UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Thales Godinho Kenne

BUZEE: PLATAFORMA PARA ACOMPANHAMENTO DOS HORÁRIOS DO TRANSPORTE COLETIVO

Santa Maria, RS
2018

Thales Godinho Kenne

Buzee: plataforma para acompanhamento dos horários do transporte coletivo

Trabalho apresentado ao Curso de Engenharia de Controle e Automação da Universidade Federal de Santa Maria como requisito para obtenção do título de Engenheiro de Controle e Automação.

Orientador: Rodrigo da Silva Guerra

Santa Maria, RS

2018

Thales Godinho Kenne

Buzee: Plataforma de acompanhamento dos horários de transporte coletivo

Trabalho apresentado ao Curso de Engenharia de Controle e Automação da Universidade Federal de Santa Maria como requisito para obtenção do título de Engenheiro de Controle e Automação

Aprovado em 26 de Fevereiro de 2018:

_____
Rodrigo da Silva Guerra, Dr. (UFSM)
(Presidente/Orientador)

_____
Frederico Menine Schaf, Dr. (UFSM)

_____
Rafael Concatto Beltrame, Dr. (UFSM)

Santa Maria, RS
2018

DEDICATÓRIA

Aos meus pais, Maria Teresinha e José Carlos, por tudo que me ensinaram e tudo que sou. Devo a vocês o meu futuro. À minha irmã, por ter estado ao meu lado sempre que precisei e pelos sábios conselhos em momentos difíceis. E à Clari, o Sol da minha vida. Que teu sorriso nos ilumine em dobro para cada momento que passei longe de ti perseguindo este sonho.

RESUMO

BUZEE: PLATAFORMA PARA ACOMPANHAMENTO DOS HORÁRIOS DO TRANSPORTE COLETIVO

AUTOR: Thales Godinho Kenne
ORIENTADOR: Rodrigo da Silva Guerra

Este trabalho apresenta o desenvolvimento de uma *startup* e seu produto, um aplicativo para acompanhamento de horários de ônibus em função de dados de posição geográfica fornecidas pelos usuários. Por meio deste, visa-se facilitar o planejamento da utilização de transporte coletivo em centros urbanos, mostrando a posição, em tempo real, de cada ônibus em sua linha. O aplicativo utiliza conceitos de *geofencing* para o controle de movimentação dos usuários. Com o intuito de ser facilmente aplicável, o aplicativo possui baixo custo de expansão, pois não requer instalação de um dispositivo GPS em cada ônibus. O projeto também propõe implementar um sistema de crescimento orgânico baseado em *crowdsourcing*, no qual os usuários contribuem ativamente com o processo de cadastramento de paradas e linhas, além de participar no refino das informações, indicando erros no sistema, atrasos, acidentes e outras situações comuns no trânsito urbano. Este relatório descreve o processo de desenvolvimento do aplicativo, como os diagramas UML, a programação e o processo de criação da *startup*, analisando o mercado e os planos de negócios.

Palavras-chave: Android. *Startup*. *Geofencing*. Transporte coletivo. Mobilidade Urbana.

ABSTRACT

BUZEE: A PUBLIC TRANSPORTATION TRACKING PLATFORM

AUTHOR: Thales Godinho Kenne
ADVISOR: Rodrigo da Silva Guerra

This work presents the development of an app to track bus times according to geographical positioning (data provided by the users). The aim is to facilitate the usage of urban transportation methods, showing the position, in real time, of each bus in its line. The app uses a geofencing concept to control the movement of the users. Seeking to be easily set up, the app aims to be a low cost solution, as it does not need a GPS device installed in the buses. The project also proposes an organic growth system based on crowdsourcing, in which the users actively take part in the registration of lines and bus stops, as well as in the refining of the system, by reporting delays, accidents, and other situations common to urban traffic. This thesis reports the development of the app, such as UML diagrams and programming, as well as the creation of the startup, analyzing the market and the business plans.

Keywords: Android. Startup. Geofencing. Public transportation. Urban mobility.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

CAPÍTULO 1 INTRODUÇÃO

O desenvolvimento dos grandes centros implica na necessidade de meios de transporte. Os veículos de transporte particular receberam ênfase social, sendo o meio mais utilizado. Investimentos do governo e das empresas criaram a cultura de que o transporte particular seria o meio do futuro, com mais conforto e confiabilidade de horários. Entretanto, o grande número de carros em circulação nas grandes cidades, hoje em dia, cria engarrafamentos e complicações no perímetro urbano.

> "Isso é resultado de uma política de Estado de valorização do automóvel. [...] O Estado brasileiro fez uma opção, com legitimidade social, de universalizar o acesso ao uso de automóvel. Com as medidas de incentivo – que não são pequenas –, temos mais carros na rua, a velocidade do transporte diminui e as pessoas andam mais devagar de ônibus e, estes, por sua vez, gastam mais combustível." (MANO, 2011)

O tráfego intenso e a grande quantidade de veículos também diminui a qualidade de vida na cidade com a poluição do ar e a poluição sonora (FREITAS, 2003), bem como o estresse vinculado à necessidade de atenção constante ao volante e os riscos envolvidos. Muitas pessoas passam horas, diariamente, no trânsito para chegar ao trabalho (MANO, 2011). Os engarrafamentos fazem parte da vida dos trabalhadores nos centros urbanos. Mais recentemente, há um apelo para que as pessoas utilizem mais o transporte público do que o particular (CAMPOS, 2006). O transporte público oferece a vantagem de diminuir o trânsito, pois possui maior capacidade de passageiros (CINTRA, 2008).

Mais especificamente focado para o transporte viário, tem-se dado incentivos para a eficiência do ônibus (CASTILHO, 1997), como a criação de corredores específicos para seu tráfego, evitando a competição por espaço com os demais carros. Com isso, a opção do transporte coletivo se torna mais atraente ao usuário, visto que é possível evitar as filas e congestionamentos.

Entretanto, com o aumento da utilização do transporte coletivo, um problema se torna evidente: o transporte coletivo no Brasil não é confiável. Muitas vezes, os horários fornecidos em tabelas não são cumpridos (PENA, 2013). O trânsito caótico

apresenta um empecilho ao movimento dos ônibus, alterando completamente sua rotina. Além disso, muitas vezes o usuário só tem acesso ao horário de partida do ônibus de seu ponto inicial, sem informações da atual posição do ônibus ao longo de seu trajeto, sendo apenas capaz de prever o horário de chegada do ônibus baseado em uma estimativa de tempo. Esta estimativa é, obviamente, isenta de dados como a situação do trânsito, acidentes ou até mesmo um atraso na própria partida do ônibus.

De acordo com uma pesquisa feita pela Moovit em 2014, 77% dos mercados pesquisados relataram que o principal problema dos transportes coletivos era a falta de informações sobre o horário de chegada nas paradas (HERRICK, 2015). Na cidade de Santa Maria, 110.000 pessoas utilizam o transporte coletivo diariamente (NUNES e PINHEIRO). Aplicando esta porcentagem aos usuários diários de ônibus desta cidade, seriam 84.700 pessoas insatisfeitas com a falta de informação sobre o posicionamento do ônibus.

Para resolver este problema, diversas soluções foram propostas: A mais comum e aplicada em diversos lugares, são dispositivos GPS instalados nos ônibus, que então apresentam os dados de tempo estimado em telões nas paradas. Essa solução, embora simples e satisfatória, não é aplicável em muitos lugares devido ao alto custo de instalação das telas e a falta de infraestrutura, bem como o recorrente vandalismo. A outra solução é o desenvolvimento de aplicativos, tais como o *Moovit* e o *CittaMobi*, que serão descritos mais adiante.

1.1 SOLUÇÃO PROPOSTA

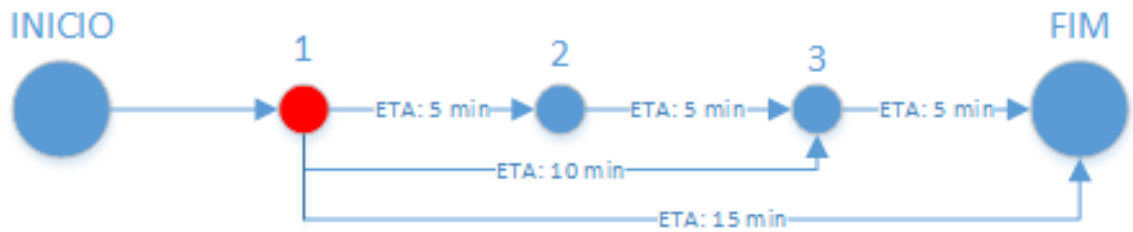
Este projeto propõe o desenvolvimento de um sistema colaborativo de localização de ônibus, no qual os usuários são a fonte da informação de coordenadas geográficas. Neste sistema, os usuários embarcados no ônibus fornecem sua localização em tempo real, sendo possível mostrar aos outros usuários do aplicativo a posição do ônibus, facilitando o planejamento da rotina diária dos usuários. Na falta de usuários embarcados com o aplicativo ativo, utiliza-se um filtro de partículas para estimar a posição do ônibus baseando-se em dados históricos.

Utilizando um sistema colaborativo de envio de posições por parte dos usuários, é possível contornar a necessidade de envolvimento da concessionária de ônibus responsável pelas linhas, que é muitas vezes o fator limitador na instalação de

tais tecnologias. Assim, não é necessária a instalação de um dispositivo GPS nos ônibus, e os custos de instalação são eliminados.

Tomando como exemplo uma situação em que um ônibus já se encontra na parada 1, como pode ser visto na Figura 1, um usuário que se encontra na parada 3 terá a informação de que faltam 10 minutos para o ônibus chegar em sua parada, enquanto um usuário na parada 2 verá que faltam 5 minutos.

Figura 1 - Diagrama de funcionamento (ETA – Estimated Time of Arrival).



Fonte: Autor.

1.2 OBJETIVOS

Os objetivos apresentados nesta seção são abstratos, não necessariamente aplicáveis ao estado atual do projeto. Eles são apenas idealizados para o projeto finalizado, quando for lançado ao mercado.

O objetivo deste projeto é desenvolver um aplicativo *android* de fácil utilização e interface simples que acompanhe a localização do ônibus ao longo do seu trajeto para informar aos usuários sobre o instante em que o ônibus chegará a cada parada. O aplicativo também poderia enviar notificações, baseadas no padrão de utilização do usuário, de que o ônibus que ele sempre toma neste determinado horário já está a caminho.

Sendo um aplicativo com o intuito de facilitar a vida dos usuários de transporte coletivo, o aplicativo também poderá incluir um sistema de compra *online* de passagens e pagamento utilizando NFC, sem a necessidade da utilização do cartão de embarque do ônibus. Assim, o usuário poderia simplesmente passar na roleta utilizando seu próprio celular.

Um problema observado é o desconhecimento dos itinerários das linhas. Nem sempre os usuários sabem qual ônibus pegar para chegar em um determinado lugar. Assim, o aplicativo ofereceria um sistema de pesquisa de rotas, mostrando como chegar em um determinado lugar utilizando o sistema de transporte público.

Por ser um sistema de fácil expansão, o projeto visa conquistar o mercado de aplicativos de mobilidade urbana das cidades de médio porte do sul do Brasil antes de começar a expansão para as ouras regiões.

1.3 METAS

Meta 1 - Uma das metas deste projeto é desenvolver um sistema gráfico de representação da movimentação dos ônibus, o que o diferenciaria da concorrência. Os aplicativos existentes no mercado, na falta da um dispositivo GPS, trabalham apenas com estimativas de tempo para o ônibus chegar em uma parada, o que não transmite confiança para o usuário. Com o sistema proposto, é possível representar graficamente o ônibus no mapa quando um usuário embarcado estiver transmitindo sua localização.

Meta 2 - Baseando-se na localização dos ônibus com usuários ativos deve-se calcular também o tempo estimado de chegada deste na parada. Esta informação será apresentada ao usuário para facilitar o seu planejamento diário.

Meta 3 -Desenvolvimento de um sistema colaborativo no qual os usuários são parte do processo e ativamente contribuem para a localização dos ônibus. Nesta parte, deve-se também correlacionar os dados fornecidos por diferentes usuários para a geração do sistema gráfico.

Meta 4 -Visto que o aplicativo é base para o desenvolvimento de uma *startup*, deve-se desenvolver um plano de negócios sólido com estratégias para conquista do mercado, bem como incentivos para a utilização do aplicativo e entregar ao usuário o valor proposto.

1.4 EQUIPE

Embora o desenvolvimento inicial tenha sido feito principalmente pelo autor, ao longo do trajeto deste projeto outros membros se juntaram a equipe. Atualmente, a

equipe é formada por 6 integrantes: quatro alunos de Engenharia de Controle e Automação, incluindo o autor; um aluno da Engenharia Elétrica e um aluno de Tecnologia da Informação. Ainda em desenvolvimento e à procura da sua área específica dentro do projeto, são responsáveis pelo desenvolvimento das ideias e atividades que realizamos em desafios de empreendedorismo, como o Desfio i9 de Empreendedorismo e o Renault Experience.

1.5 ORGANIZAÇÃO DO TRABALHO

O CAPÍTULO 2 apresenta os conceitos teóricos utilizados no desenvolvimento. No CAPÍTULO 3 serão abordadas as soluções já existentes no mercado para o problema proposto, analisando suas vantagens e desvantagens. O CAPÍTULO 4 apresenta as técnicas de programação utilizadas, as ferramentas de planejamento, um formulário feito para estudar o público alvo e o lançamento de um grupo de WhatsApp tratado como *Minimum Viable Product* (MVP) para estudar o comportamento do público sobre o compartilhamento de localização. O CAPÍTULO 5 descreve o aplicativo desenvolvido no projeto anterior. O CAPÍTULO 6 apresenta o desenvolvimento técnico do projeto atual e a sua programação. O CAPÍTULO 7 demonstra o modelo de negócios e a estratégia de mercado planejada. O CAPÍTULO 8 apresenta as conclusões do desenvolvimento do projeto.

CAPÍTULO 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos teóricos utilizados no desenvolvimento deste projeto.

2.1 GPS – *GLOBAL POSITIONING SYSTEM* (SISTEMA DE POSICIONAMENTO GLOBAL)

O GPS, *Global Positioning System*, ou Sistema de Posicionamento Global, é um sistema de localização baseado em trilateração por satélite, sendo necessários três satélites para retornar posição, ou quatro para dados de altitude. Em 2015, existiam 74 satélites de posicionamento global em órbita, sendo 28 do sistema russo GLONASS, 32 do sistema americano, 8 europeus e 17 chineses (GEOEDUC, 2015).

O desenvolvimento do GPS iniciou-se em 1960, em um projeto militar chamado NAVSTAR, mas só foi completado em 1995, com um custo total de 10 bilhões de dólares. Inicialmente de uso exclusivamente militar, hoje são utilizados gratuitamente para aplicações civis.

Dispositivos receptores de sinal GPS são variados, mas este projeto foca especialmente nos celulares. Celulares são comuns hoje em dia, e portanto, um receptor GPS está no bolso de cada usuário.

2.1.1 Trilateração

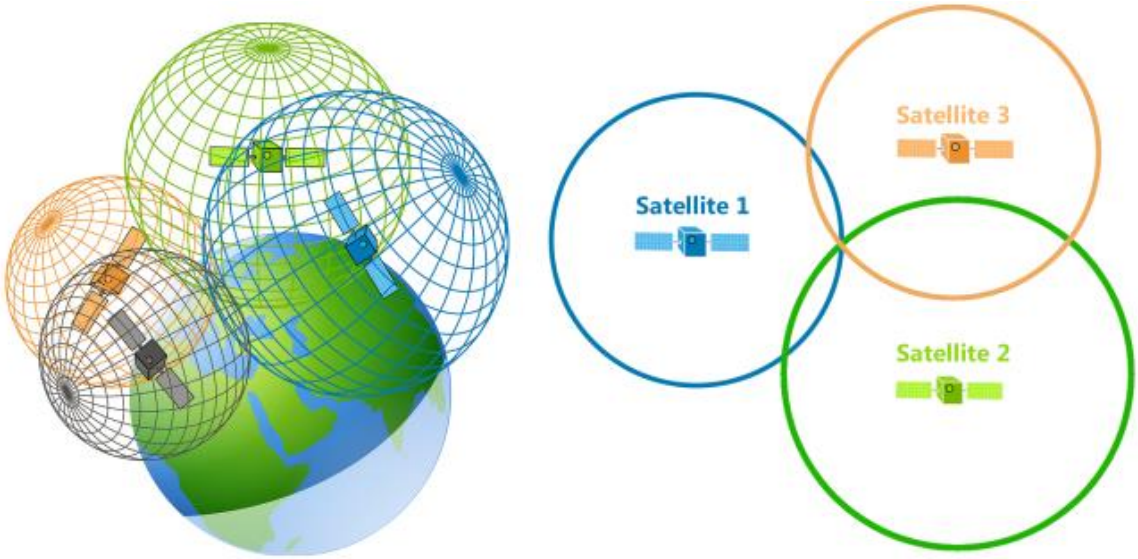
A trilateração é um conceito matemático utilizado nos Sistemas de Posicionamento Global. Utilizando os satélites GPS como pontos centrais de esferas, pode-se aferir a posição de um receptor GPS baseando-se na intersecção entre três destas esferas.

Um dispositivo receptor de GPS baseia-se em informações de tempo para calcular sua distância até cada um dos satélites. Assim, a diferença de tempo entre o sinal enviado do satélite e o momento de recepção do sinal indica a distância. Para garantir precisão na leitura, o horário interno do dispositivo e do satélite devem estar em sincronia. Os satélites são equipados com relógios atômicos e são responsáveis

pela atualização da hora dos dispositivos receptores, assegurando a qualidade da leitura.

Comumente confundido com triangulação, a trilateração utiliza diferenças de tempo para medir distâncias e funciona em um plano de três dimensões, enquanto a triangulação utiliza ângulos para a distância e funciona em um plano de duas dimensões (MELO, SOUZA e DA SILVA, 2012). Na Figura 2 pode ser observado um exemplo gráfico de trilateração.

Figura 2 – Trilateração.



Fonte: (www.gisgeography.com)

2.2 GEOFENCING

Geofencing é um conceito de *Location Based Services* (*LBS*). Françoso (2016) define LBS como serviços que utilizam a localização de um alvo para agregar valor ao serviço. A utilização inicial de LBS foi para serviços de emergência, logo sendo aderida à utilização cotidiana.

Geofencing é um conceito que se baseia em criar “cercas virtuais” ao redor de pontos de interesse. Ele se baseia na informação de posição do usuário provida pelo sistema GPS para ativar eventos quando um dispositivo entra ou sai do alcance da “cerca” (NAMIOT, 2013).

Como exemplo, pode-se usar uma situação em que uma pessoa está próxima de uma cafeteria que tem um sistema de *geofencing*. Esta pessoa, então, poderia receber uma notificação anunciando alguma promoção desta cafeteria, por exemplo. Na Figura 3, uma figura representando o conceito de *geofencing.*

Figura 3 – Geofencing.

Fonte: (www.geofencing.com)

2.3 HAVERSINE

A fórmula de Haversine é utilizada para calcular distância entre dois pontos ao longo da superfície terrestre. Inicialmente utilizado para navegação, na qual o ápice de tecnologia eram tabelas logarítmicas, o seno verso (*versine*, origem da palavra haversine) permitia manter as funções trigonométricas positivas. Além disso, tabelas de Haversine permitiam realizar os cálculos sem necessidade de resolver raízes, atividade com alto índice de erro (VENESS, 2017).

A equação é eficiente até mesmo para distâncias curtas. Para um par de pontos de latitude e longitude (φ_1,λ_1) e (φ_2,λ_2), respectivamente, a distância é dada por:

$$\Delta\varphi = (\varphi_2 - \varphi_1) * \frac{\pi}{180} \quad (1)$$

$$\Delta\lambda = (\lambda_2 - \lambda_1) * \frac{\pi}{180} \quad (2)$$

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cos\varphi_2 \sin^2(\Delta\lambda/2) \quad (3)$$

$$c = 2arctg\left(\frac{\sqrt{a}}{\sqrt{(1-a)}}\right) \quad (4)$$

$$d = Rc \quad (5)$$

Em que R representa o raio da Terra (6371 km), “c” é a distância angular em radianos e “a” é o quadrado da metade da distância entre os pontos.

2.4 MÉDIA ARITMÉTICA DE COORDENADAS GEOGRÁFICAS

O cálculo das médias é necessário para obter a posição estimada do ônibus quando mais de um usuário estiver embarcado. Com apenas um usuário embarcado, a posição do ônibus é considerada como a posição do próprio usuário, e esta é dependente da precisão de um único dispositivo celular. Quando mais usuários estiverem embarcados, a média das posições geográficas é realizada e o erro de precisão de cada dispositivo é mitigado.

Para distâncias de até 400 km, pode-se utilizar um modelo plano da esfera terrestre e conseguir resultados aproximados da média entre pontos de coordenadas geográficas. Entretanto, a média feita deste modo é dependente da localização a ser utilizada. Como o plano terrestre é dividido longitudinalmente de -180° a 180°, cálculos feitos deste modo sobre pontos próximos à linha internacional de data podem resultar em erros tremendos, visto que o ponto -180° e 180° representam a mesma longitude.

Para a aplicação atual, como desenvolvimento do sistema para o território brasileiro e com distâncias urbanas pequenas, este problema não é relevante. O sistema de média aritmética é de mais rápido processamento que o método que compensa o plano esférico, apresentado abaixo, e por isso foi escolhido.

Neste caso a média é simplesmente calculada como nas equações (6) e (7):

$$\varphi = \frac{\varphi_1 + \varphi_2 + \cdots + \varphi_n}{n} \quad (6)$$

$$\lambda = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_n}{n} \quad (7)$$

2.4.1 Cálculo de ponto médio de coordenadas geográficas compensando o plano esférico

Como descrito anteriormente, o sistema de média aritmética entre dois pontos de coordenadas geográficas pode levar a erros quando executado próximo a linha internacional de data. Embora para o sistema atual do aplicativo tenha-se escolhido utilizar a média aritmética, pode-se também utilizar um cálculo mais complexo para obter resultados mais confiáveis.

Neste caso, converte-se as coordenadas geográficas em um sistema cartesiano (x,y,z) e uma linha é traçada do centro da terra até esta nova coordenada. Onde esta linha cruzar com a superfície da terra, encontra-se a média entre os pontos (http://www.geomidpoint.com/).

Primeiro, converte-se as coordenadas de graus para radianos:

$$\varphi_1 = \varphi_1 \frac{\pi}{180} \tag{8}$$

$$\lambda_1 = \lambda_1 \frac{\pi}{180} \tag{9}$$

Em seguida, converte-se as coordenadas em radianos para o sistema cartesiano (x,y,z):

$$X_1 = \cos(\varphi_1)\, cos(\lambda_1) \tag{10}$$

$$Y_1 = \cos(\varphi_1)\, sen(\lambda_1) \tag{11}$$

$$Z_1 = \operatorname{sen}(\varphi_1) \tag{12}$$

Basta repetir estas equações para o número total de coordenadas e fazer a média da soma dos resultados:

$$X = \frac{X_1 + X_2 + \cdots + X_n}{n} \tag{13}$$

$$Y = \frac{Y_1 + Y_2 + \cdots + Y_n}{n} \tag{14}$$

$$Z = \frac{Z_1 + Z_2 + \cdots + Z_n}{n} \tag{15}$$

Com as médias calculadas, deve-se retornar os valores das coordenadas cartesianas para coordenadas geográficas:

$$\lambda = atan2(X, Y) \tag{16}$$

$$Hyp = sqrt(X^2 + Y^2) \quad (17)$$

$$\varphi = atan2(Hyp, Z) \quad (18)$$

As coordenadas calculadas até então, são em radianos. Devemos convertê-las para graus e o ponto médio entre as duas coordenadas iniciais foi estabelecido.

2.5 *BIG DATA*: ANÁLISE PREDITIVA

Big Data é um conceito que representa uma grande volume de dados armazenados, que podem ser analisados com algum fim específico. Existem quatro tipos de análises de *Big Data*, das quais a mais conhecida é a análise preditiva.

Este tipo de análise se vale de informações históricas para tentar prever o futuro de uma forma analítica. Assim, sabendo os padrões de determinados eventos, é possível prever com relativa precisão as chances destes eventos acontecerem novamente no futuro.

Um exemplo deste tipo de análise foi o trabalho realizado pela empresa Hekima para o Governo Federal, no qual a empresa analisava dados de movimentação nas ruas e informações das redes sociais para prever quais manifestações seriam mais ou menos agressivas durante a Copa do Mundo. Assim, era possível alocar forças policias analiticamente para os locais de maior risco, aumentando a eficiência do sistema de segurança (HEKIMA, 2016).

Aplicando este sistema de análise preditiva ao projeto proposto, pode-se fazer a análise dos dados salvos no banco de dados para prever os tempos de deslocamento do ônibus em função de diferentes condições, como horário, época do ano e condições climáticas, por exemplo.

2.6 *BIG DATA*: ANÁLISE DIAGNÓSTICA

A análise diagnóstica tem como objetivo compreender de maneira causal todas as suas possibilidades (HEKIMA, 2016). Quando esta análise é executada sobre os dados de um banco de dados, pode-se perceber as relações entre eventos e suas consequências. Isto permite que uma empresa analise se suas decisões foram efetivas e se devem ou não ser revertidas de acordo com seus objetivos.

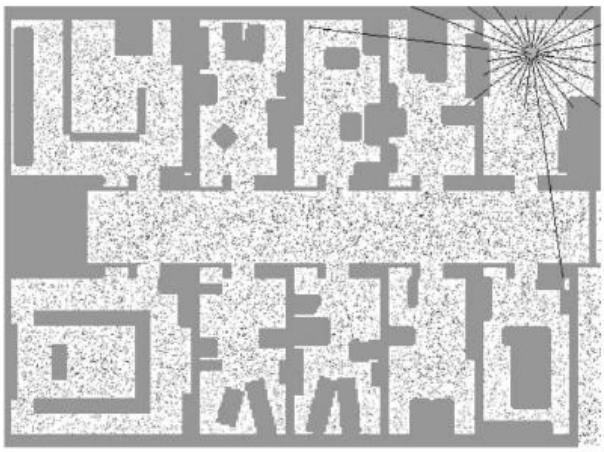
Este tipo de análise poderia ser aplicado a mudanças no trânsito, por exemplo. Tendo informações sobre como o trânsito fluía em uma determinada região, pode-se analisar os impactos de uma mudança no trânsito comparando os dados gerados na nova situação com os dados antes da mudança.

2.7 FILTRO DE PARTÍCULAS

Filtro de partículas é uma ferramenta utilizada para estimar a localização de um objeto, baseando-se em um modelo de como o sistema muda ao longo do tempo, e possivelmente *input* externos (HSIAO, DE PLINVAL-SALGUES e MILLER, 2005). Outros métodos, como o filtro de Kalman, tentam utilizar modelos simplificados do problema, visando obter resultados concretos. O filtro de partículas, por outro lado, utiliza modelos mais complexos, mas retorna resultar aproximados, ao invés de específicos.

Costumeiramente utilizados para localização de robôs, os filtros de partículas geram uma grande quantidade de partículas em uma área, tentando estimar a localização do robô. Estas partículas são analisadas pelo sistema, que determina se as partículas representam, ou não, uma solução próxima à localização estimada do robô. Partículas ruins são excluídas e as melhores são replicadas. O sistema também pode fazer uso dos sensores do robô, por exemplo, para identificar obstáculos ou padrões na localização e comparar com as partículas existentes.

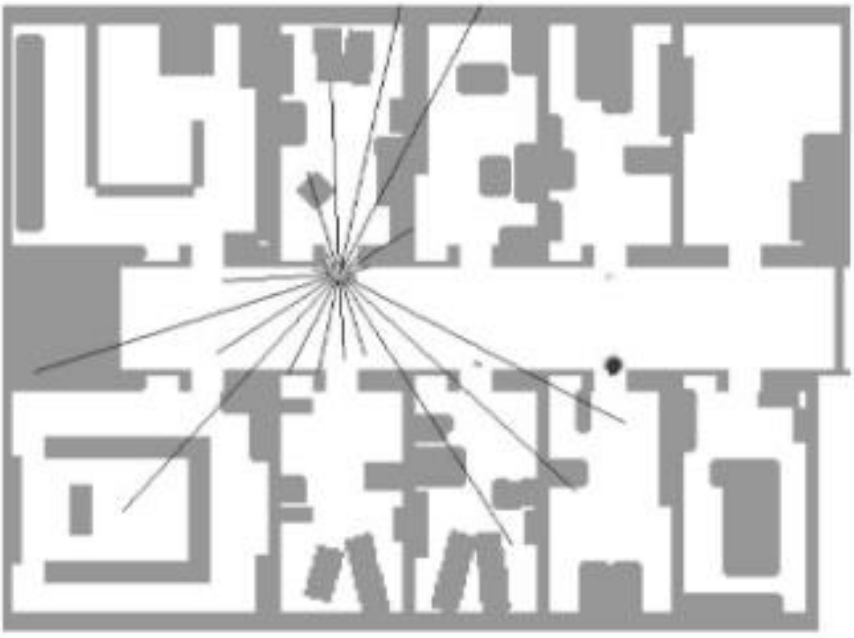
Figura 4 - Filtro de partículas em estado inicial.



Fonte: (HSIAO, DE PLINVAL-SALGUES e MILLER, 2005).

Na Figura 4 - Filtro de partículas em estado inicial.pode-se observar um modelo de filtro de partículas em estado inicial, com diversas partículas tentando estimar a posição do robô, que se encontra no campo superior direito, no meio das linhas. Neste exemplo, as linhas representam os sensores de proximidade do robô. Elas atravessam as paredes pois a posição representada é apenas estimada. Como é possível observar, diversas partículas na figura estão muito longe de representar a posição real do robô e o sistema perceberá isso. Baseando-se nas informações obtidas pelos sensores, como a distância das paredes, estas partículas serão removidas, e as que parecem representar melhor a posição do robô serão replicadas. Assim, após algumas iterações, existirão “nuvens” que podem representar a localização do objeto. Esta situação pode ser observada na Figura 5. Neste estado, o sistema encontra-se muito próximo de estimar a localização real. Pode-se observar que, baseando-se na leitura dos sensores, o sistema estima duas posições possíveis do objeto: uma, representada no centro das linhas dos sensores, e outra um pouco abaixo, à direita. O sistema considera que a melhor representação para a localização é a primeira das duas, e o experimento está pronto com um resultado aproximado.

Figura 5 - Filtro de partículas em estágio avançado.

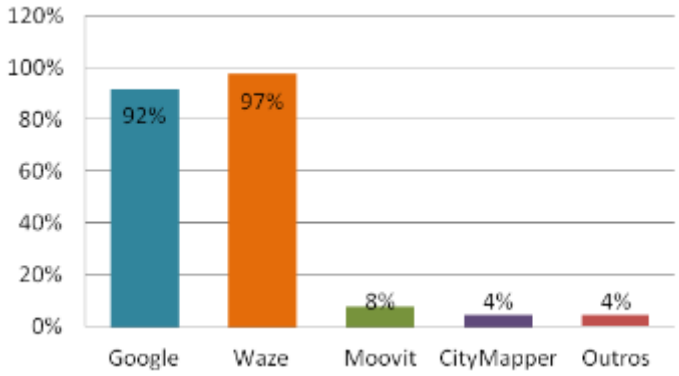


Fonte: (HSIAO, DE PLINVAL-SALGUES e MILLER, 2005).

CAPÍTULO 3 REVISÃO BIBLIOGRÁFICA

Uma pesquisa realizada por Françoso (2016) mostra que dos 312 entrevistados, 266 alegaram utilizar aplicativos de trânsito. Aproximadamente 90% dos entrevistados eram brasileiros. Pode-se notar uma utilização majoritária de aplicativos de transporte privado, como mostrado na Figura 6.

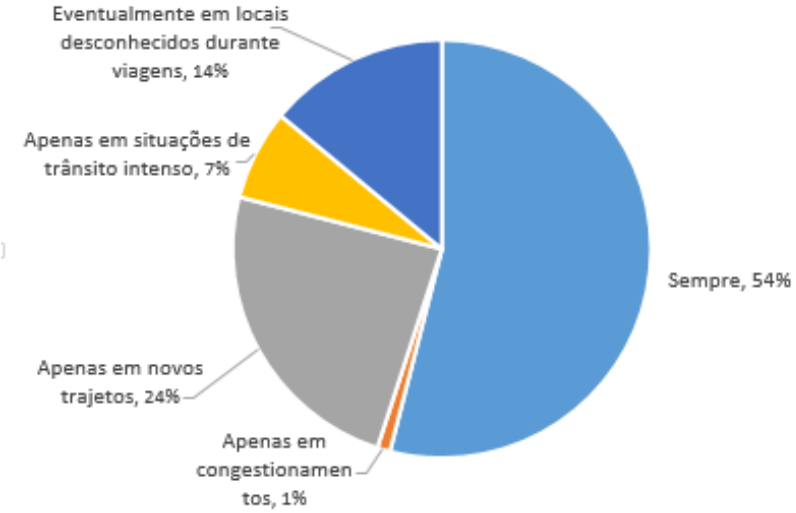
Figura 6 - Comparação de usuários por aplicativo.



Fonte: Françoso (2016).

Ainda de acordo com Françoso (2016), pode-se notar que mais da metade dos usuários utiliza os aplicativos para *smartphones* com frequência, como mostrado na Figura 7. Segundo o estudo, os usuários têm um alto nível de confiança nas informações providas pelos aplicativos.

Figura 7 - Frequência de utilização de aplicativos.



Fonte: Françoso (2016).

Pode-se notar através dos estudos que a maioria dos usuários entrevistados costuma utilizar aplicativos em todas as suas jornadas. Os dados indicam que os usuários utilizam os aplicativos para transporte privado com mais frequência que os de transporte coletivo. Isso talvez se deva aos usuários entrevistados pelo estudo não utilizarem transporte público com frequência ou que os aplicativos de transporte coletivo ainda não tenham conquistado o consumidor brasileiro. Se este for o caso, o aplicativo desenvolvido no projeto tem espaço de mercado para explorar.

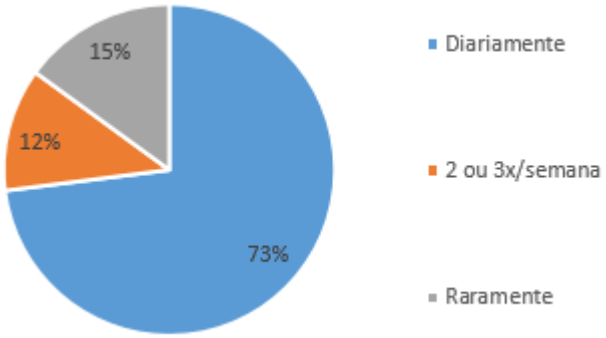
3.1 FORMULÁRIO DE VALIDAÇÃO

Para validar a necessidade de um projeto nesta área foi efetuada uma pesquisa com formulário sobre a situação do transporte público em Santa Maria e a relação dos usuários com aplicativos de mobilidade urbana neste contexto.

Nesta seção serão apresentadas as análises das respostas dos usuários para cada uma das perguntas do formulário realizado. O propósito do formulário era ajudar a analisar a necessidade de um aplicativo de mobilidade urbana e qual a situação deste ramo na cidade de Santa Maria. No total, 135 pessoas foram entrevistadas, sendo 90 pelo formulário online e 45 por entrevista oral executada nas paradas de ônibus da cidade.

Na primeira pergunta, na qual os usuários foram questionados sobre a sua frequência de utilização do transporte coletivo, 73% responderam que o utilizam diariamente, como pode ser visto na Figura 8.

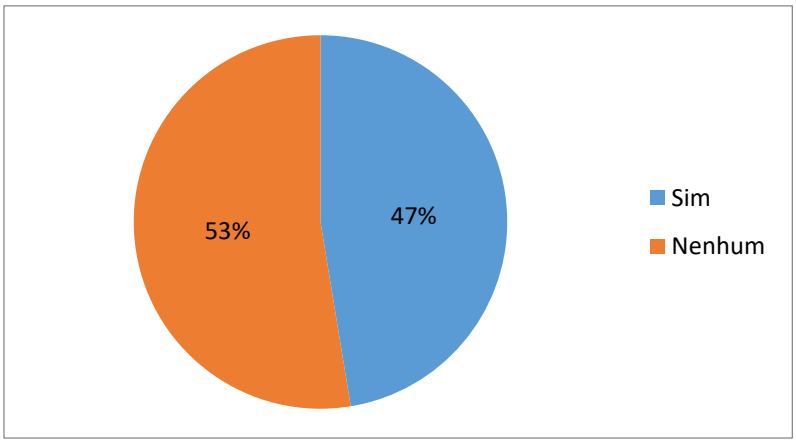
Figura 8 - Frequência de utilização do transporte coletivo dos usuários entrevistados.



Fonte: Autor

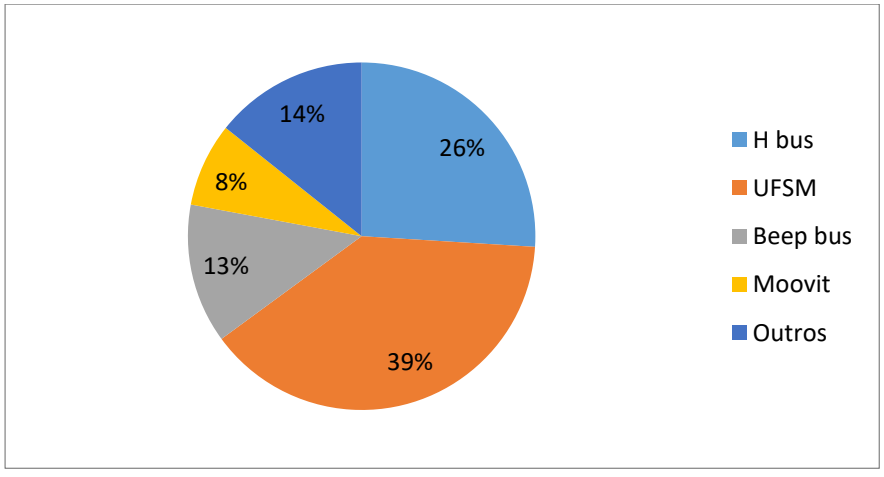
Como mostrado na Figura 9, das 135 pessoas, 64 conhecem aplicativos úteis para o sistema de transporte coletivo, totalizando 47% do total de entrevistados. É interessante comparar este resultado com os dados mostrados na Figura 10. É possível observar que uma grande parte dos usuários relatou o conhecimento do aplicativo da UFSM, que além de outras funcionalidades relacionadas à universidade, também mostra os horários de ônibus. Tal resultado era esperado, visto que grande parte dos usuários do transporte coletivo são estudantes. O HBus, outro aplicativo mencionado frequentemente recebeu críticas por não se manter atualizado. Enquanto isso, o Moovit, que é o aplicativo líder no mercado como mencionado anteriormente, recebeu menção de apenas 8% dos usuários pesquisados.

Figura 9 - Usuários que conhecem aplicativos de transporte coletivo de acordo com a pesquisa



Fonte: Autor

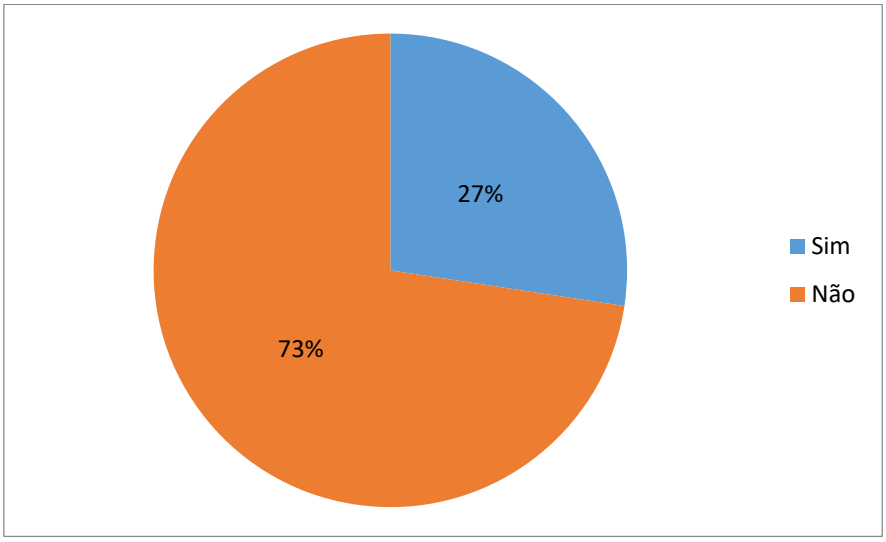
Figura 10 - Aplicativos conhecidos relatados a pesquisa



Fonte: Autor

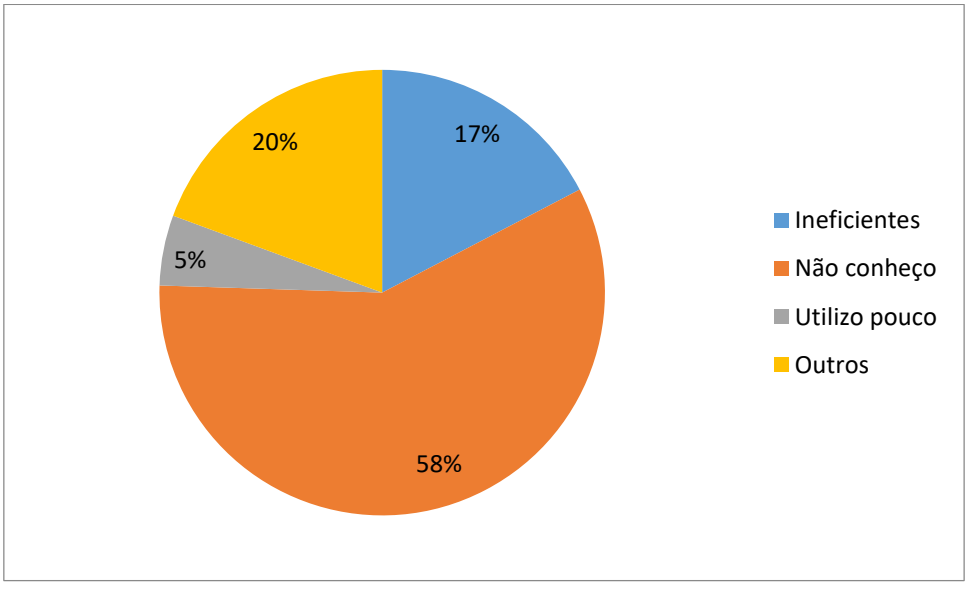
Na Figura 11 é possível perceber a pequena adesão dos pesquisados aos aplicativos de transporte coletivo. Já na Figura 12 mais da metade dos pesquisados relataram que o motivo de não utilizarem estes aplicativos é o fato de não conhecê-los. Isto remete ao assunto discutido anteriormente, de que os aplicativos existentes na área não quebraram a barreira de publicidade e se mantêm desconhecidos por parte dos usuários. Outro motivo é a imprecisão dos aplicativos existentes, tanto pelos horários que não são atualizados quanto pela falta de informação.

Figura 11 - Usuários que utilizam aplicativos de transporte coletivo



Fonte: Autor

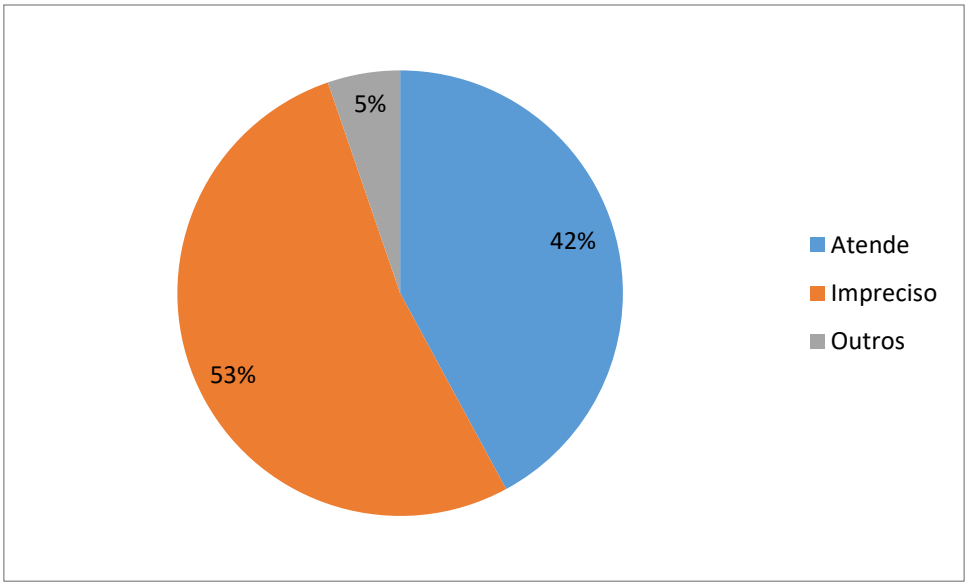
Figura 12 - Motivos da não utilização de aplicativos de transporte coletivo



Fonte: Autor

Aos usuários que responderam que utilizam aplicativos do gênero, uma pergunta sobre sua satisfação com o a qualidade de serviço foi feita. Destes, 42% responderam que o aplicativo utilizado atende à sua necessidade, enquanto o restante teve reclamações, mas os utiliza do mesmo modo. As reclamações consistiam de imprecisão no horário, interface complicada, falta de atualização dos horários e apenas horários de saída indicados.

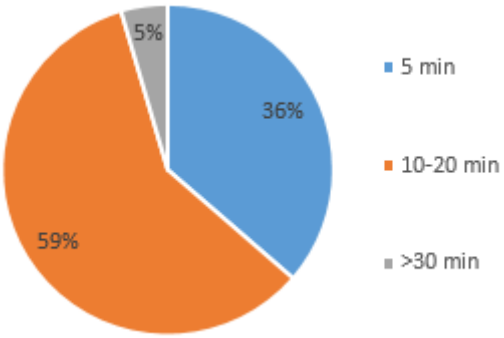
Figura 13 - Satisfação com o aplicativo utilizado



Fonte: Autor

Como um dos valores do aplicativo proposto por este projeto é que as pessoas possam se programar para perder o menor tempo possível nas paradas, foi feita uma pergunta sobre o tempo de espera dos entrevistados.

Figura 14 - Tempo de espera nas paradas de ônibus relatado



Fonte: Autor

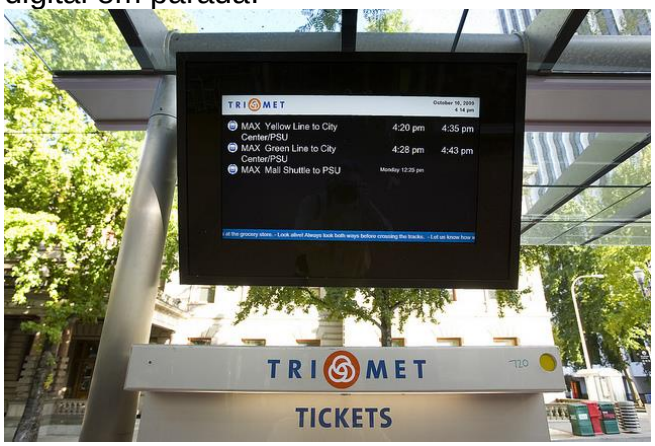
É possível perceber que mais da metade dos entrevistados espera mais de 10 minutos na parada. Estas pessoas poderiam se beneficiar de um sistema como o proposto. Também é importante ressaltar que mesmo utilizando aplicativos, algumas destas pessoas ainda mantinham uma boa margem de tempo de espera por falta de confiança nos horários apresentados.

Boa parte das pessoas revelou se programar para chegar mais cedo para aguardar o ônibus. Ainda, nas entrevistas orais foi adicionada a seguinte pergunta: Você costuma esperar muito tempo pelo seu ônibus?. Como resposta, cerca de 60% das pessoas afirmaram que sim, sendo que algumas chegam a esperar mais de uma hora. Isso piora quando perdem o seu ônibus. As pessoas que informaram não esperar muito tempo também disseram que não têm esse problema devido a possibilidade de poderem pegar vários ônibus diferentes.

3.2 SOLUÇÕES TRADICIONAIS

Uma das soluções comuns para o problema proposto é a utilização de telas nas paradas, como pode ser visto na Figura 15. Embora eficientes, são uma solução de alto custo e muito vulneráveis a vandalismo. Esse sistema também requer a instalação de um dispositivo GPS nos ônibus a serem monitorados.

Figura 15 - Tela digital em parada.

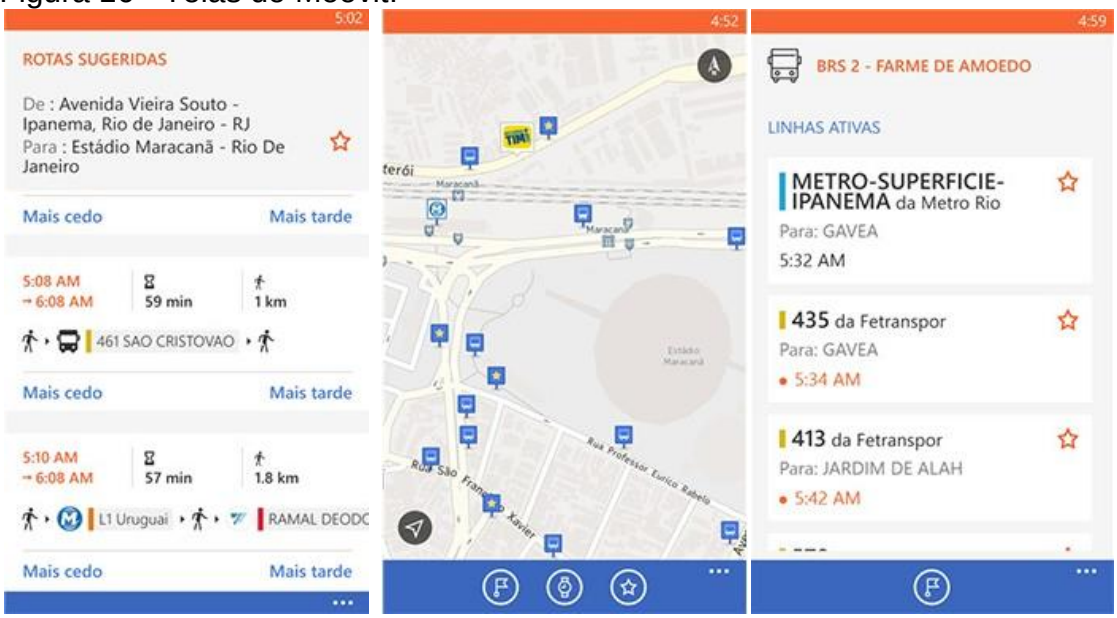


Fonte: (www.trimet.org).

O Moovit é um aplicativo consolidado no mercado que combina todas as opções de transporte público, tais como ônibus e metrô. Com ele, é possível planejar

as rotas, consultar os horários de linhas, tempo real de chegada, bem como ser informado da parada em qual o usuário deve descer para chegar ao seu destino. Uma funcionalidade especial deste aplicativo são os alertas de serviço que avisam o usuário sobre algo inesperado que tenha acontecido em sua linha. Uma compilação de telas deste aplicativo podem ser vistas na Figura 16.

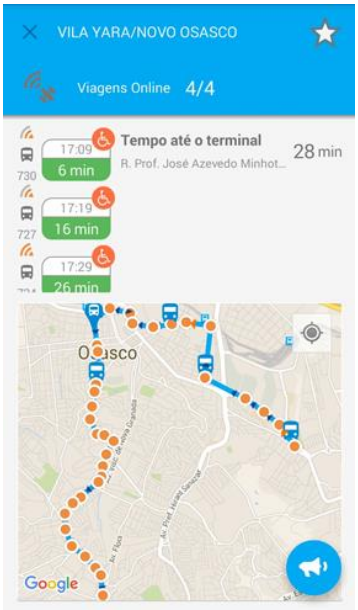
Figura 16 - Telas do Moovit.



Fonte: Divulgação/Windows Phone Store

O CittaMobi é um outro aplicativo do gênero, que mostra a posição dos ônibus em tempo real. O aplicativo apresenta um mapa com as paradas de ônibus na região e as linhas que passam por elas. Possui também um sistema de busca de ônibus adaptados para cadeirantes, o que se mostra um diferencial frente aos concorrentes. Este aplicativo também apresenta uma funcionalidade chamada “Subir no ônibus”, que calcula o tempo estimado para sua chegada no ponto final.

Figura 17 - Tela do CittaMobi.



Fonte: (http://www.osascobus.com.br/)

Embora seja apenas possível estimar como funciona o algoritmo de cada um dos aplicativos, apenas o CittaMobi parece oferecer uma funcionalidade semelhante à proposta neste trabalho, que é a função “Subir no ônibus”. Entretanto, essa funcionalidade não parece ser o ponto central do aplicativo, diferentemente da solução proposta.

Um outro diferencial da solução proposta é mostrar os ônibus se locomovendo no mapa em tempo real quando um usuário estiver embarcado com o sistema ativo. Assim, os usuários que estão consultando o aplicativo terão mais confiança no sistema do que se estivessem apenas observando estimativas de horário.

CAPÍTULO 4 MATERIAIS E MÉTODOS

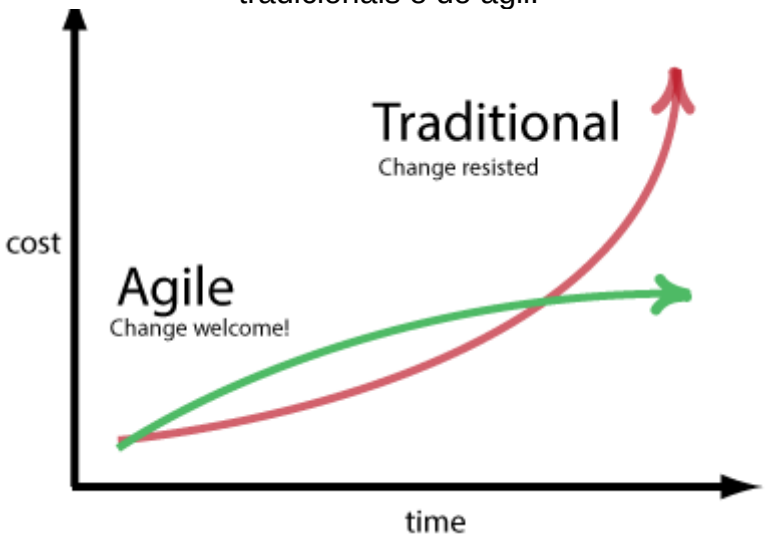
No desenvolvimento, foram utilizadas duas metodologias: *Agile Development* e *Software Engineering.* Estas metodologias serão analisadas ao longo deste capítulo.

4.1 AGILE DEVELOPMENT

Agile Development é uma metodologia que preza pelo desenvolvimento rápido e lançamento do projeto em partes. Por este motivo, escolheu-se desenvolver um *Minimum Viable Product*, descrito como MVP ao longo do trabalho. Esta decisão será discutida na seção 4.3. As metodologias de desenvolvimento ágil costumam se basear em diminuição de riscos por desenvolvimento de software em curtos períodos.

Em projetos grandes, quando desenvolvidos em um único bloco, alterações posteriores em partes iniciais do código podem necessitar da reescrita de grandes porções do projeto, aumentando os custos de produção (PRESSMAN, 2010). Desenvolvendo em parcelas, o custo de mudanças é mitigado. A comparação pode ser vista na Figura 18.

Figura 18 - Comparação do custo de mudanças no código entre métodos tradicionais e do ágil.



Fonte: agilenutshell.com.

Por enfatizar o desenvolvimento rápido do projeto, o sistema é criticado por não produzir muita documentação. Este é um dos principais pontos fracos e será discutido posteriormente.

Das diversas modalidades de *agile development* existentes, a utilizada está descrita na seção a seguir.

4.1.1 FDD – *Feature Driven Development*

Feature Driven Development, ou Desenvolvimento Guiado por Funcionalidades, é um modelo de desenvolvimento de códigos focado em quebrar o código em seções simples, as *features*. No contexto de FDD, uma *feature* pode ser definida por “uma função que pode ser implementada em duas semanas ou menos” (PRESSMAN, 2010).

Como a ideia de desenvolvimento ágil preza a publicação incremental de códigos, utilizando esta metodologia é possível que uma equipe publique partes funcionais do código de duas em duas semanas. Ainda, por separar o código em porções menores, é mais fácil de inspecionar por erros de projeto.

Features são organizados do seguinte modo: Primeiro, descreve-se o projeto em frases com o formato: <Ação><Resultado><Objeto>. Essas ações são então agrupadas em seções de funcionalidades semelhantes, que podem ser passadas para os programadores.

Neste projeto, as *features* foram definidas e agrupadas do seguinte modo:

- MINIMUM VIABLE PROJECT:
    - Set 1 – Etapa de seleção
        - Selecionar a linha de ônibus
        - Pesquisar as paradas da linha
        - Mostrar paradas da linha
        - Pesquisar horários da parada selecionada
        - Mostrar horários da parada
    - Set 2 – Controle de movimentação do usuário
        - Verificar se usuário esta embarcado
        - Confirmar embarque com pergunta
        - Atualizar servidor com posição atual

        - Verificar desembarque do usuário

- EXTENSÃO DO PROJETO
    - Set 3 – Cadastro Orgânico
        - Perguntar se usuário está na parada
        - Perguntar se linha passa nesta parada
        - Atualizar servidor com informações
        - Cadastrar nova linha

4.2 ENGENHARIA DE SOFTWARE TRADICIONAL

Engenharia de software se refere ao desenvolvimento sistemático de *software*. (ABRAN, MOORE, *et al.*, 2004). Diferentemente de *Agile Development,* a engenharia de *software* preza por documentação do processo e é considerado um sistema tradicional.

É um sistema de desenvolvimento de *software* muito mais rigoroso e prevê o planejamento prévio do código inteiro.

4.2.1 CRC *Cards* – Cartões de Classe-responsabilidade-colaboração

A primeira atividade realizada foram os CRC *Cards*. Os CRC *Cards* são cartões, geralmente desenvolvido por um grupo de pessoas, que representam as classes a serem utilizadas na programação do projeto. Nos cartões, os desenvolvedores nomeiam as classes, suas responsabilidades e as relações entre si. Assim, é possível ter uma visão geral do projeto finalizado.

Eles foram propostos por Ward Cunningham e Kent Beck como uma ferramenta de ensino de programação (BECK e CUNNINGHAM, 1989), mas também são utilizados para o desenvolvimento de projetos complicados por desenvolvedores experientes. Como são portáveis eles podem ser rearranjados na mesa com facilidade, enquanto questões de *design* são discutidas pelos programadores. Seu design compacto também força os desenvolvedores a focarem nas questões centrais das classes a serem desenvolvidas sem entrar muito em especificidades, já que este não é o objetivo da atividade.

O primeiro passo para desenvolver os cartões foi escrever, detalhadamente, o funcionamento do código. Neste passo, a linguagem utilizada não necessita ser necessariamente precisa, apenas passar a ideia do projeto. Com o texto explicativo pronto, reuniu-se uma equipe para realizar a atividade, em um estilo de *brainstorm*. Com a equipe reunida, o próximo passo foi destacar os verbos e substantivos do texto. Os substantivos iniciaram a dar forma aos nomes das classes, e os verbos às relações entre elas. Então, os cartões começaram a ser escritos. As classes foram nomeadas e as relações entre si delineadas. Na Figura 19 pode-se ver um dos cartões desenvolvidos pela equipe. No total, foram desenvolvidos 14 cartões nesta atividade.

Figura 19 - Exemplo de CRC *Card* desenvolvido.

Classe – Linha	
Responsabilidades • Registrar posição de paradas • Registrar nome da linha • Registrar horários	**Colaboração** • App • Base de dados • Paradas

Fonte: Autor.

Com todos os cartões prontos, um refinamento das classes foi feito, preparando-as para a próxima atividade. Dos 14 cartões desenvolvidos, apenas 10 seguiram sendo utilizados. Esta etapa proporcionou uma melhor compreensão do rumo a ser tomado em direção ao projeto finalizado e criou a base das classes que compõem o aplicativo. Estes cartões podem ser vistos no

Apêndice A – Cartões CRC.

4.2.2 UML 2.0 - Diagrama de classes

O diagrama de classes é um diagrama estrutural. É um dos diagramas centrais do UML 2.0, servindo de base para outros diagramas, como o diagrama de sequência, mostrado na seção 4.2.4. Um diagrama estrutural é a representação do esqueleto do sistema, descrevendo as partes relativamente estáveis.

O diagrama de classes foi resultado da primeira atividade, os CRC *Cards*. Com o esboço de todas as classes prontas, as classes começaram a ser desenvolvidas como no exemplo da Figura 20.

Enquanto no CRC o objetivo era manter a linguagem menos técnica possível, no diagrama de classes o objetivo é começar a pensar mais em termos de programação. Logo, atributos começaram a ser denominados e recebem seus tipos de variáveis e métodos começam a ser descritos, incluindo seu tipo de variável de retorno.

As classes foram, então, relacionadas entre si. O diagrama completo mostra, por exemplo, a quantidade de instâncias de uma classe que podem existir em um determinado momento. Assim, podemos imaginar que poderia existir somente um elemento do tipo “Main”, mas diversos objetos do tipo “Posicao” podem ser criados.

Figura 20 - Exemplo de classe do diagrama de classes.

Classe: Posicao
Atributos: **-** Latitude: Float - Longitude: Float
Métodos: **+**getPosition(): Posicao

Fonte: Autor.

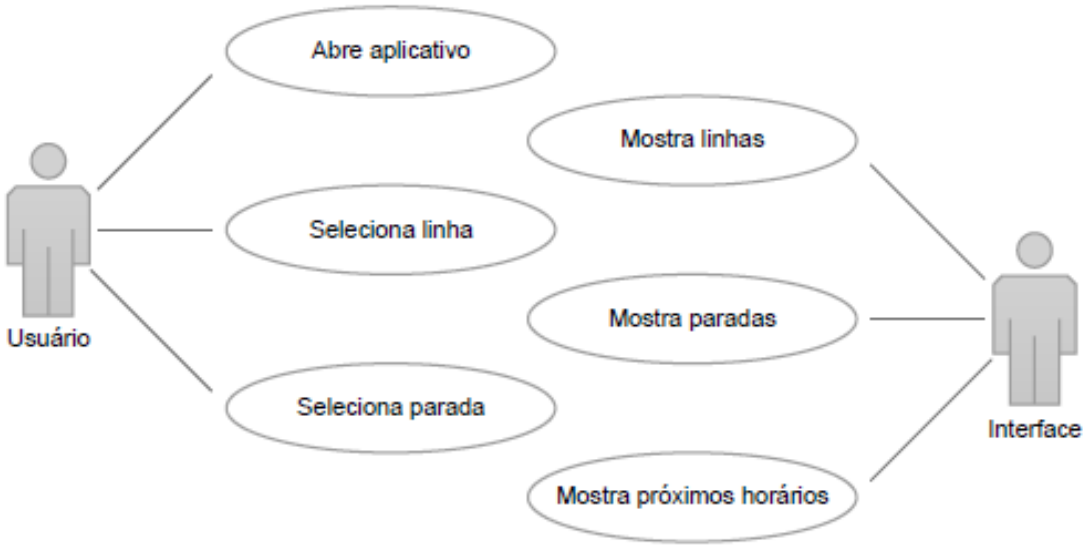
No fim desta etapa, o esqueleto do projeto estava pronto. Agora foi possível analisar como as classes se comunicam e como, além de ser possível perceber as

responsabilidades e armazenamentos de cada uma. O diagrama completo pode ser visto no Apêndice B.

4.2.3 UML 2.0 – Caso de uso

Caso de uso é um diagrama UML que faz parte dos diagramas comportamentais. Ele mostra a relação entre atores e casos de uso dentro de um sistema (AMBLER, 2005). Um exemplo de caso de uso pode ser visto na Figura 21. Os demais casos de uso podem ser vistos no Apêndice C

Figura 21 - Exemplo de caso de uso.



Fonte: Autor

4.2.4 UML 2.0 – Diagrama de sequência

Diagramas de sequência enfatizam a ordem temporal das mensagens. Neste diagrama, as classes são postas lado a lado para mostrar quando objetos são criados e destruídos em uma “linha de vida”. As comunicações entre os objetos representadas por setas.

De acordo com Ambler (2005), os diagramas de sequência são uteis para:

- Explorar o design, pois o diagrama dá visualização de invocação das classes;
- Dar percepção de quais classes serão complexas;
- Perceber gargalos no programa em função de quanto tempo as operações podem demorar.

Neste projeto, o diagrama de sequência foi divido em duas partes, para facilitar a compreensão do sistema. A primeira parte representa o corpo do aplicativo. A segunda, as extensões do aplicativo.

Os diagramas podem ser vistos nos Apêndices D e E.

4.3 CONEXÃO ENTRE AGILE DEVELOPMENT E ENGENHARIA DE SOFTWARE

Agile development e engenharia de software são metodologias opostas. Então como elas se relacionam? Segundo Pressman (2010, p. 70), em tradução livre:

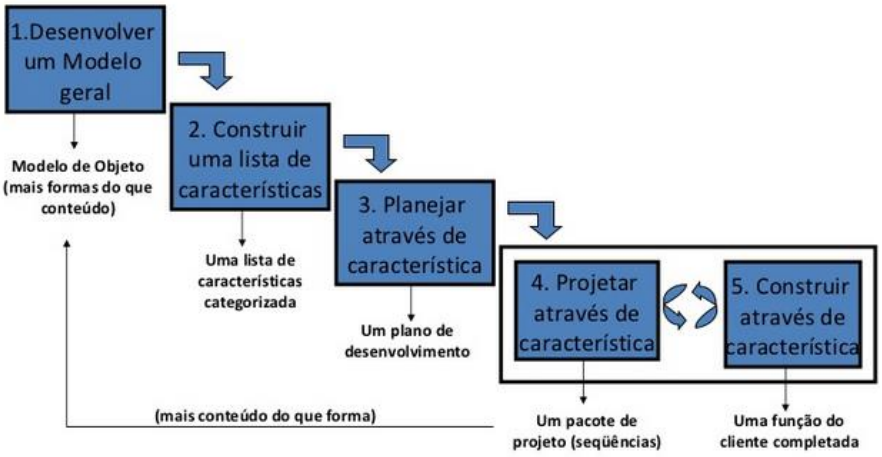
> ‘Metodologistas tradicionais são uns atolados que prefeririam produzir documentação impecável do que um sistema funcional que atende as necessidades de mercado’. Por outro lado, ele adiciona: ‘Metodologistas leves, são um bando de hackers glorificados que vão ter uma surpresa quando tentarem elevar seus brinquedos no mundo de negócios de *software*.’

Como pode-se perceber, embora mutuamente excludentes, as duas metodologias possuem suas falhas. Logo, optou-se por desenvolver o projeto utilizando o melhor de suas características: o desenvolvimento simples de *ágil development* com a documentação clara da engenharia de software.

Utilizando o esquemático característico do FDD, é possível verificar que na verdade, os dois sistemas acabam se sobrepondo em etapas e adicionando ainda mais à documentação final.

Tanto em *Agile Development* quanto em Engenharia de Software, a etapa inicial de desenvolver um modelo genérico se sobrepõem e se complementam. Entretanto, as suas diferenças ficam visíveis no momento de programação. O desenvolvimento ágil permite visualizar bem quais etapas devem ser programadas em sequência, enquanto a engenharia permite visualizar a correlação entre as etapas programadas.

Figura 22 - Esquemático de *Feature Driven Development.*



Fonte: (RIBEIRO,2016).

Na Figura 22 pode-se observar a interação entre as duas metodologias. A primeira etapa, de desenvolvimento do modelo é feita em ambas. A segunda da separação em listas e sets é feita apenas na ágil, mas não desenvolve a metodologia de programação. A seção de *"design by feature"* fica por conta da engenharia de software, no caso deste projeto. É nesta seção em que as classes são desenvolvidas, exatamente como foram utilizando as técnicas de UML.

4.4 MVP – MINIMUM VIABLE PRODUCT

O *Minimum Viable Product* deste projeto foi desenvolvido durante sua participação no Desafio I9 de empreendedorismo, no qual recebeu a premiação de primeiro lugar. Um MVP, como o nome sugere, é o produto mínimo viável que poderia ser lançado no mercado para atingir a expectativa dos consumidores. Neste caso, o MVP foi um sistema colaborativo de compartilhamento de localização utilizando o aplicativo para celulares *WhatsApp*.

Este MVP foi desenvolvido para testar a opinião dos possíveis usuários do aplicativo finalizado sobre sua disponibilidade quanto ao compartilhamento de localização. É importante saber o que os usuários esperam de retorno do aplicativo para que estejam dispostos a compartilhar a sua localização pessoal.

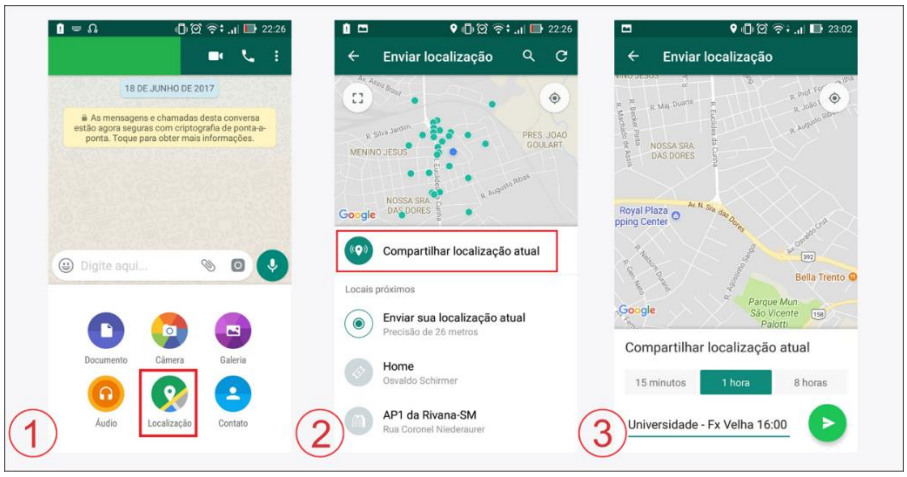
Para isso, um grupo de WhatsApp foi montado com aproximadamente 56 pessoas. A elas, foi pedido que utilizassem o sistema de compartilhamento de localização em tempo real dentro do próprio aplicativo e que dissessem qual ônibus

eles estavam representando. Na Figura 23 pode-se observar o tutorial repassado aos membros do grupo com as orientações de como compartilhar a localização

Este grupo foi mantido em funcionamento por aproximadamente uma semana. Nos dois primeiros dias, apenas três pessoas compartilharam suas localizações. Todas elas tinham algum nível de envolvimento com o projeto.

O sistema apresentava muitas falhas, como falta de anonimato, número muito pequeno de pessoas para criar um mapa confiável de ônibus em tempo real, falta de clareza nas informações do chat e a necessidade de pró-atividade do membro de abrir o aplicativo por puro altruísmo, sem retorno nenhum, e compartilhar sua localização.

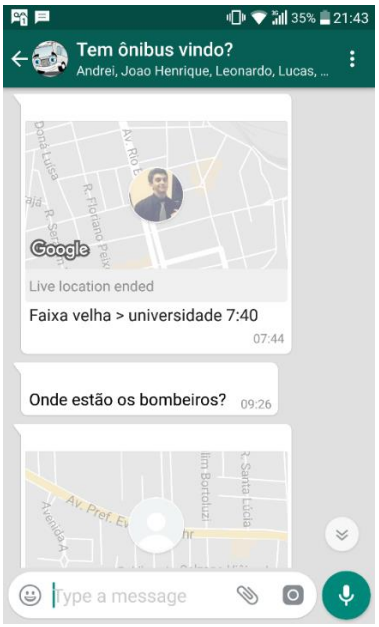
Figura 23 - Instruções dadas aos membros do grupo.



Fonte: Autor.

Mesmo com todos estes empecilhos, ainda era necessário validar a disposição das pessoas a compartilharem sua localização. A pequena adesão dos membros ao sistema nos primeiros dias foi preocupante. Em uma reunião de grupo, foi sugerida um bônus a pessoa que compartilhasse a localização mais vezes durante a semana de testes. Isso ajudaria a incentivar os usuários a compartilhar, apesar de todas as fraquezas do MVP. A pessoa que compartilhasse a localização em tempo real mais vezes até o fim da semana, receberia quatro litros de cerveja. A Figura 24 é um exemplo do grupo após o enunciado da premiação.

Figura 24 - Tela do MVP.



Fonte: Autor.

Nos dias seguintes ao anúncio, houve mais compartilhamentos de localização, inclusive de pessoas que não tinham envolvimento com o projeto. Esta proposta indicou que as pessoas estariam dispostas a compartilhar sua localização, mesmo na plataforma menos intuitiva, desde que recebessem benefícios.

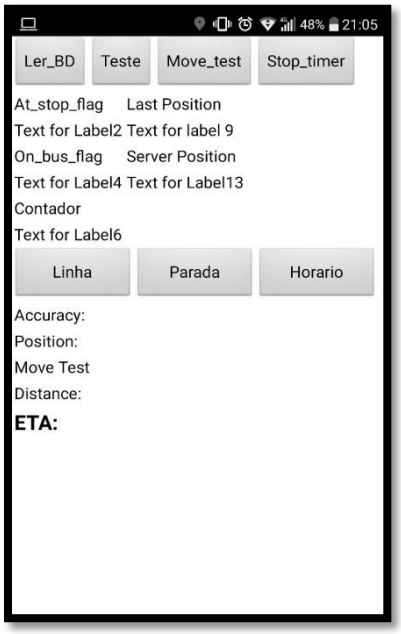
Este resultado indicou que os usuários do aplicativo estariam dispostos a utilizar o sistema, mas necessitariam de um incentivo extra para contribuir para ele. Por ser um sistema colaborativo, é necessário que os usuários do sistema disponibilizem sua localização. Assim, desenvolveu-se a ideia de gerar cupons de descontos em lojas e serviços da região para os usuários que contribuírem para o sistema.

CAPÍTULO 5 DESENVOLVIMENTO DA PRIMEIRA VERSÃO

Este capítulo descreve a versão anterior do aplicativo, desenvolvido na disciplina de Projeto Integrador. A proposta deste capítulo é comparar quais as características da versão anterior que foram desenvolvidas e se mantiveram na versão atual e quais funcionalidades foram descartadas ou melhoradas. O aplicativo apresentado neste capítulo foi desenvolvido antes do estudo realizado para este projeto, e portanto as técnicas de engenharia de *software* apresentadas não foram aplicadas.

Nesta versão, programada no MIT App Inventor, ao abrir o aplicativo, o usuário visualiza três botões: selecionar linha, selecionar parada e selecionar horário. Isto pode ser visto na Figura 25.

Figura 25 - Tela de testes desenvolvida.

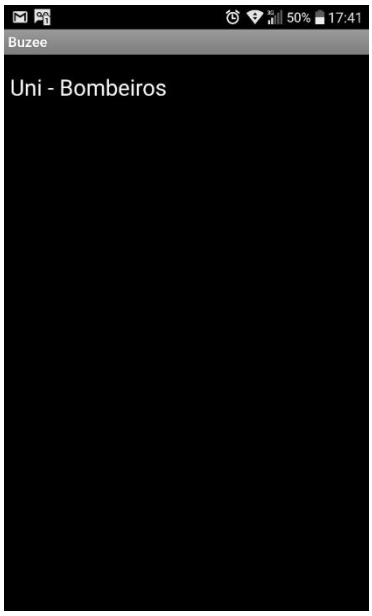


Fonte: Autor

Ao clicar em selecionar linha, opções como “Bombeiros - Faixa velha” e “Universidade - Faixa nova” são apresentadas, como mostrado na Figura 26. Clicando em uma das seleções, o menu de paradas é atualizado automaticamente em função das paradas contempladas pelo trajeto, como pode ser visto na Figura 27. As paradas foram apenas numeradas sequencialmente a partir do início da linha nesta versão do

aplicativo. Neste momento o usuário requisita também os dados do servidor sobre as posições das paradas e horários da linha, para que sejam gravados na memória interna do celular.

Figura 26 - Tela de seleção de linha.



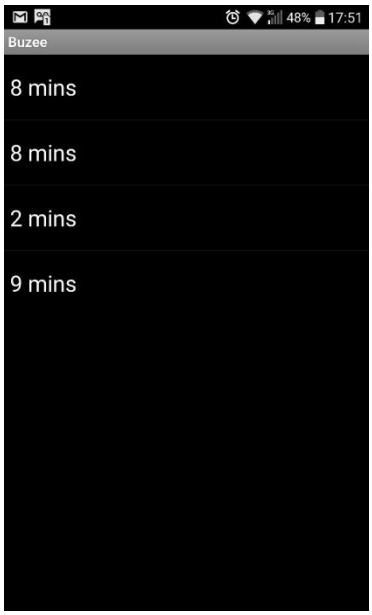
Fonte:Autor.

Figura 27 - Tela de lista de paradas.

Fonte: Autor.

Então, ao selecionar sua parada, as seleções do menu de horários são atualizados em função dos ônibus já em circulação e do tempo estimado até sua chegada na parada, representado na Figura 28. Nesta versão, é necessário que o usuário selecione esses parâmetros para confirmar qual ônibus será tomado para que o código possa processar os dados.

Figura 28 - Tela de tempo estimado de chegada.



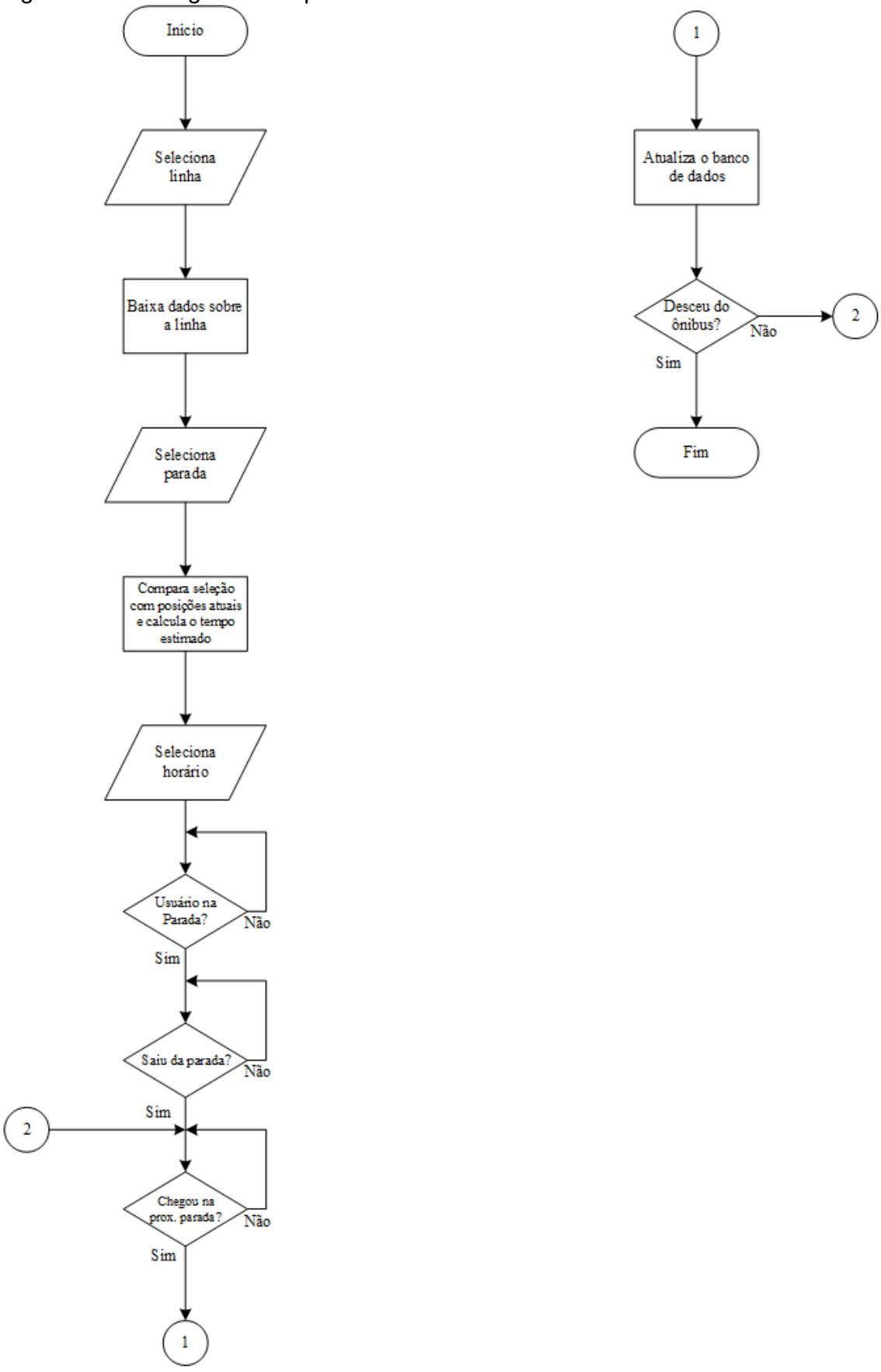
Fonte: Autor.

Para confirmar que o usuário esteja embarcado no ônibus, o código baseia-se em *geofencing*. Ao selecionar a parada, o código testa a posição do usuário a cada segundo. Assim que o usuário se aproximar de no mínimo 30 metros da parada, um contador de 10 segundos se inicia. Isso é uma medida para evitar que erros de GPS acusem que o usuário esteja na parada. Ao passar 10 segundos dentro do perímetro, uma *flag* é ativada para avisar que o usuário se encontra na parada. Assim, quando o usuário sair de um raio de 50 metros da parada, significa que o usuário embarcou no ônibus desejado.

A partir de então, a posição do usuário é lida pelo aplicativo a cada segundo. Quando o usuário se encontrar dentro de um raio de 50 metros de qualquer parada da linha, o aplicativo envia a posição do ônibus para o banco de dados. Utilizando este método, as paradas funcionam como *checkpoints*, nos quais os dados são enviados e a última posição do ônibus é gravada. Utilizando este método é possível poupar o

plano de dados dos usuários, pois a atualização constante da posição no servidor não é necessária. Por outro lado, o aplicativo apenas fornece uma estimativa de horário de chegada, visto que apenas o momento em que o ônibus passou na última parada é gravado. Entretanto, utilizando dados sobre o trânsito providos pelo Google, é possível prever o tempo estimado com maior precisão, levando em conta engarrafamentos, por exemplo. Sendo apenas uma estimativa de tempo, escolheu-se utilizar uma margem de erro de três minutos para a chegada do ônibus. Sendo assim, quando o tempo estimado de chegada estiver esgotado, por três minutos será mostrada uma mensagem dizendo “chegando”. Caso o atraso seja maior, será declarado o atraso do ônibus. Como o desvio de tempo é recalculado a cada parada e não é cumulativo ao longo da linha, se um ônibus se atrasar em um trecho não terá impacto no cálculo de tempo estimado para o próximo trecho. Um fluxograma do funcionamento pode ser visto na Figura 29.

Figura 29 - Fluxograma simplificado de funcionamento.



Fonte: Autor.

Esta versão do aplicativo funciona sem processamento centralizado. Cada celular é responsável pelo tratamento de seus próprios dados e atualização do banco de dados. A vantagem deste método é não necessitar de um servidor, mas isso limita o aplicativo, forçando a necessidade dos botões mencionados anteriormente.

Quanto ao banco de dados, escolheu-se utilizar cada horário como âncora para a posição de cada ônibus. Assim, cada ônibus que sai do terminal inicial possui o mesmo índice do seu horário, sendo mais fácil a atualização de sua posição. Isso pode ser visto na Figura 30.

Figura 30 - Tabela do banco de dados mostrando os horários de saída e posições.

bus_id	Uni_Bomb_pos	Bus_time
1	1	01:00:00
2	9	02:00:00
3	9	03:00:00
4	9	04:00:00

Fonte: Autor.

O banco de dados também possui uma tabela com as coordenadas geográficas de cada parada ao longo da linha, como pode ser visto na Figura 31

Figura 31 - Tabela do banco de dados com posições das paradas.

ID ▲ 1	Uni_Bomb
1	-29.719122,-53.714646
2	-29.713809, -53.715897
3	-29.712099,-53.716172
4	-29.709404, -53.716428
5	-29.707848, -53.716180
6	-29.703444, -53.715464
7	-29.701912, -53.716775
8	-29.701409, -53.718886
9	-29.701270, -53.721386

Fonte: Autor.

Da primeira versão para a versão atual, muito foi mudado no aplicativo. A plataforma escolhida para a primeira versão, *o App Inventor*, apesar de ser de programação simples, o que auxiliou o desenvolvimento em um primeiro momento, limitou a capacidade de desenvolvimento do *app*, especialmente na parte gráfica.

Uma das grandes mudanças entre as duas versões é o local de processamento. A primeira versão utilizava um processamento descentralizado, no qual cada celular era responsável pelo seus próprios cálculos, apenas lendo os dados do banco de dados e processando-os de acordo. Na versão atual, todo o processamento é feito pelo *Firebase*, com os celulares apenas provendo o servidor com as informações necessárias para o sistema. A vantagem do sistema descentralizado é a simplicidade do sistema, mas deixa muito a desejar em confiabilidade. O sistema centralizado, além de permitir mais controle e atualização mais simples de software, permite um nível de confiabilidade muito maior, sem dependência das características específicas de cada celular.

O sistema anterior também era muito mais dependente de *input* do usuário. O usuário necessitava executar diversos cliques de maneira não intuitiva para poder obter a informação desejada. O sistema atual, com a interface gráfica melhorada não necessita de tanta interação do usuário, mostrando a informação com apenas um clique na parada desejada.

Uma característica importante que se manteve do sistema foi a utilização de *geofences* nas paradas de ônibus. As *geofences* continuam sendo utilizadas para demarcar as paradas e controlar a chegada ou saída do usuário do seu alcance, mas a programação foi alterada de alguns modos. Na primeira versão, o sistema de *geofencing* foi todo programando manualmente, enquanto na versão atual uma API é utilizada para obter resultados mais precisos e facilitar a programação.

Diferentemente do sistema anterior, que criava uma *geofence* em torno de cada parada ao longo da linha e as tratava como um *checkpoint,* o sistema atual apenas a cria na primeira parada, e não utiliza as outras como controle. O sistema não trabalha mais com previsão de tempo baseado na última informação obtida e sim em tempo real. Agora, os usuários enviam constantemente sua posição para o sistema, provendo o servidor com a posição do ônibus ponto a ponto.

CAPÍTULO 6 SISTEMA DESENVOLVIDO

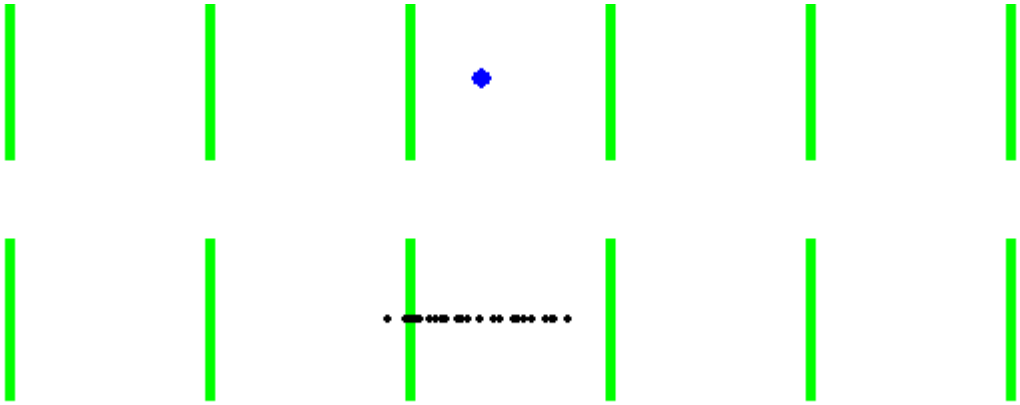
6.1 FILTRO DE PARTÍCULAS

O filtro de partículas foi desenvolvido para ser utilizado como inteligência artificial para o servidor quando o aplicativo estiver pronto. A ideia é que o servidor simule a localização de um ônibus utilizando partículas que se deslocam ao longo do trajeto, tentando prever o padrão de movimentação do ônibus. Quando um usuário embarca no ônibus com o aplicativo ligado, as partículas que estiverem mais próximas do seu sinal são partículas de “qualidade melhor”, ou seja, representam com mais fidelidade a localização do ônibus. Logo, qualidades como velocidade e aceleração destas partículas boas são clonadas, e outras partículas são criadas com estas qualidades. As partículas que estiverem mais longe do sinal, por outro lado, são destruídas, pois não representam bem o ônibus.

Tomando estas decisões de destruir ou clonar partículas, a simulação vai refinando a localização dos ônibus, pois ao longo do tempo as partículas de melhor qualidade são as que se replicam. Para a geração inicial das partículas, pode-se levar em conta os dados históricos de movimentação da linha em casos específicos, como data e hora, ou situação climática, o que poderia alterar o padrão de movimento.

A simulação desenvolvida, entretanto, ainda não possui integração com o aplicativo ou qualquer módulo externo, funcionando apenas isoladamente. Para esta simulação foi desenvolvido um modelo gráfico do sistema para facilitar a compreensão. A linha superior, como pode ser visto na Figura 32, representa um ônibus navegando ao longo de seu trajeto. Os traços verticais representam as paradas. A linha inferior representa as partículas que tentam simular a localização do ônibus.

Figura 32 - Simulação do filtro de partículas.



Fonte: Autor.

6.1.1 Modelagem

Para a movimentação das partículas, dois modelos foram desenvolvidos. Um dos modelos faria com que a partícula tivesse 50% de chance de parar em cada parada, com esta chance sendo recalculada toda vez que a partícula passasse por uma parada, independentemente de parar, ou não. Toda vez que a partícula parar em uma parada, sua velocidade e aceleração para o próximo trecho são recalculados. Este modelo visa simular que um passageiro solicitou que o ônibus parasse na parada, seja para descer ou subir. Este modelo poderia ainda se beneficiar da informação gerada pelos usuários atualizando a chance de parada para a simulação. Assim, paradas que têm uma maior frequência de abordagem teriam uma chance maior de que a partícula na simulação parasse também, enquanto paradas menos utilizadas também teriam chances menores de influenciar a movimentação da partícula.

O segundo modelo é muito semelhante ao primeiro. Entretanto, neste modo as partículas paravam em todas as paradas da linha. Este segundo modelo foi apenas utilizado para as partículas da segunda linha da interface, ou seja, as partículas de simulação. O primeiro modelo foi utilizado tanto para a simulação do ônibus quanto das partículas.

6.1.2 Resultados

Utilizando estes dois modelos, testes foram executados para verificar qual dos dois modelos melhor representava a movimentação do ônibus em simulação. Naturalmente, estes modelos estavam rodando em uma modalidade “crua”, ou seja, sem *input* externo ou simulação baseada em histórico. Nestes testes, o código foi executado 100 vezes para cada modalidade.

O primeiro teste fez a partícula ônibus e as simulatórias terem 50% de chance de parar na parada, seguindo o primeiro modelo. Com um total de 5.000 amostras, o modelo teve um média de 90,9m de distância entre as partículas e o ônibus. Esta distância era calculada para cada partícula toda vez que esta se mexia. O desvio médio calculado foi de 74,8m.

O segundo teste utilizou a partícula ônibus com 50% de chance de parada e as partículas de simulação parando em todas as paradas. Com um total de 5.100 amostras, o modelo teve média de 163,9m de distância entre as partículas e o ônibus e o desvio médio foi de 66,2m.

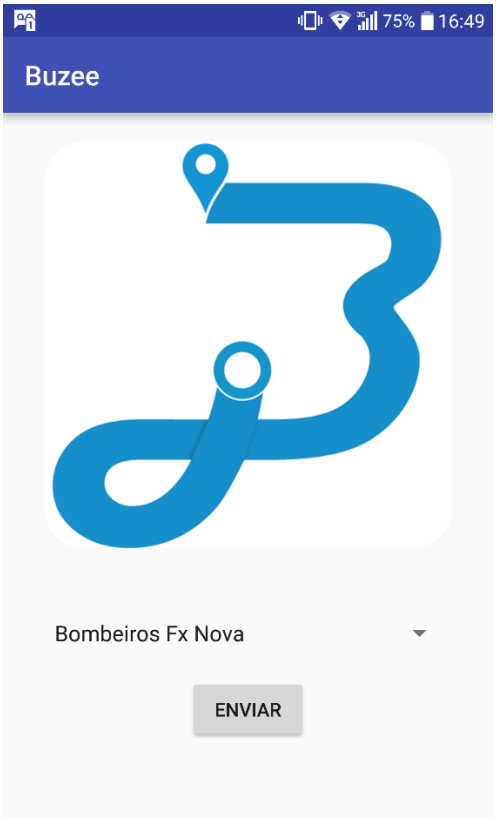
Estes resultados, embora inconclusivos em função da falta de replicação das partículas boas e da exclusão das ruins, indicou que o modelo do primeiro teste obteve resultados melhores. Os códigos utilizados para esta seção estão disponíveis no Apêndice F.

6.2 APLICATIVO ATUAL

Esta seção apresentará o desenvolvimento da programação do aplicativo e as funcionalidades atuais. O aplicativo continua em desenvolvimento e algumas das funcionalidades apresentadas já possuem substituição em andamento. Apesar disso, as funcionalidades apresentadas serão as atuais e que estão sendo utilizadas nesta versão do aplicativo. Como é o padrão em desenvolvimento *Android*, os aplicativos são divididos em atividades. Atividades costumam ser telas diferentes com as quais o usuário interage para atingir seus objetivos. No caso deste projeto, foram desenvolvidas duas atividades: A inicial, na qual o usuário seleciona a linha que deseja e a atividade principal, na qual o usuário visualiza o mapa com as paradas e ônibus ativos no sistema.

No momento em que o usuário abre o aplicativo e visualiza a primeira atividade, como pode ser visto na Figura 33, o aplicativo automaticamente busca no banco de dados as linhas cadastradas, que neste caso são apenas as linhas de teste da cidade de Santa Maria. O usuário deve, então, escolher sobre qual linha ele deseja obter informações. Ao clicar no botão enviar, a segunda atividade é iniciada e o aplicativo mostra as paradas da linha selecionada, que neste caso não estão gravadas no banco de dados, mas inseridas na programação, apenas como situação temporária. Uma das mudanças em curso é a utilização de um *web service* que buscaria as informações das linhas e paradas na internet, para que não haja necessidade de cadastramento manual de cada uma das linhas das cidades. Um fluxograma detalhando esta etapa está disponível na Figura 34.

Figura 33 - Tela inicial do aplicativo.

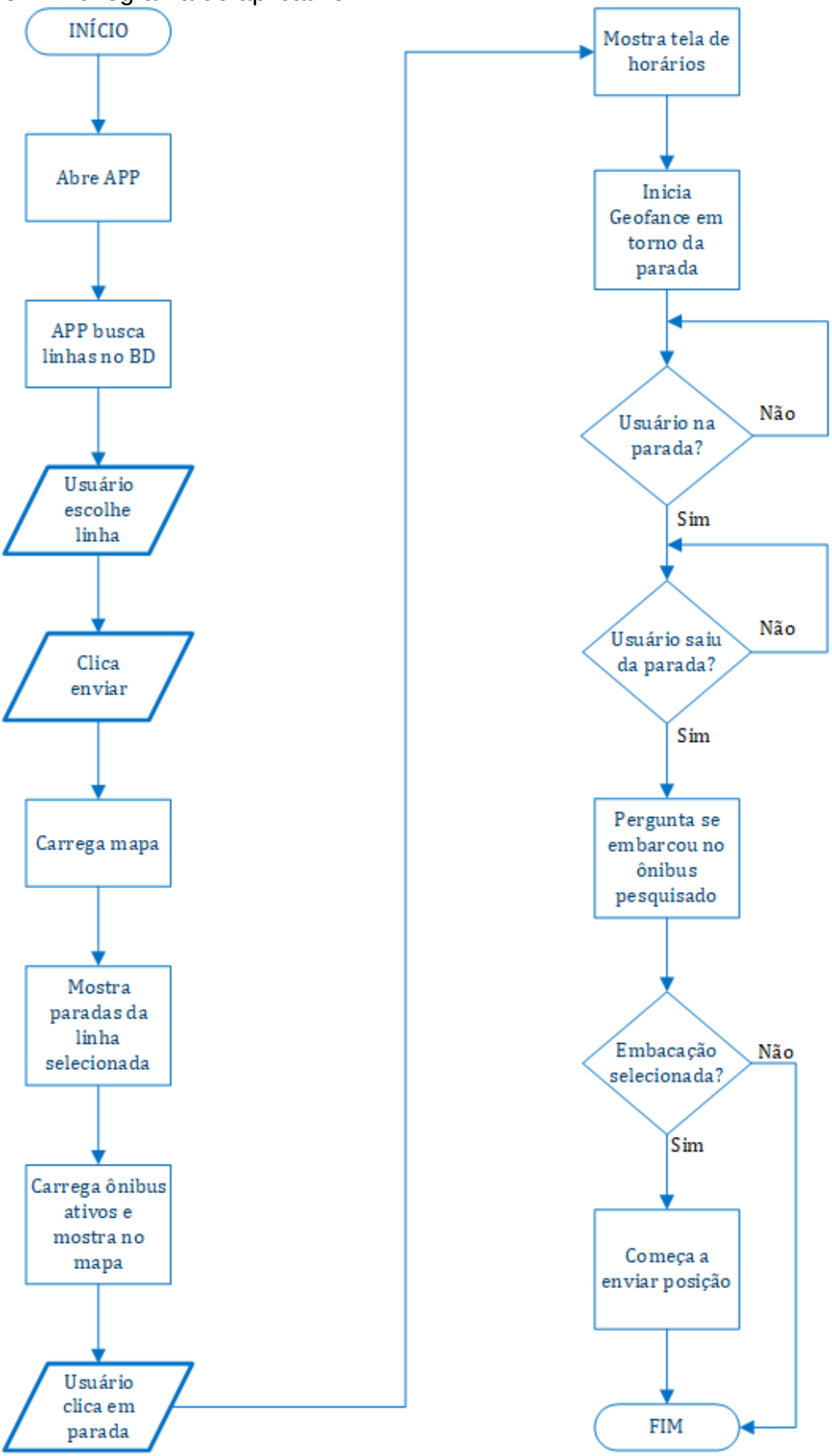


Fonte: Autor.

Nesta segunda atividade, que pode ser vista na Figura 35, na verdade é a atividade central do aplicativo, o aplicativo busca no banco de dados os ônibus que estão ativamente em circulação e os representa no mapa com um ícone. O usuário

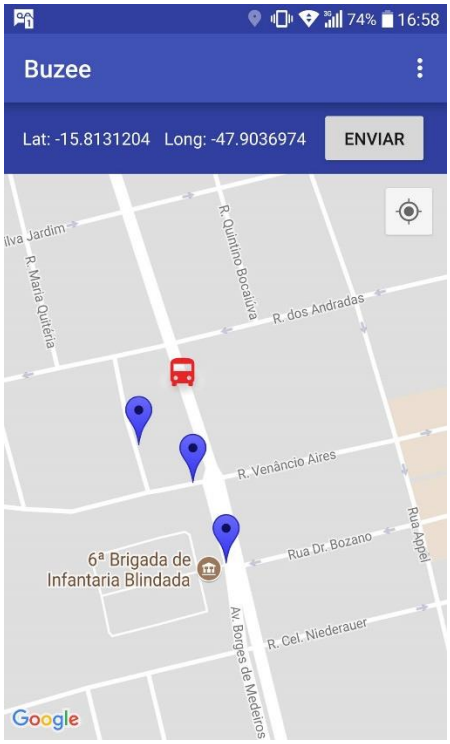
deve selecionar a parada desejada para que possa ver as estimativas de tempo para a chegada dos ônibus. Quando o usuário clica na parada, a API de *geofence* cria uma cerca virtual ao redor da parada selecionada e começa a acompanhar a localização do celular do usuário. Isto é feito para que possamos controlar se o usuário chegou ou saiu da parada selecionada. Se o usuário estiver dentro do raio da parada e logo depois se deslocar para longe dela, muito provavelmente ele tenha embarcado no ônibus desejado.

Figura 34 - Fluxograma do aplicativo.



Fonte: Autor.

Figura 35 - Tela do aplicativo em funcionamento.



Fonte: Autor.

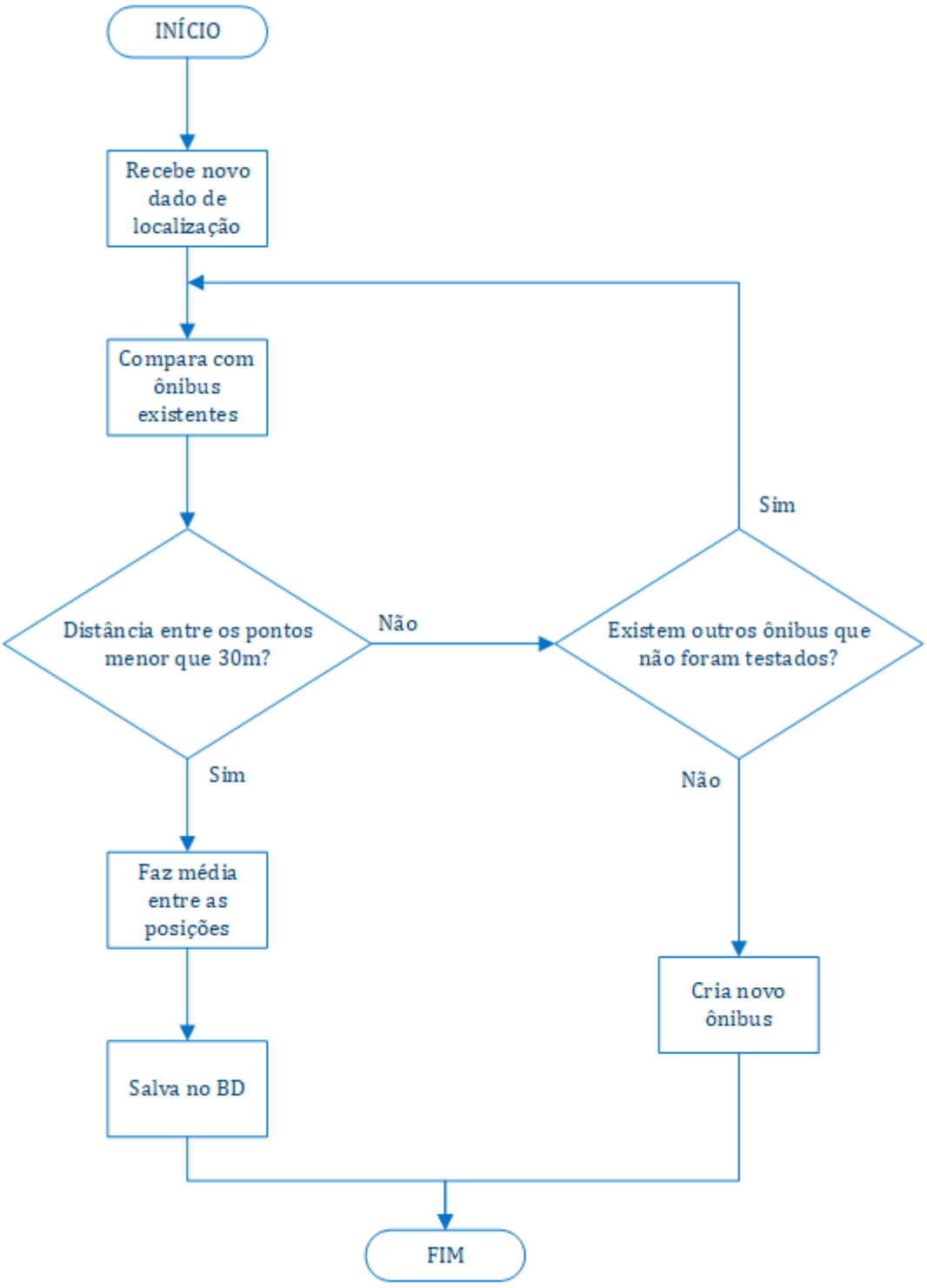
Mas e se o usuário tiver tomado outra linha? E se tiver decidido ir a pé? Para resolver esta questão, o sistema envia uma notificação ao usuário perguntando se ele embarcou no ônibus da linha que foi pesquisada no aplicativo. Para esta notificação, existem dois casos diferentes.

No primeiro caso, se o usuário responder que não está na linha consultada, o aplicativo é terminado, afinal não se sabe qual modalidade de transporte o usuário selecionou. Além disso, contribuir para o sistema com sua localização em tempo real é opcional, então o usuário pode simplesmente não querer compartilhar por questões de privacidade.

No segundo caso, se o usuário responder que sim, ou seja, que embarcou na linha que consultou no aplicativo, seus dados são utilizados para alimentar o sistema. Esta resposta ativa uma função presente no servidor que calcula a distância entre todos os ônibus existentes no banco de dados e a localização do usuário utilizando o método de Haversine. Caso o usuário esteja a uma distância inferior a 30 metros de um ônibus existente, ele começa a participar do deslocamento daquele ônibus. Isso significa que é feita uma média da localização de todos usuários embarcados no

mesmo ônibus, e esta localização é salva no banco de dados. Este sistema está explicado no fluxograma da Figura 36.

Figura 36 - Fluxograma de funcionamento do servidor.



Fonte: Autor.

O aplicativo é divido em duas atividades e três serviços independentes. Como mencionado anteriormente, a primeira atividade, que é responsável por buscar as linhas e a seleção do usuário. Em seguida, a segunda atividade é iniciada, mostrando o mapa e esperando que o usuário selecione uma parada.

Ao clicar em uma parada, o usuário inicia o serviço de *geofence* que acompanha a localização do usuário e se ele atravessou as cercas virtuais. Quando o evento de saída de uma *geofence* é percebido, um outro serviço é iniciado, o de enviar a notificação para o usuário para confirmar se está embarcado no ônibus correto.

Por fim, em função da resposta do usuário à notificação, um terceiro serviço pode ser iniciado. Caso o usuário responda que está embarcado no ônibus selecionado, sua posição é monitorada e enviada para o servidor a cada 5 segundos.

Quando o servidor recebe os dados de posição, estes dados são comparados com os usuários já embarcados na mesma linha e as médias de posição são feitas de acordo.

CAPÍTULO 7 MODELO DE NEGÓCIOS

Como este projeto envolve o desenvolvimento de uma *startup*, um modelo de negócios foi sido desenvolvido. Como comentado anteriormente, os aplicativos de mobilidade urbana costumam ser focados em torno de transporte privado. Os aplicativos de transporte coletivo existem, mas não conquistaram espaço no mercado. A ideia é tentar explorar esta brecha e captar a grande massa de usuários de transporte coletivo para o aplicativo.

Segundo dados de 2017 fornecidos pela URBS, a empresa responsável pelo sistema de transporte público de Curitiba – PR, 1.511.743 passageiros passam por suas linhas diariamente. Ainda, de acordo com Françoso (2016), apenas 16% dos usuários de aplicativos utilizam ou tem conhecimento de aplicativos de transporte coletivo, como pode ser visto na Figura 6 - Comparação de usuários por aplicativo. Assim, aplicando esta estatística nos usuários do transporte coletivo de Curitiba, apenas 241.879 usuários teriam conhecimento ou utilizariam algum aplicativo desta área.

É possível relacionar também a faixa etária dos usuários de transporte coletivo e dos usuários de aplicativos. Um estudo feito pela Associação Brasileira de *online to off-line* (O2O), sobre 2.500 voluntários, mostrou que cerca de 70% dos usuários de aplicativos de O2O, como táxis ou *delivery* de comida, tem menos de 34 anos. Em dados de (2013), uma pesquisa feita pelo Metrô da Grande São Paulo, concluiu que os passageiros do transporte coletivo têm entre 18 e 34 anos.

7.1 ESTRATÉGIA DE CONQUISTA DE MERCADO

Levando estas informações em consideração, é possível prever que o mercado é passível de ser explorado, apenas a barreira de comunicação com os usuários ainda não foi quebrada. Por este motivo, têm-se pensado em investir pesadamente em *marketing* como meio de relacionamento com os usuários.

Como a instalação do sistema é gratuita, é possível negociar com as concessionárias de transporte público a colocação de adesivos e cartazes nos ônibus, para incentivar o maior número possível de usuários a aderir ao aplicativo.

Marketing digital também é de extrema importância, utilizando meios como *Facebook,* para atingir uma grande massa de usuários. Assim como no *design* do aplicativo, é importante levar fatores como simplicidade e objetividade nas publicações a serem feitas, para atrair a visão dos clientes sem necessidade de leituras muito extensas, por exemplo.

É de extrema importância que o sistema apresente informações confiáveis, pois, como mostrado na Figura 12, um dos grandes motivos da não utilização de aplicativos do gênero é a falta de confiança nos dados apresentados. Portanto, é imprescindível conquistar o público com horários precisos.

Como se trata de um sistema colaborativo, é importante que os usuários estejam dispostos a compartilhar sua localização. Para isso, planeja-se utilizar um sistema de cupons, com o qual os usuários que contribuem para o sistema são recompensados com cupons de descontos, o que também incentivaria os usuários a se manterem fiéis ao aplicativo.

Na Figura 37 - Matriz FOFA. pode-se ver a matriz de Forças, Fraquezas, Oportunidades e Ameaças (FOFA). Esta matriz detalha as condições de uma empresa, nas quais ela deve se apoiar, como as forças e oportunidades, e as quais ela deve melhorar ou se precaver. Isto ajuda a formar um caminho a ser seguido e analisar os pontos fortes e fracos da empresa.

Figura 37 - Matriz FOFA.

Forças	**Fraquezas**
• Metodologia inovadora • Facilidade de mudanças • Facilidade de expansão	• Dependência de usuários • Falta de experiência da equipe neste mercado
Oportunidades	**Ameaças**
• Mercado em crescimento • Mercado pouco explorado • Aplicável globalmente	• Concorrente dominante no mercado • Sistemas com rastreadores • Falta de cooperação de concessionárias

Fonte: Autor.

7.2 ANÁLISE DE CENÁRIOS

Geralmente a análise de cenários é feita sobre tempos de produção, custos de produção e venda e outros tópicos relativos ao funcionamento econômico de uma empresa. Entretanto, como é o caso do Buzee, a “produção” está diretamente ligada à quantidade de usuários ativos e a sua frequência de uso. Assim, a análise não será feita sobre os custos de operação, e sim sobre a capacidade de expandir e conquistar o mercado.

7.2.1 Cenário otimista

No cenário otimista, o Buzee consegue o apoio da concessionária para sua expansão, facilitando a propaganda do aplicativo através do meio, como cartazes nos ônibus, por exemplo.

Com o apoio da concessionária, a expansão é rápida e uma grande massa de usuários é atingida logo após a instalação do sistema na cidade. Com mais usuários o usando, o sistema se torna mais efetivo e mais atraente para novos usuários, conquistando o interesse do público.

Neste cenário, os usuários também se mostram engajados e interessados no sistema, o utilizando com frequência em seus trajetos de ônibus, talvez com interesse em fazer o sistema funcionar, talvez com interesse nas recompensas.

7.2.2 Cenário realista

No cenário realista, o Buzee não consegue o apoio da concessionária, pois esta não recebe vantagem direta da existência do aplicativo e por isto não tem interesse em contribuir para seu crescimento.

Com isto, o crescimento é mais gradual e depende de outros meios de comunicação, como propagandas nas redes sociais, notícias nos jornais locais e comentários entre amigos. Isso faz com que os momentos iniciais após a instalação do sistema tenham poucos usuários ativos, se tornando menos atrativo para novos usuários. Esta situação deve ser compensada pelo sistema de recompensas, na esperança de que os usuários tenham um interesse extra de fazer o sistema funcionar.

Neste cenário, os usuários se mostram relativamente engajados, utilizando o aplicativo algumas vezes na semana durante seus trajetos.

7.2.3 Cenário pessimista

No cenário pessimista, o Buzee também não consegue o apoio da concessionária, atrasando o crescimento inicial, como no cenário realista.

O crescimento se dá da mesma forma, sendo gradual e dependente dos meios tradicionais e da conversa entre amigos.

Entretanto, neste cenário os usuários se mostram relutantes em utilizar o aplicativo, talvez por receio de privacidade, por exemplo. Neste caso, o aplicativo funcionará basicamente com simulações das posições dos ônibus, baseando-se nas informações históricas dependentes de usuários esporádicos.

No caso dos cenários realistas e pessimistas, especialmente no segundo caso, será necessária uma revisão do mercado a ser atuado para perceber qual a origem do problema em particular e reinventar o modo com o qual se lida com este mercado. Soluções específicas para problemas regionais podem ser adotadas.

7.3 LUCROS

O sistema possui três fontes de renda principais: coleta, análise e venda de informações de mobilidade urbana; Venda de passagens online e pagamentos de passagem com NFC; e publicidade.

A principal fonte de renda projetada para o aplicativo é a coleta e venda de informações sobre mobilidade urbana. Estas informações são de extrema importância para os governos e órgãos de transporte para organizar e melhorar a qualidade dos serviços de transporte e do trânsito nas cidades.

Dados como o fluxo de pessoas, velocidade das vias, aceleração, intensidade de tráfego, todos poderiam ser coletados através das informações providas automaticamente pelos usuários.

Assim, os principais clientes seriam governos, órgãos de transporte e concessionárias de transporte público. Do ponto de vista das concessionárias, é possível perceber a valia das informações em função de economia de gastos, já que

paradas de ônibus que têm pouca frequência podem ser removidas do sistema ou pode-se monitorar o modo de dirigir de algum motorista que cause problemas mais frequentemente nos ônibus. É possível também analisar como mudanças no trânsito, como fechamento de ruas ou mudanças na direção das vias afetam o fluxo na região.

A outra fonte de renda planejada é a venda de passagens online. Em cidades como Santa Maria, a venda de passagens online ainda não está disponível. É necessário que os usuários se desloquem até um estabelecimento da concessionária para realizar a recarga do cartão de transporte. Com a venda online, os usuários poderiam comprar sua passagem pela internet e utilizar o celular para passar na roleta do ônibus. Para este serviço seria cobrada uma taxa de manutenção, de onde tiraríamos o lucro.

E a terceira fonte seria proveniente de publicidade. Neste caso, existem duas opções. Como o sistema trabalha com a geração de cupons para os usuários que contribuem para o aplicativo, empresas que concordassem com ceder cupons de desconto teriam direito a publicidade grátis no aplicativo. Empresas que não cederem cupons, também podem fazer publicidade, mas pagando uma taxa.

7.4 PITCH

Um *pitch* é uma apresentação rápida de um projeto que delineie bem os propósitos e valores entregados por alguma empresa. Deve ser uma apresentação sucinta que seja capaz de atrair o interesse de potenciais clientes ou investidores.

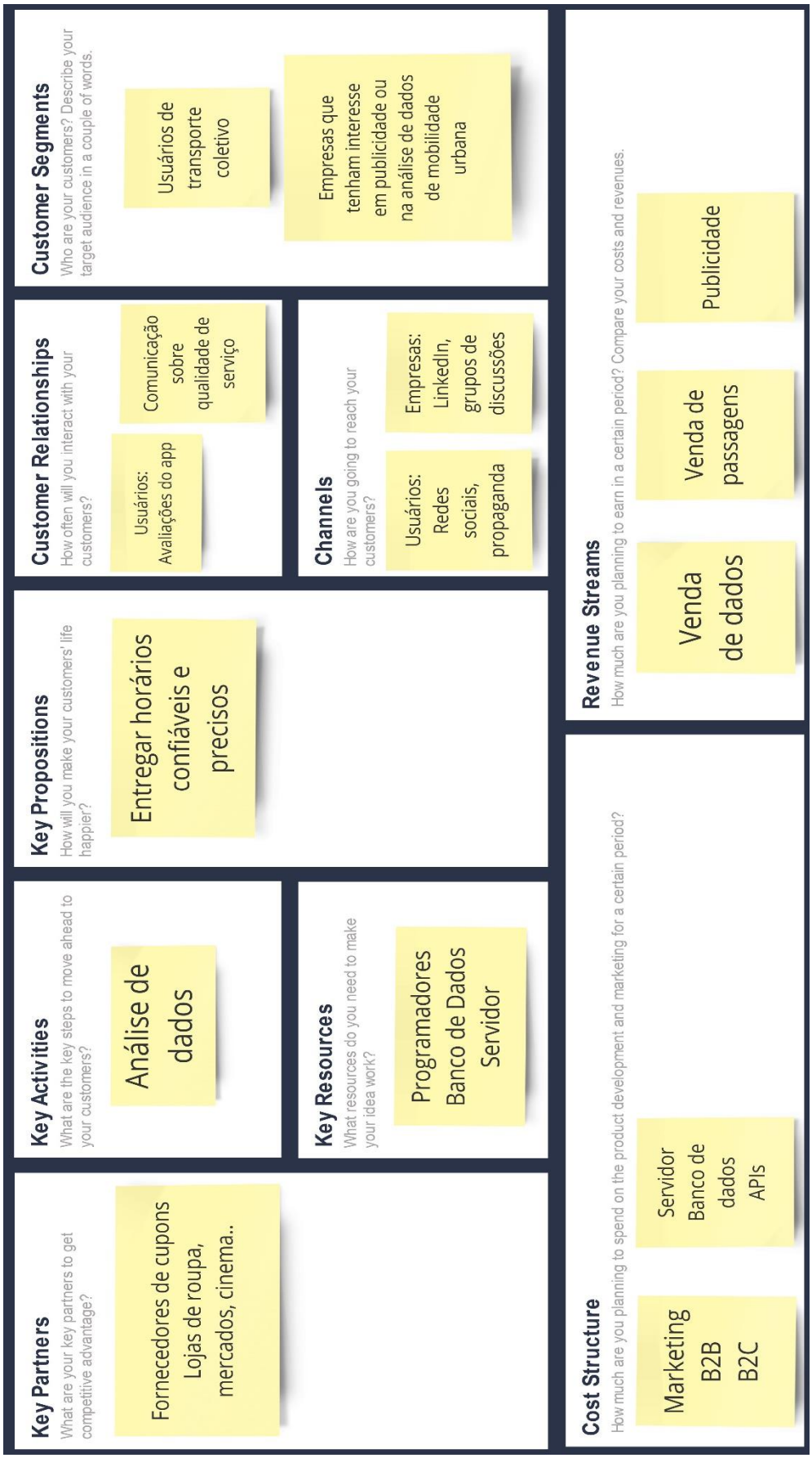
Um *pitch* de 3 minutos está disponível no seguinte link:

https://youtu.be/IrXTPCmei1w

7.5 CANVAS

O *Business Model Canvas* é uma ferramenta de planejamento estratégico que ajuda empreendedores a alinhar o modelo de negócios de um projeto novo ou já existente. Nele, diferentes campos podem ser alterados facilmente em um tipo de *brainstorming,* o que auxilia na criação de ideias e percepção de caminhos a tomar. O *canvas* desenvolvido para este projeto pode ser visto na Figura 38.

Figura 38 - Canvas utilizado



Fonte:Autor

CAPÍTULO 8 CONCLUSÃO

Este projeto iniciou-se como foco apenas na programação, sem a intenção de se tornar uma *startup*. Durante o Desafio I9, explorou-se mais a fundo o potencial da ideia e desenvolveu-se o lado empreendedor do negócio. Esta etapa foi de extrema importância no projeto, visto que possibilitou expandir caminhos que antes não eram cogitados.

O formulário realizado, resultante da participação no Desafio, analisou superficialmente o mercado e a necessidade de um aplicativo nesta área na cidade de Santa Maria. Foi possível perceber que o problema percebido é real e diversas soluções para este problema têm sido propostas. Recentemente, Santa Maria fechou acordo do desenvolvimento de um sistema na área (ZOLIN, 2017).

Especificamente por a cidade ser um polo universitário, o aplicativo da UFSM se mostrou ser o principal aplicativo utilizado pelos usuários para obter os horários de ônibus. O aplicativo utiliza o modelo tradicional de horários, mostrando apenas os horários de saída do ônibus. O Moovit, maior empresa da área, não se mostrou reconhecido com frequência neste estudo, o que aponta a necessidade de estudo de cada mercado para se obter um *marketing* eficiente e especifico.

O *Minimum Viable Product* desenvolvido, embora rudimentar, foi útil para analisar o comportamento dos usuários em relação à necessidade de compartilhar a localização. O MVP não era, de nenhum modo, tão simples de utilizar quando o conceito do aplicativo finalizado e isto pode ter afetado os resultados de maneira negativa. Entretanto, a reação dos membros do grupo a um estímulo de premiação mostrou que as pessoas estariam mais dispostas a fazer parte de um sistema colaborativo se este oferecesse vantagens externas além da sua função original de mostrar horários. Isto também pode ser visto como uma recompensa por colaborar com o sistema, pois os usuários estariam dispondo do seu plano de dados e bateria.

Os resultados tirados do filtro de partículas não foram satisfatórios, mas talvez indiquem que o modelo de 50% de chance de parada para as partículas seja o mais indicado. Este modelo obteve melhores resultados mesmo sendo realizado sem *input* externo, e poderia ser otimizado atualizando a chance de parada em função da frequência de utilização de cada parada. Adicionalmente, o modelo poderia ser otimizado para que as partículas obtivessem uma velocidade relativa a cada seção do

trajeto, obtendo médias de velocidade mais altas em avenidas, por exemplo, e velocidades menores em áreas com mais curva ou intensidade de tráfego. A utilização do filtro de partículas no trabalho finalizado se mostra um bom caminho a seguir, mas ainda é necessário muito trabalho em cima do modelo para obtermos estimativas precisas.

Diferentemente do desenvolvimento do projeto do semestre anterior, o projeto atual começou sua programação com praticamente todo o aplicativo conceito desenvolvido. Isto foi feito utilizando os diagramas UML, o que facilitou muito a percepção dos caminhos a serem tomados em direção ao objetivo final. Assim, não foram muitas as mudanças de código durante o desenvolvimento. O aplicativo desenvolvido não encontra-se finalizado, mas já é um sistema de compartilhamento de localização coletiva, o que é o cerne da ideia. No estado atual, as pessoas são capazes de compartilhar sua localização e ver a localização dos ônibus existentes. O sistema também já é capaz de identificar quais usuários estão embarcados em quais ônibus.

Quanto ao plano de negócios, este foi rigorosamente discutido durante uma disciplina de empreendedorismo e ainda mais aprofundado durante o Desafio I9. As ferramentas como o *canvas* se mostraram valiosas para o desenvolvimento de ideias, pondo tópicos em discussão e propondo caminhos para a *startup.* Entretanto, como é comum nesta área, todo este planejamento deverá ser revisando quando posto a prova e quando suas fraquezas forem encontradas. Este é um dos motivos de o canvas ser uma ferramenta tão versátil, pois possibilita revisão das metas de uma empresa de acordo com as mudanças do mercado.

No geral, o projeto abrangeu diversas áreas de conhecimento e as englobou em apenas um sistema, obtendo resultados muito mais concretos do que os obtidos com o sistema anterior. Embora o aplicativo não esteja finalizado, seu desenvolvimento continua e mesmo que as notícias para a cidade de Santa Maria sejam desanimadoras em função do acordo com a empresa de instalação de dispositivos GPS nos ônibus, diversas outras cidades apresentam o mesmo problema e podem ser tidas como objetivo.

CAPÍTULO 9 BIBLIOGRAFIA

G1, 2013. Disponivel em: <http://g1.globo.com/sao-paulo/anda-sp/noticia/2013/06/pesquisa-traca-perfil-dos-usuarios-do-transporte-publico-em-sao-paulo.html>. Acesso em: 26 set. 2017.

StartUpi, 2016. Disponivel em: <https://startupi.com.br/2016/08/estudo-revela-preferencias-dos-usuarios-brasileiros-que-usam-servicos-via-aplicativos/>. Acesso em: 26 set. 2017.

ABRAN, A. et al. **Guide to the Software Engineering Body of Knowledge**. [S.l.]: [s.n.], 2004.

AMBLER, S. W. **The elementos of UML 2.0 Style**. [S.l.]: Cambridge University Press, 2005.

BECK, K.; CUNNINGHAM, W. **A laboratory for teaching object-oriented thinking**. OOPSLA'89 Conference Proceedings. Louisiana, New Orleans: [s.n.]. 1989.

CAMPOS, V. B. G. **UMA VISÃO DA MOBILIDADE URBANA SUSTENTÁVEL**. [S.l.]: [s.n.]. 2006.

CASTILHO, R. A. D. **Análise e simulação de operação de ônibus em corredores exclusivos**. [S.l.]: [s.n.]. 1997.

CINTRA, M. Os custos do congestionamento na capital paulista. **Conjuntura econômica**, junho 2008.

FRANÇOSO, M. T.; CUSTÓDIO DE MELLO, N. **INFLUÊNCIA DOS APLICATIVOS DE SMARTPHONES PARA TRANSPORTE URBANO NO TRANSITO**. 7° Congresso Luso Brasileiro para o planejamento urbano, regional, integrado e sustentável. Maceió - Brasil: [s.n.]. 2016.

FREITAS, M. K. **Investigação da produção e dispersão de poluentes do ar no ambiente urbano:** determinação empírica e modelagem em rede neural da concentração de CO. [S.l.]: [s.n.]. 2003.

GEOEDUC. **InstitutoGeoEduc**, 2015. Disponivel em: <http://www.geoeduc.com/mais-de-70-satelites-de-posicionamento-global-ja-estao-em-orbita-entenda/>. Acesso em: 30 jun. 2017.

HEKIMA. Big Data Business: Hekima, 2016. Disponivel em: <http://www.bigdatabusiness.com.br/o-que-e-analise-preditiva/>. Acesso em: 24 jan. 2018.

HERRICK, J. **Talk Android**, 2015. Disponivel em: <http://www.talkandroid.com/276516-moovit-live-directions-update/>. Acesso em: 28 nov. 2017.

HSIAO, K.; DE PLINVAL-SALGUES, H.; MILLER, J. Particle Filters and their applications, 11 abr. 2005. Disponivel em: <http://web.mit.edu/16.412j/www/html/Advanced%20lectures/Slides/Hsaio_plinval_miller_ParticleFiltersPrint.pdf>. Acesso em: 20 fev. 2018.

HTTP://WWW.GEOMIDPOINT.COM/. **GeoMidPoint**. Disponivel em: <http://www.geomidpoint.com/calculation.html>. Acesso em: 20 jan. 2018.

MANO, M. K. IPEA - Insituto de Pesquisa Econômica Aplicada, 2011. Disponivel em: <http://www.ipea.gov.br/desafios/index.php?option=com_content&id=2578:catid=28&Itemid=23>. Acesso em: 23 jun. 2017.

MELO, W. D. A.; SOUZA, D. N. A.; DA SILVA, C. D. **UTILIZAÇÃO DO PROGRAMA ADJUST EM AJUSTAMENTO DE TRIANGULAÇÕES E TRILATERAÇÕES**. IV Simpósio Brasileiro de Ciências Geodésicas e Tecnologias da Geoinformação. Recife - PE: [s.n.]. 2012.

NAMIOT, D. **GeoFence Services**. International Journal of Open Information Technologies. [S.l.]: [s.n.]. 2013.

NUNES, R.; PINHEIRO, B. Transporte coletivo em Santa Maria: uma questão pública? **Revista TxT**, Santa Maria, n. 22.

PENA, R. A. Mundo Educação, 2013. Disponivel em: <http://mundoeducacao.bol.uol.com.br/geografia/a-qualidade-transporte-publico-no-brasil-os-protestos.htm>. Acesso em: 23 jun. 2017.

PRESSMAN, R. S. **Software Engineering A practitioner's approach**. 7. ed. [S.l.]: McGraw-Hill, 2010.

URBS. **URBS**. Disponivel em: <https://www.urbs.curitiba.pr.gov.br/institucional/urbs-em-numeros>. Acesso em: 26 set. 2017.

VENESS, C. **Movable Type Scripts**, 2017. Disponivel em: <http://www.movable-type.co.uk/scripts/latlong.html>. Acesso em: 30 jun. 2017.

ZOLIN, D. Diário de Santa Maria, 2017. Disponivel em: <http://diariosm.com.br/not%C3%ADcias/economia/come%C3%A7am-neste-ano-os-

testes-com-app-que-informar%C3%A1-sobre-o-tempo-de-espera-do-%C3%B4nibus-1.2037458>. Acesso em: 02 fev. 2018.

APÊNDICE A – CARTÕES CRC

USUÁRIO	
Responsabilities	Collaboration
• Abrir o aplicativo • Registrar a linha selecionada • Chamar notificação • Incluir usuário em ônibus daquela linha ou criar ônibus daquela linha para monitoramento	• Aplicativo • Cadastro orgânico • Menu de linhas • Ônibus • Notificação

NOTIFICAÇÃO	
• Confirmar se o usuário esta embarcado na linha selecionada pelo menu de linhas.	• Usuário

TIMER	
• Envia com uma certa frequência um atualização do ônibus ao servidor	• Usuário • Servidor • Ônibus

MENU DE LINHAS	
• Mostrar linhas existentes com prioridade para linhas próximas a posição do usuário • Oferecer opção de cadastro de nova linha apenas se usuário estiver em uma parada • Ao selecionar linha já cadastrada mas que não há registro de parar naquela parada, perguntar se linha para ali. Usuário deve se encontrar na parada.	• APP • Usuário • Linha • Cadastro orgânico

MAPA	
• Mostrar as paradas da linha selecionada pelo usuário • Permitir que o usuário selecione uma parada e mostrar janela de horários para aquela parada	• Usuário • Paradas • Janela de horários

CADASTRO ORGÂNICO	
• Verificar se a posição do usuário corresponde a posição de uma parada cadastrada • Se a posição do usuário não corresponde a uma parada e se essa posição não estiver no cadastro de “não parada”, então abre pop-up perguntando se aquela posição é uma parada • Cadastro de novas linhas quando selecionada opção no menu de linhas	• Usuário • Parada • Aplicativo • Menu de linhas

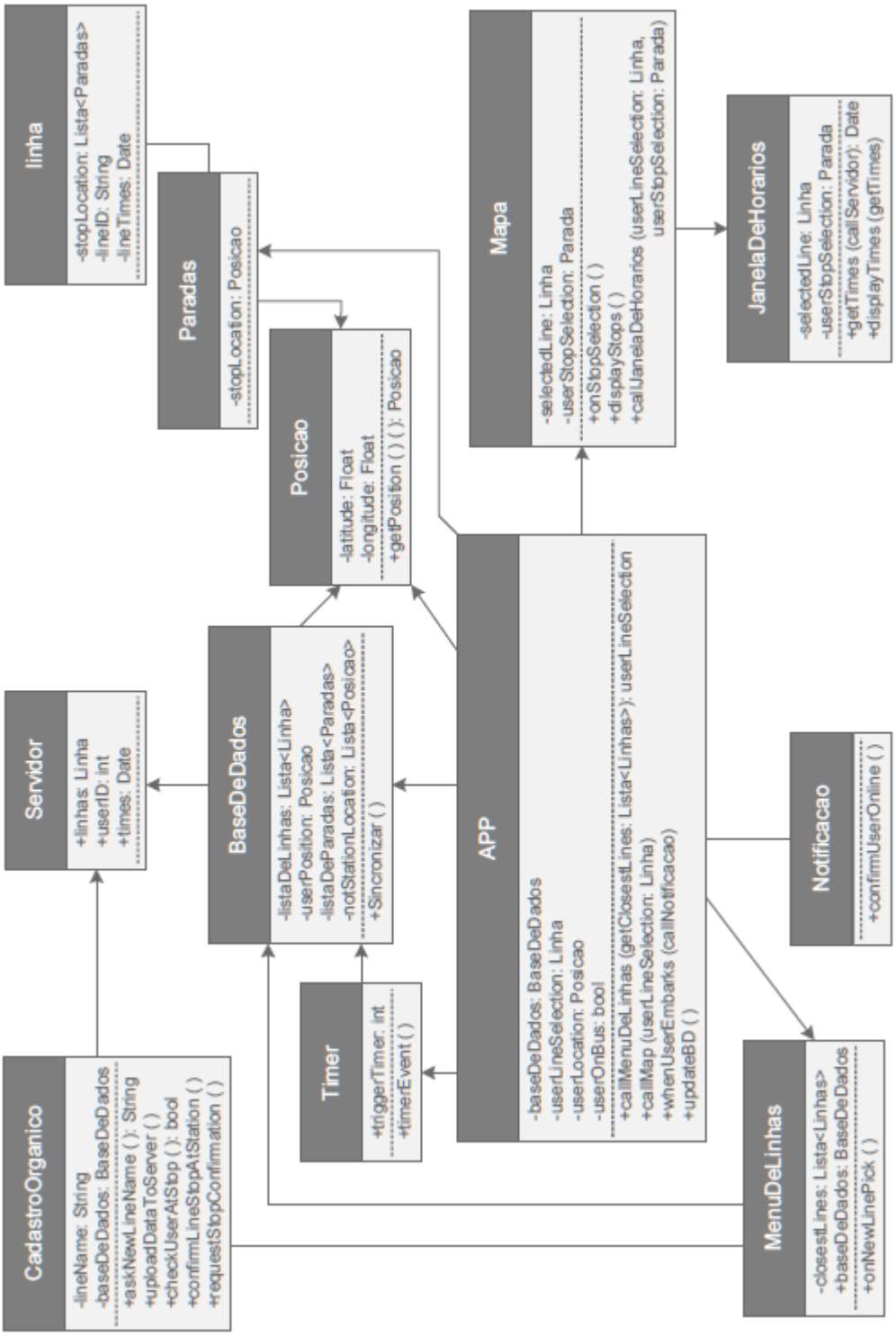
ÔNIBUS	
• Monitorar posição de gps quando usuário está embarcado. • Registrar linha a qual pertence.	• Usuário • Timer • Linha

JANELA DE HORÁRIOS	
• Mostrar horários para parada selecionada.	• Mapa • Linha

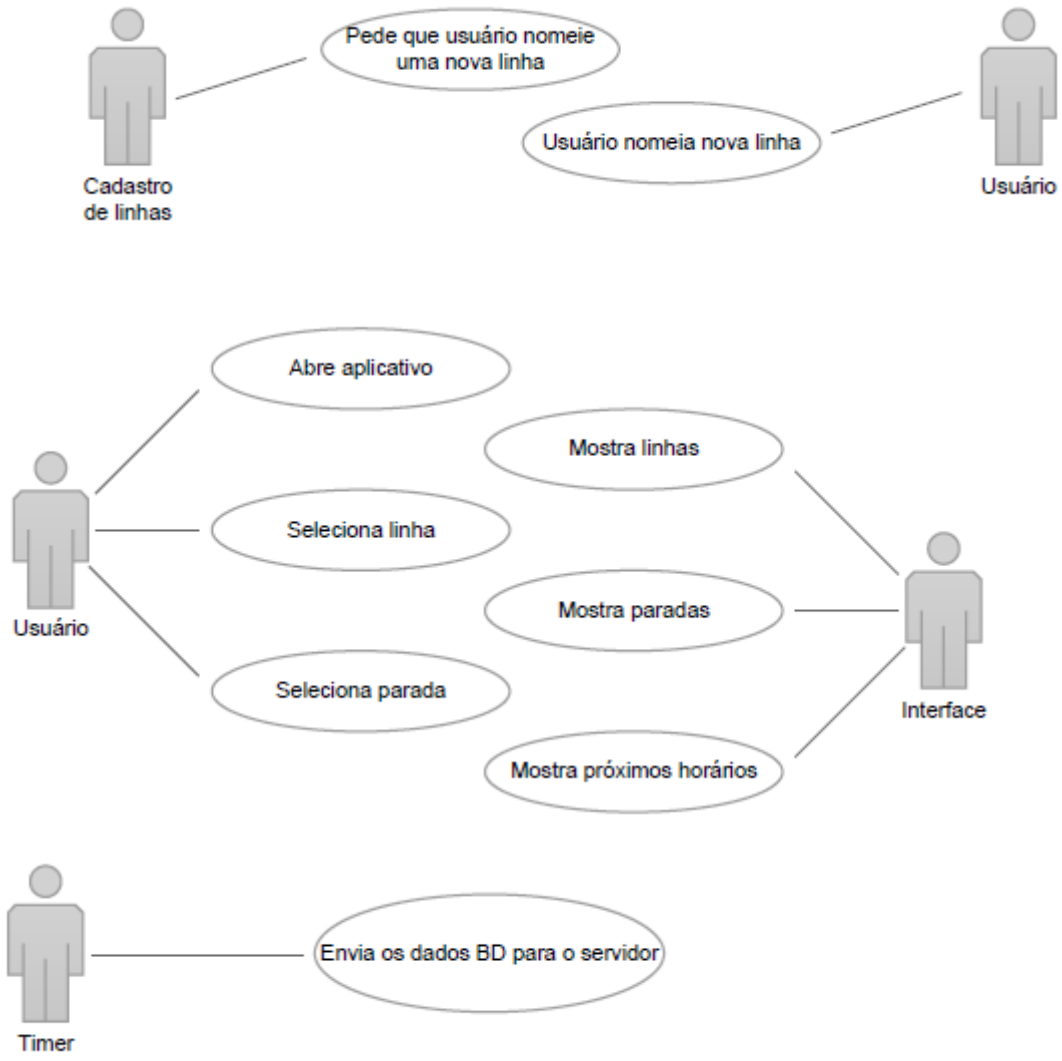
SERVIDOR	
• Registra linhas, usuário, paradas, ônibus.	• Timer • Linha

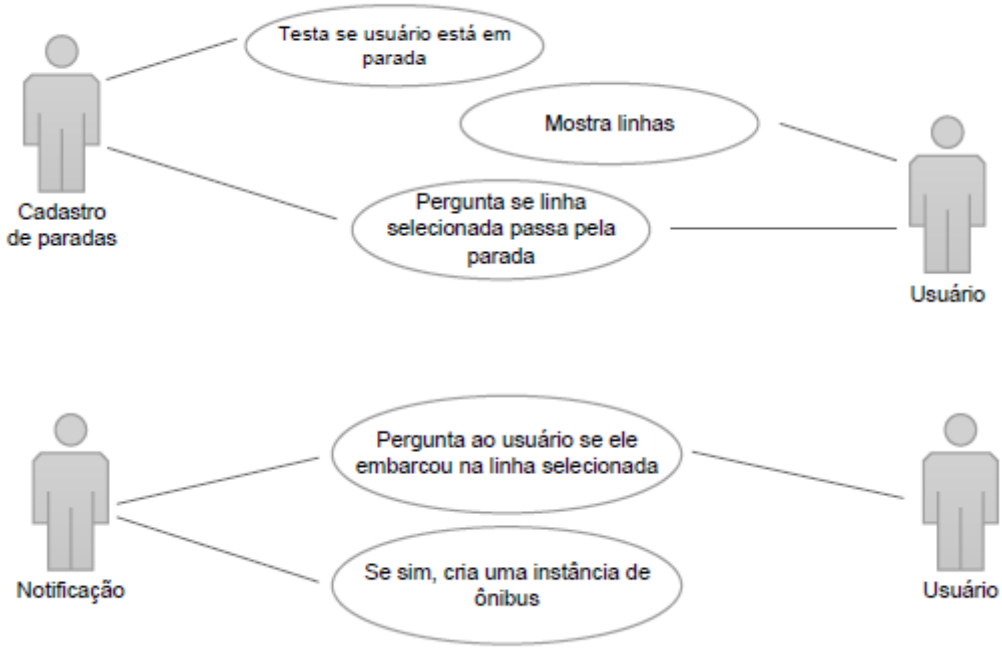
APP	
• Chama o cadastro org. • Chama o menu de linhas (calcular linhas próximas). • Verifica se o usuário está em uma parada. • Chama o mapa.	• Usuário • Cadastro org. • Menu de linhas • Parada • Mapa • Base de dados

APÊNDICE B - DIAGRAMA DE CLASSES.

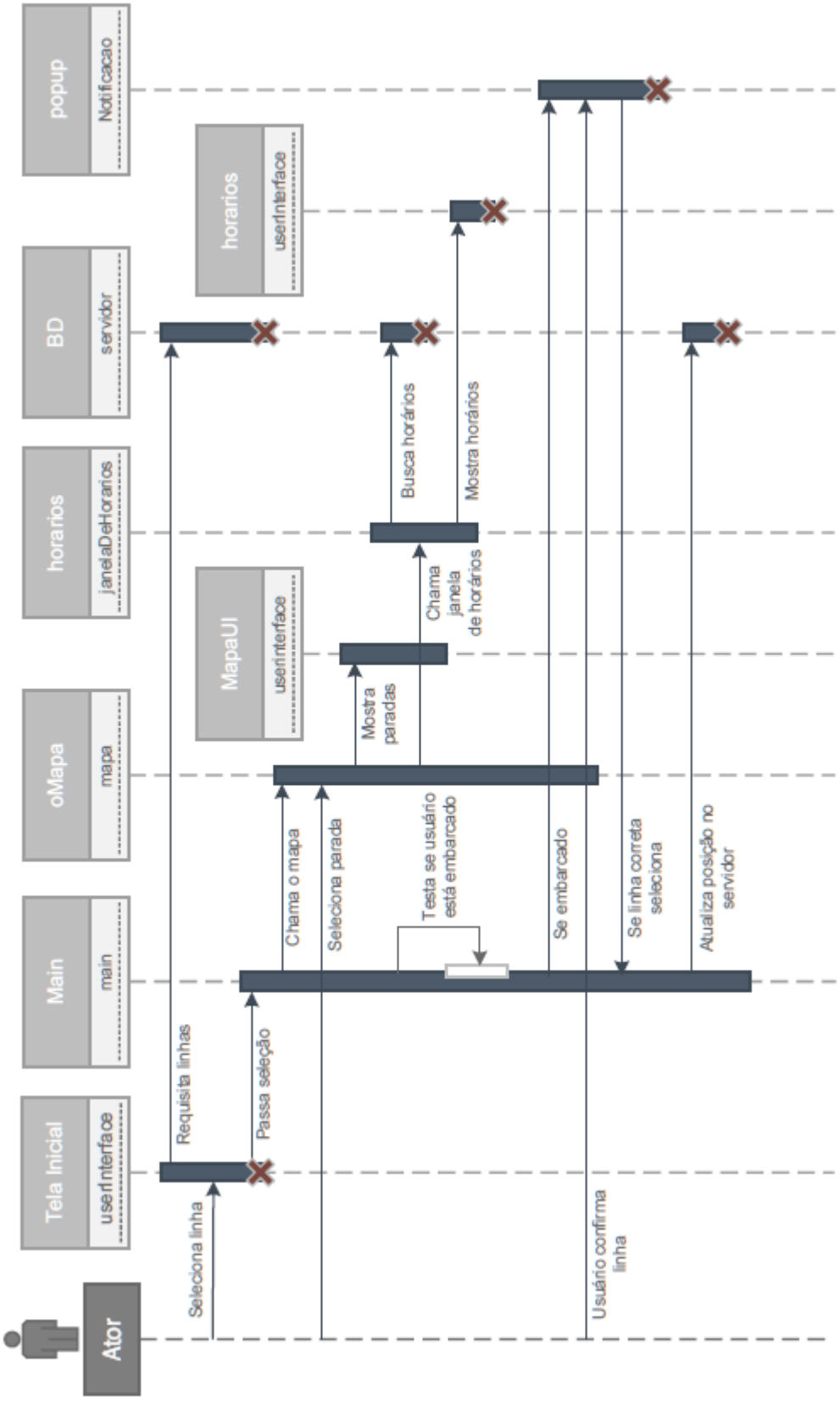


APÊNDICE C - DIAGRAMA DE CASO DE USO.

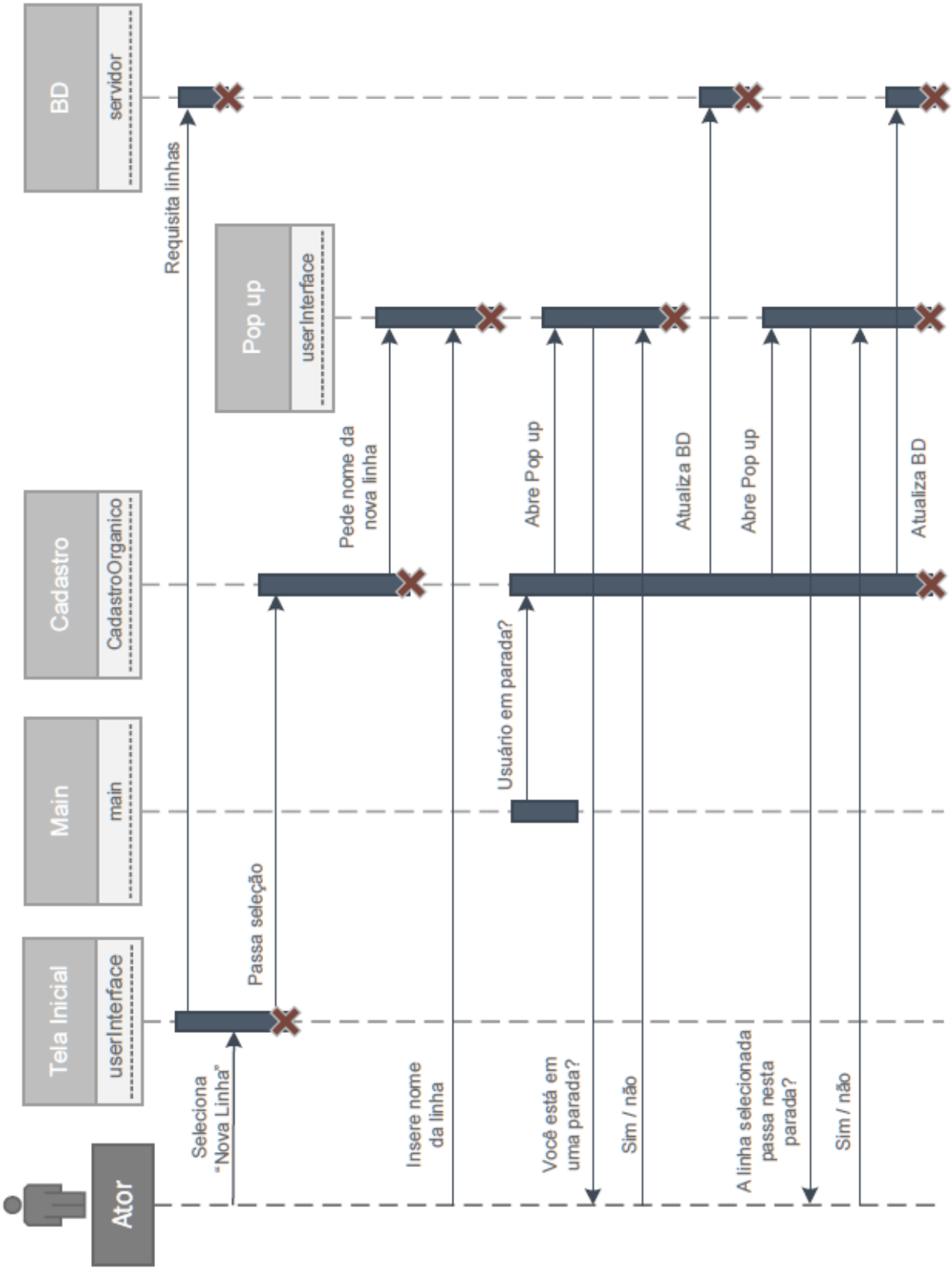



Testa se usuário está em parada
Mostra linhas
Cadastro de paradas
Pergunta se linha selecionada passa pela parada
Usuário
Pergunta ao usuário se ele embarcou na linha selecionada
Notificação
Se sim, cria uma instância de ônibus
Usuário

APÊNDICE D – DIAGRAMAS DE SEQUÊNCIA



APÊNDICE E – DIAGRAMAS DE SEQUÊNCIA



APÊNDICE F – CÓDIGO DO FILTRO DE PARTÍCULAS.

```
from constants import *
from random import uniform
from time import sleep, time
import numpy as np
import pygame
from pygame.locals import *
from random import random
import sys
pygame.init()

size = (700, 500)
screen = pygame.display.set_mode(size)
n = 0
global busPos
busPos = 0
global carryOn
carryOn = True
restart = False

# The clock will be used to control how fast the screen updates
clock = pygame.time.Clock()
class Particle(object):
      def __init__(self, name, vel, t):
            self.velocity = vel
            self.position = 10
            self.timeStop = t
            self.time = 0.0
            self.name = name
            self.currentSector = 0
            self.atStop = True
            self.endStop = False
            self.willStop = True
            self.changedSector = False
            self.random = 0.0
            self.hasCalledRandom = False
            self.lastStop = 0
            self.running = False
            self.distance = 0
```

```
            def __repr__(self):
                  return '[Name = %s Position = %.6s Velocity = %.6s TimeStop
= %.6s Time = %s]' % (str(self.name), \
                   str(self.position),                   str(self.velocity),
str(self.timeStop), str(self.time))

      class ParticleFilter(object):
            def __init__(self, num, mainBus = False):
                  self.particles = []
                  self.mainBus = mainBus
                  for i in range(num):
                        self.particles.append(Particle(i,
self.sample(VELOCITIES[0][1])                   +          VELOCITIES[0][0],
self.sample(STOPTIMES[0][1]) + STOPTIMES[0][0]))
                  self.time = time()

            def sample(self, var):
                  s = 0
                  for i in range(12):
                        s += uniform(-var,var)
                  return 0.5*s

            def update(self, p, i):
                  p.velocity    =    self.sample(VELOCITIES[i][1])        +
VELOCITIES[i][0]
                  p.timeStop      =      self.sample(STOPTIMES[i][1])      +
STOPTIMES[i][0]
                  p.time = 0.0

            def stopAtAllStops(self):
                  global n
                  newTime = time()
                  timeDiff = newTime - self.time
                  self.time = newTime

                  for p in self.particles:
                        i = np.abs(DISTANCES-p.position).argmin()
                        '''if (p.lastStop == STOPNUMBER - 1):
                              p.endStop = True'''
                        if (p.endStop == True):
```

```
                        n += 1
                        p.velocity = 0.0
                        if self.mainBus:
                              print "FIM"
                  else:

                        #print np.abs(p.time - p.timeStop)
                        if self.mainBus:
                              pass
                        else:
                              if p.position > 10:
                                    distance = mainBus.position -
p.position
                                    p.distance = (distance +
p.distance)/2
                                    #with
open("stopAtAllStopsResults.txt", "a") as myfile:

                                          #myfile.write("\n" +
str(abs(distance)))


                        #Se esta na parada, recalcula parametros
                        if (p.atStop == True):
                              p.changedSector = False
                              p.running = False
                              p.lastStop = i
                              #secao abaixo para em todas as paradas
                              '''if self.mainBus == True:

                                    p.random = random()
                                    if (p.random >= 0.5):
                                          p.willStop = True
                                    else:
                                          p.willStop = False


                              else:
                                    p.willStop = True'''
                              #secao abaixo faz paradas aleatorias
                                          p.random = random()
```

```
                                    if (p.random >= 0.5):
                                          p.willStop = True
                                    else:
                                          p.willStop = False

                                    p.time = p.time + timeDiff
                                    #Se passar tempo de espera na parada,
inicia a corrida.
                                    if (np.abs(p.time - p.timeStop) < 0.1 or
p.time > p.timeStop):
                                          p.running = True
                                          self.update(p, i)
                              if (p.running == True):
                                    p.position += p.velocity * timeDiff
                                    p.atStop = False
                                    if self.mainBus:
                                          mainBus.position = p.position

                                    if (p.currentSector == STOPNUMBER - 2):
                                          p.willStop = True
                                          p.endstop = True
                                          if(p.position                    >
DISTANCES[p.currentSector + 1]):
                                                p.atStop = True
                                                #Se onibus chegou na parada
final
                                                if (self.mainBus):
                                                      for        p        in
part.particles:
                                                            #em qual arquivo
salvar
                                                            with
open("randomResults.txt", "a") as myfile:
            myfile.write("\n" + str(abs(p.distance)))
                                                      global carryOn
                                                      carryOn = False
                                    else:
                              #Se passar de uma parada e precisar parar, para
                                          if(p.position                    >
DISTANCES[p.currentSector + 1] and p.willStop == True):
                                          p.currentSector += 1
```

```
                                        p.atStop = True
                                        if (p.position >
      DISTANCES[p.currentSector+1]):
                                              p.currentSector += 1
                                              p.changedSector = True
                        #Se passar de uma parada, decide se quer parar na
proxima
                              if (p.changedSector == True and self.mainBus ==
True):
                                        p.changedSector = False

                                        p.random = random()
                                        if (p.random >= 0.5):
                                              p.willStop = True
                                        else:
                                              p.willStop = False
                                              #print p.willStop
            def move(self):
                  self.stopAtAllStops()

            def draw(self):
                  if self.mainBus:
                        y = 40
                        r = 5
                        color = BLUE

                  else:
                        y = 160
                        r = 2
                        color = BLACK

                  for p in self.particles:
                        pos = int(p.position)
                        pygame.draw.circle(screen, color, [pos, y], r, 0)

      class MainBus():
            def __init__(self):
                  self.mainBus = ParticleFilter(1, mainBus = True)
                  self.position = 10

            def move(self):
```

```
                self.mainBus.move()

        def draw(self):
                self.mainBus.draw()

        def embark(self):
                self.mainBus.particles[0].willStop = True

    class People():
        def __init__(self):
                self.pos = 0

        def draw(self, pos):
                pass
    if __name__ == "__main__":
        part = ParticleFilter(NUMPARTICLES)
        mainBus = MainBus()

        while carryOn:
        # --- Main event loop
                for event in pygame.event.get(): # User did something
                        if event.type == pygame.QUIT: # If user clicked close
                                carryOn = False # Flag that we are done so we
exit this loop
                # --- Game logic should go here

                part.move()
                mainBus.move()
                if (n >= NUMPARTICLES):
                        carryOn = False
                        #print "TERMINOU"
                # --- Drawing code should go here
                # First, clear the screen to white.
                screen.fill(WHITE)
                #The you can draw different shapes and lines or add text
to your background stage.
                for i in range(6):
                        pygame.draw.line(screen, GREEN, [DISTANCES[i], 0],
[DISTANCES[i],80], 5)
                        pygame.draw.line(screen, GREEN, [DISTANCES[i], 120],
[DISTANCES[i],200], 5)
```

```
                part.draw()
                mainBus.draw()
                # --- Go ahead and update the screen with what we've drawn.
                pygame.display.flip()
                # --- Limit to 60 frames per second
                clock.tick(60)


        #Once we have exited the main program loop we can stop the game
engine:
        pygame.quit()
```