciado o módulo	
Pós-condições: -	
Fluxo de eventos	
Ator	**Sistema**
1. Usuário confirma o inicio da rota	2. Módulo começa a enviar dados de geolocalização
	3. Sistema exibe tela com opção de parar envio de dados

Tabela 4 – Tabela com caso de uso UC004.

Identificação: UC004	
Nome: Finalizar uma rota	
Atores: Motorista	
Pré-condições: O motorista precisa ter iniciado uma rota	
Pós-condições: -	
Fluxo de eventos	
Ator	**Sistema**
1. Motorista confirma o fim da rota	2. Módulo para de enviar dados de geolocalização
	3. Sistema exibe tela com opção de iniciar envio de dados

Tabela 5 – Tabela com caso de uso UC005.

Identificação: UC005	
Nome: Avisar chegada no ponto	
Atores: Motorista	
Pré-condições: O motorista precisa ter iniciado uma rota	
Pós-condições: -	
Fluxo de eventos	
Ator	**Sistema**
1. Motorista confirma chegada no ponto de ônibus	2. Módulo enviar aviso de chegada
	3. Sistema exibe tela com opção de parar envio de dados

3.1.5 Interfaces com Sistema

Neste tópico, serão apresentados as telas criadas tanto para o aplicativo, quanto para o módulo do ônibus.

3.1.5.1 Aplicativo

3.1.5.1.1 Busca por um ponto próximo

Figura 13 – Tela do aplicativo ao buscar por um ponto de ônibus próximo.

Tabela 6 – Descrição dos elementos da tela de busca por ponto de ônibus próximo.

Número	Nome	Descrição	Requisitos	Grupo
1	Símbolo da busca do ponto de ônibus	Indica que o aplicativo está procurando um ponto de ônibus	-	Imagem e texto
2	Áudio sobre busca	Indica ao usuário que está sendo feito uma busca por algum ponto próximo	Função Talkback ativa	Áudio

3.1.5.1.2 Busca por um ponto na API

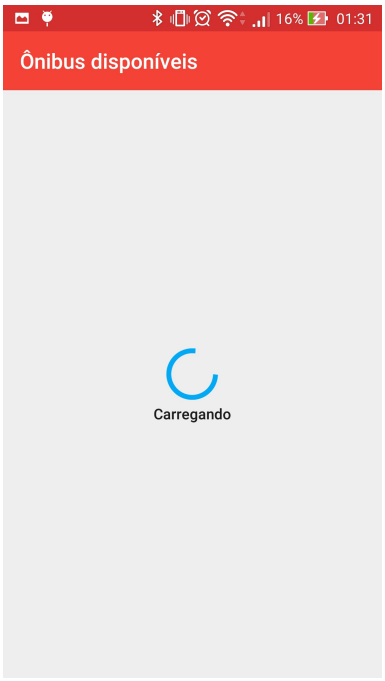

Figura 14 – Tela do aplicativo ao buscar por um ponto de ônibus na API.

Tabela 7 – Descrição dos elementos da tela de busca por ponto de ônibus na API.

Número	Nome	Descrição	Requisitos	Grupo
1	Símbolo da busca das linhas de ônibus	Indica que o aplicativo procura as linhas de ônibus	O sistema deve ter detectado um ponto de ônibus	Imagem e texto
2	Áudio sobre busca	Indica ao usuário que está sendo feito uma busca dos ônibus disponíveis	Função Talkback ativa	Áudio

3.1.5.1.3 Lista de ônibus disponíveis

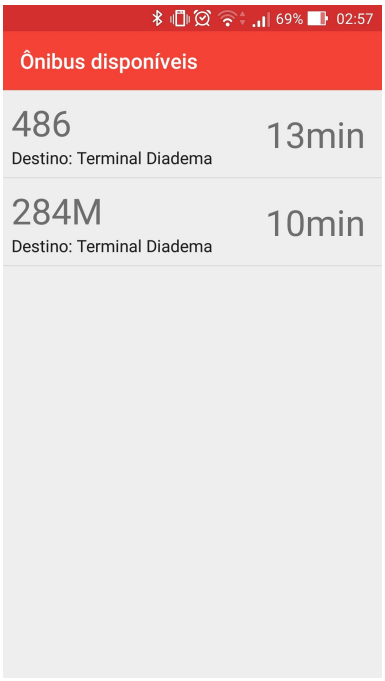

Figura 15 – Tela do aplicativo com ônibus disponíveis.

Tabela 8 – Descrição dos elementos da tela de busca por ponto de ônibus na API.

Número	Nome	Descrição	Requisitos	Grupo
1	Lista de linhas	Lista de linhas que o usuário pode escolher	Ter recebido uma lista da API	Botão
2	Áudio sobre escolha de um item	Indica que a lista de ônibus já está disponível	Função Talkback ativa	Áudio

3.1.5.1.4 Detalhes do ônibus

Figura 16 – Tela do aplicativo com detalhes de um ônibus.

Tabela 9 – Descrição dos elementos da tela de detalhes do ônibus.

Número	Nome	Descrição	Requisitos	Grupo
1	Linha X1	Mostra a linha selecionada	Receber previsão da API	Texto
2	Origem	Exibe o ponto inicial da linha	Receber previsão da API	Texto
3	Destino	Exibe o ponto final da linha	Receber previsão da API	Texto
4	Chegada em	Exibe a previsão de chegada da linha	Receber previsão da API	Texto
5	Voltar	Volta para a seleção de linhas	Receber previsão da API	Botão
6	Solicitar ônibus	Solicita que o ônibus pare no seu ponto e acionar o acompanhamento dele	Receber previsão da API	Botão
7	Áudio sobre previsão	Alerta ao usuário a previsão do ônibus	Receber previsão da API	Áudio

3.1.5.1.5 Aguardando um ônibus

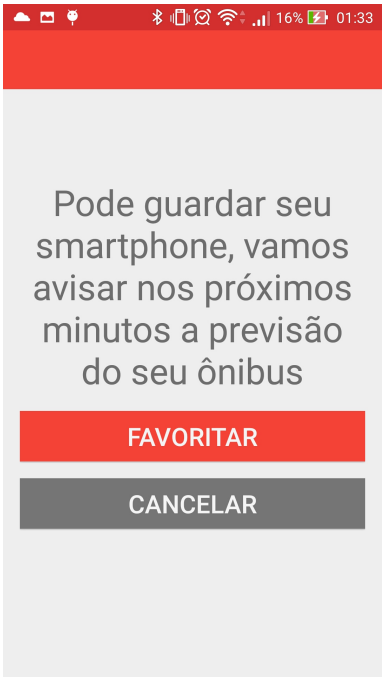

Figura 17 – Tela do aplicativo sobre a solicitação de um ônibus.

Tabela 10 – Descrição dos elementos da tela sobre a solicitação de um ônibus.

Número	Nome	Descrição	Requisitos	Grupo
1	Informação de previsão	O sistema irá informar o usuário até a chegada do ônibus	Ter selecionado botão Solicitar ônibus	Texto
2	Favoritar	Adiciona ônibus como favorito	O ônibus não pode estar cadastrado como favorito. Caso esteja o botão não é exibido	Botão
3	Cancelar	Cancela o acompanhamento do ônibus e solicita que não pare mais no ponto	Ter selecionado botão Solicitar ônibus	Botão
4	Áudio sobre o acompanhamento	Informa ao usuário que está sendo feito o acompanhamento do ônibus	Ter escolhido acompanhar um ônibus. Função Talkback ativa	Áudio

3.1.5.2 Módulo do ônibus

3.1.5.2.1 Carregando Informações

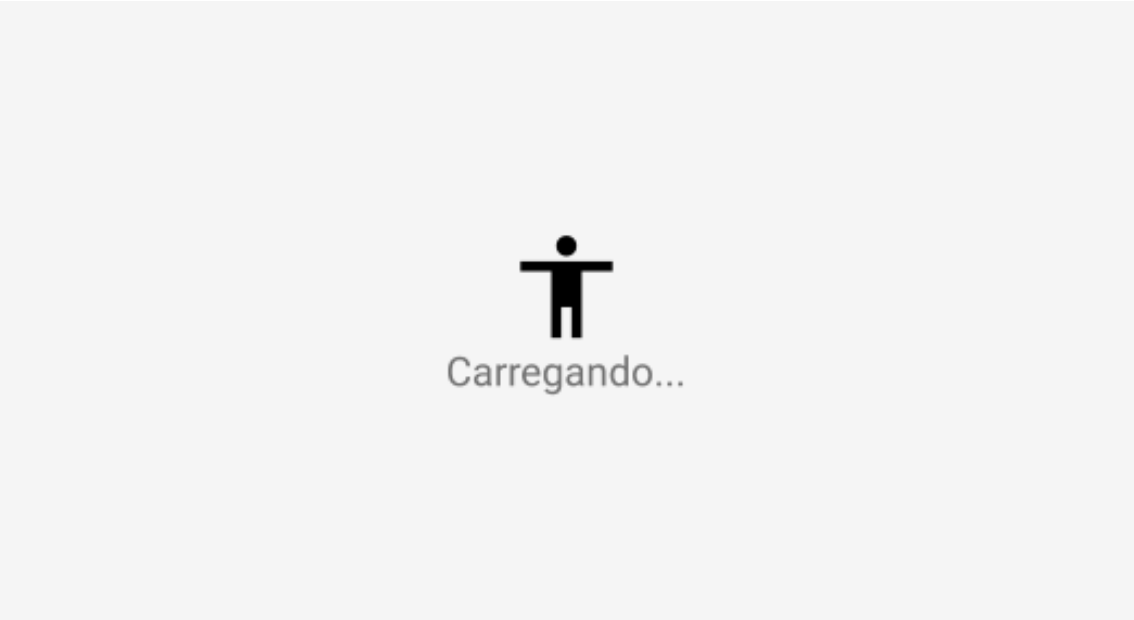

Figura 18 – Tela do módulo do ônibus quando está carregando informações.

Tabela 11 – Descrição dos elementos da tela sobre estar carregando informações.

Número	Nome	Descrição	Requisitos	Grupo
1	Mensagem	Informa que está carregando informações do servidor	Ter iniciado o sistema	Texto

3.1.5.2.2 Aguardando iniciar a rota

Figura 19 – Tela do módulo do ônibus aguardando motorista iniciar rota.

Tabela 12 – Descrição dos elementos da tela sobre começar uma rota.

Número	Nome	Descrição	Requisitos	Grupo
1	Informação para começar rota	Informar que precisa avisar quando começar a rota	-	Texto
2	Começar rota	Começar rotina de envio da posição	-	Botão

3.1.5.2.3 Realizando uma rota

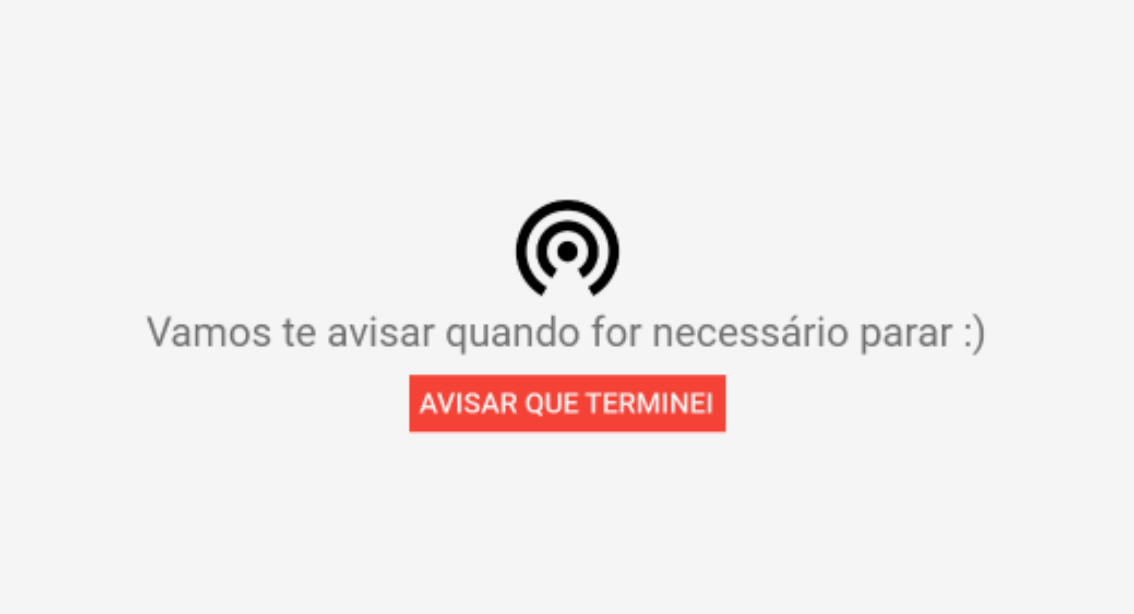

Figura 20 – Tela do módulo do ônibus quando motorista está fazendo a rota.

Tabela 13 – Descrição dos elementos da tela sobre terminar uma rota.

Número	Nome	Descrição	Requisitos	Grupo
1	Informação sobre término da rota	Informar que precisa avisar quando terminar a rota	Ter iniciado uma rota	Texto
2	Terminar rota	Terminar rotina de envio da posição	Ter iniciado uma rota	Botão

3.1.5.2.4 Aviso para parar no próximo ponto

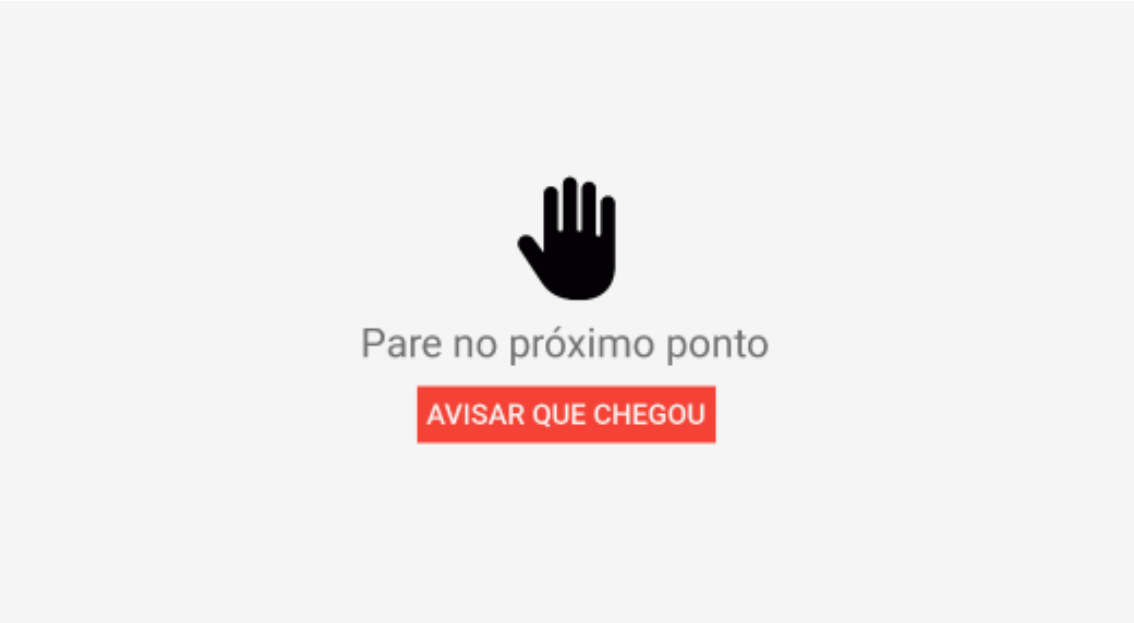

Figura 21 – Tela do módulo do ônibus quando motorista precisar parar no próximo ponto.

Tabela 14 – Descrição dos elementos da tela sobre terminar uma rota.

Número	Nome	Descrição	Requisitos	Grupo
1	Símbolo de alerta para parar	Informar que precisa parar no próximo ponto	Servidor informar a necessidade	Imagem e Texto
2	Informar chegada	Envia informação que chegou no ponto	Ter chegado no ponto	Botão

3.2 Módulo do Ponto de Ônibus

3.2.1 Hardware

1. HM-10 - Bluetooth 4.0 BLE module
2. Arduino Uno

3.2.2 Software

- Arduino IDE 1.8.3 ou superior, encontrado em <https://www.arduino.cc/en/main/software>.

3.2.3 Configuração

Para configurar o módulo HM-10 utilizamos o Arduino como ponte. Para realizar tais configurações, foi montado o circuito conforme a Figura 22.

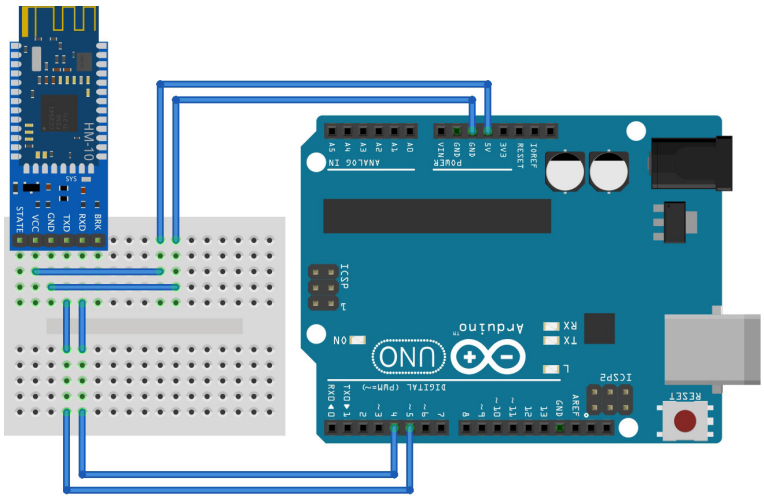

Figura 22 – Configurando o módulo *Bluetooth* HM-10 via Arduino UNO.

Após conectar o Arduino ao computador, foi utilizado sua *IDE* para escrever o código, que está no apêndice A. No código do Arduino, foi estabelecido uma comunicação serial com o módulo HM-10 para enviar os comandos AT necessários. Esses comandos são para otimizar o uso da bateria e ativar a função *Beacon* do módulo. A seguir, a descrição dos comandos.

Código	Descrição
AT+RENEW	Coloca nos padrões de fábrica
AT+RESET	Reinicia para aplicar os padrões de fábrica
AT+MARJ0xNNNN	Define o valor Marjor
AT+MINO0xNNNN	Define o valor Minor
AT+NAMEMeuBeacon	Define o nome do dispositivo
AT+ADVI5	Define tempo de envio. 5 = 546.25 millisegundos
AT+ADTY3	Define como não pareável
AT+IBEA1	Habilita como *Beacon*
AT+DELO2	Configura para apenas emitir sinal
AT+PWRM0	Habilita função auto-sleep para economizar energia
AT+RESET	Reinicia para aplicar as configurações

Após configurado, o módulo pode ser ligado em uma bateria 3,3v para utilização.

3.3 Módulo do Ônibus

3.3.1 Hardware

- Raspberry Pi 3
- Tela LCD 7"*Touch Screen*
- NEO u-blox 6 GPS Modules

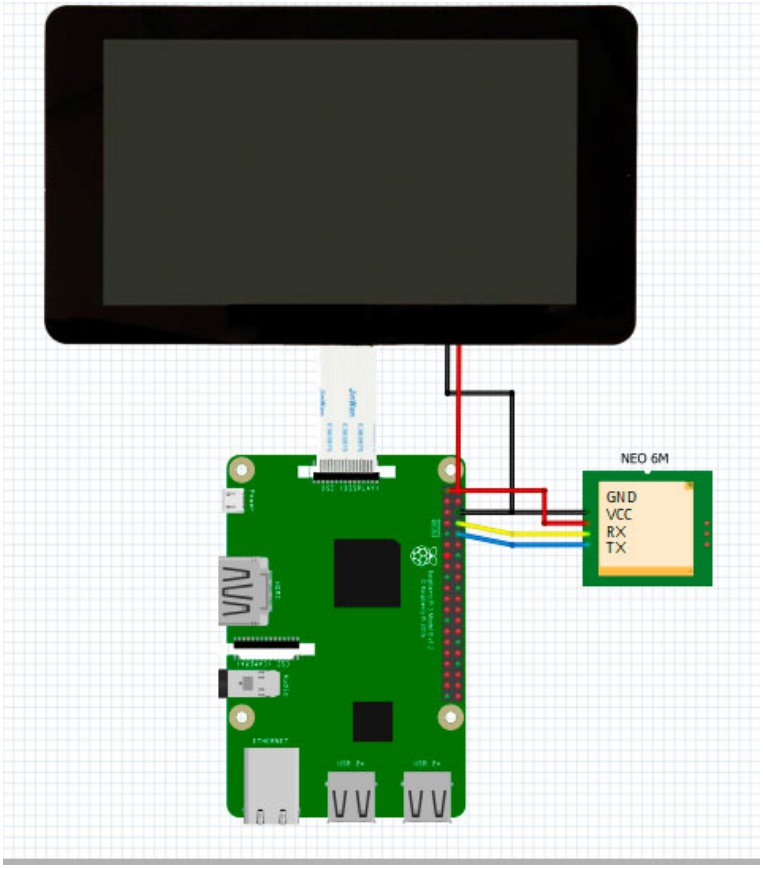

Figura 23 – Hardware do módulo do ônibus.

3.3.1.1 Intel Edison

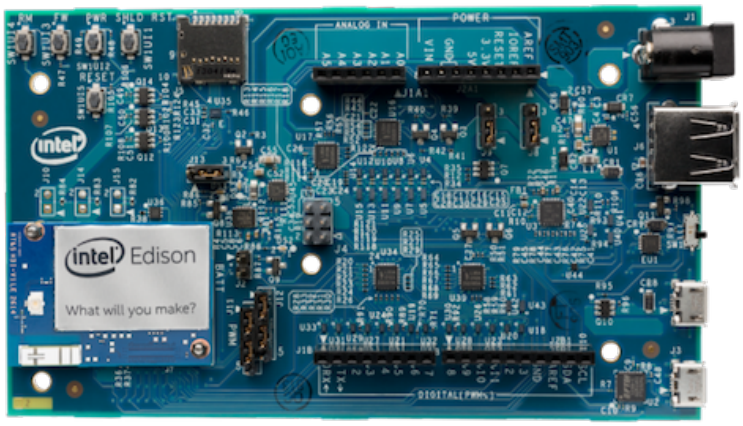

Figura 24 – Intel Edison

Fonte: DEVELOPER

Inicialmente foi adotado o *Intel Edison* com placa de expansão Arduino. Foi escolhido devido a fácil acesso a um exemplar e ótimo hardware. Ele conta com *WiFi*, *Bluetooth*, portas I/O, processador Intel Atom de 500 MHz, 1GB de memória RAM DDR3 e 4GB eMMC.

Sua utilização foi fácil e não obtivemos nenhuma dificuldade em instalar o sistema escolhido.

Problemas encontrados em adotar como solução:

Preço

Embora tenha um ótimo hardware e uma empresa séria por trás da sua construção, o preço, em 07/2017, que gira em torno de R$ 600,00, não justifica sua adoção como a melhor solução para o projeto já que existem alternativas com preços melhores e bom desempenho, como o *Raspberry Pi 3*.

Ausência de controlador gráfico

Este projeto foi elaborado pensando em fazer alertas por meio de um *display* gráfico. A placa Intel Edison nos permite fazer alertas visuais utilizando LEDs ou pequenos *displays*

OLED, o que não satisfaz a exigência do projeto.

Descontinuidade da placa pela Intel

Em 07/2017, a Intel anunciou a descontinuidade do desenvolvimento de algumas placas que fabrica, incluindo a placa Intel Edison.

3.3.1.2 Raspberry Pi 3

Testes realizados no Raspberry Pi 3 demonstraram ser uma boa alternativa ao Intel Edison. A instalação do sistema Android Things foi realizada sem dificuldade e a placa vem com saída HDMI e *display port* que permite utilizar telas LCD para fazer os alertas visuais. Seu preço, em 08/2017, gira em torno de R$ 150,00, 1/4 do preço do Intel Edison Seu hardware contém boas especificações.

Embora tenha um hardware com especificações superiores ao Intel Edison, não houve ganho de desempenho ao rodar o sistema, devido a ausência de algoritmos complexos no sistema. Assim, a grande vantagem de se utilizar o Raspberry Pi 3 ao invés do Intel Edison, é seu baixo custo e recurso de chip gráfico.

3.3.2 Software

3.3.2.1 Android Things

Ele é, atualmente, uma versão do Android O reduzida. Sua escolha foi devido a facilidade de embarcar em placas como o Raspberry Pi e Intel Edison, além da variedade de recursos que já estão disponíveis no SO que facilitam o desenvolvimento do módulo, como serviços de geolocalização.

Para obter uma imagem do Sistema Operacional, foi preciso realizar um cadastro na plataforma *Android Things Console*, encontrado no link <https://partner.android.com/things/-console/>. Foi necesssário cadastrar um produto, neste caso chamamos de *BeaconBusTracker*, conseguindo acesso ao download da imagem do SO.

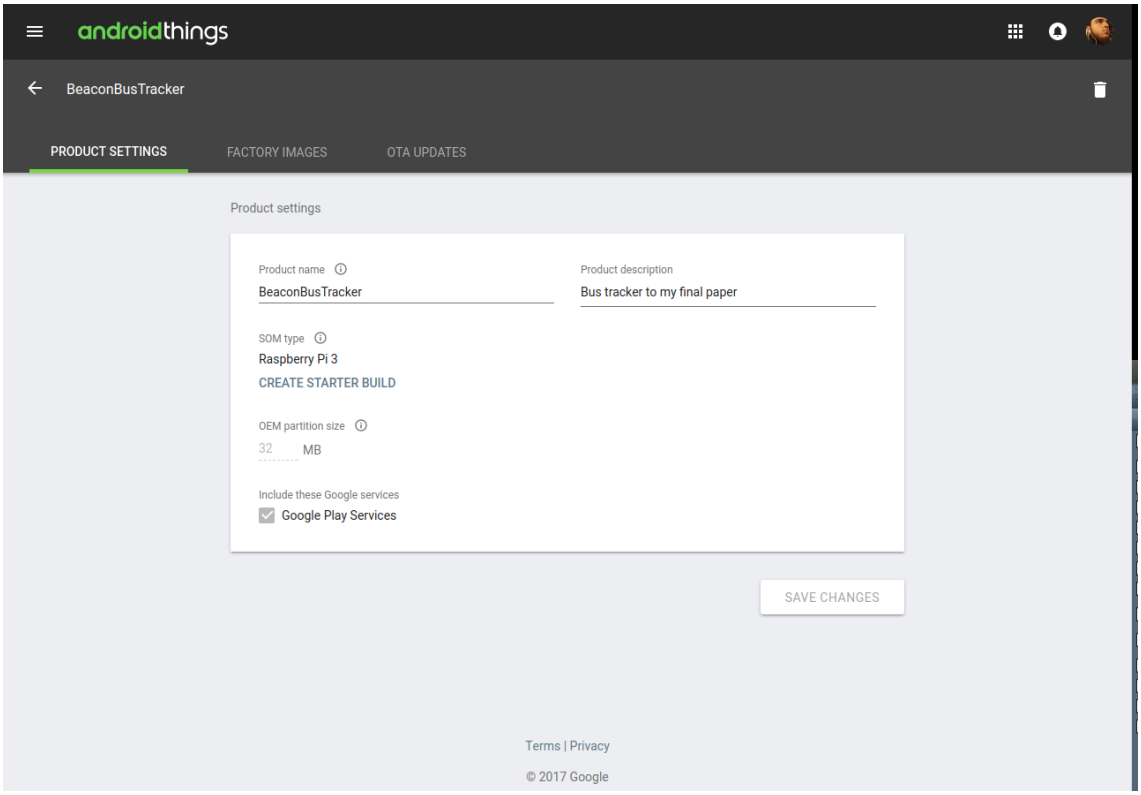

Figura 25 – Página de configurações do produto criado.

LocationManager

O LocationManager é um serviço nativo que observa constantemente os dados de geolocalização do dispositivo. Com ele é possível ser notificado sobre as mudanças da posição, podendo configurar ser notificado quando se entra em determinada região, a cada tempo determinado ou quando se move por uma distância pré-definida.

Este serviço abstrai toda lógica de acesso ao hardware e interpretação dos dados. Neste trabalho foi configurado para que ele notifique a geolocalização a cada 10 segundos.

Peripheral Driver Library

Não disponível na versão *Android* para *smartphones*, esta biblioteca oferece suporte ao *Android Things* para que possa adicionar sensores e atuadores a placa utilizada, neste caso o *Raspberry Pi 3*. Neste trabalho foi utilizado o *Driver GPS* desta biblioteca para poder configurar o módulo *NEO 6M*. O que possibilita o *LocationManager* obter os dados de geolocalização, já que não existe sensor *GPS* no *Raspberry Pi 3*.

3.3.2.2 IDE

Foi escolhido o Android Studio como IDE do projeto. Ela é desenvolvida pela IntelliJ e tida pelo Google como ferramenta oficial de desenvolvimento para aplicativos Android.

3.3.2.3 Animações

Um dos tópicos mais presentes sobre melhorar experiência do usuário em *Smartphones*, são as animações. Elas deixam o uso mais fluído e agradável para o usuário.

Como cita [LECHETA, 2015], "animações dão um acabamento profissional ao aplicativo, e costumam enriquecer a experiência do usuário". Um dos tópicos mais presentes sobre melhorar experiência do usuário em *smartphones*, são as animações, pois deixam o uso mais fluído e agradável para o usuário.

O *Android Things* provê uma *API* para animações que herdou da versão do *Android* de *smartphones*. Ela foi utilizada para melhorar a experiência de uso dos motoristas com o módulo, porém, foi observado uma baixa qualidade nas animações. O que fez ter o efeito contrário, pois passa a impressão de ser um sistema de baixo desempenho.

3.4 Aplicativo

3.4.1 IDE

Foi escolhido o Android Studio como IDE do projeto. Ela é desenvolvida pela IntelliJ e descrita pelo Google como a ferramenta oficial de desenvolvimento para aplicativos Android.

3.4.2 Áudio Descrição

Uma das funcionalidades do aplicativo é descrição em áudio da tela que o usuário está. O *TalkBack* fala para o usuário em qual componente ele está tocando, porém, não descreve em qual tela ele acabou de entrar.

A implementação por áudio descrição foi simples com uso da API nativa *TextToSpeech*, onde podemos passar textos personalizados e o serviço se encarrega de sintetizar a voz.

O uso de uma camada de DI (Injeção de Dependências) facilitou o processo de implementação, fazendo ela na camada de serviço e configurando a instanciação no padrão *Singleton*

para que todos que vão utilizar (nesse caso são os *presenters*), apenas solicitem a instância sendo passada por construtor.

3.4.3 Usabilidade

É uma boa prática no desenvolvimento de softwares, sempre confirmar se o usuário tem certeza que deseja executar alguma alteração que possa ter algum impacto no sistema ou em alguma funcionalidade, normalmente lançando alertas para garantir que o usuário não clicou sem querer em algum determinado botão, por exemplo.

Inicialmente foi pensado em usar um *dialog*, um pequeno quadrado que surge acima de todos componentes da tela com uma mensagem, para que o usuário confirme a ação de adicionar um ônibus como favorito, na tela de confirmação de acompanhamento. Ao testar a aplicação funcionando com *Talkback*, foi observado que o sistema descreve o botão com o seguinte texto: *"Adicionar aos favoritos. Botão, para acionar toque duas vezes"*. Esse texto já faz o usuário se assegurar da sua ação, tornando a prática de lançar um outro alerta ser algo desnecessário, fazendo o usuário ter um trabalho a mais de deslizar o dedo pela tela para encontrar os botões de *OK* e cancelar do *dialog*.

Com base nessas observações, não foi implementado *dialogs* de confirmação. Deixando a responsabilidade de afirmar as ações do usuário para o *Talkback* fazer. Embora seja uma pequena ação, tem grande impacto na usabilidade no quesito facilidade de uso do aplicativo por parte do usuário final, que são pessoas com deficiência visual.

Outro ponto observado durante os testes foi a rotação da tela. O sistema operacional Android permite que aplicativos implementem variações da tela de acordo com a orientação do dispositivo, retrato ou paisagem. Isso permite que o layout do aplicativo se adapte a nova disposição de espaço.

Pensando no usuário final, para saber onde está cada elemento, ele precisa deslizar o dedo pela tela para conhecer a localização de cada um. Se ao rotacionar o aparelho, a disposição dos elementos mudar, o usuário precisa verificar novamente onde está cada um.

Como este trabalho desenvolve um aplicativo com poucos elementos na tela para simplificar o uso por pessoas com deficiência visual, foi bloqueado a mudança de tela ao rotacionar o aparelho. Isso garante um melhor conforto ao usar o aplicativo.

3.5 Web service

3.5.1 Provedor de servidor

Para tornar a aplicação acessível ao público, torna-se necessário que o servidor fique disponível constantemente, mantendo a API sempre ativa.

Foram pesquisados três provedores de servidores, analisando qual deles apresentava melhores condições para realização do *deploy* da aplicação. Foram eles a GoDaddy (https://br.godaddy.com/), Umbler (https://www.umbler.com/) e LocaWeb (https://www.locaweb.com.br).

Primeiro fator, que fez o serviço da GoDaddy ser excluído da lista, foi a questão de período de teste. A GoDaddy possui um período nulo de teste do serviço. Já a Umbler e LocaWeb fornecem créditos para novos desenvolvedores.

Em seguida, foi observada a disponibilidade de informações sobre como realizar o *deploy* de um servidor em Node.js. Nesse aspecto, a LocaWeb não demonstra com clareza os procedimentos a serem seguidos.

A Umbler acabou sendo selecionada por ser muito intuitiva na hora de selecionar a linguagem da aplicação e qual o banco de dados a ser utilizado, com poucos cliques é possível iniciar um servidor funcional.

3.5.2 Arquitetura

Devido o uso do Express.js como intermediário para carregar os *middlewares*, a arquitetura utilizada para o desenvolvimento da aplicação segue o padrão MVC.

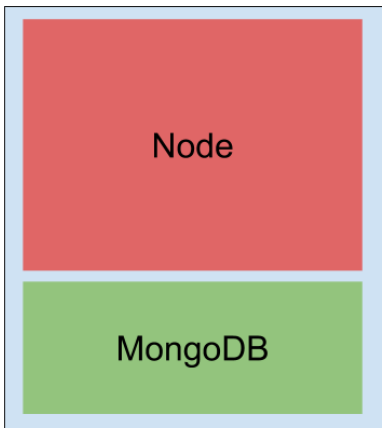

Figura 26 – Diagrama de blocos do Web Service.

Node Cuida da parte de escalabilidade e conexão do servidor com os usuários da aplicação.

MongoDB Representa o banco de dados da aplicação, com as *collections* necessárias para o funcionamento da aplicação.

O Node está subdividido da seguinte forma:

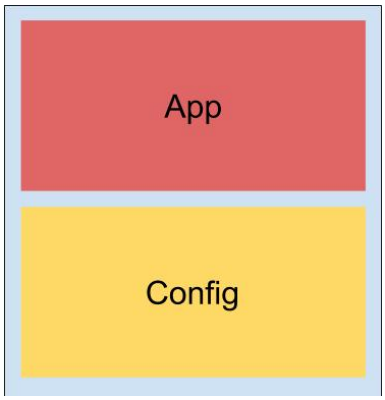

Figura 27 – Diagrama de blocos do Node.

App Conjunto com todas as funções da aplicação.

Config Possui os arquivos necessários para realizar as conexões com o servidor, banco de dados e o sistema de notificações.

Dentro do app é onde se observa a utilização da arquitetura MVC, ilustrada na Figura 28, representada da seguinte forma:

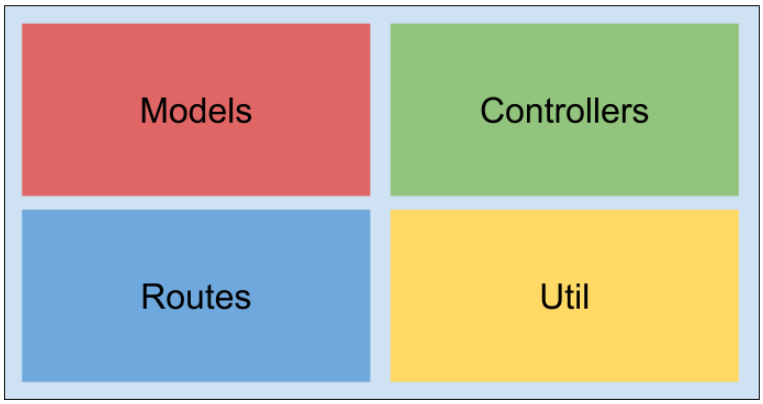

Figura 28 – Diagrama de blocos do diretório app.

Models Possui as regras de negócio da aplicação, com as informações necessárias para a criação das *collections* do MongoDB através do mongoose e os arquivos com as funções de previsão de tempo e de correção de posição.

Controllers Possui as funções que utilizam os dados do banco de dados para gerar as informações adequadas aos usuários.

Routes Possui os arquivos com as rotas dos *endpoints*, indicando qual arquivo e função dos *controllers* será utilizado para a operação necessária pelo usuário, seguindo o padrão REST.

Util Possui as fórmulas necessárias para o desenvolvimento de certas funções da aplicação.

3.5.3 Localização do ônibus

Ter conhecimento da posição do ônibus se faz necessário para realizar a previsão de chegada em um ponto de parada. Neste trabalho, sempre que o *Web Service* recebe as coordenadas geográficas do ônibus, elas são corrigidas para um ponto válido da rota que este está fazendo.

Quando o *Web Service* recupera as informações da rota do veículo, ele obtém todos os pontos referentes a ela, onde cada um tem informações de latitude, longitude e se este é um ponto de parada. Então é verificado qual ponto está mais próximo a localização recebida, o mais próximo é associado ao ônibus.

O cálculo de distância é feito a partir da fórmula *Haversine*, encontrado em [SCRIPTS, 2017].

$$a = \sin^2(\Delta\phi) + \cos\phi 1 * \cos\phi 2 * \sin^2(\Delta\lambda)$$
$$c = 2 * \mathrm{atan2}(\sqrt{a}, \sqrt{1-a})$$
$$d = R * c$$

Onde: $\Delta\phi = (latitude2 - latitude1), \phi 1 = latitude1, \phi 2 = latitude2, \Delta\lambda = longitude1 - longitude2$

Foi observado que seria gasto muito processamento nessa parte. Uma tentativa de diminuir isso, foi o uso de um algoritmo de busca binária.

Busca Binária

"Binary search is to algorithms what a wheel is to mechanics: It is simple, elegant, and immensely important." Udi Manber, Introduction to Algorithms

Para otimizar a busca pelo ponto mais apto, foi estudado a busca binária. Ela segue o paradigma de dividir para conquistar. Tendo como premissa um *array* de dados ordenados, primeiro observa os extremos do array, para em seguida observar o ponto médio entre eles. Caso um desses pontos seja igual ao valor procurado, assume ele como solução. Caso contrário se observa qual sub *array* é mais apto, início até o centro ou do centro até o fim. O mais apto é submetido a uma nova busca binária.

A vantagem em se utilizar uma busca binária ao invés de uma busca sequencial pode ser observado pelas suas complexidades. A busca binária possui complexidade $O(\log_2 n)$, enquanto a sequencial possui complexidade $O(n)$.

A tabela a seguir mostra o número de vezes que cada busca iria precisar fazer para chegar no ponto mais apto.

Rota	Número de Pontos	Busca Binária ($O(\log_2 n)$)	Busca Sequencial ($O(n)$)
1	410	$\simeq$9	410
2	500	$\simeq$9	500
3	675	$\simeq$10	675
4	708	$\simeq$10	708
5	725	$\simeq$10	725

A implementação da busca ficou da seguinte forma:

Primeiro é calculado a distância entre a localização recebida do ônibus e o primeiro ponto da rota. Em seguida, calculamos a distância entre a localização recebida e o último ponto da rota. Com isso temos a distânciaInicio e distânciaFim.

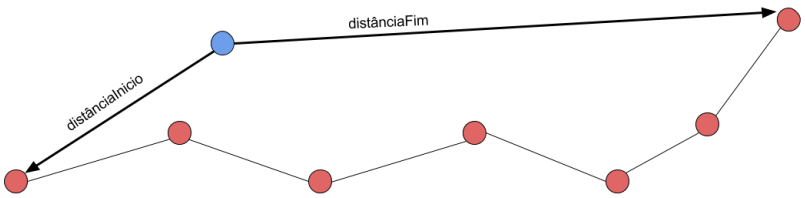

Figura 29 – Representação da busca binária I.

Um terceiro cálculo é feito, a distância entre a localização recebida e o ponto do meio da rota.

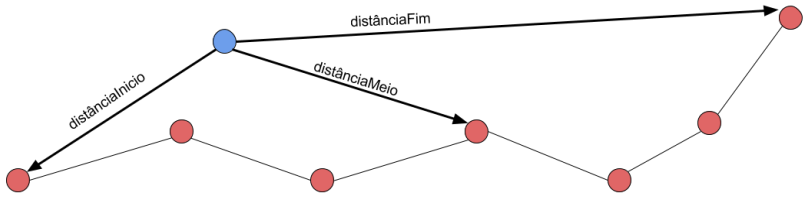

Figura 30 – Representação da busca binária II.

O algoritmo então verifica qual metade é mais apta, baseando-se nas distâncias mais curtas. Se distânciaFim é maior que distânciaInicio, é feito uma recursividade assumindo distânciaMeio como fim.

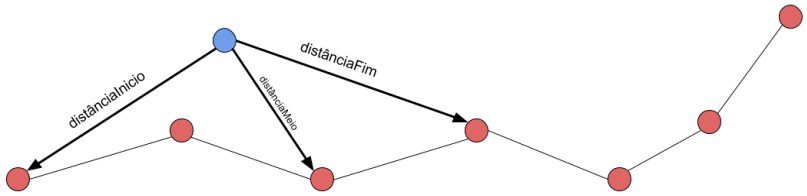

Figura 31 – Representação da busca binária III.

Fazendo isso sucessivamente, chega uma hora que não é mais possível dividir, então o ponto mais próximo assume o posto de mais apto e é associado ao ônibus.

Resultados

Após realizar testes, foi comprovado que embora exista a melhora em diminuir drasticamente o processamento, foi encontrado o problema do ponto correto estar em um sub *array* que não é considerado o mais apto, como ilustra a Figura 32.

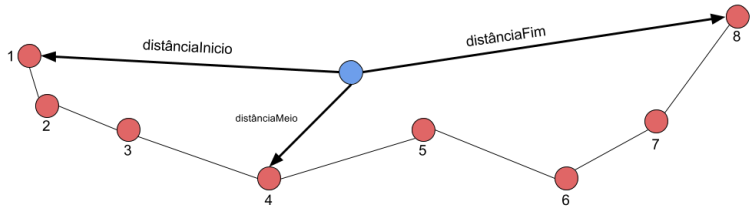

Figura 32 – Representação do problema da busca binária.

Como pode se observar na Figura 32, o ponto mais próximo em relação a posição recebida é o número 5. Quando se inicia a busca binária, o ponto número 4 é considerado o centro, tornando o sub *array* distânciaInicio até distânciaMeio ser o mais apto para se realizar uma busca interna, fazendo o ponto número 5 ser desconsiderado.

Embora a busca binária demonstre ser mais eficiente que a busca sequencial, a implementação neste problema, tendo como heurística a menor distância, demonstrou não ser suficiente para diminuir o processamento e encontrar a melhor solução.

3.5.4 Previsão de chegada

Para calcular a previsão de chegada de um ônibus, é necessário saber a última localização conhecida dele e o ponto de parada escolhido. Com essas duas informações pode-se recuperar todos dados necessários para o cálculo.

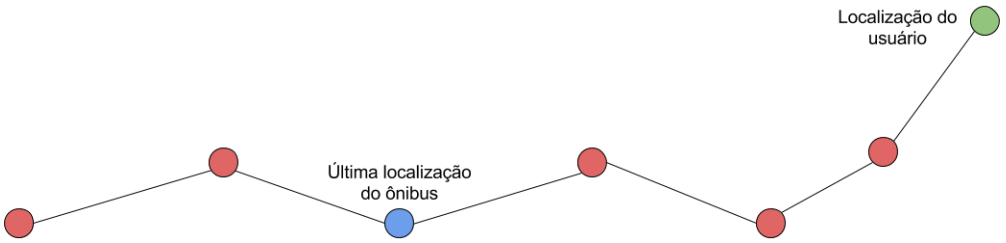

Figura 33 – Representação de uma rota.

Primeiro é recuperado todos os dados da rota que o ônibus está fazendo. Cada rota possui N pontos associados a ela, onde cada um tem a informação de latitude, longitude e se este é um ponto de parada. Sabendo em qual ponto da rota o ônibus está e qual ponto de parada o usuário se encontra, podemos pegar todos os pontos que estão entre eles e calcular a distância com a seguinte fórmula.

$$t = \frac{s}{v}$$

A partir disso se soma a distância dentre todos esses pontos, a partir do resultado se divide pela velocidade da pista e se obtém quanto tempo aproximado falta para o ônibus chegar na parada.

3.5.5 Alertar motorista sobre parar

O *Android Things* não suporta serviço de *push notification*. Para enviar mensagens instantâneas ao módulo foi utilizado o *Realtime Database*, do *Firebase*, que permite observar um nó. Sempre que existe uma alteração deste, quem está observando é notificado.

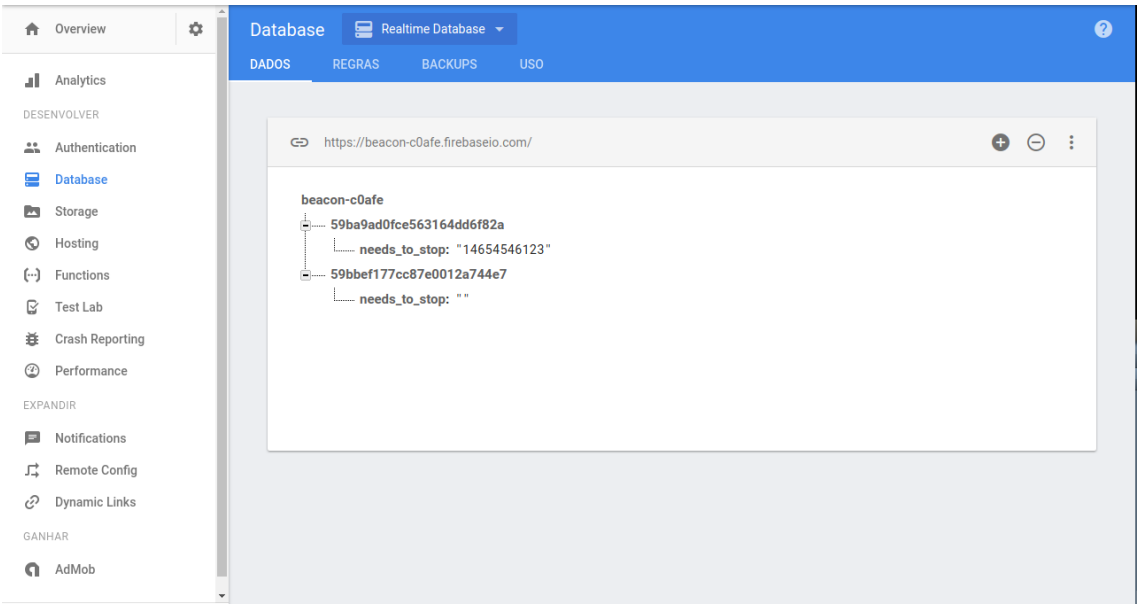

Figura 34 – Painel de controle do Firebase Realtime Database.

Quando um módulo do ônibus é conectado a internet, ele cria um nó com seu ID na base de dados e um nó filho intitulado *needs_to_stop*, e passa a observá-lo. Quando o valor do nó filho é alterado, o módulo então dispara uma notificação para o motorista sobre a necessidade de parar no próximo ponto de ônibus.

O responsável por alterar o valor do nó filho é o *Web Service*. Como visto na seção *Localização do Ônibus*, é feito uma varredura para descobrir o ponto da rota que o ônibus se encontra. Em seguida também é verificado qual o próximo ponto que ele vai passar. Caso o próximo ponto esteja na lista de requisições de parada, é então alterado o valor do nó filho para que aquele módulo alerte o motorista.

3.5.6 Alertar usuário que ônibus chegou

Quando o motorista chega na parada, ele pressiona um botão para anunciar sua chegada. Quando *Web Service* recebe esse alerta por meio de uma requisição *HTTP*, ele verifica quais

usuários estão aguardando aquele ônibus, então é disparado uma *push notification*, por meio do *Firebase*, avisando que o ônibus que ele aguarda está parado.

4 Considerações Finais

Com os avanços tecnológicos, surgem diversas ferramentas que facilitam a vida de seus usuários. Porém, nem sempre essas ferramentas alcançam a todos, sendo de difícil manuseio para determinados grupos, como, por exemplo, o de pessoas com deficiência visual. Os aplicativos de monitoramento de transportes públicos, que indicam uma previsão de quanto tempo até que o ônibus chegue na parada em que o usuário se encontra é um desses casos. Começando com o fato de ser necessário que o usuário ajuste manualmente a sua posição inicial num mapa de modo a melhorar a precisão de sua localização. Esse passo dificulta muito a utilização do aplicativo para um usuário deficiente visual. Mesmo os aplicativos que possuem uma integração com algum tipo de tecnologia assistiva, como o Talkback da Google, acabam possuindo alguma dificuldade de uso.

Neste trabalho foi possível desenvolver um aplicativo nativo em Android, adaptado com o Talkback e com feedback descritivo das telas por meio de áudio, uma aplicação Web implementando a API do sistema, além dos protótipos dos módulos do ponto de ônibus e dos ônibus. O módulo do ponto envia um sinal constante com o ID do ponto, com isso, o aplicativo pede para a aplicação Web uma lista com os ônibus que estão em circulação e que passam por aquele ponto, além de obter uma previsão para sua chegada. Após o usuário confirmar qual ônibus ele deseja esperar, é criada uma requisição que analisa as informações, alimentadas pelo módulo do ônibus, para disparar um aviso ao módulo do ônibus dizendo que é preciso parar na próxima parada por existir um deficiente visual esperando-o. O sistema se comportou como esperado, provendo facilidades de uso na hora de escolher o ônibus, demonstrando ser uma boa solução para facilitar o uso deste transporte público.

O objetivo principal deste trabalho foi desenvolver uma solução para facilitar a forma como pessoas com deficiência visual utilizam os ônibus, sem o foco de elaborar um sistema de previsão de horários mais preciso. Isso gerou a possibilidade de diversos aprimoramentos futuros que podem ser adicionados a este sistema, como: alterar a linha quando termina uma rota; melhorar a tratativa de velocidade do ônibus para considerar velocidades médias de um trecho possibilitando uma previsão de chegada mais precisa; e a implementação de uma interface gráfica para controlar os registros que existem no banco de dados da aplicação Web, a fim de

facilitar seu uso.

Referências

ALMEIDA, F. *MEAN - Full stack JavaScript para aplicações web com MongoDB, Express, Angular e Node*. [S.l.: s.n.], 2016. ISBN 9788555190469. Citado 2 vezes nas páginas 23 e 25.

ANDROID THINGS. *The ease and power of Android*. 2015. Disponível em: <https://developer.android.com/things/index.html>. Citado na página 21.

ANDROID, V. o. *O QUE É ANDROID*. 2017. Disponível em: <https://www.vivaoandroid.com.br/android/>. Citado na página 20.

APPLE INC. *Acessibilidade - Visão*. 2017. Disponível em: <https://www.apple.com/br/accessibility/iphone/vision/>. Citado na página 12.

ARDUINO. *Arduino Uno Rev3*. Disponível em: <https://store-cdn.arduino.cc/usa/catalog/product/cache/1/image/1800x/ea1ef423b933d797cfca49bc5855eef6/A/0/A000066_front_2.jpg>. Citado na página 18.

AVRAM ABEL TRADUZIDO POR MALARA, R. *Kotlin agora é uma linguagem oficial no Android*. 2017. Disponível em: <https://www.infoq.com/br/news/2017/06/android-kotlin>. Citado na página 23.

BERSCH, R. *Introdução à tecnologia assistiva*. 2013. Disponível em: <http://www.assistiva.com.br/Introducao_Tecnologia_Assistiva.pdf>. Citado na página 10.

CARNEIRO, C. *Beacon: o que é e quais suas utilizações mais inusitadas*. 2016. Disponível em: <http://usemobile.com.br/conheca-beacon/>. Citado na página 15.

CARNEIRO, C. *iBeacon Tudo que você precisa saber*. 2016. Disponível em: <http://ibeacon.blog.br/ibeacon-guia-completo/>. Citado na página 15.

CASTILHO, R. *Princípios SOLID: Princípio da Responsabilidade Única (SRP)*. 2013. Disponível em: <https://robsoncastilho.com.br/2013/02/06/principios-solid-principio-da-responsabilidade-unica-srp/>. Citado na página 22.

CCM. *Formatos e extensões de arquivos - Tipo MIME*. 2017. Disponível em: <http://br.ccm.net/contents/649-formatos-e-extensoes-de-arquivos-tipo-mime>. Citado na página 14.

DEVELOPER, A. *Intel Edison*. 2014. Disponível em: <https://developer.android.com/things/images/intel-edison-arduino-kit.png>. Citado na página 48.

DEVELOPERS, G. *Firebase Realtime Database*. 2017. Disponível em: <https://firebase.google.com/docs/database/?hl=pt-br>. Citado na página 16.

DORINA, F. *O que é deficiência*. 2017. Disponível em: <https://www.fundacaodorina.org.br/a-fundacao/deficiencia-visual/o-que-e-deficiencia/>. Citado na página 10.

FINZI, E. *O que é Push Notification*. 2016. Disponível em: <http://blog.cedrotech.com/o-que-e-push-notification/>. Citado na página 15.

GIANG, H. *How to Communicate Module Bluetooth HM-10 (4.0) With Android Smartphone and MCU*. 2016. Disponível em: <http://www.instructables.com/id/How-to-Communicate-Module-Bluetooth-HM-10-40-With-/>. Citado na página 17.

GIT. *Primeiros passos sobre controle de versão*. 2016. Disponível em: <https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o>. Citado na página 26.

GOMES ANDRé FARIA APUD TIKOV, S. *Uma rápida introdução ao REST*. 2009. Disponível em: <https://www.infoq.com/br/articles/rest-introduction>. Citado na página 25.

GOMES, H. S. *Moovit avisa em que parada usuário de transporte público deve descer*. 2015. Disponível em: <http://g1.globo.com/tecnologia/tem-um-aplicativo/noticia/2015/04/moovit-avisa-em-que-parada-usuario-de-transporte-publico-deve-descer.html>. Citado na página 11.

GOOGLE LLC. *Google Talkback*. 2017. Disponível em: <https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&hl=pt-BR>. Citado na página 12.

GOOGLE PLAY. *Acessibilidade - CittaMobi*. 2017. Disponível em: <https://play.google.com/store/apps/details?id=com.cittabus.acessibilidade&hl=pt_BR>. Citado na página 11.

IBGE. *Características Gerais, Religião, Deficiência*. 2010. Disponível em: <ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/xls/Brasil_xls.zip>. Citado na página 10.

IDC. *Smartphone OS Market Share, 2017 Q1*. 2017. Disponível em: <https://www.idc.com/promo/smartphone-market-share/os>. Citado na página 20.

INSTITUTO PADRE CHICO. *Home*. 2017. Disponível em: <http://www.padrechico.org.br/>. Citado na página 11.

INSTRUCTABLES. *Raspberry Pi & the Neo 6M GPS*. 2015. Disponível em: <https://cdn.instructables.com/FKL/Q62O/ICJLZ1YR/FKLQ62OICJLZ1YR.MEDIUM.jpg>. Citado na página 19.

JINAN HUAMAO. *Bluetooth*. 2017. Disponível em: <http://www.jnhuamao.cn/bluetooth.asp>. Citado na página 17.

KEWERSON, H. *Por que Kotlin foi criado?* 2017. Disponível em: <http://blog.triadworks.com.br/por-que-o-kotlin-existe>. Citado na página 23.

LECHETA, R. *Google Android*. [S.l.: s.n.], 2015. ISBN 9788575224403. Citado na página 51.

MARTIN, R. C. *Clean Architecture*. [S.l.]: Prentice Hall, 2012. Citado na página 22.

MCROBERTS, M. Arduino básico. *São Paulo: Novatec*, 2011. Citado na página 18.

MEIRELLES, F. S. *Pesquisa Anual do uso de TI-2016*. 2016. Disponível em: <http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/pesti2016gvciappt.pdf>. Citado na página 10.

NODEBR. *O que é Node.js*. 2016. Disponível em: <http://nodebr.com/o-que-e-node-js/>. Citado na página 24.

PIEKARSKI, W. *Announcing updates to Googles Internet of Things platform: Android Things and Weave*. 2016. Disponível em: <https://android-developers.googleblog.com/2016/12/announcing-googles-new-internet-of-things-platform-with-weave-and-android-things.html>. Citado na página 21.

RASPBERRY PI. *Raspberry Pi 3 imagem*. 2017. Disponível em: <https://www.raspberrypi.org/app/uploads/2017/05/Raspberry-Pi-3-462x322.jpg>. Citado na página 16.

SAES, B. *O que são Beacons e como eles mudarão a sua rotina?* 2014. Disponível em: <http://www.impacta.com.br/blog/2014/12/09/o-que-sao-beacons-como-mudarao-rotina/>. Citado na página 15.

SAÚDE, P. 2014. Disponível em: <http://portalsaude.saude.gov.br/index.php/o-ministerio/principal/secretarias/509-sas-raiz/dapes/saude-da-pessoa-com-deficiencia/l2-saude-da-pessoa-com-deficiencia/10250-comite-de-ajudas-tecnicas>. Citado na página 10.

SCHMITZ, D. *Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar*. 2015. Disponível em: <https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/>. Citado na página 27.

SCRIPTS, M. T. *Calculate distance, bearing and more between Latitude/Longitude points*. [S.l.], 2017. Disponível em: <http://www.movable-type.co.uk/scripts/latlong.html>. Citado na página 55.

SOUSA, J. P. C. d. Extração e análise de memória volátil em ambientes android: uma abordagem voltada à reconstrução de trajetórias. 2017. Citado na página 19.

TEIXEIRA, F. *Tudo o que você precisa saber para começar a brincar com iBeacons*. 2014. Disponível em: <https://brasil.uxdesign.cc/tudo-o-que-voc%C3%AA-precisa-saber-para-come%C3%A7ar-a-brincar-com-ibeacons-fdf5847e640b>. Citado na página 12.

UPTON, E.; HALFACREE, G. *Raspberry Pi – Manual do Usuário*. [S.l.], 2017. Disponível em: <https://books.google.com.br/books?id=xD41DwAAQBAJ>. Citado na página 16.

VIEIRA, L. *CittaMobi calcula em tempo real quanto seu ônibus vai demorar a chegar*. 2015. Disponível em: <http://www.techtudo.com.br/tudo-sobre/cittamobi.html>. Citado na página 11.

Apêndices

APÊNDICE A – Diagrama de Classes

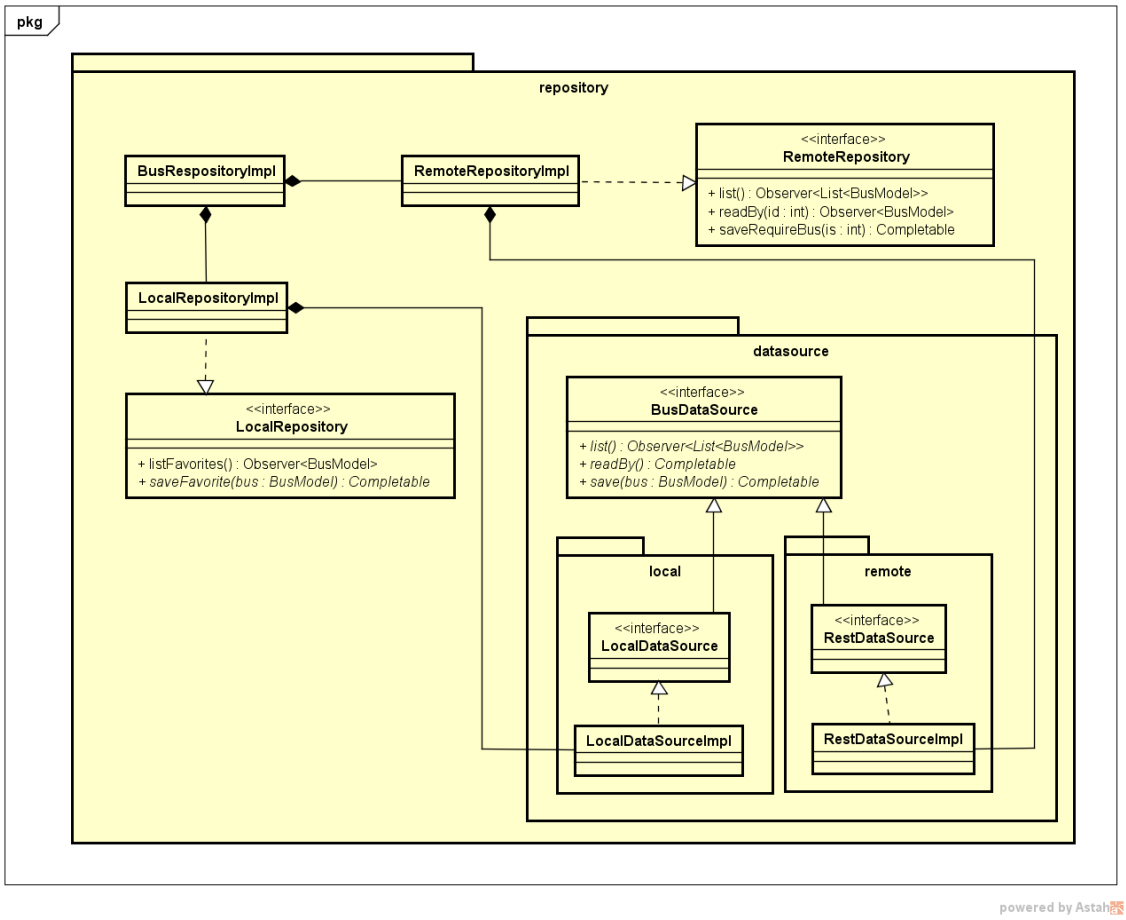

Figura 35 – Diagrama de classes da camada de dados.

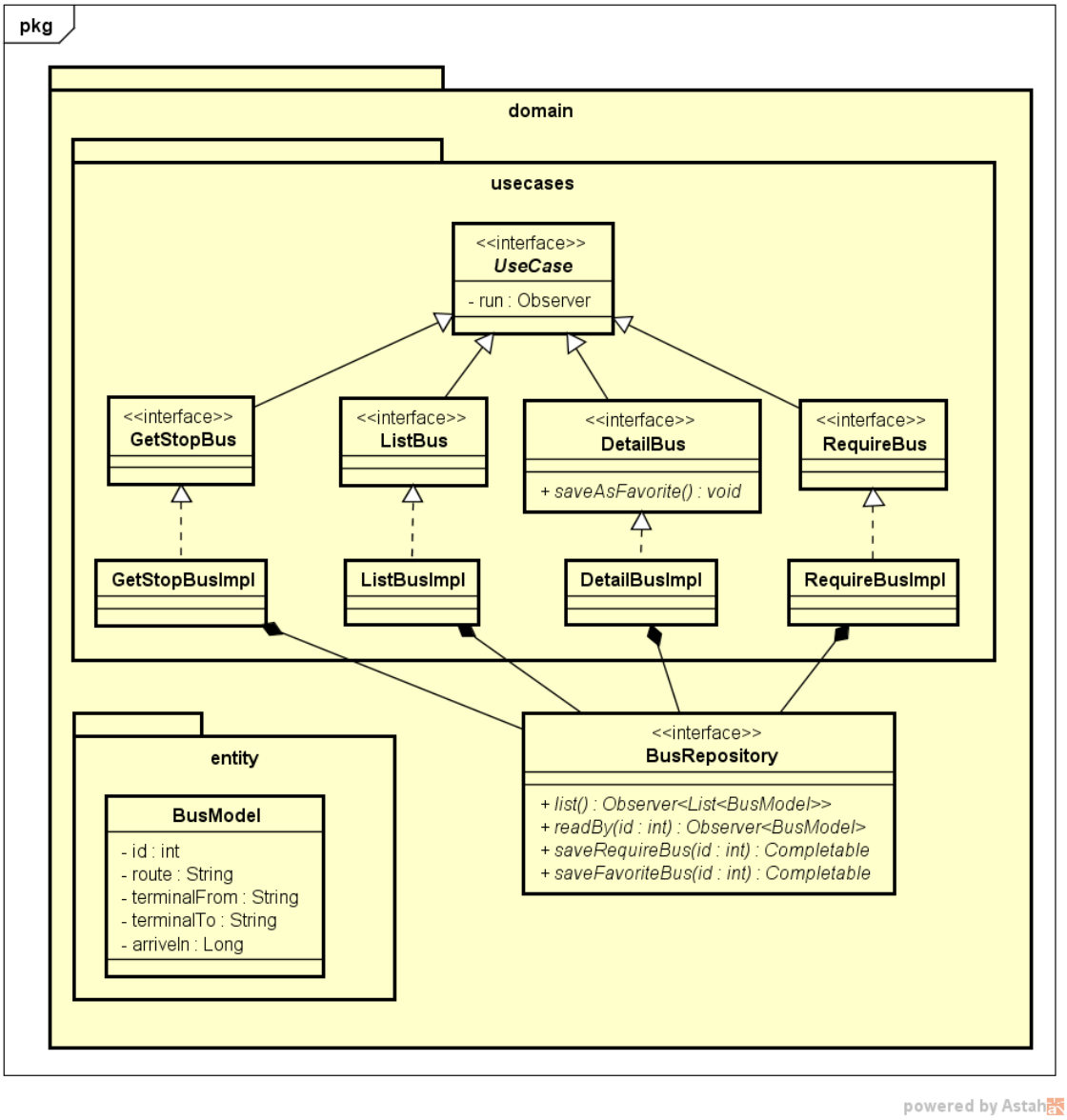

Figura 36 – Diagrama de classes da camada de domínio.

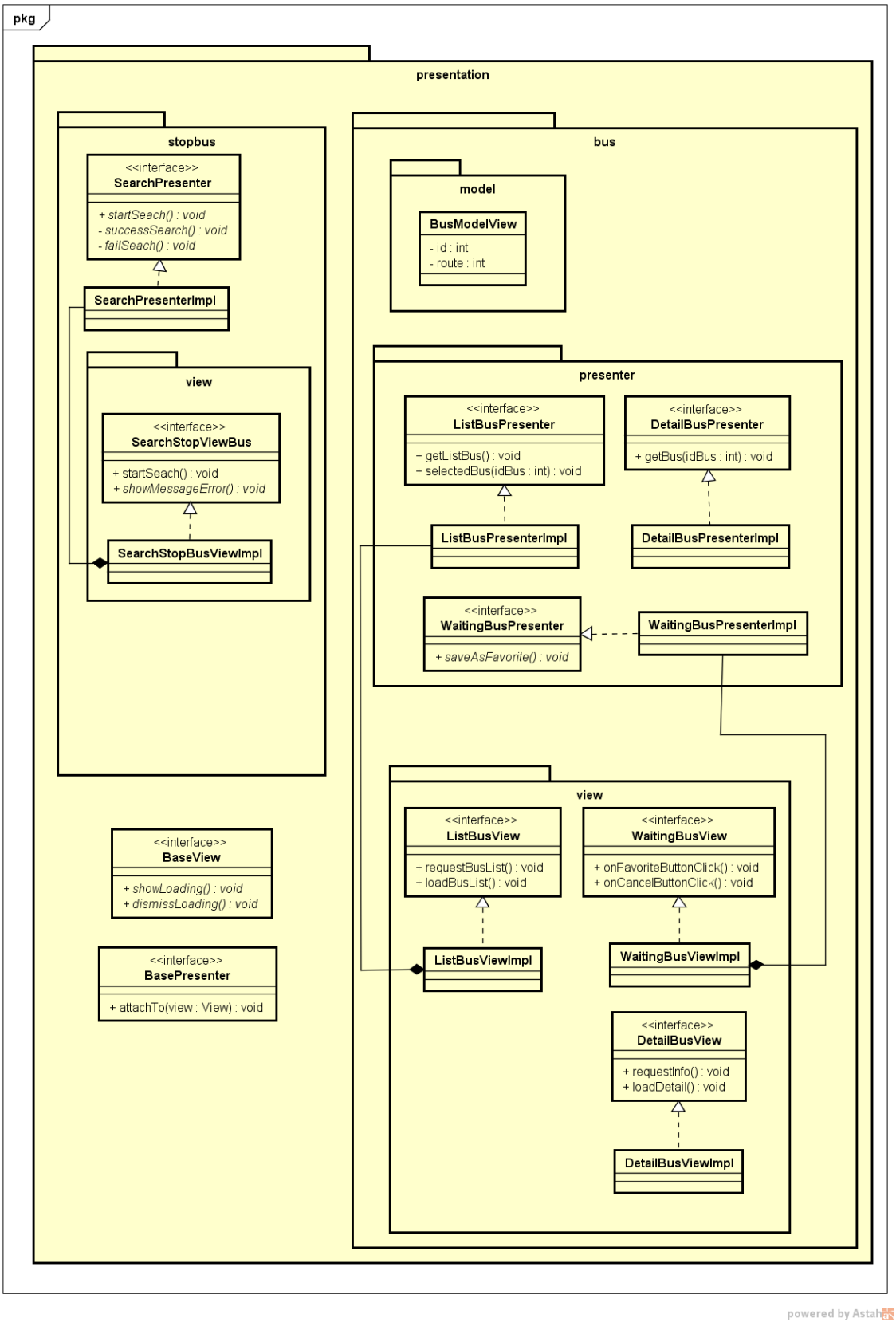

Figura 37 – Diagrama de classes da camada de apresentação.

APÊNDICE B – Wireframes

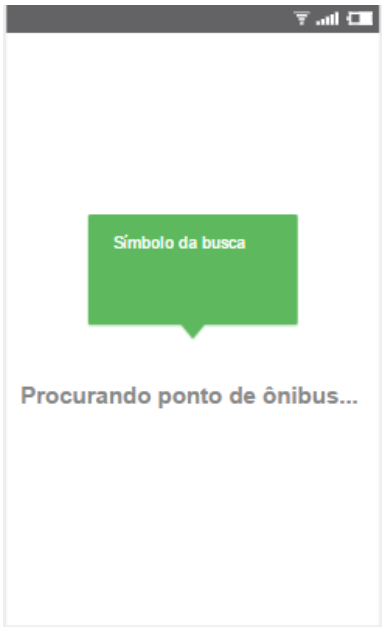

Figura 38 – Wireframe da tela do aplicativo ao buscar por um ponto de ônibus próximo.

Figura 39 – Wireframe da tela do aplicativo ao buscar por um ponto de ônibus na API.

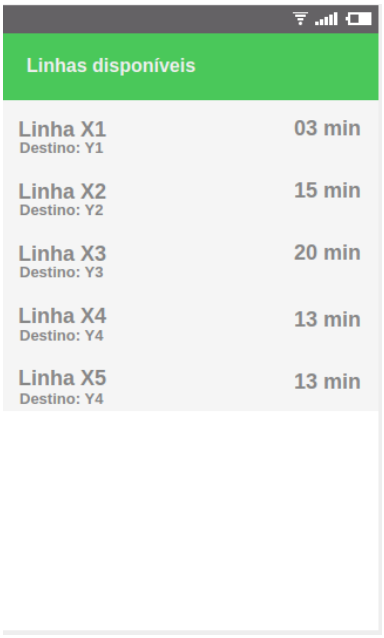

Figura 40 – Wireframe da tela do aplicativo com ônibus disponíveis.

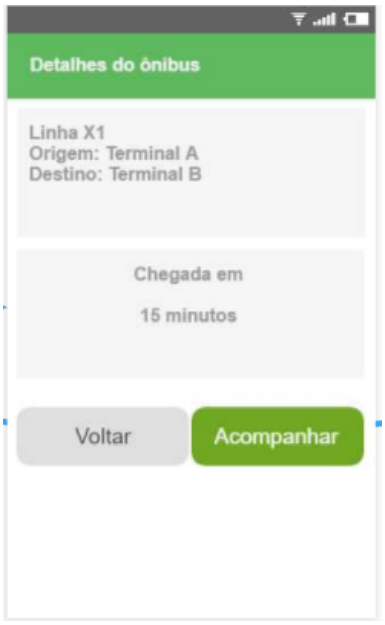

Figura 41 – Wireframe da tela do aplicativo com detalhes de um ônibus.

Figura 42 – Wireframe da tela do aplicativo sobre a solicitação de um ônibus.

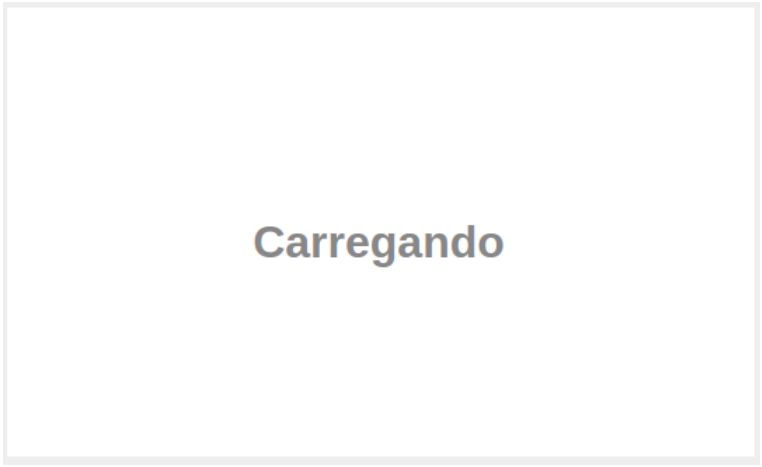

Figura 43 – Wireframe da tela do módulo do ônibus quando está carregando informações.

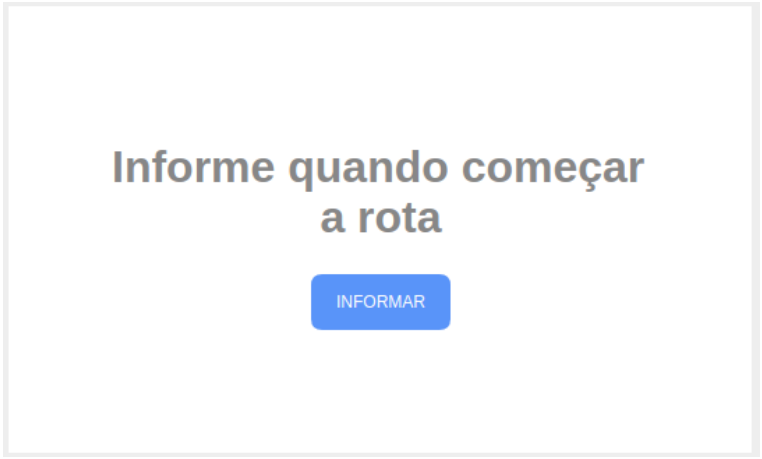

Figura 44 – Wireframe da tela do módulo do ônibus aguardando motorista iniciar rota.

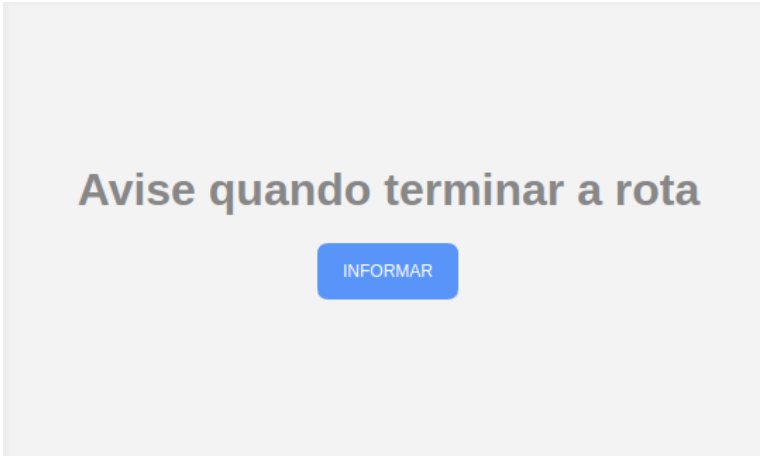

Figura 45 – Wireframe da tela do módulo do ônibus quando motorista está fazendo a rota.

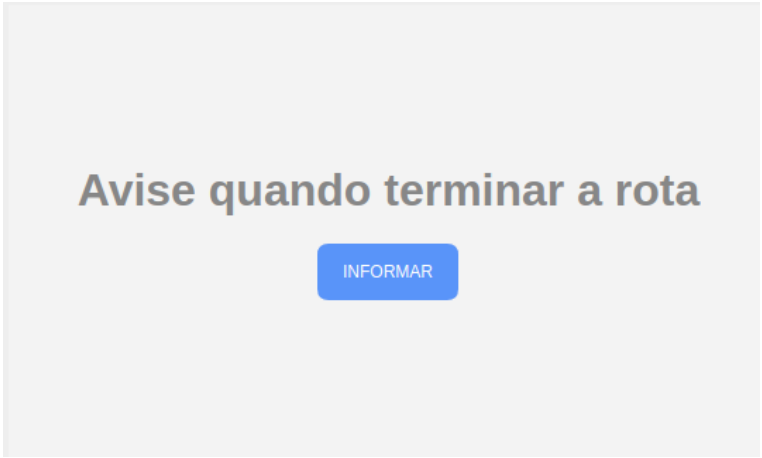

Figura 46 – Wireframe da tela do módulo do ônibus quando motorista está fazendo a rota.

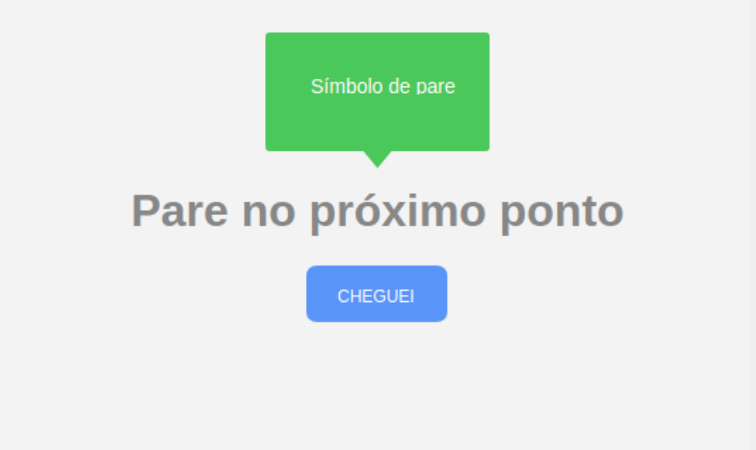

Figura 47 – Wireframe da tela do módulo do ônibus quando motorista precisar parar no próximo ponto.

APÊNDICE C – Testes de aceitação

Os testes de aceitação visam fazer um conjunto de testes isolados uns dos outros para garantir a qualidade e as funcionalidades do aplicativo.

Cada teste está organizado na seguinte estrutura:

ID: x.x.x	**Título:**	**Nível do teste**
Descrição:		
Necessário para:		
Nota de teste:		
Etapas:		
Resultado: Sucesso/Falha		

- **ID:** Identificador único para um teste.
- **Título:** Texto breve que identifica o que será testado.
- **Nível do teste:** Identifica se o teste é crítico para o sistema.
- **Descrição:** Texto explicando com mais detalhes o que será testado.
- **Necessário para:** Cliente que fará uso da funcionalidade.
- **Nota de teste:** Observações a serem feitas antes de realizar o teste.
- **Etapas:** Passo a passo para reprodução do teste.
- **Resultado:** Apresenta o resultado do teste realizado.

Nível de teste

Os sistemas possuem diversas funcionalidades e pré-requisitos para funcionar corretamente. Cada uma tem seu grau de importância para o sistema, que dependendo deste grau, pode afetar totalmente ou parcialmente seu uso caso esteja com algum problema na implementação. Por isso, é de extrema importância identificar o quanto um teste pode impactar no sistema, para quando falhar, gerar um alerta de mau funcionamento.

Os níveis dos testes, indicados no canto superior direito da tabela de cada teste, servem para alertar o quão importante o teste significa para o correto funcionamento do sistema que a funcionalidade pertence, podendo apresentar dois níveis: Crítico ou Atenção.

Crítico: Representa uma funcionalidade de extrema importância para o correto funcionamento do sistema. Caso este teste falhe, o sistema não terá o comportamento necessário para funcionar como o usuário deseja. Sua indicação apresenta um fundo vermelho.

Atenção: Representa uma funcionalidade importante para o sistema. Caso o teste falhe, não afetará o funcionamento do sistema. Pode impactar em performance ou usabilidade reduzida, mas não em valores entregues ao usuário. Sua indicação apresenta um fundo azul.

Módulo do ponto de ônibus (*Beacon*)

Emissão de sinal

ID: 1.1.1	**Título:** Emissão de ID único	**Crítico**
Descrição: Este teste irá verificar se o módulo está emitindo um sinal com id único de acordo com padrões estabelecidos.		
Necessário para: Aplicativo		
Nota de teste: O sinal emitido é de acordo com a tecnologia *Bluetooth BLE* (4.+). O canal utilizado para a emissão do ID é o id1 de acordo com as especificações dos *Beacons*. O padrão estabelecido para o ID emitido é 655xxxx, sendo x qualquer valor do conjunto dos números Naturais.		
Etapas: 1. Ativar o módulo 2. Receber o sinal 3. Verificar se o canal "id1"está de acordo com o padrão		
Resultado:		

Módulo do ônibus

Geolocalização

ID: 2.1.1	**Título:** Emissão de geolocalização	**Crítico**
Descrição: Este teste irá verificar se o módulo está enviando a geolocalização do veículo		
Necessário para: *Web service*		
Nota de teste: O envio dos dados deve estar de acordo com a notação JSON. Deve ser enviado em determinados intervalos de tempo, conforme definido no *web service*.		
Etapas: 1. Ativar o módulo 2. Verificar se o *Web Service* está recebendo a geolocalização 3. Verificar se os dados recebidos estão chegando de acordo com o tempo definido		
Resultado:		

Interação com motorista

ID: 2.2.1	**Título:** Emissão de alerta de parada	**Crítico**
Descrição: Este teste irá verificar se o módulo está recebendo notificações de alerta e notificando ao motorista sobre a necessidade de parada no próximo ponto		
Necessário para: Módulo de ônibus		
Nota de teste: O envio do alerta se faz por meio do serviço do *Firebase Realtime Database*.		
Etapas: 1. Ativar o módulo 2. Alterar a variável no *Firebase Realtime Database* de alerta de parada 3. Verificar se houve a emissão do alerta de parada		
Resultado:		

Aplicativo Mobile

Instalação

ID: 3.1.1	**Título:** Instalação realizada com sucesso	**Crítico**
Descrição: Este teste irá verificar se o aplicativo foi instalado com sucesso		
Necessário para:		
Nota de teste: A versão do Android instalada deve ser no mínimo a versão Jelly Beans (v4.1).		
Etapas: 1. Instale a aplicação no dispositivo 2. Navegue até a lista de aplicativos 3. Verifique se o aplicativo *Beacon* está listado.		
Resultado:		

ID: 3.1.2	**Título:** Talkback ativo	**Atenção**
Descrição: Este teste irá verificar se o talkback está ativo		
Necessário para:		
Nota de teste:		
Etapas: 1. Navegue pela tela passando pelos ícones disponíveis 2. Verifique se está sendo emitido a áudio descrição dos ícones.		
Resultado:		

Tela

ID: 3.2.1	**Título:** Áudio descrições	**Atenção**
Descrição: Este teste irá verificar se o talkback está conseguindo ler os componentes da tela no aplicativo		
Necessário para: A navegação pelos componentes visuais deve ser de acordo com as regras de navegação do Talkback.		
Nota de teste:		
Etapas: 1. Abra o aplicativo 2. Navegue pelos componentes visuais. 3. Verifique se está sendo emitido a áudio descrição dos componentes.		
Resultado:		

ID: 3.2.2	**Título:** Orientação da tela	**Atenção**
Descrição: Este teste irá verificar se o aplicativo funciona corretamente ao rotacionar a tela.		
Necessário para:		
Nota de teste: Para melhorar usabilidade, a mudança de layout ao rotacionar é bloqueada.		
Etapas: 1. Segure o dispositivo na posição *portrait*. 2. Abra o aplicativo. 3. Gire o dispositivo para posição *landscape*. 4. Verifique se o aplicativo não rotacionou a tela.		
Resultado:		

Geolocalização

ID: 3.3.1	**Título:** Reconhecimento do ponto de ônibus	**Crítico**
Descrição: Este teste irá verificar se o aplicativo está reconhecendo o ID emitido pelo módulo do ponto ônibus		
Necessário para: *Web Service*		
Nota de teste: O reconhecimento do ID deve seguir as especificações do módulo do ponto de ônibus.		
Etapas: 1. Ligar o *bluetooth* do dispositivo mobile 2. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido 3. Visualizar a tela de *loading* 4. Aguardar carregar lista com ônibus		
Resultado:		

Ônibus disponíveis

ID: 3.4.1	**Título:** Listagem de ônibus disponíveis	**Crítico**
Descrição: Este teste irá verificar se o aplicativo está carregando corretamente os ônibus disponíveis para um determinado ponto de ônibus		
Necessário para: Usuário		
Nota de teste: O reconhecimento do ID deve seguir as especificações do módulo do ponto de ônibus.		
Etapas: 1. Ligar o *bluetooth* do dispositivo mobile. 2. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 3. Visualizar a tela de *loading*. 4. Aguardar carregar lista com ônibus disponíveis para aquele ID módulo de ponto de ônibus.		
Resultado:		

ID: 3.4.2	**Título:** Detalhes dos ônibus disponíveis	**Crítico**
Descrição: Este teste irá verificar se o aplicativo está exibindo os detalhes corretamente dos ônibus disponíveis.		
Necessário para: Usuário		
Nota de teste: O reconhecimento do ID deve seguir as especificações do módulo do ponto de ônibus.		
Etapas: 1. Ligar o bluetooth do dispositivo mobile. 2. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 3. Visualizar a tela de loading. 4. Aguardar carregar lista com ônibus disponíveis para aquele ID módulo de ponto de ônibus. 5. Selecionar um ônibus na lista. 6. Verificar se as informações estão de acordo com a lista.		
Resultado:		

ID: 3.4.3	**Título:** Requisitar um dos ônibus disponíveis	**Crítico**
Descrição: Este teste irá verificar se o aplicativo está realizando a requisição de um ônibus disponível.		
Necessário para: Usuário		
Nota de teste: Este teste não aborda a validação no *web service*		
Etapas: 1. Ligar o *bluetooth* do dispositivo *mobile*. 2. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 3. Aguardar carregar lista com ônibus disponíveis para aquele ID módulo de ponto de ônibus. 4. Selecionar um ônibus na lista. 5. Clicar no botão Solicitar Ônibus. 6. Aguardar mensagem de solicitação realizada com sucesso.		
Resultado:		

ID: 3.4.4	**Título:** Cancelar a requisição de um dos ônibus disponíveis	**Crítico**
Descrição: Este teste irá verificar se o aplicativo está realizando o cancelamento da requisição de um ônibus disponível.		
Necessário para: Usuário		
Nota de teste: Este teste não aborda a validação no web service		
Etapas: 1. Ligar o bluetooth do dispositivo mobile. 2. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 3. Aguardar carregar lista com ônibus disponíveis para aquele ID módulo de ponto de ônibus. 4. Selecionar um ônibus na lista. 5. Clicar no botão Solicitar Ônibus. 6. Aguardar mensagem de solicitação realizada com sucesso. 7. Na tela de requisição bem sucedida, clicar no botão Cancelar. 8. Aguardar mensagem de sucesso.		
Resultado:		

Rede

ID: 3.5.1	**Título:** Alerta sobre ausência de conexão	**Crítico**
Descrição: Este teste irá verificar se o aplicativo identifica o estado da conexão com a rede. Caso esteja desligado, deve alertar o usuário		
Necessário para: Usuário		
Nota de teste:		
Etapas: 1. Abrir o aplicativo *Beacon* 2. Visualizar um dialog com alerta sobre ausência de conexão.		
Resultado:		

ID:	**Título:** Alerta sobre *bluetooth* desligado	**Crítico**
Descrição: Este teste irá verificar se o aplicativo identifica o estado do *Bluetooth*. Caso esteja desligado, deve alertar o usuário		
Necessário para: Usuário		
Nota de teste:		
Etapas: 1. Desligar o *Bluetooth* do dispositivo *mobile* 2. Abrir o aplicativo *Beacon* 3. Visualizar um dialog com alerta sobre o *Bluetooth* desligado		
Resultado:		

Acessibilidade

ID: 3.5.1	**Título:** Alerta sonoro sobre busca do ponto de ônibus	**Atenção**
Descrição: Este teste irá verificar se o aplicativo emite um aviso sonoro indicando ter encontrado um ponto de ônibus.		
Necessário para: Usuário		
Nota de teste:		
Etapas: 1. Ligar o bluetooth do dispositivo mobile 2. Deixar o alto falante do dispositivo mobile ligado 3. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 4. Verificar se foi emitido um aviso sonoro indicando qual ponto de ônibus foi encontrado.		
Resultado:		

ID: 3.5.2	**Título:** Alerta sonoro sobre quantidade de ônibus disponíveis	**Atenção**
Descrição: Este teste irá verificar se o aplicativo emite um aviso sonoro indicando o número de ônibus disponíveis.		
Necessário para: Usuário		
Nota de teste:		
Etapas: 1. Ligar o bluetooth do dispositivo mobile 2. Deixar o alto falante do dispositivo mobile ligado 3. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 4. Verificar se foi emitido um aviso sonoro indicando qual ponto de ônibus foi encontrado. 5. Verificar se foi emitido um aviso sonoro indicando o número de ônibus disponíveis.		
Resultado:		

ID:	**Título:** Alerta sonoro sobre detalhes de um ônibus disponível	**Atenção**
Descrição: Este teste irá verificar se o aplicativo emite um aviso sonoro indicando os detalhes de um ônibus disponível.		
Necessário para: Usuário		
Nota de teste:		
Etapas: 1. Ligar o bluetooth do dispositivo mobile 2. Deixar o alto falante do dispositivo mobile ligado 3. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 4. Selecionar um ônibus disponível. 5. Na tela de detalhes, verificar se foi emitido aviso sonoro sobre os detalhes do ônibus e sua previsão de chegada.		
Resultado:		

ID: 3.5.4	**Título:** Alerta sonoro Requisitando	**Atenção**
Descrição: Este teste irá verificar se o aplicativo emite um aviso sonoro indicando estar realizando a requisição de um ônibus.		
Necessário para: Usuário		
Nota de teste:		
Etapas: 1. Ligar o *Bluetooth* do dispositivo *mobile*. 2. Deixar o alto falante do dispositivo mobile ligado. 3. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 4. Selecionar um ônibus disponível. 5. Na tela de detalhes, clicar em Solicitar Ônibus. 6. Verificar se foi emitido o aviso sonoro alertando estar requisitando um ônibus		
Resultado:		

ID: 3.5.5	**Título:** Alerta sonoro sobre sucesso da requisição	**Atenção**
Descrição: Este teste irá verificar se o aplicativo emite um aviso sonoro indicando ter tido sucesso para fazer requisição.		
Necessário para: Usuário		
Nota de teste:		
Etapas: 1. Ligar o *Bluetooth* do dispositivo *mobile*. 2. Deixar o alto falante do dispositivo mobile ligado. 3. Abrir o aplicativo *Beacon* próximo a um módulo de ponto de ônibus válido. 4. Selecionar um ônibus disponível. 5. Na tela de detalhes, clicar em Solicitar Ônibus. 6. Aguardar emitir sinal sonoro sobre o status Requisitando. 7. Verificar se foi emitido sinal sonoro sobre sucesso da requisição.		
Resultado:		

Web Service

ID: 4.1.1	**Título:** Teste de listagem de ônibus	**Crítico**
Descrição: Este teste irá verificar se o *web service* consegue obter a lista com os ônibus próximos		
Necessário para: Aplicativo		
Nota de teste: Esse teste verificará se o *web service* obtém a lista de ônibus que passarão pela parada correspondente		
Etapas: 1. Enviar o ID da parada. 2. Buscar os ônibus que possuem o mesmo ID da parada. 3. Verificar as informações enviadas para o aplicativo.		
Resultado:		

ID: 4.1.2	**Título:** Teste de resposta de previsão de chegada	**Crítico**
Descrição: Esse teste verificará o tempo demorado para calcular a previsão de chegada		
Necessário para: Aplicativo		
Nota de teste: Deve ser calculado com antecedência o tempo de chegada entre a ultima posição conhecida do ônibus e a posição de um ponto de ônibus para comparar os resultados do teste.		
Etapas: 1. Enviar o ID do ônibus escolhido 2. Verificar se a previsão de chegada é igual a calculada antes de iniciar o teste.		
Resultado:		

Requisições

ID: 4.2.1	**Título:** Teste de cadastro de requisição de acompanhamento	**Crítico**
Descrição: Esse teste verificará a funcionalidade de cadastro de requisição do ônibus escolhido pelo usuário		
Necessário para: Aplicativo		
Nota de teste:		
Etapas: 1. Enviar os IDs da parada e do ônibus escolhido pelo usuário. 2. Gerar um ID de requisição de acompanhamento para o usuário. 3. Verificar o ID de requisição de acompanhamento no aplicativo.		
Resultado:		

ID: 4.2.2	**Título:** Teste de cancelamento de requisição de acompanhamento	**Atenção**
Descrição: Esse teste verificará a funcionalidade de cancelamento de requisição do ônibus escolhido pelo usuário		
Necessário para: Aplicativo		
Nota de teste: Este teste não é crítico pois não compromete na performance do sistema.		
Etapas: 1. Cancelar a requisição de acompanhamento do ônibus 2. Enviar o ID de requisição de acompanhamento 3. Remover a requisição referente ao ID recebido 4. Verificar a confirmação de cancelamento		
Resultado:		

ID: 4.2.3	**Título:** Teste de notificação de parada	**Crítico**
Descrição: Esse teste verificará o envio de avisos ao módulo do ônibus sobre a existência de passageiros na próxima parada		
Necessário para: Módulo do ônibus		
Nota de teste:		
Etapas: 1. Enviar uma requisição para acompanhamento de veículo 2. Verificar se o *Web Service* alterou o parâmetro corretamente no *Firebase Realtime Database*, conforme documentação.		
Resultado:		