**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**Faculdade de Tecnologia Rubens Lara**

**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

**BRUNO MENEZES**
**DIEGO DA SILVA LIMA**
**FELIPE SCARPITTA SUCCI**
**NATHAN DE MATOS BARROS**

# APLICAÇÃO DE AUXÍLIO PARA TRANSPORTE PÚBLICO

**Santos, SP**
**2016**

**BRUNO MENEZES**
**DIEGO DA SILVA LIMA**
**FELIPE SCARPITTA SUCCI**
**NATHAN DE MATOS BARROS**

# APLICAÇÃO DE AUXÍLIO PARA TRANSPORTE PÚBLICO

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia Rubens Lara, como exigência para a obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

**Orientador: Prof. Me. Alexandre Garcia de Oliveira**

**Santos, SP**
**2016**

*“Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar a onde a maioria não chega, faça o que a maioria não faz.”*

**Bill Gates**

# DEDICATÓRIA

O presente trabalho é dedicado aos nossos pais, por todo o apoio, conforto e incentivo que recebemos ao longo de nossas vidas e principalmente, por acreditarem em nós. A vocês, todo o nosso amor.

À nossa família, por ser nossa base, alicerce e responsável pela transmissão de valores que hoje fazem parte de nós.

A todos os entes queridos, por todo apoio, carinho e compreensão ao longo deste período.

# AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte desta fase importante de nossas vidas. Portanto, desde já pedimos desculpas àquelas que não estão presentes entre estas palavras, mas que podem estar certas de que fazem parte de nossos pensamentos e de nossa gratidão.

Agradecemos primeiramente a Deus, por nos amparar e nos dar força para superar qualquer dificuldade.

A nossas famílias, que estiveram sempre presentes nos incentivando e apoiando.

Aos nossos professores, que contribuíram grandemente para nosso desenvolvimento como profissionais.

Agradecemos também ao Diego Braga e Diego Vieira pelo suporte na utilização de algumas tecnologias usadas no projeto.

Ao nosso orientador Alexandre Garcia por se disponibilizar e nos auxiliar nesta importante fase.

# RESUMO

Este projeto trata do desenvolvimento de uma aplicação móvel híbrida capaz de auxiliar a população que utiliza o transporte público na cidade de Santos. Esta ferramenta identifica a localização atual do usuário e sugere as melhores opções de ônibus para que a pessoa possa alcançar seu destino, tendo mais de uma opção, podendo escolher o trajeto que mais lhe convém em certa situação. Foi feita uma pesquisa de campo, na qual foi verificado que parte dos cidadãos que frequentam a cidade possuem dificuldade ao saber quais linhas poderiam pegar e em qual parada deveriam descer para se locomover de um ponto a outro da cidade. Com base nisso, foram estudadas tecnologias recentes de grande desempenho para criação da aplicação, bem como metodologias ágeis para desenvolvimento do sistema móvel.

**Palavras-chave:** aplicação, transporte público, móvel, híbrido.

# ABSTRACT

This project is about the development of a hybrid mobile application capable of helping the population that uses public transportation in the city of Santos. This tool identifies the user's current location and suggests the best bus options for the person to reach its destination, having more than one option, choosing the path that suits you in a certain situation. A field research was conducted, where it was found that citizens who attend the city have difficulty to know which lines they could take and which bus stop they should get off the bus to get around from one point to another of the city. Based on this, recent technologies of high-performance implementation for creating the application were studied, as well as agile methodologies for mobile development system.



**Keywords:** application, public transportation, mobile, hybrid.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE GRÁFICOS

# SUMÁRIO

# 1 INTRODUÇÃO

Em Santos, grande parte da população usufrui do transporte público oferecido pela Piracicabana para se locomover de um ponto ao outro da cidade e várias linhas são disponibilizadas para que todo o público consiga alcançar seu destino, seja ele próximo, ou distante.

Existem alguns métodos para encontrar uma boa alternativa de linha, porém, há uma defasagem na facilidade com que a informação chega às pessoas, o que impede que tenham mais assertividade em saber qual linha circular melhor se encaixa para aquela determinada situação. A população da cidade não conhece a localização de todos os pontos de ônibus e talvez, não possuam o conhecimento sobre os ônibus que circulam por aquela determinada região.

Segundo IBGE (2010), Santos é a cidade mais populosa da Baixada Santista, com 25% do total da região, movimentando aproximadamente 1,5% da economia do estado, quase o dobro do segundo colocado Guarujá, com 0,81%. Além disso, uma grande parte da população das cidades vizinhas frequenta a cidade de Santos (seja para estudos, trabalho ou lazer) diariamente, e por ser uma cidade turística e portuária, atrai muitos visitantes, até mesmo de fora do país, e parte deles utiliza o transporte público para conhecer diferentes pontos turísticos da cidade.

Considerando estas informações, o serviço reduz o tempo de espera da população baseado na posição atual do usuário e verificando sua distância em relação aos pontos mais próximos, tomando como referência a quantidade de ônibus disponíveis para aquele destino solicitado, facilitando, assim, o acesso dos usuários ao transporte público da viação Piracicabana, sejam eles moradores da cidade de Santos ou pessoas vindas de outras cidades, mapeando os pontos de ônibus da cidade e informando a melhor rota até o destino.

## 1.1 Justificativa Do Tema

Atualmente, está disponível no site oficial da Piracicabana um guia de linhas circulares onde é possível verificar todas as linhas e os seus trajetos completos pelo mapa de Santos, porém a visualização é feita através de uma página web.

A proposta é desenvolver um aplicativo que tenha toda essa praticidade em um dispositivo móvel, listando não somente as linhas e seus trajetos como também pontos turísticos. Dentro da solução proposta, a grande vantagem é conseguir ter uma gama de informações extremamente úteis de uma maneira simples e acessível, através de um aparelho celular.

## 1.2 Objetivo Geral

Desenvolver uma ferramenta móvel que disponibilizará informações de todos os pontos de ônibus ativos em Santos, bem como todas as linhas circulares que por eles passam.

### *1.2.1 Objetivos Específicos*

As etapas do projeto a serem cumpridas para se atingir o objetivo são:

- Elaboração do diagrama de caso de uso;
- Elaboração de objetos JSON como armazenamento de dados;
- Planejamento, elaboração e desenvolvimento documentais de todas as telas;
- Desenvolvimento da aplicação móvel.

# 2 REFERENCIAL TEÓRICO

Foi estudada a possibilidade de usar várias tecnologias para o desenvolvimento da aplicação móvel e a que melhor se adaptou foi a utilização do Ionic Framework (que utiliza dentro de seu núcleo HTML, CSS e JavaScript) juntamente com o Cordova (que permite utilizar funções nativas do dispositivo do usuário) e Firebase para armazenagem de dados.

Segundo documentação oficial do Ionic (2016), a criação de aplicativos se torna bem mais abrangente, pois seu desenvolvimento traz uma gama maior de usuários finais por trabalhar com linguagens padronizadas e utilizando PhoneGap, transforma sua aplicação em um sistema multi-plataforma, ou seja, rodando nos principais sistemas operacionais móveis (Android, iOS, Windows Phone), criando o conceito de Aplicativos Híbridos.

O conceito é bem simples, de acordo com Arvind Ravulavaru (2015): cada sistema operacional tem sua própria API (Application Programming Interface – Interface de Programação de Aplicativos) para desenvolvimento de aplicações, onde cada uma possui uma Web View (uma espécie de navegador que roda dentro do escopo da aplicação). Esta Web View roda HTML, CSS e JavaScript, que por sua vez, tem capacidade de ser criada usando tecnologias precedentes e então rodar o código em seu aplicativo.

Pensando nativamente, cada sistema operacional móvel (Android, iOS, Windows Phone, dentre outros) utiliza uma linguagem de programação específica em sua criação e para o desenvolvimento de seus aplicativos.

Com base na literatura de Sérgio Lopes (2016), o Android, desenvolvido pela Google e lançado em 2008, tem grande parte do seu desenvolvimento feito em Java em código aberto (apesar de possuir também linguagens como C e C++), que, por sua vez, utiliza dessa linguagem para criação de aplicativos nativos para esta plataforma.

O sistema operacional móvel iOS, fabricado e distribuído pela Apple desde 2007, possui seu código fechado, porém utiliza Swift (open source) como linguagem nativa para seu desenvolvimento e para criação de aplicativos nativos para seus aparelhos, como constado em documentação divulgada pela Apple (2016). Anteriormente, utilizava do Objective-C (baseado em C) como linguagem padrão, porém, sofreu alteração para a atual linguagem mais simples e mais poderosa.

Por fim, outro dos principais sistemas operacionais móveis é o Windows Phone, criado pela Microsoft em 2010, também possui código fechado e é feito em C#, linguagem criada pela Microsoft baseada em C.

## 2.1 Tecnologias Utilizadas

Utilizando Ionic, é possível criar aplicações para todas as plataformas por meio de linguagens web (HTML, CSS e JavaScript) sem modificar o conteúdo em seu código e com ele, todas as funcionalidades da aplicação são feitas através de programação web e demonstradas através da Web View.

### *2.1.1 Ionic*

Conforme verificado em documentação dos criadores da tecnologia (2016), Ionic é uma poderosa tecnologia *open source* criada no final de 2013 que aglomera características móveis otimizadas de HTML (linguagem de marcação utilizada para inserção de conteúdo na página), CSS (que tem como principal função estilizar o conteúdo em HTML para deixa-lo mais atrativo visualmente ao usuário) e JavaScript (responsável por toda a parte lógica do site). Com tais características, a criação da interface móvel ganha desempenho de processamento lógico e gráfico, além de ser capaz de transformar a aplicação nela criada em aplicações multi-plataforma devido à tecnologia e Phonegap e com o Apache Cordova, é possível utilizar funções nativas do aparelho, sendo as principais: câmera, vibração, acelerômetro, informações sobre bateria, geolocalização, mídia, transferência de arquivo.

Aprofundando um pouco mais na documentação do Ionic (2016), uma poderosa SDK (software development kit – ou Kit de desenvolvimento de software em português) é utilizada em HTML5 para criar as aplicações, focando principalmente no visual e sensitivo, além de, principalmente, na interação que o usuário terá com sua interface. Além disso, também possui um gerenciamento de notificações Push para seus usuários baseado em filtros selecionados pela programação da aplicação para que atinja apenas as pessoas que necessitam receber tais informações. Conta também com um sistema pré-implementado tradicional de login via e-mail e senha além de atualizar automaticamente a interface do usuário caso alguma alteração seja feita pelo desenvolvedor.

Através da sessão de HTML5 na documentação do Ionic (2016), complementadas portal W3Schools (2016) que é responsável por padronizar tecnologias, foi verificado que a framework toma vantagem dos múltiplos tipos de entrada de teclado fornecidos em HTML5 para beneficiar ainda mais a experiência de usuário, facilitando o tipo de entrada (texto, números, e-mail, senhas, etc.), configurando previamente o teclado para tal inserção. É um pequeno detalhe que faz grande diferença, pois se a informação solicitada no formulário são números, não há motivo para o teclado estar padronizado em texto. Através do HTML5, o Ionic consegue manter essa boa prática e manter sua aplicação muito mais limpa, de fácil acessibilidade e mais agradável para o usuário.

Na ala que diz respeito a CSS3 da documentação do Ionic (2016), constatou o poder que o Ionic tem para construir aplicações de grande escalão visual de maneira organizada e de boa visualização através da linguagem. Suas principais aplicações são na criação de títulos, subtítulos, rodapés (todos com estilos de letras e cores diferentes), botões (de diversas cores, tamanhos, modelos, etc.), ícones de botões, botões em cabeçalhos, listas, listas com ícones, listas com botões, formatação de formulários e botão ativado/desativado.

De acordo com documentação do Mozilla Developer Network (2015), toda a parte de processamento lógico é feito devido ao uso do JavaScript. Com ele, é possível desfrutar de todo o poder da linguagem para diversos tipos de funcionalidades, como validação de formulário, pop-ups, scroll e navegação entre telas. O JavaScript roda no lado do cliente e tem função primordial pois é através dele que toda a parte lógica propriamente dita é executada, desde uma simples alteração de variável, até grandes processamentos internos de acordo com a aplicação. Utilizando Ionic juntamente com o AngularJS, as possibilidades de desenvolvimento são muito mais abrangentes para desenvolvimento híbrido.

Ionic foi escolhido como principal framework no uso do trabalho devido às grandes facilidades de programação via HTML, CSS e JavaScript, pela versatilidade oferecida por ele e a capacidade de criar aplicações multi-plataforma, ou seja, com praticamente o mesmo código, é possível rodar a aplicação nas principais plataformas mobiles (Android, iOS, Windows Phone).

De acordo com o próprio site do Ionic (2016), o aplicativo oficial das Olimpíadas Rio 2016 foi inteiramente desenvolvido com sua framework, o que acaba trazendo consagração à tecnologia.

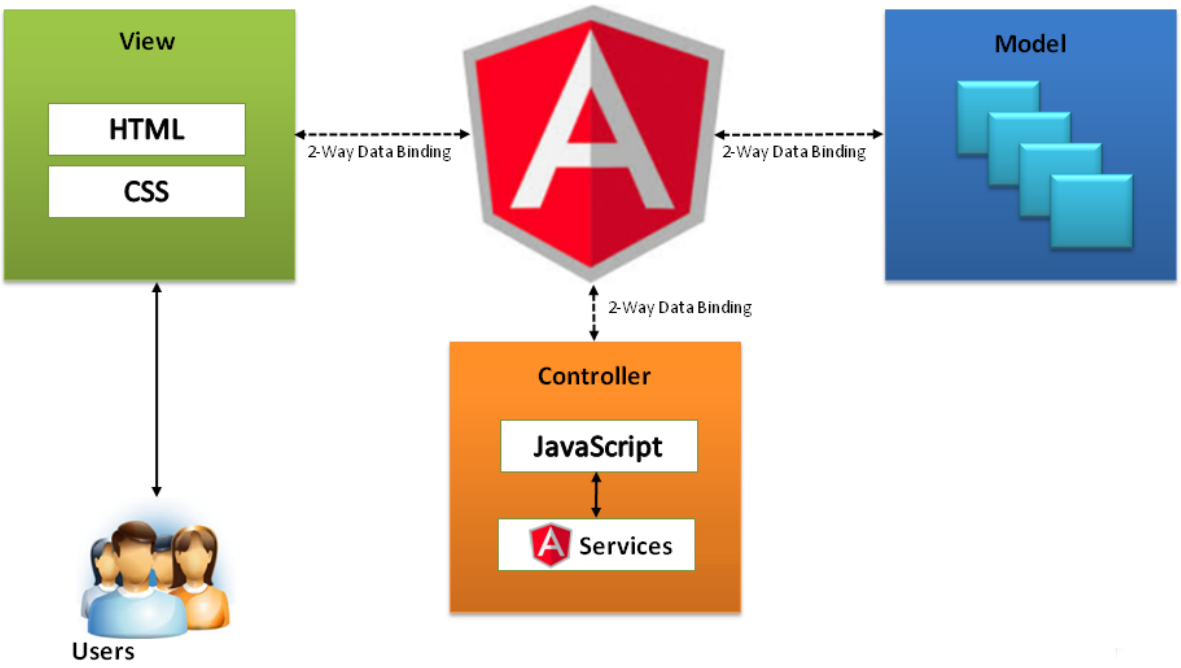
### *2.1.2 Angular JS*

Ionic utiliza em seu núcleo lógico principalmente o Angular JS, que foi criado em 2009 por Miško Hevery e Adam Abrons como uma framework JavaScript open source client-side que promove alta produtividade em desenvolvimento web, de acordo com Rodrigo Branas (2014). Foi criada com a intenção de delegar o HTML tradicional para transformar o trabalho de seus programadores mais fácil, resultando em reaproveitamento de código, componentes de aplicações sustentáveis, deixando a codificação mais limpa e clara para manter o foco apenas nos fatores importantes e valiosos.

Segundo Shyam Seshadri e Brad Green (2014), as aplicações em Angular são desenvolvidas seguindo o padrão MVC – Model (modelo que corresponde aos dados por trás da aplicação, ou seja, todas as informações que o usuário visualiza, vêm da camada Model), View (é a interface gráfica que corresponde àquilo que o usuário vê e com o qual ele interage e é gerada de acordo com um modelo da aplicação) e Controller (representa a parte lógica, ou seja, realiza todas as ações de acesso a dados. Por exemplo, mostra como e quais elementos da Model serão apresentados na View).

Com isso, segundo Adam Freeman (2013), ganha desempenho extensível (onde o programador consegue facilmente incrementar sua aplicação, criando novas funcionalidades para seus usuários), sustentabilidade (são de fácil manutenção), testável (possui grande suporte de teste ponta-a-ponta, o que permite ao desenvolvedor localizar erros antes mesmo que o usuário os encontre) e padronizado (criado com capacidade natural dos navegadores, permite criar aplicações web compatíveis com o padrão, tirando vantagem das grandes atualizações recentes como, por exemplo, do HTML5).

**Figura 1 –** Estrutura MVC

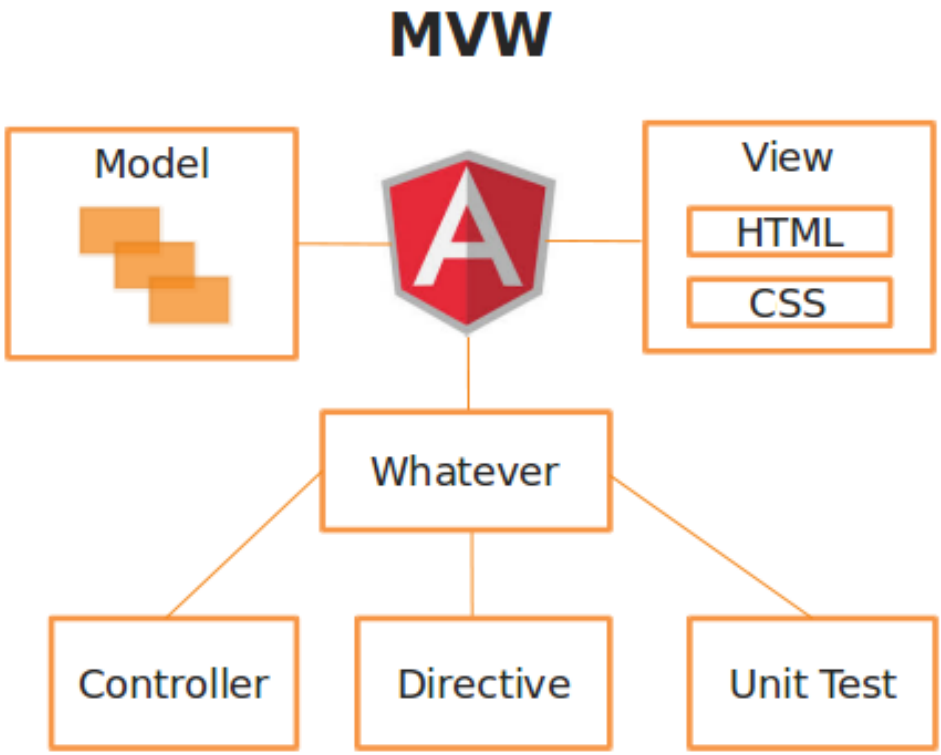



Fonte: Shinde (2016)

Por outro lado, complementa Branas (2014), o AngularJS apresenta um novo conceito chamado MVW (model-view-whatever) onde a Model e a View podem ser automaticamente sincronizadas sem a utilização de uma Controller (embora possa ser utilizada). Neste padrão, cada camada é responsável por apenas uma tarefa, facilitando a manutenção do código e novas implementações, precisando saber apenas em que camada a parte a ser desenvolvida se encontrará, além de ter uma característica mais modular e reutilizável.

De acordo com Micheal Henrique R. Pereira (2014), a função da Controller é obrigatoriamente organizar a Model e a View (ou seja, os dados e a interface do usuário) e não possui nenhuma outra responsabilidade além desta. Com a apresentação do Two-Way Data Binding, a Controller se torna desnecessária.

**Figura 2 –** Estrutura MVW




Fonte: Shinde (2016)

Verificado em documentação oficial sobre AngularJS (2016) complementado por Pereira (2014), a maioria dos modelos de sistemas trabalha com dados de apenas uma maneira (One-Way Data Binding), onde juntam os componentes das camadas “Template” e “Model” na camada “View” e depois dessa junção, as mudanças efetuadas não são automaticamente mostradas na View. A framework utiliza Two-Way Data Binding e nela não há essa junção das camadas onde o Template é compilado primeiramente e todas as mudanças passam previamente pela camada View.

Ainda em sua documentação (2016) e com informações também complementadas pelo W3 (2016), o AngularJS compila todos os elementos em HTML antes de exibí-los ao usuário, incluindo Directives. Directives são marcas no Modelo de Objeto de Documento (DOM) que faz o compilador de HTML no Angular juntar um comportamento específico deste elemento DOM, ou até mesmo modificá-lo e seus herdeiros, o que o faz oferecer mais funcionalidades à sua aplicação

através de um conjunto de diretivas embutidas. Também permite ao desenvolvedor definir suas próprias diretivas.

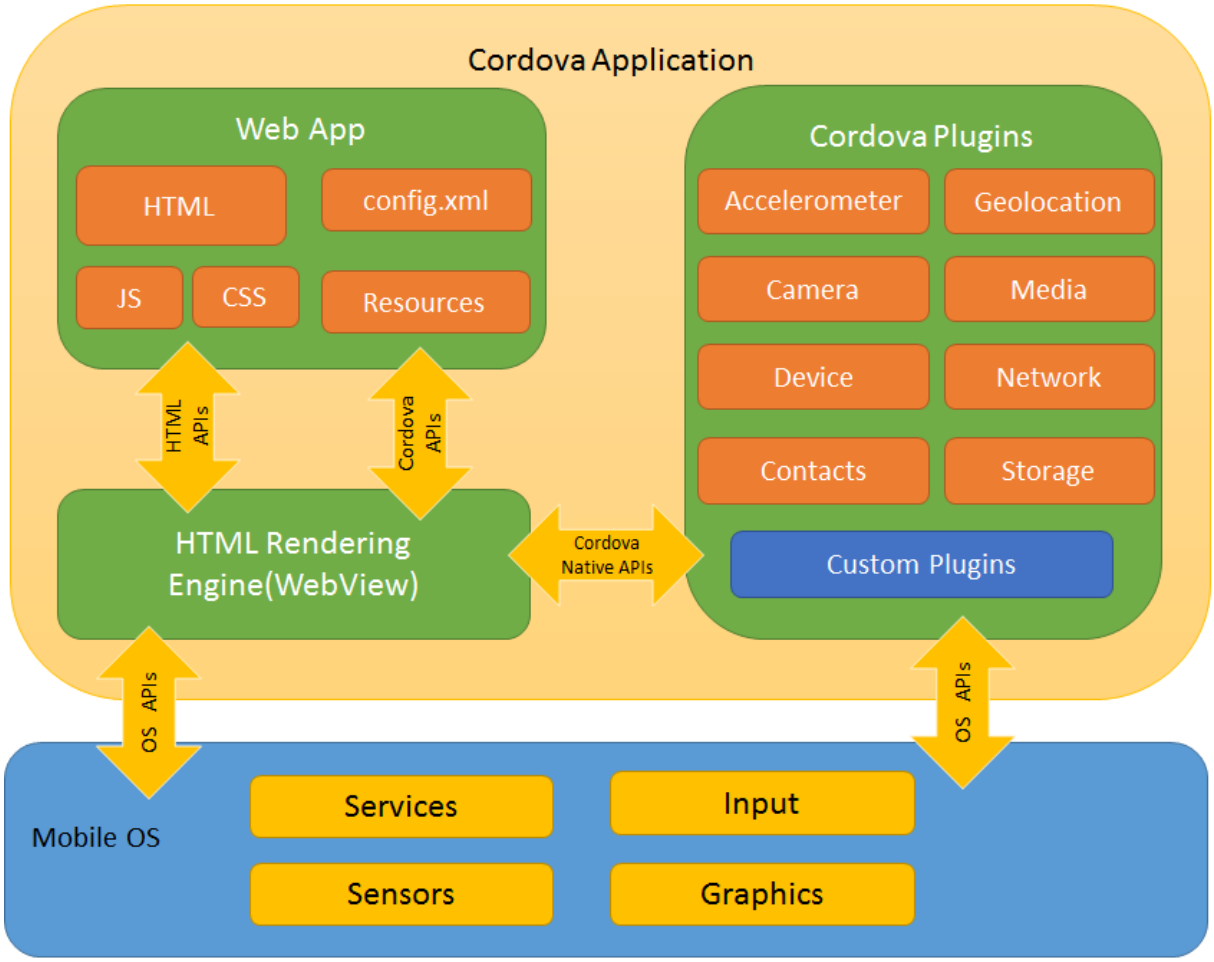
A grande vantagem do AngularJS é que com ele, evita-se redundâncias no código e faz todo o trabalho maçante e pesado (já que com o template do lado do cliente e o uso intenso do JavaScript, o desenvolvimento da aplicação se torna algo bem trabalhoso) seja poupado para utilizá-lo de forma melhor, exclusivamente no núcleo da aplicação. Com isso, se torna umas das principais responsáveis pela abstração e limpeza na implementação do código.

### *2.1.3 Cordova*

Apache Cordova, que foi criada pela Nitobi, é uma framework gratuita e de código aberto para desenvolvimento multi-plataforma de aplicações nativas usando HTML, CSS e JavaScript. Conforme dito por John M. Wargo (2015), o principal intuito do Cordova é desenvolver aplicações para várias plataformas combinando tecnologias nativas do aparelho e web. Esse tipo de abordagem, onde se pode reutilizar o código para abranger uma gama maior de usuários (e seus respectivos sistemas operacionais móveis), tem sido muito utilizado para ganhar tempo de desenvolvimento e manter uma aplicação visual e com grande poder de processamento. Basicamente, Cordova consiste em:

- Renderizar os conteúdos da aplicação web no dispositivo, ou seja, todo conteúdo desenvolvido nas linguagens citadas anteriormente são mostrados em tela para o usuário através do Cordova, que por sua vez, rearranja os elementos de acordo com o sistema que o usuário está utilizando;
- Um conjunto de núcleo de API (normalmente através de plug-ins) que permite que o aplicativo web rodando com o seu conteúdo, acesse funcionalidades nativas do dispositivo, podendo, por exemplo, utilizar funções de vibração quando algum evento ocorrer.

**Figura 3 –** Arquitetura do Cordova




Fonte: Documentação do Cordova (2016)

Hazem Saleh (2014) diz que um grande desafio no universo de desenvolvimento móvel é que cada sistema operacional possui uma linguagem de programação específica para seu desenvolvimento, além de padrões de linguagem e ferramentas diferentes para cada tipo de plataforma. Por exemplo, para Android, é necessário Eclipse (ou algum IDE similar), para iOS, é necessário Xcode e para Windows Phone, o Visual Studio. Considerando que será desenvolvido o mesmo aplicativo para cada plataforma separadamente, testar, encontrar erros e implementar diferentes funções tende a se tornar bem difícil e com custo alto. Além disso, outro desafio é prever situações onde será necessário interagir com outra plataforma. Uma mensagem de texto, por exemplo: se ela for enviada de um Android, o iOS precisa ter essa “compatibilidade” para receber esta mensagem vinda de outra API.

Em aplicativos móveis, o Cordova reduzirá a complexidade destes desafios. Com ele, Saleh (2014) complementa que toda a parte lógica é feita em JavaScript e será executada em qualquer uma das plataformas, ou seja, não é necessário o uso de outras linguagens depois de usar o Cordova, pois ele provê a habilidade de ter um código base comum para todas as implementações da aplicação. Com isso, a aplicação usando Cordova acaba se tornando o código todo centralizado, o que acaba fazendo com que o código fique mais legível e com uma facilidade maior de manutenção.

Considerando a aplicação nativa em Cordova, John Wargo (2015) diz que a interface do usuário consiste em apenas uma tela não contendo nada além de uma única web view (que é um componente nativo da aplicação que costuma renderizar e exibir o conteúdo da web para o usuário) redimensionada de acordo com a resolução da tela. A partir disso, quando a aplicação é aberta, a página inicial (normalmente index.html) é carregada dentro da web view para que o usuário consiga interagir com a aplicação. Assim que uma interação acontece, códigos em JavaScript ou links podem carregar outros conteúdos do pacote da aplicação tanto quanto informações externas, caso tenha sido programado para isso.

Wargo (2015) complementa ao dizer que uma aplicação comum em Ionic não acessa funções nativas de um dispositivo. Por exemplo, a câmera não pode nem consegue ser acessada pela aplicação web diretamente, ou não consegue fazer uso do microfone para qualquer tipo de evento, logo, a aplicação precisa ter acesso a estes tipos de capacidades. Com o Cordova, é possível ter todos estes tipos de acessos, sendo os principais status de bateria, câmera, notificação, geolocalização, vibração, mídia, informações de rede, orientação do dispositivo e contatos.

Como verificado em sua documentação oficial do Cordova (2015), todos esses recursos estão disponíveis (ou em desenvolvimento) para as grandes plataformas móveis como Android, iOS e Windows Phone.

Apesar de muito útil para aplicações híbridas, Sérgio Lopes (2016) acredita que o Cordova possui data de validade. Considerando que os navegadores já estão trazendo lentamente algumas funções nativas, como Service Worker (que é um tipo de execução em background, ou seja, continua sendo executado mesmo off-line) e Notification Push (que são as notificações ao usuário), Cordova poderá ser extinto em um futuro principalmente porque algumas páginas já podem criar WebApps a

partir de lojas dentro do próprio navegador. Lopes comenta que até o próprio criador do Cordova e do Phonegap disse que esta tecnologia talvez seja desnecessária com a evolução dos navegadores. Até que os navegadores se tornem dominantes para criação de aplicações web, o Cordova é uma excelente alternativa para desenvolvimento nesta área.

### *2.1.4 Firebase*

De todas as opções estudadas, o Firebase foi a melhor escolha para a utilização de mecanismos de persistência de armazenagem.

De acordo com documentação oficial do Firebase(2016), é uma ferramenta que disponibiliza vários serviços. A primeira delas foi sua autenticação. Com ela, o próprio Firebase cria todo um ambiente de serviços back-end (servidor) para criar a autenticação do usuário no sistema. Além de disponibilizar o ambiente para criação de login personalizado, ele também oferece suporte já integrado de opção de login através de contas do Google, Facebook e Twitter como outra opção, abrangendo uma gama maior de usuários e facilitando sua usabilidade.

Outra grande ferramenta usada no projeto foi o banco de dados na nuvem, chamado de Firebase Realtime Databate, onde os dados são armazenados em JSON e são distribuídos para todos os usuários conectados, ou seja, quando uma aplicação multi-plataforma é criada implementando o Firebase, todos os clientes utilizam de uma mesma instância do banco de dados e recebem as informações mais recentes e atualizadas.

Com o Firebase, os dados são mantidos localmente no dispositivo do usuário, logo, se houver alguma interrupção na conexão, o usuário ainda terá acesso aos dados já carregados anteriormente, sendo assim, logo que a conexão é reestabelecida, o Realtime Database se encarrega de sincronizar todas as alterações para que o usuário não perca nada e mantenha suas informações recebidas atualizadas.

A base de dados é considerada como NoSQL (Not Only SQL – Não Apenas SQL), ou seja, possui alguns recursos e diferentes otimizações quando comparado a um banco de dados relacional. De acordo com documentação oficial do banco de dados MongoDB (2016), NoSQL é usado para trabalhar com uma grande quantidade de dados para que sejam de fácil modificação. Quando comparado com

bancos de dados relacionais, a base de dados em NoSQL é bem mais escalável e provê uma performance muito superior, o que torna uma vantagem muito grande, além de ser orientada a objetos.

**Quadro 1 –** Comparativo de NoSQL e SQL

| | **NoSQL** | **SQL** |
|---|---|---|
| **Modelo** | Não relacional | Relacional |
| | Armazena os dados em um documento JSON | Armazenada os dados em tabelas |
| **Dados** | Novas propriedades podem ser adicionadas em tempo real | Adicionar uma propriedade requer alteração do esquema do Banco de Dados |
| **Esquema** | Esquema dinâmico e flexível | Esquema restrito |

Fonte: Microsoft Azure (2016)

Conforme documentação oficial do Firebase (2016) possui bibliotecas para várias linguagens diferentes (Javascript inclusa), porém, possui um framework totalmente em client-side usada em AngularJS para plataformas móveis como Android e iOS, que no caso, armazena todas as informações que seriam inseridas no banco de dados, direto na aplicação, dentro do lado do cliente em formado JSON (onde o que é visto é exatamente o que está sendo armazenado).

De acordo com Manoj Waikar (2015), o Firebase possui algumas vantagens para usabilidade. Por ser um serviço de nuvem, não é preciso necessariamente de instalação e possui nativamente uma criptografia segura de todos os dados transferidos, que por sua vez, cria uma réplica de si mesmo (como uma espécie de backup) em diferentes lugares seguros para que a chance de perda de dados seja o menor possível. Combinado com AngularJS, ele cria um Three-Way data Binding entre HTML, JavaScript e os dados.

## 2.2 Posicionamento No Mercado

A Piracicabana possui uma aplicação própria baseada em QR Code onde em cada ponto de ônibus possui um código e, ao escaneá-lo através de um smartphone, o aplicativo retorna as linhas que estão prestes a passar por aquele ponto. O sistema se baseia neste escaneamento e retorna basicamente o tempo de espera dos ônibus que passam por aquele determinado ponto.

Disponível nas lojas móveis, constava um aplicativo chamado Meu Ônibus, na qual o usuário conseguia selecionar um ônibus e ver o percurso completo dele pela cidade, mas não havia interação com o usuário, sendo meramente informativo. Esta aplicação não está mais disponível para download.

O CittaMobi é outro aplicativo muito conhecido no mundo e que está disponível para download nas lojas móveis. Conforme informações do aplicativo em loja de aplicativos (2016), este sistema oferece funcionalidades diversas que se assemelham muito às funcionalidades oferecidas pelo projeto. Neste aplicativo, é possível visualizar os pontos mais próximos do usuário e a previsão de chegada dos ônibus em tempo real. Além disso, é possível também consultar itinerários e buscar por veículos adaptados para cadeirantes. Entretanto, o software não atende a cidade de Santos, e ainda não há nenhuma previsão de chegada à cidade. A ferramenta está disponível apenas em 12 cidades brasileiras. São elas: Colatina (ES), Diadema (SP), Juiz de Fora (MG), Maceió (AL), Recife (PE), Rio Branco (AC), Rio Grande (RS), São Caetano do Sul (SP), Santo André (SP), Santa Rita (PB) e Volta Redonda (RJ).

A ideia do projeto foi desenvolver uma aplicação móvel onde o usuário pudesse ter mais de uma alternativa para chegar a seu destino com base de sua localização, informando no mapa quais pontos passam ônibus que chegam até algum ponto próximo do destino solicitado pelo usuário. O site atualmente encontra-se fora do ar.

De acordo com site da própria Piracicabana (2016), a aplicação web “Quanto Tempo Falta?” possui uma interface simples onde o usuário insere o código do ponto de ônibus ou seleciona um em uma lista completa. Ao selecionar um ponto, é exibido ao usuário uma pequena lista com os ônibus que estão prestes a chegar naquela parada, indicando a previsão de chegada de cada ônibus. A aplicação também informa, através de um ícone ao lado do número da linha, se o veículo

possui ou não ar condicionado. Não há nenhuma informação sobre as linhas ou sobre trajetos.

Foi feito um quadro comparativo para criar tal posicionamento de mercado para a aplicação final do projeto:

**Quadro 2 –** Comparativo de aplicações

| | **Projeto** | **Meu Ônibus** | **Quanto Tempo Falta** | **CittaMobi** |
|---|---|---|---|---|
| Tempo de Espera | | X | X | X |
| Informações sobre pontos | X | | X | |
| Informações sobre linhas | X | X | | X |
| Demonstração de trajeto | X | | | X |
| Compatibilidade com a cidade de Santos | X | X | X | |

Fonte: autores (2016)

O quadro acima mostra alguns pontos importantes sobre as funcionalidades de cada aplicação.

A aplicação da Piracicabana – Meu Ônibus (que não se encontra mais disponível para download) possuía informações consideravelmente importantes como por exemplo, o tempo restante aproximado para determinado ônibus chegar àquele ponto. Com essa informação, o usuário poderia planejar melhor sua saída com base neste tempo. A aplicação também oferecia uma lista completa contendo todos os ônibus que passam pela cidade, porém não disponibiliza nenhuma informação sobre os pontos de ônibus ou sobre os trajetos pela cidade.

De acordo com informações retiradas do site da CittaMobi (2015), a aplicação oferece previsões de chegada dos ônibus de acordo com cada ponto de ônibus. Também possui uma funcionalidade de localização onde ele recebe a posição atual do usuário e informa alguns pontos de ônibus próximos a ele para que o mesmo possa se localizar melhor na cidade. Possui informações sobre as linhas e

demonstra o trajeto delas no mapa, porém ainda não oferece informações adicionais sobre os pontos e não possui suporte para a cidade de Santos.

O projeto traz informações de todos os pontos de ônibus na cidade, exibindo ao usuário, todas as linhas que passam por aquele ponto para que o usuário tenha conhecimento e consiga usar essa informação a seu favor. Também traz uma lista completa com todas as linhas e ao selecionar uma linha, indica no mapa, todo o seu trajeto ao redor da cidade. O sistema tem como foco a cidade de Santos, trazendo as informações mais importantes para que o usuário consiga se situar da melhor maneira possível em relação aos ônibus e suas paradas, em formato de informação pura ou visual, através de sua exibição no mapa. Como a Piracicabana não disponibiliza informações internas como previsão de chegada dos ônibus para terceiros, não foi possível implementar esta função ao sistema.

### *2.2.1 Pesquisa*

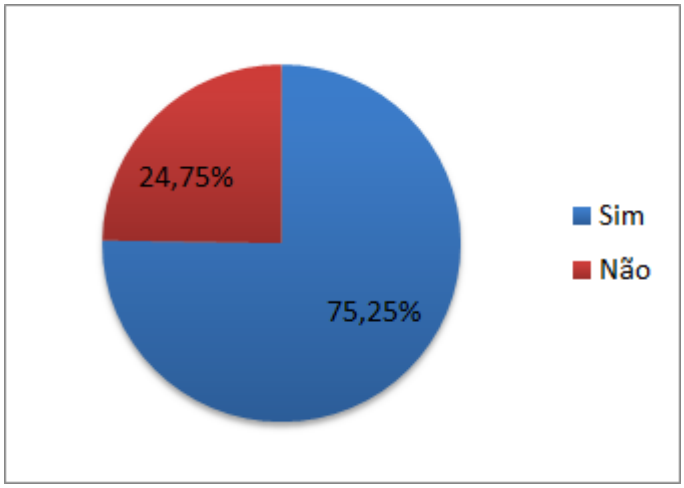
A metodologia usada se enquadra em uma pesquisa quantitativa, pois de acordo Pinheiro (2011), é usada para levantamento de dados, gerando estatística para uma conclusão mais assertiva sobre determinado escopo.

Foi criada uma pesquisa online via Google Docs que está disponível no Apêndice B, com o intuito de descobrir o nível de informação sobre as linhas que a população regional possui em relação aos ônibus da cidade. Essa técnica foi escolhida pelos integrantes, pois se mostrou a maneira mais simples e eficaz de conseguir levantar os dados necessários sobre as pessoas da região para abordar as questões tratadas no projeto para, então, no final, conseguir entregar uma aplicação que oferecesse mais informação à população.

O questionário foi aplicado aos cidadãos da cidade de Santos, São Vicente e Guarujá para saber se eles conhecem alternativas para chegar em determinado lugar, fora aquela que elas estão acostumadas a pegar, bem como o trajeto completo do ônibus que elas pegam para saber se podem utilizar esta mesma linha para outros fins.

Ao todo, foram entrevistadas 295 pessoas, e após coleta e processamento de dados, identificamos que 75% delas utilizam o transporte público da cidade, conforme imagem abaixo:

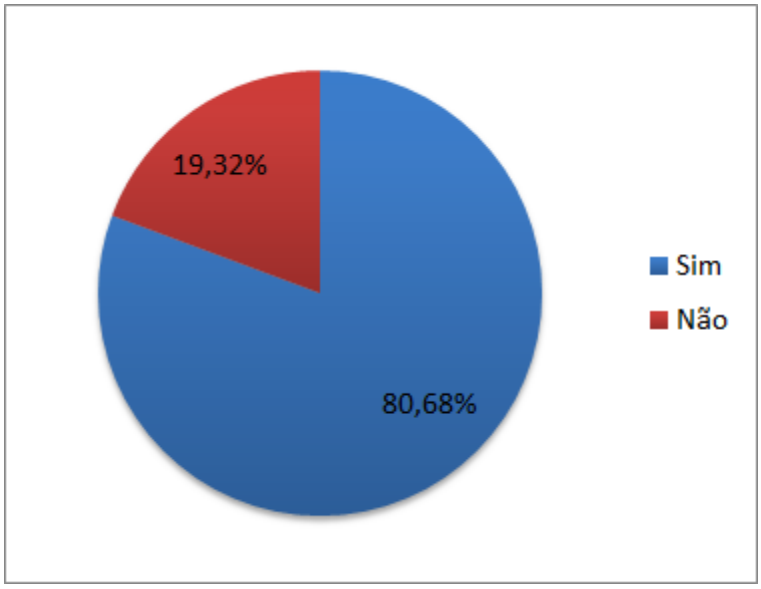
**Figura 4 –** Gráfico 3: Pesquisa - Você utiliza o transporte público em Santos?




Fonte: autores (2016)

De todos os que utilizam o transporte, 80% das pessoas têm dificuldade em saber onde parar quando vão para um destino desconhecido, seja por falta de informação ou por não identificar as referências recebidas de outros. A imagem abaixo mostra a pesquisa feita com as pessoas que têm essa dificuldade:

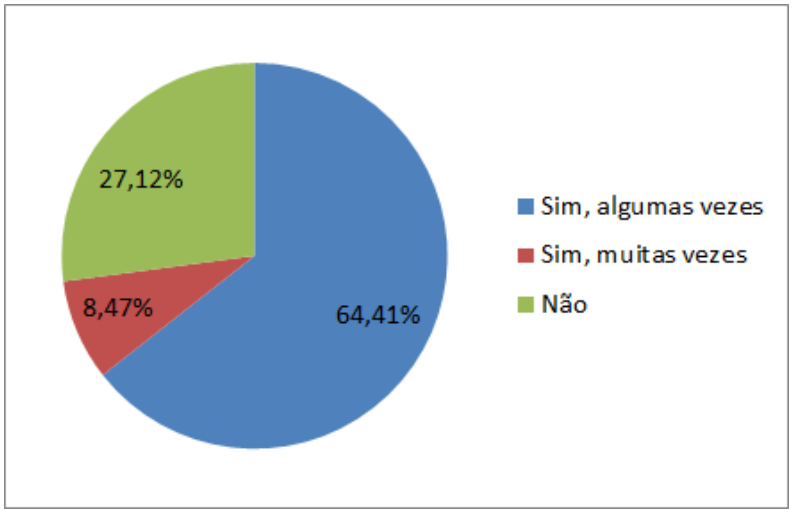
**Figura 5 –** Gráfico 7: Pesquisa - Quando pega um ônibus, tem dificuldade em saber em qual ponto parar, com um destino nunca feito anteriormente?




Fonte: autores, (2016)

Foi averiguada também, a quantidade média de conduções erradas que as pessoas já pegaram e elas somam pouco mais de 72%, conforme imagem a seguir:

**Figura 6 –** Gráfico 8: Pesquisa - Já pegou alguma condução errada?




Fonte: autores (2016)

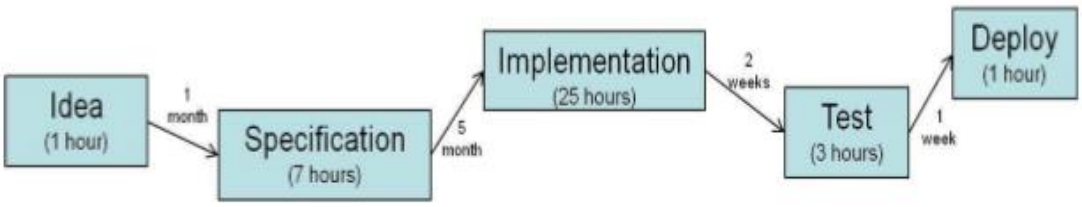
# 3 METODOLOGIA

Dentre vários métodos de desenvolvimento, escolhemos o modelo ágil Kanban, pois segundo Boeg (2012), com ele é possível ter uma flexibilidade que vai além das interações fixas e planejadas de outras metodologias. Também se aplica fortemente para projetos onde as prioridades de desenvolvimento podem se alterar diariamente.

Kanban é uma metodologia ágil que foca principalmente em:

- Visualizar cada passo do projeto a partir de uma vaga concepção até a entrega do sistema, ou seja, em cada etapa do desenvolvimento, é feito uma visualização futura do sistema completo com o módulo em questão. Com isso, é criado um fluxo de trabalho para acompanhar o planejamento do projeto inteiro e como ele vai fluir conforme o tempo. Em sua fase de testes, além do sistema ser propriamente testado de fato, ele também é corrigido, refatorado e discutido melhorias para esta mesma fase da aplicação.

**Figura 7 –** Exemplo de fluxo de trabalho




Fonte: Boeg (2012)

Como mostrado no exemplo de fluxo, cada estágio tem seu tempo diário de investimento especificado e o mesmo é feito durante o período indicado nas setas, ligando o fluxo;

Visualizar o trabalho através do fluxo gera benefícios. Todos ficam a par exatamente do que está acontecendo durante o processo de cada etapa e todo o trabalho e o que ele afeta fica visível para todos os envolvidos.

- Definir limites de trabalho (WIP – Work-In-Progress) para cada etapa onde o desenvolvimento não fique estagnado em questão de qualidade com alguma ideia viciada durante sua criação. Nele, é definido uma quantidade

de tarefas que poderão ser efetuadas. Com isso definido, não poderá ser tratado mais tarefas do que a quantidade definida;

- Tornar explícitas todas as políticas para que as mesmas sejam seguidas a finco, tornando assim o desenvolvimento do projeto focado em cada estágio da produção. É de extrema importância que todos se comprometam com as políticas estabelecidas para que o projeto flua com facilidade. Boeg (2012) também afirma que ao infringir esse acordo, o processo se degenera rapidamente;
- Verificar todas as decisões tomadas anteriormente e as consequências causadas por ela para ter um embasamento maior em relação a futuros caminhos a serem seguidos;
- Identificar melhorias em todas as etapas. Estas melhorias acabam sendo sempre responsabilidade de todos os envolvidos, onde todos atuam na otimização dos processos.

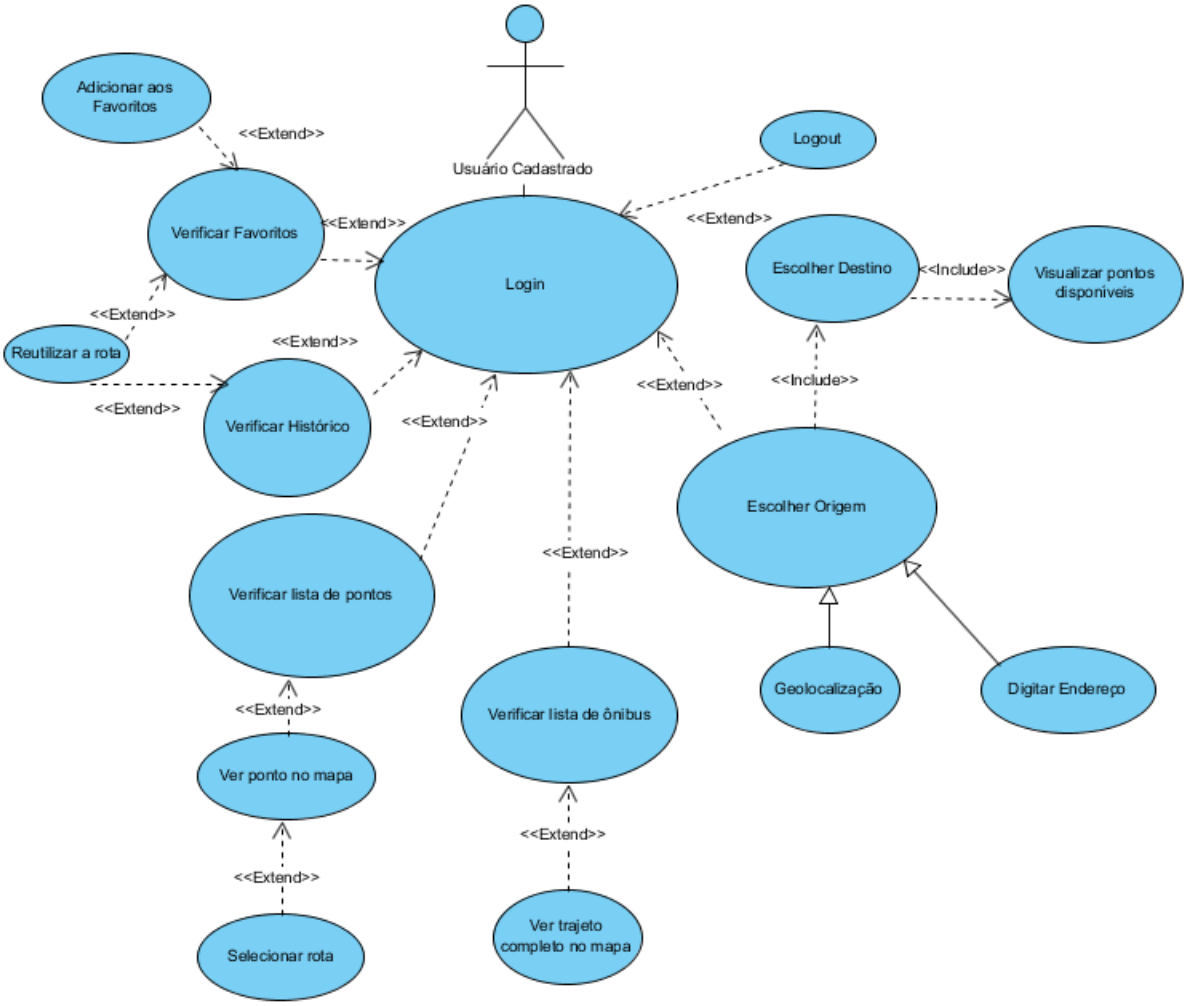
Basicamente, seguindo a filosofia do Kanban descrita por Boeg (2012), deve-se começar sempre com o que já está sendo feito, procurar sempre por atualizações e respeitar a etapa atual com a responsabilidade de cada ator durante o desenvolvimento.

## 3.1 Casos De Uso

Foram elaborados dois grupos de casos de usos, um para cada tipo de usuário que pode utilizar a aplicação: cadastrados e não cadastrados.

### *3.1.1 Cadastrados*

**Figura 8 –** Casos de Uso – Usuário Cadastrado




Fonte: autores (2016)

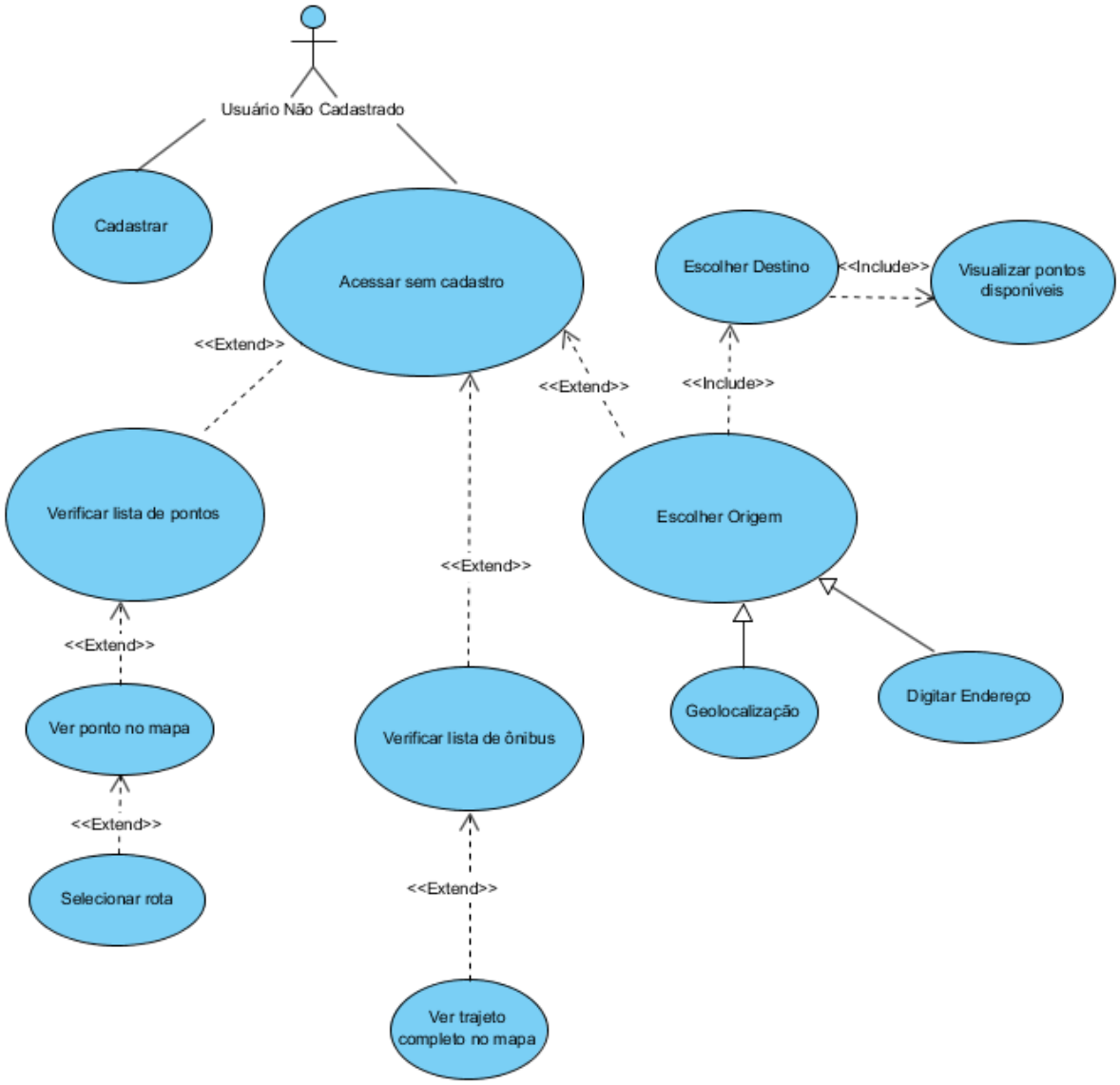
Algumas funcionalidades estão disponíveis apenas para usuários que possuam uma conta registrada na aplicação sendo elas, “Favoritos” e “Histórico”.

A figura 12 representa todo o fluxo de ações que o usuário cadastrado pode executar seguindo os caminhos estabelecidos pelo sistema e mantendo a sequência lógica do sistema.

A descrição dos casos de uso apresentados acima (contendo número do caso, nome do caso, atores, descrição, inclusões e exclusões) se encontra no apêndice deste trabalho (APÊNDICE A).

### *3.1.2 Não Cadastrados*

**Figura 9 –** Casos de Uso – Usuário Não Cadastrado




(Fonte: autores, 2016)

O diagrama de Caso de Uso dos usuários não cadastrados é bem semelhante ao dos cadastrados, contudo, o ator que excuta as ações neste cenário é sempre “Usuário Não Cadastrado”, além de possuir módulos específicos como “Acessar sem

cadastro” e “Cadastrar”. Não possui “Favoritos” nem “Histórico”, ou seja, todos os Casos são iguais porém, não salvam nenhum registro de uso. Os detalhes dos casos de uso estão localizados no apêndice do trabalho (APÊNDICE A).

## 3.2 Banco De Dados

A base de dados NoSQL foi utilizada com base na estrutura do Firebase, que possui nativamente uma sessão exclusiva para autenticação de usuários (através de login e senha. Como toda a estrutura de Banco de Dados do NoSQL é feita através de JSON, criou-se a seguinte estrutura para login do usuário:

**Figura 10 –** Objeto JSON – USUARIOS

```
-- USUARIOS
    {
        "_id" : "integer",
        "email" : "string",
        "password" : "string"
    }
```

Fonte: autores (2016)

Cada campo possui seu nome identificador e seu tipo específico de dado para seu preenchimento, onde:

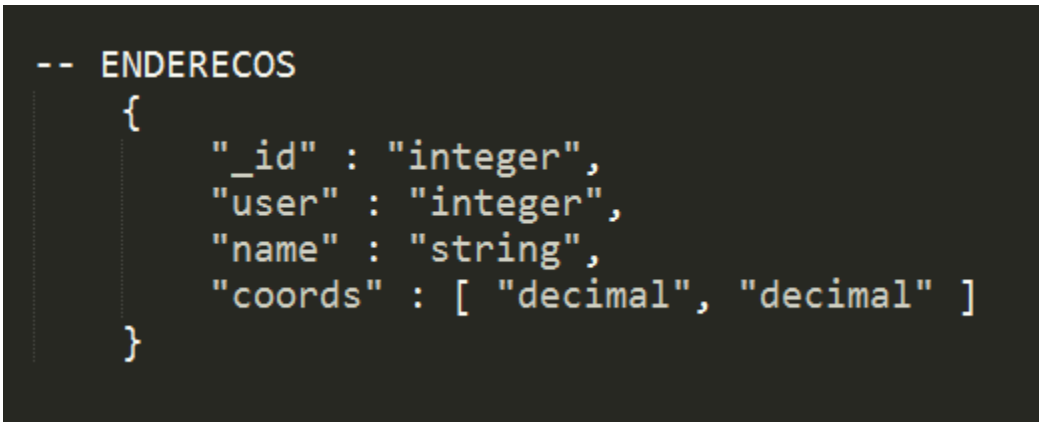
- **_id:** código único do usuário (não pode haver duplicatas) utilizado como chave primária. Seu tipo “integer” é um valor numérico inteiro;

- **email:** e-mail utilizado para cadastro do usuário, também utilizado como login no sistema. Possui um tipo “string” que armazena valores texto;

- **password:** senha do usuário para acesso ao sistema, também de valor texto.

Firebase também possui a sessão de Database (ou base de dados) onde foram criadas objetos (conhecido como tabelas em Banco de Dados relacionais) para dar estrutura ao banco de dados NoSQL utilizado no sistema. Desta forma, tem-se a seguinte estrutura de dados JSON:

**Figura 11 –** Objeto JSON – ENDERECOS




Fonte: autores (2016)

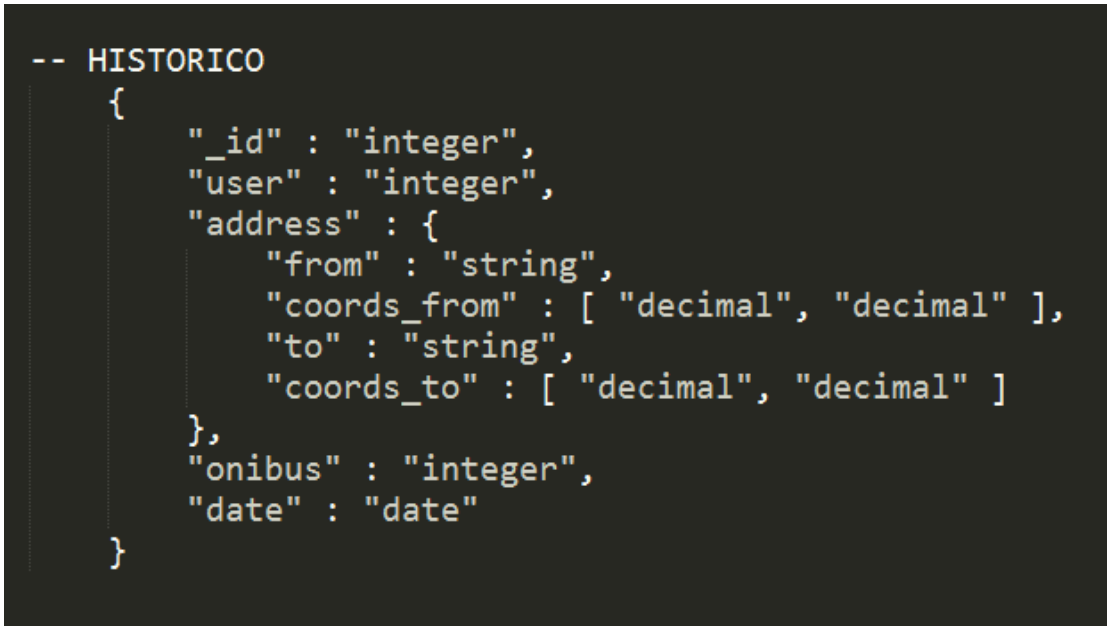
- **_id:** código único do endereço (não podendo também haver duplicatas) utilizado como chave primária. Seu tipo “integer” é um valor numérico inteiro;

- **user:** campo que usa o valor chave inteiro da tabela USUARIOS para vincular os endereços rápidos salvos pelo usuário, fazendo com que ele possua atalhos de endereços para fácil acesso futuro;

- **address:** atributo com valor texto contendo o logradouro do endereço rápido salvo pelo usuário;

- **coords:** possui um valor duplo, com dois atributos do tipo decimal, armazenando as coordenadas (latitude para o primeiro decimal e longitude para o segundo) do endereço rápido.

**Figura 12 –** Objeto JSON – HISTORICO



```
-- HISTORICO
    {
        "_id" : "integer",
        "user" : "integer",
        "address" : {
            "from" : "string",
            "coords_from" : [ "decimal", "decimal" ],
            "to" : "string",
            "coords_to" : [ "decimal", "decimal" ]
        },
        "onibus" : "integer",
        "date" : "date"
    }
```

Fonte: autores (2016)

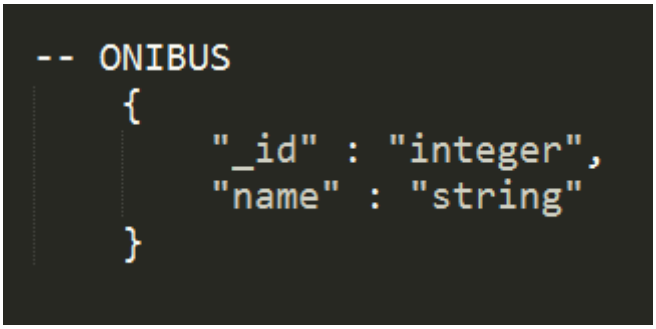
- **_id:** código do tipo numérico único do histórico utilizado como chave primária;
- **user:** campo que usa o valor chave inteiro da tabela USUARIOS para vincular os endereços rápidos salvos pelo usuário, fazendo com que ele possua atalhos de endereços para fácil acesso futuro;
- **address:** campo que armazena o histórico do usuário. Dentro dele, existe outro objeto com algumas informações contendo detalhes do endereço salvo no histórico:
    - **from:** campo que armazena um tipo “string” de texto com o logradouro de origem do usuário;
    - **coords_from:** valor duplo contendo decimais (latitude e longitude) da origem do usuário;
    - **to:** campo que armazena um tipo “string” de texto com o logradouro de destino do usuário;

- **coords_to:** valor duplo contendo decimais (latitude e longitude) do destino do usuário;

- **bus:** campo de valor inteiro que utiliza o valor chave do ônibus da tabela ONIBUS, gravando no objeto HISTORICO, a linha utilizada pelo usuário;

- **date:** campo do tipo “date” que armazenada a data que o usuário efetuou utilizou a linha acima.

**Figura 13 –** Objeto JSON – ONIBUS

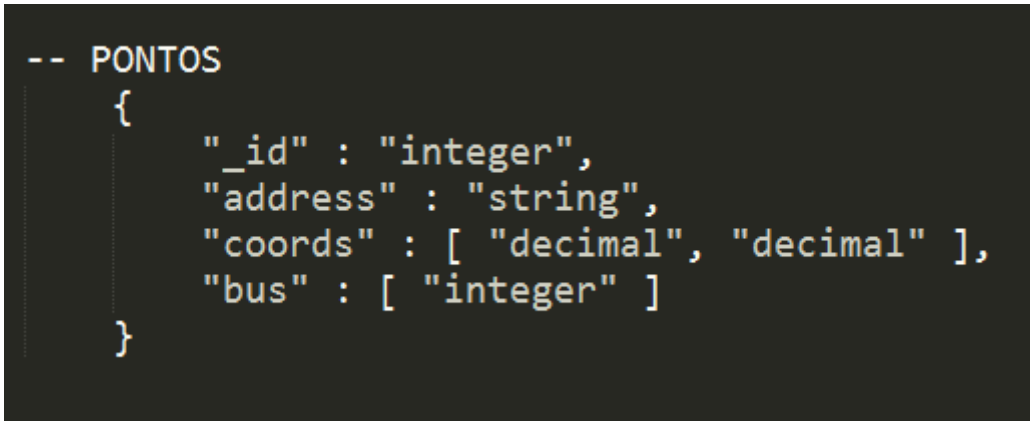



Fonte: autores (2016)

- **_id:** código único de cada ônibus (cada linha possuindo um código específico) utilizado como chave primária;

- **name:** valor do tipo texto utilizado para nomear a linha circular (ex.: Linha 29).

**Figura 14 –** Objeto JSON – PONTOS




Fonte: autores (2016)

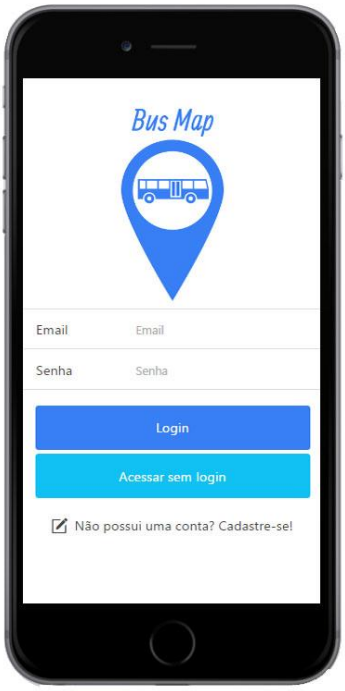
- **_id:** código único de cada ponto de ônibus utilizado como chave primária;
- **address_name:** atributo com valor texto contendo o logradouro do endereço onde se encontra o ponto de ônibus em questão;
- **coords:** valor duplo, com dois atributos do tipo decimal, armazenando as coordenadas (latitude e longitude) do endereço do ponto de ônibus;
- **bus:** campo contendo um vetor de atributos inteiros, onde os valores possuem a chave do ônibus do objeto ONIBUS, listando todas as linhas que passam por aquele ponto específico.

## 3.3 Telas

A demonstração das telas obtidas com o desenvolvimento representa as funcionalidades aplicadas, explicando como ocorrem as suas interações. Todas as telas foram feitas através de HTML e AngularJS e foram verificadas em diferentes dispositivos para testar o design responsivo da aplicação.

### *3.3.1 Login*

**Figura 15 –** Login




Fonte: autores **(**2016)

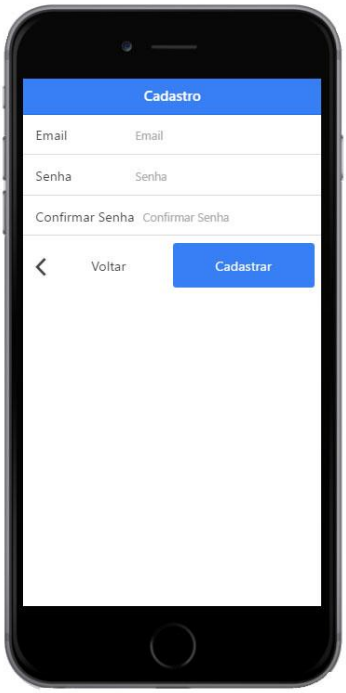
Na primeira tela, ao abrir a aplicação, são exibidos algumas opções para o usuário, sendo elas:

- **Email:** campo utilizado para ser preenchido com o e-mail cadastrado na aplicação;
- **Senha:** campo para entrada da senha criada pelo usuário em seu cadastro;
- **Login:** botão que valida a autenticação do usuário com base no email e senha inseridos e comparando com a base de dados de usuários cadastrados;

- **Acessar sem login:** botão que habilita as funções principais da aplicação sem a necessidade de um cadastro, fazendo com que o usuário consiga usar o sistema, porém, não será registrado histórico nem será possível adicionar ou verificar endereços favoritos.
- **Não possui uma conta? Cadastre-se:** botão que leva o usuário a uma tela de cadastro para que o mesmo possa se registrar no sistema.

### *3.3.2 Cadastro*

**Figura 16 –** Tela de cadastro




Fonte: autores (2016)

Para efetuar o registro, o sistema exibe a tela de cadastro onde é necessário preencher alguns campos obrigatórios para completar o evento. A tela conta com:

- **Email:** campo para inserção do e-mail que o usuário irá usar para efetuar login futuro no sistema;
- **Senha:** campo para criação da senha do usuário;
- **Confirmar Senha:** campo para que o usuário repita a senha para efetuar uma garantia de que a senha está sendo digitada corretamente;
- **Voltar:** botão com finalidade de voltar à tela inicial da aplicação (login);

- **Cadastrar:** botão que verifica o e-mail do usuário e compara a igualdade dos dois campos de senha digitados. Caso não haja nenhuma divergência, o novo usuário é registrado na aplicação.

### *3.3.3 Página Inicial*

**Figura 17 –** Página inicial


Fonte: autores (2016)

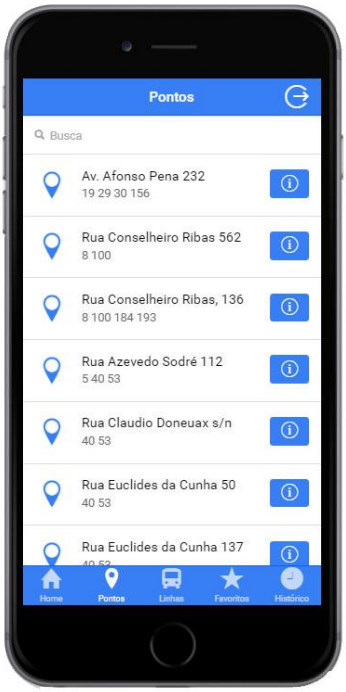
Ao efetuar o login ou acessar sem ele, é exibida ao usuário a tela Home. O mapa com a localização atual do usuário é carregado automaticamente sem nenhuma informação adicional e na tela existem os seguintes componentes:

- **Logout:** botão localizado na barra superior de navegação que ao tocado, exibe um pop-up pedindo uma confirmação ao usuário para sair do perfil usado para login;
- **Origem:** campo de texto onde se pode inserir o endereço de origem (endereço que por sua vez, é pesquisado na base de dados da API do Google Maps, ou seja, traz apenas endereços válidos) ou utilizar o endereço da localização atual do usuário para definir o ponto inicial da pesquisa;

- **Destino:** campo semelhante ao item anterior, porém com obrigatoriedade de preenchimento, pois se refere ao endereço de destino do usuário, utilizado para calcular a rota de ônibus e sugerir a ele;
- **Mostrar todos os pontos:** caixa de seleção que, ao ativada, exibe todos os pontos de ônibus existentes na cidade com um pin customizado para caso o usuário queira ver os pontos no mapa;
- **Home:** sessão na aba inferior para se direcionar ao Home;
- **Pontos:** sessão onde exibe todas as paradas em uma lista;
- **Linhas:** sessão contendo todas as linhas circulares em uma lista;
- **Favoritos:** ala dedicada aos endereços salvos pelo usuário;
- **Histórico:** ala que salva as últimas pesquisas feitas pelo usuário.

### *3.3.4 Pontos De Ônibus*

**Figura 18 –** Pontos de ônibus



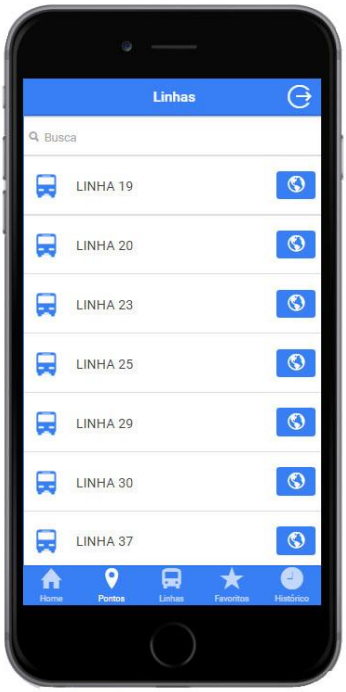

Fonte: autores (2016)

Com a aba de “Pontos” selecionada, a tela acima é exibida ao usuário. Nela temos os seguintes elementos:

- **Busca:** campo onde o usuário pode procurar um ponto a partir do endereço. Com base nisso, a pesquisa é feita automaticamente exibindo apenas resultados compatíveis com o texto digitado;
- **Lista:** na lista, consta o endereço de todos os pontos de ônibus da cidade em destaque e subsequentemente, as linhas que passam por aquele ponto;
- **Informação:** botão que ao ser tocado, exibe no mapa apenas aquele ponto para indicar ao usuário a localização da parada em questão.

### *3.3.5 Linhas*

**Figura 19 –** Linhas circulares




Fonte: autores (2016)

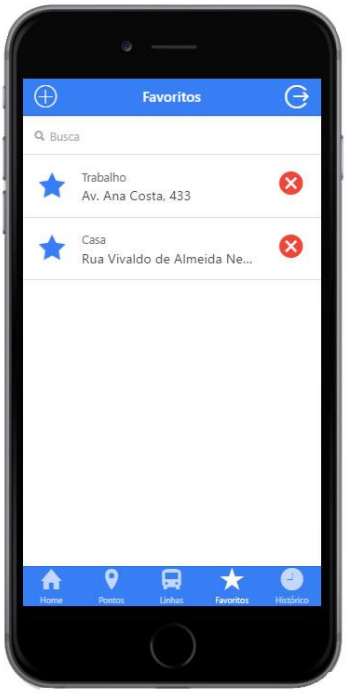
Na aba de “Linhas”, a aplicação exibe para o usuário todas as linhas circulares que passam na cidade. A tela possui:

- **Busca:** campo para procurar uma linha através da barra de pesquisa, que é feita dinamicamente;
- **Lista:** lista com todas as linhas que passam pela cidade;

- **Botão Mundo:** botão que exibe todo o percurso daquela linha específica no mapa, mostrando ao usuário todo o trajeto percorrido por aquele ônibus na cidade.

### *3.3.6 Favoritos*

**Figura 20 –** Favoritos




Fonte: autores (2016)

Em “Favoritos”, o usuário pode salvar endereços utilizados frequentemente para acesso rápido. Esta tela contém:

- **Adicionar:** botão usado para exibir um pop-up onde o usuário define um apelido da rota e um endereço salvando como Favorito para futuro acesso rápido;
- **Busca:** campo para procurar todos os endereços já salvos previamente pelo usuário;
- **Lista:** lista contendo todos os endereços salvos pelo usuário;
- **Excluir:** botão existente em cada linha onde é exibido um pop-up para excluir um endereço favorito.

### *3.3.7 Histórico*

**Figura 21 –** Histórico


Fonte: autores (2016)

A tela de Histórico exibe todas as pesquisas que o usuário efetuou, mostrando a origem e o destino usado na busca. Nela, temos:

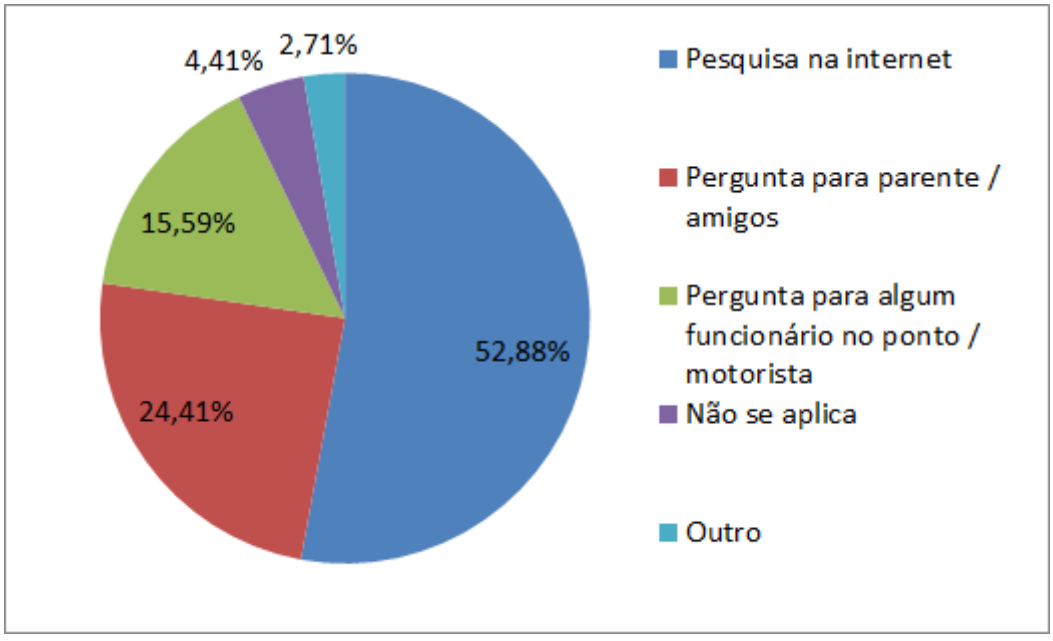
- **Busca:** campo para procurar todos os endereços buscados previamente pelo usuário;
- **Lista:** lista contendo todos os endereços buscados pelo usuário;
- **Reutilizar rota:** botão para reutilizar a rota salva no Histórico exibindo no mapa novamente o trajeto.

# 4 CONCLUSÃO

A utilização dos smartphones vem crescendo cada vez mais tanto quanto as tecnologias oferecidas para esta plataforma. Novos serviços com grandes utilidades superam as expectativas dos usuários, tornando suas vidas mais simples e mais práticas.

Com o trabalho desenvolvido foi possível concluir que muitas pessoas se perdem quando necessitam ir a algum destino diferente do que estão acostumadas. De todas as pessoas que utilizam o transporte, quase 53% das pessoas utilizam a internet para procurar meios de se localizar mediante às possíveis rotas para um determinado destino conforme figura 22 abaixo. Com isso em mente, a aplicação gerada no projeto é de fácil utilização e de grande ajuda para as pessoas que utilizam este serviço de transporte público.

**Figura 22 –** Gráfico 6: Pesquisa - Quando você precisa ir a um lugar que nunca foi antes, como faz para descobrir o ônibus que precisa pegar?




Fonte: autores (2016)

Durante o desenvolvimento, conclui-se facilmente que aplicações híbridas são muito mais simples e rápidas de serem desenvolvidas, além de possuir um custo muito menor que aplicações nativas, onde exigem conhecimentos mais específicos e equipes focadas em cada plataforma. Por outro lado, aplicações híbridas costumam

ser mais pesadas, não sendo recomendada a utilização desta tecnologia para sistemas muito grandes.

A aplicação utiliza um baixo consumo de dados, pois seu único fluxo de internet é devido ao download do mapa na tela e todas as demais informações são carregadas internamente no sistema e com isso, o aplicativo permite que o usuário utilize o serviço em qualquer lugar, podendo se localizar de forma fácil e verificar informações na medida em que lhes forem conveniente.

Ao testar o aplicativo foi constatado que o sistema está intuitivo para a experiência do usuário e se mostra de fácil utilização, exibindo ao usuário, as informações necessárias para que ele consiga visualizar todas as linhas e pontos da cidade, bem como traçar a melhor rota para alcançar um determinado destino.

# 5 REFERÊNCIAS BIBLIOGRÁFICAS

Angular JS – Creating Custom Directives. Disponível em: <https://docs.angularjs.org/guide/directive>. Acesso em: 21 jun. 2016 – 23:20.

Angular JS – Data Binding. Disponível em: <https://docs.angularjs.org/guide/databinding>. Acesso em: 15 jun. 2016 – 15:15.

APACHE CORDOVA – Platform Support. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/support/index.html>. Acesso em: 21 jun. 2016 – 16:00.

APPLE – About Swift. Disponível em: <https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/>. Acesso em: 26 jun. 2016 – 15:50.

BOEG, Jesper. Priming Kanban, Copenhagen, Dinamarca: Info Q, 2012. pág. 12.
BRANAS, Rodrigo. AngularJS Essencials, Birmingham, UK: PACKT, 2014. pág. 6.

CITTAMOBI – CittaMobi. Disponível em: <http://www.cittamobi.com.br/?1#getTheApp>. Acesso em 04 nov. 2016 – 15:30.

FIREBASE – AngularFire. Disponível em: <https://www.firebase.com/docs/web/libraries/angular/>. Acesso em: 25 jul. 2016 – 20:00.

FIREBASE – Firebase Realtime Database. Disponível em: <https://firebase.google.com/docs/database/>. Acesso em: 25 jul. 2016 – 19:30.

FREEMAN, Adam. Pro AngularJS: APRESS, 2013. pág. 1.

IBGE – Síntese das Informações. Disponível em: <http://cidades.ibge.gov.br/xtras/temas.php?lang=&codmun=354850&idtema=16&search=||s%EDntese-das-informa%E7%F5es> Acesso em: 24/09/2016 – 13:45.

IONIC – Create mobile apps with the web technologies you love. Disponível em: <http://ionicframework.com/>. Acesso em: 24 mai. 2016 – 00:05.

IONIC – CSS Components. Disponível em: <http://ionicframework.com/docs/components>. Acesso em: 12 jun. 2016 – 12:30.

IONIC – HTML5 Input Types. Disponível em: <http://ionicframework.com/html5-input-types>. Acesso em: 11 jun. 2016 – 18:45.

IONIC – Overview. Disponível em: <http://ionicframework.com/docs/overview/>. Acesso em: 09 jun. 2016 – 19:15.

IONIC SHOWCASE – Showcase. Disponível em <http://showcase.ionicframework.com/>. Acesso em: 10 ago. 2016 – 20:45.

JSON – Introducing JSON. Disponível em: <http://www.json.org/>. Acesso em: 04 nov. 2016 – 10:10.

LOPES, Sérgio. Aplicações mobile híbridas com Cordova e PhoneGap, São Paulo: CASA DO CÓDIGO, 2016. pág. 1.

MICROSOFT AZURE – NoSQL vs SQL. Disponível em: <https://azure.microsoft.com/en-us/documentation/articles/documentdb-nosql-vs-sql/>. Acesso em: 27 out. 2016 – 15:00.

MOBILE TIME. Disponível em: <http://www.mobiletime.com.br/05/01/2016/uso-dos-aplicativos-moveis-cresceu-58-em-2015/>. Acesso em: 10 nov. 2016 – 15:10.

MONGODB – NoSQL Databases Explained. Disponível em: <https://www.mongodb.com/nosql-explained>. Acesso em: 27 jul. 2016 – 15:30.

MOZILLA DEVELOPER NETWORD – About JavaScript. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript>. Acesso em: 14 jun. 2016 – 19:20.

MOZILLA DEVELOPER NETWORD – Introduction. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript>. Acesso em: 14 jun. 2016 – 19:35.

PEREIRA, Michael Henrique R. AngularJS – Uma Abordagem Prática e Objetiva, São Paulo, São Paulo: NOVATEC, 2014. pág. 14.

PINHEIRO, Roberto. Pesquisa de Mercado, Rio de Janeiro, RJ: FGV, 2011. pág. 69.

PIRACICABANA – Quanto Tempo Falta? Disponível em: <http://quantotempofalta.piracicabana.com.br>. Acesso em: 15 ago. 2016 – 18:45.

PIRACICABANA. Disponível em: <http://geosismo.piracicabana.com.br/>. Acesso em: 12 nov. 2015 – 18:50.

PLAYSTORE – CittaMobi. Disponível em: <https://play.google.com/store/apps/details?id=br.com.cittabus&hl=pt_BR>. Acesso em 15 ago. 2016 – 19:00.

RAVULAVARU, Arvind. Learning Ionic, Birmingham, UK: Packt, 2015. pág. 19.

SALEH, Hazem. JavaScript Mobile Application Development: PACKT, 2014. pág. 8.

SESHADRI, Shyam & GREEN, Brad. Desenvolvendo com AngularJS, São Paulo: NOVATEC, 2014. pág. 20.

W3 Resource – Tutorial JSON. Disponível em: <http://www.w3resource.com/JSON/introduction.php>. Acesso em: 04 nov. 2016 – 10:00.

W3C – HTML5. Disponível em: <https://www.w3.org/TR/html5/introduction.html#background>. Acesso em: 09 jun. 2016 – 13:00.

W3C Schools – AngularJS Directives. Disponível em: <http://www.w3schools.com/angular/angular_directives.asp>. Acesso em: 21 jun. 2016 – 23:35.

W3C Schools – CSS Introduction. Disponível em: <http://www.w3schools.com/css/css_intro.asp>. Acesso em: 11 jun. 2016 – 20:10.

W3C Schools – CSS3 Introduction. Disponível em: <http://www.w3schools.com/css/css3_intro.asp>. Acesso em: 11 jun. 2016 – 20:20.

W3C Schools – HTML5 Introduction. Disponível em: <http://www.w3schools.com/html/html5_intro.asp>. Acesso em: 09 jun. 2016 – 13:45.

W3C Schools – JavaScript Introduction. Disponível em: <http://www.w3schools.com/js/js_intro.asp>. Acesso em: 14 jun. 2016 – 19:45.

WAIKAR, Manoj. Data-Oriented Development With AngularJS, Birmingham, UK: PACKT, 2015. pág. 52.

WARGO, John M. Apache Cordova 4 Programming, USA: Addison Wesley, 2015. pág. 5.

WARGO, John M. Apache Cordova API Cookbook, USA: Addison Wesley, 2015. pág. 1.

What is a MVW framework?. Disponível em: <http://www.santoshshinde.com/2016/01/what-exactly-means-mvw-design-pattern.html>. Acesso em: 26 out. 2016 – 10:30.

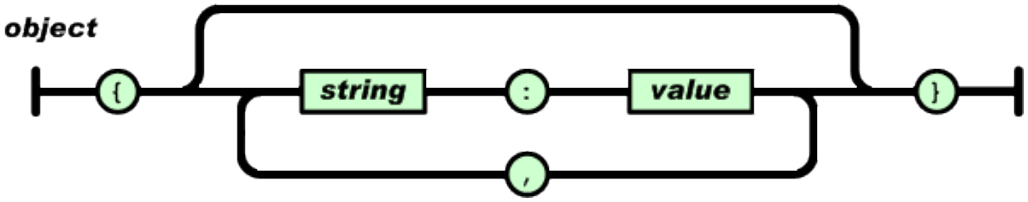
# ANEXO

# ANEXO A – Objetos JSON de Armazenamento de Dados

De acordo com sua documentação (2016), JSON (JavaScript Object Notation ou Objeto de Notação JavaScript) é um formato muito leve de troca de dados que é de fácil leitura e escrita para humanos tanto quanto é de fácil entendimento pela máquinas, pois é baseada em um subconjunto da linguagem Javascript. Em poucas palavras JSON é um formato de texto que é completamente independente de alguma linguagem de programação, porém, usa algumas bases que são bem familiares para programadores, tornando-se assim, ideal para transferência de dados.

De acordo com a página de pesquisas da W3 (2015) complementado por sua documentação (2016), a base de dados JSON funciona por meio de criação de objetos contendo um rótulo (ou identificação do atributo) e um valor respectivo onde este valor pode assumir vários tipos diferentes como texto, numérico, formato de data, array (ou lista), booleano (verdadeiro ou falso) ou até outros objetos.

**Figura 23 –** Estrutura de objeto JSON




Fonte: Documentação de JSON (2016)

A estrutura é frequentemente usada para serializar e transferir dados através de uma conexão, reforça a W3 (2015), como por exemplo, passar informações entre um servidor e uma aplicação web. Neste ambiente, serializar é o processo de transformar estruturas de dados e objetos em formatos que possam ser utilizados em sistema, com a informação já extraída, podendo trabalhar com esses dados para se encaixar em uma lógica dentro da aplicação, logo depois que os dados são recuperados. Por causa deste comportamento dos dados estruturados, JSON é fortemente utilizado para armazenar e representar dados semiestruturados.

Conforme W3 (2015) as duas maiores usabilidades do JSON é em APIs (Application Programming Interface ou Interface de Programação de Aplicação) e em base de dados NoSQL.

Seu uso nas APIs consiste em ser usada para troca de informação onde empresas possuem internamente um JSON dentro e suas interfaces para que o desenvolvedor tenha acesso a uma gama de dados para que seja possível criar aplicações derivadas.

Outra área para o uso de JSON é em NoSQL, complementa W3 (2015). Considerando que estruturas JSON podem ser transformadas em objetos em JavaScript (que são trabalhadas dentro do ambiente do navegador), se torna muito mais fácil trabalhar com ele, podendo até ser integrado com o lado do servidor em JavaScript, tornando a armazenagem dos dados de fácil entendimento para a aplicação.

# APÊNDICE

# APÊNDICE A

## Descrição Dos Diagramas De Caso De Uso

**Quadro 3 –** Caso de Uso 001

| | |
|---|---|
| Número do Caso de Uso | **UC001** |
| Nome do Caso de Uso | Login |
| Ator(es) | Usuário Cadastrado |
| Descrição | Efetua a entrada do usuário através de um e-mail e uma senha cadastrada |
| Inclusões | Não há |
| Extensões | UC002 - Verificar Favoritos<br>UC003 - Verificar Histórico<br>UC004 - Verificar lista de pontos<br>UC005 - Verificar lista de ônibus<br>UC006 - Escolher Origem<br>UC007 - Logout |

Fonte: autores (2016)

**Quadro 4 –** Caso de Uso 002

| | |
|---|---|
| Número do Caso de Uso | **UC002** |
| Nome do Caso de Uso | Verificar Favoritos |
| Ator(es) | Usuário Cadastrado |
| Descrição | Exibe para o usuário uma lista com todos os endereços salvos por ele |
| Inclusões | Não há |
| Extensões | UC008 - Adicionar aos Favoritos<br>UC009 - Reutilizar a rota |

Fonte: autores (2016)

**Quadro 5 –** Caso de Uso 003

| | |
|---|---|
| Número do Caso de Uso | **UC003** |
| Nome do Caso de Uso | Verificar Histórico |
| Ator(es) | Usuário Cadastrado |
| Descrição | Exibe para o usuário uma lista com as últimas pesquisas efetuadas por ele |
| Inclusões | Não há |
| Extensões | UC009 - Reutilizar Rota |

Fonte: autores (2016)

**Quadro 6 –** Caso de Uso 004

| | |
|---|---|
| Número do Caso de Uso | **UC004** |
| Nome do Caso de Uso | Verificar lista de pontos |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Exibe ao usuário uma lista com o endereço exato de todos os pontos de ônibus da cidade |
| Inclusões | Não há |
| Extensões | UC010 - Ver ponto no mapa |

Fonte: autores (2016)

**Quadro 7 –** Caso de Uso 005

| | |
|---|---|
| Número do Caso de Uso | **UC005** |
| Nome do Caso de Uso | Verificar lista de ônibus |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Exibe ao usuário uma lista com todas as linhas circulares da cidade |
| Inclusões | Não há |
| Extensões | UC012 - Ver trajeto completo no mapa |

Fonte: autores (2016)

**Quadro 8 –** Caso de Uso 006

| | |
|---|---|
| Número do Caso de Uso | **UC006** |
| Nome do Caso de Uso | Escolher Origem |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Ação efetuada ao definir seu ponto de origem (generalizado por endereço digitado ou por geolocalização) |
| Inclusões | UC013 - Escolher Destino |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 9 –** Caso de Uso 007

| | |
|---|---|
| Número do Caso de Uso | **UC007** |
| Nome do Caso de Uso | Logout |
| Ator(es) | Usuário Cadastrado |
| Descrição | Efetua a saída do seu perfil cadastrado |
| Inclusões | Não há |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 10 –** Caso de Uso 008

| | |
|---|---|
| Número do Caso de Uso | **UC008** |
| Nome do Caso de Uso | Adicionar aos Favoritos |
| Ator(es) | Usuário Cadastrado |
| Descrição | Usuário preenche o campo de texto e ao clicar no botão de salvar, o endereço é salvo na lista de Favoritos |
| Inclusões | Não há |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 11 –** Caso de Uso 009

| | |
|---|---|
| Número do Caso de Uso | **UC009** |
| Nome do Caso de Uso | Reutilizar a rota |
| Ator(es) | Usuário Cadastrado |
| Descrição | Utiliza novamente a rota salva previamente pelo usuário e registra no Histórico |
| Inclusões | Não há |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 12 –** Caso de Uso 010

| | |
|---|---|
| Número do Caso de Uso | **UC010** |
| Nome do Caso de Uso | Ver ponto no mapa |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Utiliza somente o ponto selecionado da lista para exibir no mapa |
| Inclusões | Não há |
| Extensões | UC011 - Selecionar rota |

Fonte: autores (2016)

**Quadro 13 –** Caso de Uso 011

| | |
|---|---|
| Número do Caso de Uso | **UC011** |
| Nome do Caso de Uso | Selecionar rota |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Utiliza ponto de ônibus selecionado para traçar uma rota até ele e salva a ação no Histórico |
| Inclusões | Não há |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 14 –** Caso de Uso 012

| | |
|---|---|
| Número do Caso de Uso | **UC012** |
| Nome do Caso de Uso | Ver trajeto completo no mapa |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Utiliza o ônibus selecionado da lista e exibe seu trajeto completo no mapa |
| Inclusões | Não há |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 15 –** Caso de Uso 013

| | |
|---|---|
| Número do Caso de Uso | **UC013** |
| Nome do Caso de Uso | Escolher Destino |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Define o endereço de destino |
| Inclusões | UC014 - Visualizar pontos disponíveis |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 16 –** Caso de Uso 014

| | |
|---|---|
| Número do Caso de Uso | **UC014** |
| Nome do Caso de Uso | Visualizar pontos disponíveis |
| Ator(es) | Usuário Cadastrado<br>Usuário Não Cadastrado |
| Descrição | Exibe os pontos disponíveis que possuem rotas para o destino selecionado |
| Inclusões | Não há |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 17 –** Caso de Uso 015

| | |
|---|---|
| Número do Caso de Uso | **UC015** |
| Nome do Caso de Uso | Cadastrar |
| Ator(es) | Usuário Não Cadastrado |
| Descrição | Efetua o cadastro através de um e-mail e uma senha de escolha do usuário |
| Inclusões | Não há |
| Extensões | Não há |

Fonte: autores (2016)

**Quadro 18 –** Caso de Uso 016

| | |
|---|---|
| Número do Caso de Uso | **UC016** |
| Nome do Caso de Uso | Acessar sem cadastro |
| Ator(es) | Usuário Não Cadastrado |
| Descrição | Acessa todas as funções da aplicação (com exceção de Favoritos e Histórico) sem necessidade de um login |
| Inclusões | Não há |
| Exclusões | UC004 - Verificar lista de pontos<br>UC005 - Verificar lista de ônibus<br>UC006 - Escolher Origem |

Fonte: autores (2016)

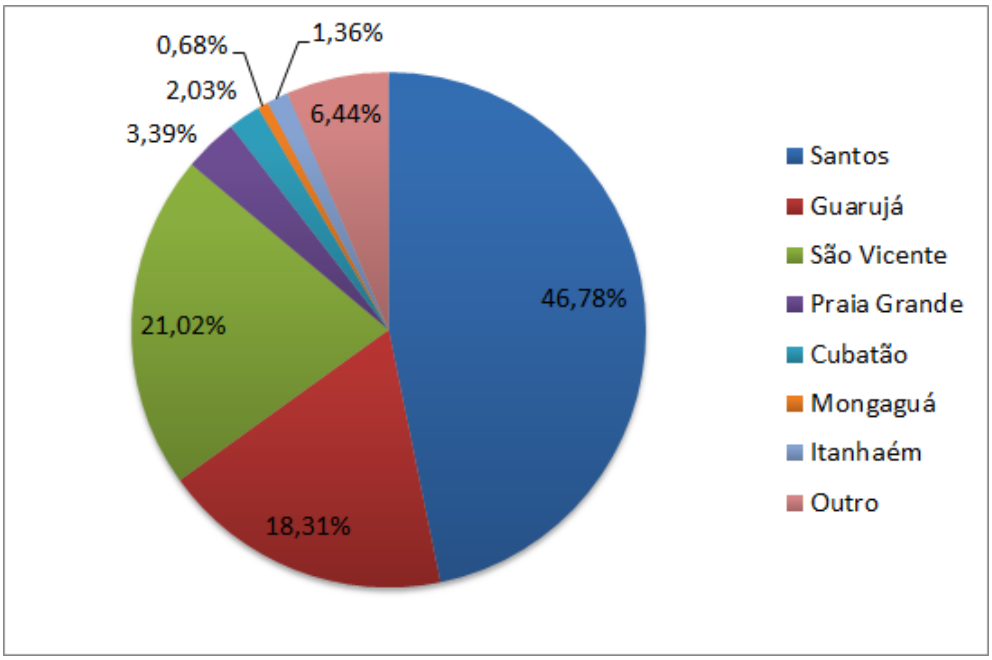
# APÊNDICE B

## Pesquisa de Campo

Foram entrevistadas um total de 295 pessoas e abaixo, a pesquisa completa juntamente com suas estatísticas:

1) Em que cidade você mora?
    a) Santos;
    b) Guarujá;
    c) São Vicente;
    d) Praia Grande;
    e) Cubatão;
    f) Mongaguá;
    g) Itanhaém;
    h) Outro.

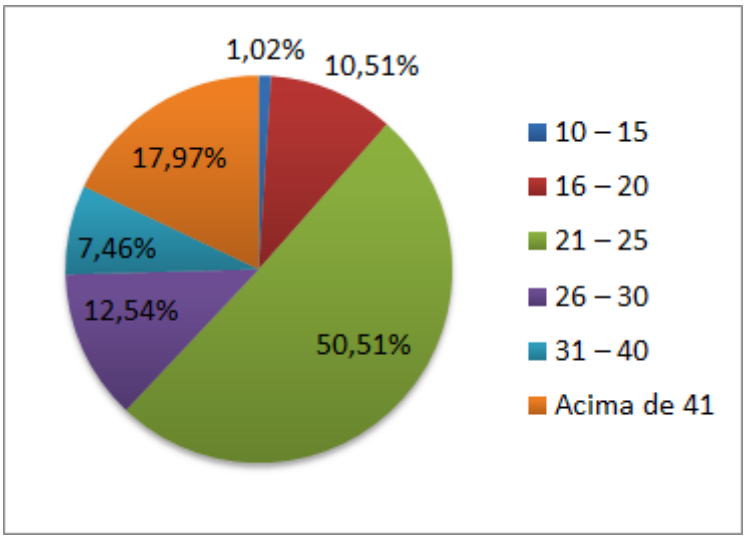
**Gráfico 1 –** Pesquisa – Em que cidade você mora?




Fonte: autores (2016)

2) Qual sua idade?
    a) 10 – 15;
    b) 16 – 20;
    c) 21 – 25;
    d) 26 – 30;
    e) 31 – 40;
    f) Acima de 41.

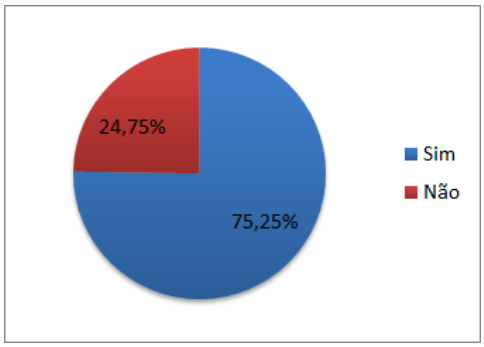
**Gráfico 2 –** Pesquisa – Qual sua idade?




Fonte: autores (2016)

3) Você utiliza o transporte público em Santos?
    a) Sim;
    b) Não.

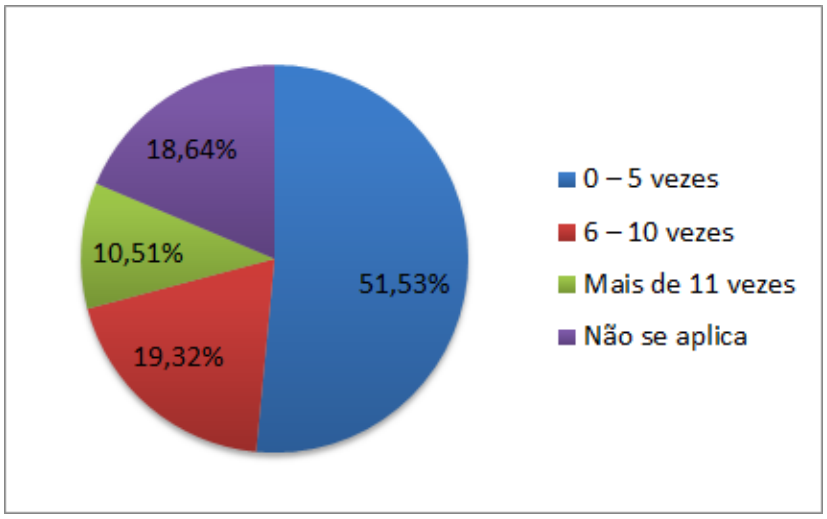
**Gráfico 3 –** Pesquisa - Você utiliza o transporte público em Santos?




Fonte: autores (2016)

4) Quantas conduções você utiliza por semana?
    a) 0 – 5 vezes;
    b) 6 – 10 vezes;
    c) Mais de 11 vezes;
    d) Não se aplica.

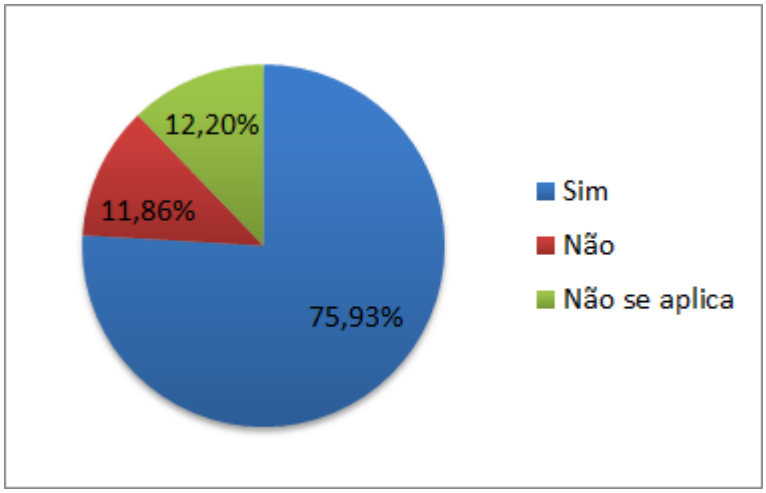
**Gráfico 4 –** Pesquisa - Quantas conduções você utiliza por semana




Fonte: autores (2016)

5) Você conhece o trajeto dos ônibus que você costuma pegar?
    a) Sim;
    b) Não;
    c) Não se aplica.

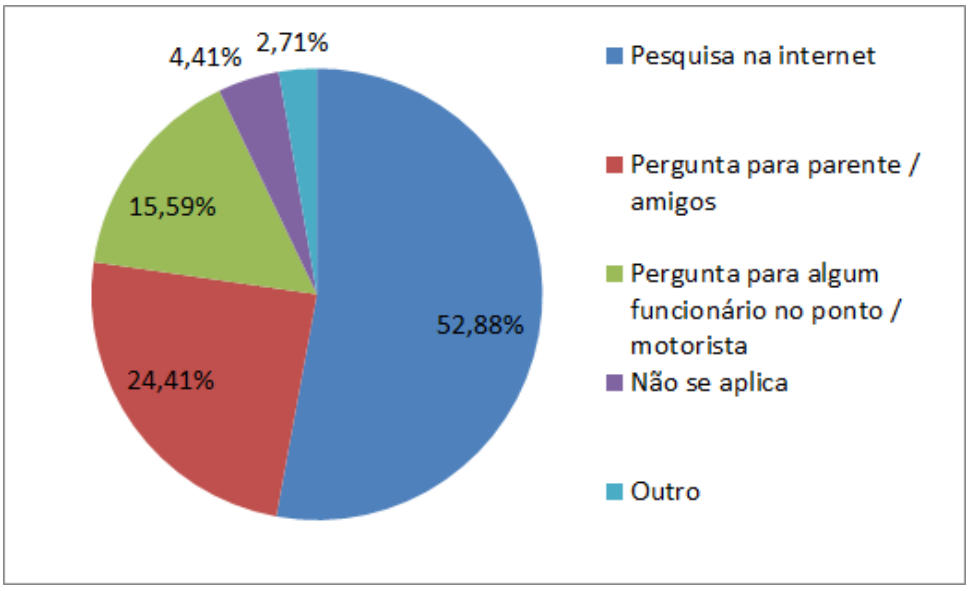
**Gráfico 5 –** Pesquisa - Você conhece o trajeto dos ônibus que você costuma pegar?




Fonte: autores (2016)

6) Quando você precisa ir a um lugar que nunca foi antes, como faz para descobrir o ônibus que precisa pegar?
    a) Pesquisa na internet;
    b) Pergunta para parente/amigos;
    c) Pergunta para algum funcionário no ponto / motorista;
    d) Não se aplica;
    e) Outro.

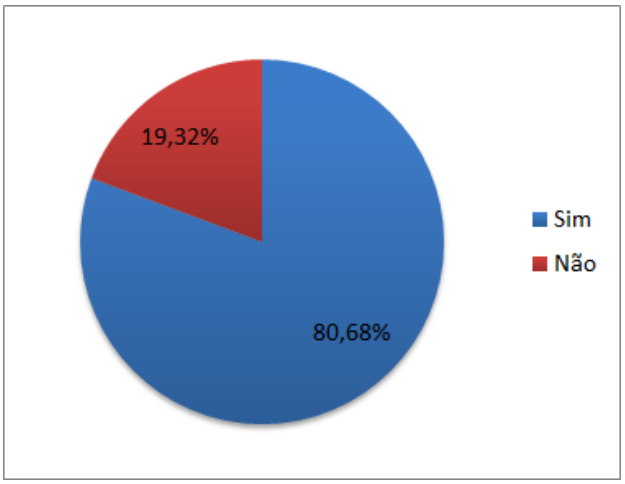
**Gráfico 6 –** Pesquisa - Quando você precisa ir a um lugar que nunca foi antes, como faz para descobrir o ônibus que precisa pegar?




Fonte: autores (2016)

7) Quando pega um ônibus, tem dificuldade em saber em qual ponto parar, com um destino nunca feito anteriormente?
    a) Sim;
    b) Não.

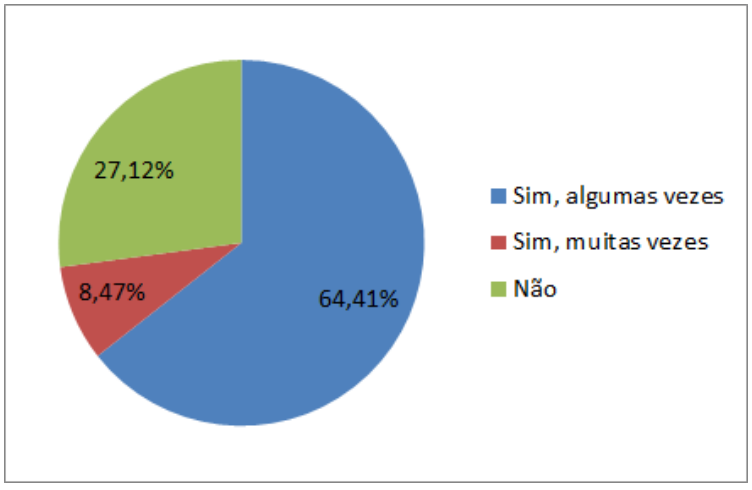
**Gráfico 7 –** Pesquisa - Quando pega um ônibus, tem dificuldade em saber em qual ponto parar, com um destino nunca feito anteriormente?




Fonte: autores (2016)

8) Já pegou alguma condução errada?
    a) Sim, algumas vezes;
    b) Sim, muitas vezes;
    c) Não, nunca.

**Gráfico 8 –** Pesquisa - Já pegou alguma condução errada?




Fonte: autores (2016)