UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

GIONATTA MARCON MOCELLIN
GIUSEPE PIETRO NIQUELE

UTILIZAÇÃO DE APLICATIVO MÓVEL E *CROWDSOURCING* PARA RASTREIO DE ÔNIBUS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2014

GIONATTA MARCON MOCELLIN
GIUSEPE PIETRO NIQUELE

UTILIZAÇÃO DE APLICATIVO MÓVEL E *CROWDSOURCING* PARA RASTREIO DE ÔNIBUS

Trabalho de Conclusão de Curso de Engenharia de Computação apresentado ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Engenheiro em Computação".

Orientadora: Prof^a. Dr^a. Ana Cristina Kochem Vendramin

Co-orientadora: Prof^a. Dr^a. Anelise Munaretto Fonseca

CURITIBA

2014

RESUMO

MOCELLIN, Gionatta Marcon. NIQUELE, Giusepe Pietro. UTILIZAÇÃO DE APLICATIVO MÓVEL E *CROWDSOURCING* PARA RASTREIO DE ÔNIBUS. 44 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

O incessante ritmo ditado pelos grandes centros urbanos gera na população a necessidade de minimizar o tempo gasto em seus deslocamentos, principalmente quando o transporte público é utilizado para este propósito. Contudo, é muito difícil determinar com precisão quando um ônibus chegará ao ponto ou ao terminal. Por vezes, isso causa longos períodos de espera, ou porque o ônibus acabou de partir, ou porque se atrasou devido a um imprevisto. Felizmente, a tecnologia evolui a tal ponto que os chamados *smartphones* se tornaram um artefato muito popular. Ainda, o advento de redes móveis e sem fio, tais como o 3G e o WiFi, permite que pessoas fiquem conectadas a maior parte do tempo, independente do lugar em que estejam. Dentre os inúmeros benefícios que os *smartphones* fornecem, a possibilidade de troca de informações entre usuários é uma delas, permitindo um fenômeno chamado *crowdsourcing*. O *crowdsourcing*, aplicado em um âmbito maior - que englobe toda ou a maior parte da cidade - poderia ser um primeiro passo para elevá-la ao nível de uma *Smart City* ou "cidade inteligente", onde a tecnologia é utilizada pela população para a solução/mitigação de um determinado problema urbano, no caso, a mobilidade. Neste projeto é proposto o desenvolvimento de um aplicativo móvel que alie conceitos, tanto de *Smart Cities* quanto de *crowdsourcing*, no qual informações em tempo real sobre o transporte coletivo, especificamente no que compete o estado atual de uma linha de ônibus, sejam disponibilizadas aos usuários.

Palavras-chave: Dispositivos móveis, Redes tolerantes a atraso, Cidades inteligentes, *Crowdsourcing*, Acompanhamento de ônibus

ABSTRACT

MOCELLIN, Gionatta Marcon. NIQUELE, Giusepe Pietro. . 44 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

The unceasing rhythm dictated by the large urban centers causes in the population the necessity of minimizing the time spent in their locomotion, especially when the public transport is used for this purpose. However, it's difficult to determine with precision when the bus will stop at the bus station. Fortunately, the technology evolved to a point where the smartphones are now a very popular artifact. Nevertheless, the mobile and wireless networks, such as 3G and WiFi, allow the people to stay connected all the time, regardless the place they are. Amongst the severeal benefits brought by the smartphones, the information exchange between the users is one of them, making possible a phenomenon called crowdsourcing. The crowdsourcing, applied in a larger environment – covering the entire, or almost the entire, city – may be the first step to evolve it to a Smart City, where the technology is used by the population to solve/mitigate a specific problem, such as mobility. In this project is proposed the development of a mobile application which integrates concepts from Smart Cities and crowdsourcing, where real time informations concerning the public transport, specifically the actual state of a bus line, is availabe to the users.

Keywords: Mobile devices, Delay Tolerant Networks, Smart Cities, Crowdsourcing, Bus tracking

LISTA DE SIGLAS

ICT	*Information and Communication Technology*
GPS	*Global Positioning System*
URBS	Companhia de Urbanização e Saneamento de Curitiba
MBTS	*Mobile Bus Tracking System*
SMS	*Short Message Service*
DTN	*Delay Tolerant Networks*
ACK	*Acknowledge*
P2P	*Peer-to-Peer*
JSP	*JavaServer Pages*
RF	Radio Frequência
CVS	*Control Version System*
ADT	*Android Developer Tools*
SDK	*Software Development Kit*
API	*Application Programming Interface*
JDK	*Java Development Kit*
PERT	*Program Evaluation and Review Technique*
CPM	*Critical Path Method*
UAW	*Unadjusted Actor Weight*
UUCW	*Unadjusted Use Case Weight*
UUCP	*Unadjusted Use Case Point*
TCF	*Technical Complexity Factor*
EF	*Environmental Factor*
UCP	*Use Case Points*

SUMÁRIO

1 INTRODUÇÃO

O conceito de *Smart Cities* (Cidades Inteligentes, em tradução livre) é um sustentáculo elementar para o desenvolvimento urbano sustentável. De certa forma, ele poderia contribuir na solução de diversos problemas críticos, originados a partir da urbanização de grandes cidades como congestionamentos, poluição do meio ambiente, e os limites dos recursos naturais. (PAN et al., 2013).

Um outro aspecto do conceito de cidades inteligentes a ser considerado, é a importância crescente das tecnologias digitais para um futuro sustentável. Assim sendo, destacam-se seis principais áreas, nas quais essas inovações digitais tem a competência para fazer a diferença: vida inteligente, governança inteligente, economia inteligente, ambiente inteligente, pessoas inteligentes e mobilidade inteligente (SCHUURMAN et al., 2012).

Com o uso da tecnologia da informação e comunicação (do inglês ICT, *Information and Communication Technology*,) a análise e mineração de dados de sensoriamento grande cidades é um passo importante para considerá-las uma cidade inteligente. Por exemplo, informações sobre a mobilidade dos veículos e pessoas tornaram-se temas em voga nos últimos tempos devido à prevalência do GPS (*Global Positioning System*) e outras tecnologias de localização. Esse conhecimento sobre a cidade e seus habitantes poderá trazer benefícios nas áreas de transporte, planejamento urbano, saúde pública, segurança pública e comércio (PAN et al., 2013).

Embora as ações de sensoriamento e mineração dos dados seja de imprescindível importância, algo deve ser feito para transformar esses dados em informações. Nesse sentido, uma abordagem interessante para a resolução desta questão consiste no conceito de *crowdsourcing* (SCHUURMAN et al., 2012).

Howe (2006) caracteriza *crowdsourcing* como o fenômeno no qual várias pessoas comuns dedicam seu tempo livre na busca de soluções para um problema de interesse coletivo. Isto posto, a massificação dos chamados *smartphones* e o advento de redes móveis

e sem fio tais como o 3G e o WiFi e recentemente, o 4G, permitem que pessoas fiquem conectadas a maior parte do tempo, em qualquer lugar. Dentre os inúmeros benefícios que os *smartphones* fornecem, a possibilidade de troca de informações entre usuários é uma delas, bem como a aplicação do conceito de *crowdsourcing*.

Considera-se o seguinte cenário: uma pessoa que acabou de chegar em um ponto de ônibus, deseja saber onde o mesmo se encontra. Questões como "será que o ônibus está chegando?", "será que vai demorar?" ou "será que o ônibus estragou?" são bastante comuns. Mas como descobrir suas respostas? Infelizmente, não é possível respondê-las apenas com a tabela de horários de chegada dos ônibus, disponível nos terminais e no *site* da URBS (Companhia de Urbanização e Saneamento de Curitiba).

Nesse sentido, é proposto nesse documento uma solução possível para esse quesito. A ideia básica é a construção de um aplicativo que alie conceitos, tanto de *Smart Cities* quanto de *crowdsourcing*, no qual informações em tempo real sobre o transporte coletivo, especificamente no que compete o estado atual de uma linha de ônibus, sejam disponibilizadas aos usuários.

Alguns aplicativos para dispositivos móveis foram desenvolvidos com o objetivo de prover, ao usuário, informações referentes ao transporte público. Um dos aplicativos mais recentes é o chamado "Busão Curitibano", lançado em 2013[1]. Os desenvolvedores – e também os usuários do aplicativo – encontraram algumas resistências por parte da URBS, pois o aplicativo usava a base de dados do transporte coletivo. Devido aos inúmeros acessos, acabava derrubando o servidor da URBS e tornando o serviço de acompanhamento de ônibus indisponível.

Outro aplicativo semelhante foi desenvolvido por Sujatha et al. (2014). Os autores propuseram um sistema chamado MBTS (*Mobile Bus Tracking System*) que ajuda qualquer pessoa a obter informações de um determinado ônibus sem fazer ligações ou perturbar passageiros com mensagens SMS (*Short Message Service*). Um banco de dados é utilizado para armazenar informações sobre os ônibus (por exemplo, as coordenadas do veículo), as quais estarão disponíveis para acesso através de um aplicativo Android..

1.1 JUSTIFICATIVA

A utilização de uma arquitetura centralizada, estilo cliente–servidor é um ponto comum nos dois trabalhos relacionados. Em ambos, nota-se que a arquitetura encontra-se

[1]Para mais informações sobre o aplicativo, consultar notícia da Gazeta do Povo, disponível em: http://www.gazetadopovo.com.br/vidaecidadania/conteudo.phtml?id=1377575

em uma relação n:1, ou seja, vários clientes para um único servidor. Alguns dos problemas que podem surgir nesse tipo de arquitetura é que o servidor pode ficar sobrecarregado e eventualmente ficar indisponível devido ao grande número de usuários utilizando o serviço, como foi o caso do "Busão Curitibano". Além disso, nesse caso, o servidor utilizado não é de propriedade dos desenvolvedores, mas sim da URBS. Caso a companhia deseje interromper o serviço de fornecimento de dados, o "Busão Curitibano" perde sua utilidade. Isso não constitui um problema para o MBTS, pois o mesmo concentra as informações em um servidor próprio, mantido pelos próprios desenvolvedores.

Sendo assim, a principal motivação para o desenvolvimento deste projeto é implementar um aplicativo de acompanhamento de ônibus para dispositivos móveis – *smartphones* – que utilize uma rede descentralizada. Nela, os próprios usuários são os nós, que compartilharão informações entre si. Para isso, utilizar como referência o aplicativo móvel "Waze"[2], que implementa uma espécie de comunidade para mapeamento do trânsito em tempo real e usufruir do GPS dos *smartphones* para fornecer aos usuários a posição dos veículos.

1.2 OBJETIVO GERAL

Desenvolver um aplicativo móvel que execute sobre o sistema operacional Android, aliando conceitos de *Smart Cities* e *Crowdsourcing*, no qual informações em tempo real sobre o transporte coletivo sejam disponibilizadas aos usuários.

1.3 OBJETIVOS ESPECÍFICOS

- Desenvolver um aplicativo móvel, exclusivamente para *smartphones* e *tablets* com sistema operacional Android, para acompanhamento de ônibus da rede de transporte de Curitiba;
- Estudar e implementar no aplicativo o conceito de *crowdsourcing*, para permitir que usuários colaborem entre si com informações pertinentes, através de *feedbacks* pré-definidos no aplicativo;
- Desenvolver o aplicativo para que o mesmo seja executado sobre uma plataforma que não exija custos de desenvolvimento;

[2]Uma análise do aplicativo pode ser encontrada em http://www.techtudo.com.br/tudo-sobre/waze.html

- Implementar uma arquitetura de rede descentralizada, a fim de evitar problemas como sobrecarga de servidor.
- Unir ideias de aplicativos móveis já existentes, como o “Waze” e o “Busão Curitibano”, no desenvolvimento do projeto.

1.4 ESTRUTURA E ORGANIZAÇÃO

O presente documento encontra-se organizado da seguinte forma. O capítulo 1 apresenta a introdução, com a formulação do problema, justificativa do projeto, objetivo geral e específicos do mesmo. O capítulo 2 apresenta o levantamento bibliográfico, importante para fundamentar o projeto e fornecer um embasamento. A metodologia, que descreve as etapas de desenvolvimento pelas quais o projeto percorrerá, é apresentada no capítulo 3. Os recursos de *hardware* e *software* para desenvolvimento do projeto, e a viabilidade do mesmo são apresentados nos capítulos 4 e 5, respectivamente. O contexto pelo qual se aplica o projeto é apresentado no capítulo 6 e a especificação do projeto, que engloba requisitos funcionais, não-funcionais e casos de uso é apresentada no capítulo 7. A gestão e o planejamento do projeto são apresentadas no capítulo 8, e no capítulo 9 apresenta-se a análise de riscos. Por fim, conclusões pertinentes ao desenvolvimento do projeto, tais como dificuldades esperadas e horizontes de projeto, são apresentados no capítulo 10.

2 LEVANTAMENTO BIBLIOGRÁFICO

Este capítulo apresenta uma breve introdução à *Smart Cities*, *Crowdsourcing*, contexto atual do desenvolvimento de programação voltada a aplicativos de transporte público e demais tópicos relativos ao desenvolvimento do trabalho.

2.1 *SMART CITIES*

O conceito de Cidades Inteligentes e o respectivo aumento no seu uso, pode ser reconhecido como o resultado do alto crescimento das tecnologias digitais na intenção de garantir um futuro sustentável. Embora o conceito seja empregado normalmente com o objetivo de elaborar estratégias que objetivem melhorar a qualidade de vida dos cidadãos, criando um futuro sustentável, o conceito por si só continua sendo usado em diferentes contextos e assim permanece um tanto ambíguo. Dessa forma faz-se necessário estabelecer um critério muito importante que o diferencia das cidades-conceito, o aspecto colaborativo entre os diversos interessados, mais comumente os cidadãos (SCHUURMAN et al., 2012).

Um outro enfoque, mais empresarial, apresentado por Walravens (2012), menciona que o conceito também pode ser empregado para caracterizar um grupo de organizações que intentam por revolucionar algum aspecto em uma região, entre os quais estão parques empresariais, o nível de escolaridade de uma população, o uso de tecnologias em contextos urbanos, o aumento da eficácia de governos e especialmente aquelas com foco em ICT. Walravens argumenta também que de maneira geral, o uso de serviços de telefonia móvel adquirem uma importância inevitável, já que é um sub aspecto de ICT e dessa forma grande importância na construção de uma Cidade Inteligente (WALRAVENS, 2012).

Em vista disso, sistemas operacionais desenvolvidos para *smartphones* inspiram desenvolvedores a conceber aplicativos e serviços que aperfeiçoam a vida nas cidades de diversas formas diferentes, como por exemplo, facilitando acesso à informação sobre o transporte publico. Além disto, à medida que os *smartphones* se tornam mais acessíveis

e populares, cria-se o desejo de que se tornem eminentes ferramentas na busca de uma cidade mais inteligente (WALRAVENS, 2012).

2.2 *CROWDSOURCING*

Outro aspecto importante a ser mencionado é a caracterização de *crowdsourcing*, no que tange inteligência coletiva. À grosso modo, entende-se inteligência coletiva como quando indivíduos de um determinado grupo, que possui interesse por determinado assunto, combinam seus conhecimentos a fim de encontrar a solução para um determinado problema. Através da interação social, o conhecimento individual é compartilhado, corrigido, processado, enriquecido e avaliado. Normalmente, os resultados colhidos são melhores que os obtidos por um único indivíduo. Talvez a aplicação mais difundida desse conceito seja a Wikipedia (SCHUURMAN et al., 2012).

Por conseguinte, a ponte que se estabelece entre *crowdsourcing* e *smart cities* acompanha o advento dos *smartphones*. Para Kanhere (2011), as melhorias no poder de processamento, sensoriamento e capacidades de armazenamento permite que telefones celulares sejam comparados a dispositivos de computação. Esse fenômeno abre margem ao surgimento de um novo paradigma, denominado pela literatura de sensoriamento participativo, cuja ideia principal gira em torno de capacitar um cidadão comum a coletar e compartilhar dados de sensoriamento do ambiente em que está inserido, a partir de seu telefone celular (KANHERE, 2011).

Ainda segundo Kanhere, sensoriamento participativo possui quatro vantagens principais sobre redes de sensores tradicionais (já que esta necessita de uma gama considerável de dispositivos sem fio, principalmente em áreas urbanizadas), sendo elas: custo de implementação baixo, já que usa telefones celulares e Wi-Fi; uma ampla mobilidade e cobertura proporcionada pelas operadoras de telefonia; economias de escala proporcionadas pelo uso de celulares; a facilidade assegurada pelas lojas de aplicativos e também as inúmeras formas de desenvolvimento de software disponíveis para sistemas operacionais de dispositivos móveis (KANHERE, 2011).

2.3 ARQUITETURAS DE REDE

A proposta de uma arquitetura para DTN (*Delay Tolerant Networks*) surgiu a partir da necessidade de que dispositivos capacitados para operar nas redes móveis e sem fio (smartphones e tablets, por exemplo) operem em qualquer lugar e instante de tempo,

mesmo na ausência de uma infra-estrutura de rede. Assim, alguns problemas como conectividade não contínua e atrasos muito longos, podem ser solucionados (VENDRAMIN, 2012).

Segundo Warthman (2012), DTNs foram projetadas originalmente para uso interplanetário, onde a tolerância a atrasos é de extrema importância. No entanto, muitas aplicações terrestres podem ser vislumbradas com o uso de DTNs, especialmente quando trata-se de tecnologias sem fio (*wireless*), tais como RF.

Entretanto, o modelo atual no qual funciona a Internet, largamente apoiada no protocolo TCP/IP, vai de encontro com alguns princípios de uma arquitetura DTN. A comunicação utilizando TCP/IP no que se refere à conectividade intermitente, na qual não há um caminho entre um nó origem e um nó destino na rede, por exemplo, não funciona (WARTHMAN, 2012). Ainda, segundo Warthman (2012), os protocolos de Internet atuais não suportam longos períodos de espera por uma resposta (um ACK, *Acknowledge*).

A arquitetura DTN se apoia num método muito antigo chamado *store-carry-forward* (armazenar, transportar e encaminhar), onde informações (ou fragmentos dela) são armazenados em um local de armazenamento em um nó e movidos para outro nó através de um caminho que eventualmente atinja o destino (WARTHMAN, 2012). Warthman (2012) enumera as razões na qual o armazenamento persistente em uma DTN é necessário, que são a possibilidade de que a comunicação entre os nós não esteja indisponível por um longo tempo, a disparidade na velocidade de transmissão e recepção de informações entre os nós e a transmissão (retransmissão) de informações em caso de erros ou de não aceitação de recebimento por um determinado nó.

No que tange arquiteturas de rede, há também a arquitetura P2P (*Peer-to-Peer*) capaz de fornecer recursos de rede com sobreposição e auto-organização distribuída, com o intuito de prover uma distribuição eficiente dos dados. O mecanismo básico de funcionamento consiste em pares, nos quais uma entidade presta e, ao mesmo tempo, consome os recursos oferecidos por outras entidades que compõe o sistema. Outras características que estes sistemas apresentam, normalmente, são auto-organização, tolerância a falhas e escalabilidade (CACCIAPUOTI; CALEFFI; PAURA, 2011).

Com base em todos esses conceitos, estabelece-se, a seguir, um breve estado da arte registrado na literatura, no que se refere à área de desenvolvimento de aplicativos para plataformas móveis dentro da ideia de *crowdsourcing* e *smart cities*.

2.4 ESTADO DA ARTE

Conforme já relacionado na Introdução, Sujatha et al. (2014) propuseram um sistema chamado MBTS para que qualquer pessoa obtenha informações de um determinado ônibus através de um aplicativo móvel.

Três são os atores do sistema: os passageiros, os chamados "coordenadores" do ônibus e as pessoas que estão aguardando em um ponto. Todos os atores necessitam de um *smartphone* com sistema operacional Android conectado à Internet. Continuamente, os passageiros e os "coordenadores" alimentam um banco de dados com informações sobre o ônibus. Essas informações podem ser as coordenadas do ônibus, obtidas através do GPS integrado ao *smartphone*, ou até mesmo a "situação" do ônibus – se ele está lotado ou vazio, se houve algum acidente, se o ônibus está atrasado, entre outros. Essas informações ficam armazenadas em um banco de dados para serem acessadas, posteriormente, por alguém que esteja utilizando o aplicativo, principalmente usuários que estão aguardando um ônibus em algum ponto.

A arquitetura do MBTS pode ser observada na Figura 1. Basicamente, consiste de duas aplicações, uma Web e outra Android. A primeira permite o registro de usuários e ônibus e a segunda é utilizada para rastreio. A aplicação Web, desenvolvida em JSP (*JavaServer Pages*), é voltada para o administrador do sistema e também funciona como um *"middleware"* para que a aplicação Android se conecte ao banco de dados e armazene informações (SUJATHA et al., 2014).

Uma outra pesquisa neste área de acompanhamento de ônibus, envolve a utilização de RF (Radio frequência). Foi desenvolvida por Paradells et al. (2014).

Os autores propõe a utilização de uma rede de dados composta por transmissores e receptores RF. Os transmissores situam-se nos ônibus, os quais transmitem informações para os receptores, que situam-se nos pontos de ônibus que, basicamente, são os nós da rede. Assim que um ônibus se aproxima de um ponto de ônibus - nó - inicia a transmissão de informações relevantes, captadas por sensores atrelados ao veículo. Essas informações podem ser usadas/acessadas por um usuário que está aguardando no referido ponto de ônibus.

Uma limitação do sistema baseado em RF, e percebida pelos próprios autores, diz respeito à distância entre o ônibus e os nós da rede (neste caso os pontos de ônibus). Uma vez que sensores baseados em radio frequência possuem limitações de distância e na transmissão dos dados, acabam limitando a velocidade desenvolvida pelo veículo, o que

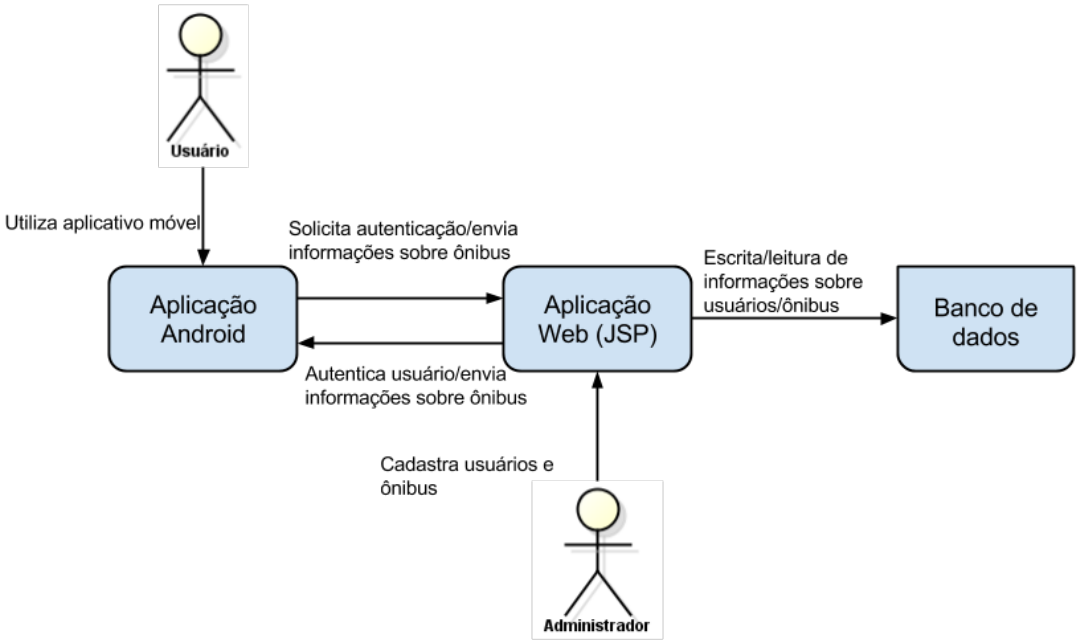

Figura 1: Arquitetura do MBTS.
Fonte: Adaptado de Sujatha et al. (2014).

torna o projeto inviável.

Ainda, as informações só podem ser coletadas por usuários que já estão próximos ao veículo (ou em pontos de ônibus próximos ao veículo). Isso acaba tirando o propósito de acompanhamento ou rastreio de ônibus, pois o mesmo já está próximo do ponto, e pode ser visualizado por um usuário sem o auxílio de um sistema específico para tal.

Alves, Martinez e Viegas (2012) apresentaram um sistema – chamado de *Trip-planner* – para planejar rotas em tempo real, para pessoas que utilizam transporte público em Lisboa. O sistema consegue informar quais são as melhores rotas e o tempo estimado de viagem para um determinado destino, baseados na estimativa de quantos veículos estão trafegando e quais são suas velocidades.

Informações de tempo-real são coletadas através do GPS equipado nos ônibus. Essas informações são utilizadas em um servidor (o que os autores chamam de *Data Center*) para atualizar históricos e melhorar as estimativas, uma vez que é utilizado um algoritmo para predição de tempos de viagem. Esses históricos se referem à relatórios de quatro meses, com informações sobre tempos de viagem e velocidades dos veículos. Essas informações são analisadas e passam por um classificador, e uma vez processadas, são repassadas para qualquer dispositivo móvel conectados em uma rede sem fio, através de

broadcast (ALVES; MARTINEZ; VIEGAS, 2012).

A Figura 2 descreve de forma simplificada o sistema proposto por Alves, Martinez e Viegas (2012). Nota-se claramente que dados históricos e de tempo-real são coletados com o auxílio do GPS instalado no ônibus. Esses dados passam pelo *Data Center* e são utilizados como entrada para um algoritmo de predição. Uma vez processados, distribuem-se os dados para dispositivos móveis através de *broadcast.*

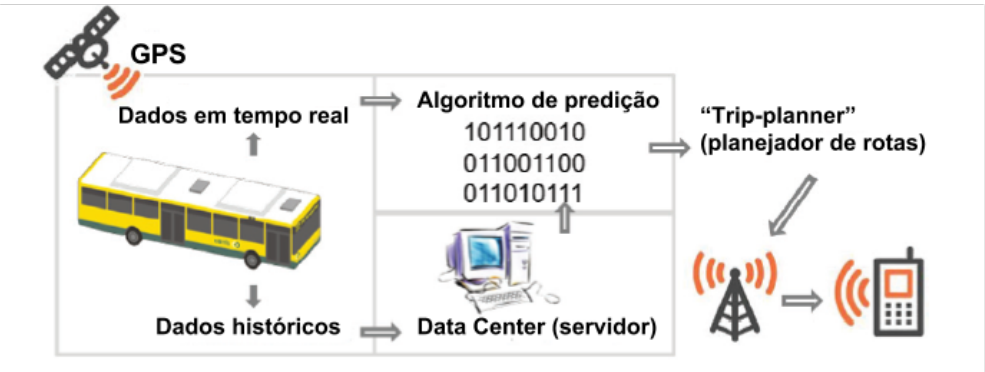

Figura 2: Arquitetura do *Trip-planner.*
Fonte: Adaptado de Alves, Martinez e Viegas (2012).

2.5 DISCUSSÕES

Conforme já discutido na Introdução, a principal motivação para o desenvolvimento deste projeto é implementar um aplicativo de acompanhamento de ônibus para dispositivos móveis - *smartphones* - que utilize uma rede descentralizada.

Espera-se que o aplicativo a ser desenvolvido seja de grande ajuda ao dia a dia dos passageiros do transporte coletivo e que, de alguma, forma traga informações de qualidade aos usuários, bem como lhes ofereça uma maior economia de tempo e torne o processo de decisão, sob qual ônibus tomar ou qual rota seguir, descomplicado.

Em termos de arquiteturas de rede, decidiu-se por utilizar neste projeto uma arquitetura que utilize conceitos tanto de DTN quanto de P2P. Acredita-se que os dois mecanismos em conjunto oferecerão uma arquitetura de rede que resultará em alta disponibilidade e boa conectividade a todos os supostos usuários do aplicativo a ser desenvolvido. A união dessas arquiteturas resulta numa arquitetura de rede descentralizada, que não necessita de infra-estrutura de rede e com capacidade de auto-organização, constituindo um modelo muito interessante para ser utilizada por dispositivos que suportam redes móveis

e sem fio, como *smartphones* e *tablets*.

3 METODOLOGIA

Neste capítulo apresenta-se a metodologia a ser empregada a fim de alcançar os desígnios do projeto. Para tanto, apresenta-se de que forma o desenvolvimento será abordado pela equipe e, na sequência, as tecnologias a serem utilizadas, tais como o ambiente de programação, a linguagem de programação, mecanismos para controle de versão e, por fim, a plataforma de *software*.

3.1 ABORDAGEM DA EQUIPE

Partindo de ideias de aplicativos já existentes para acompanhamento de tráfego (Waze) e acompanhamento de ônibus na cidade de Curitiba (Busão Curitibano), a equipe decidiu reuni-las e aperfeiçoá-las em um aplicativo único. O aperfeiçoamento será resultado do uso de arquiteturas de rede que dispensam centralização, como as Redes Tolerantes a Atrasos, (DTN) se contrapondo à arquitetura cliente-servidor, na qual operam softwares (aplicativos) móveis existentes já citados no início desta seção.

Sendo assim, o uso de uma arquitetura DTN dispensa a necessidade de conectividade contínua para transmissão de informações, uma vez que essa transmissão ocorrerá assim que um usuário, que necessita de informações, esteja próximo de um usuário que porte essas informações.

Tendo em vista a grande quantidade de usuários que estarão utilizando o aplicativo, e o emprego de uma arquitetura descentralizada como as DTNs, nota-se também a importância da arquitetura Ponto-a-ponto (P2P), que permite que um usuário (que pode ser representado aqui por um nó da rede) adquira um papel simultâneo de ”cliente”e ”servidor”. Em outras palavras, um usuário receberá/enviará informações de/para outros usuários, partindo do princípio que todos estejam utilizando o mesmo aplicativo móvel, a ser desenvolvido neste projeto. A descoberta de outros nós nas proximidades, bem como a propagação de informações por esses nós, dar-se-á de forma transparente ao usuário.

3.2 TECNOLOGIAS

A presente seção refere-se às principais tecnologias a serem empregadas no desenvolvimento do projeto, tais como ambientes de desenvolvimento, linguagens de programação, plataforma de *software* e plataforma de *hardware*.

3.2.1 AMBIENTE DE DESENVOLVIMENTO

Pelo fato de o projeto envolver um aplicativo para o sistema operacional Android, é necessário utilizar um ambiente de desenvolvimento específico para essa plataforma.

O *site* oficial para desenvolvimento de aplicativos Android, Android Developers[1], disponibiliza algumas ferramentas para a concepção e teste de aplicativos para a plataforma.

É possível utilizar um *plug-in*, juntamente com o Android SDK, e integrá-los em uma instalação existente da conhecida IDE chamada Eclipse, realizando as devidas configurações manualmente. Uma segunda opção, e que torna a preparação do ambiente de desenvolvimento mais simplificada, é baixar um *bundle* que já inclui o Android SDK, a IDE Eclipse, o *plug-in*, ferramentas de desenvolvimento e uma imagem de sistema para emular um dispositivo Android no *Desktop*. Dessa forma, é possível depurar aplicações Android diretamente no PC, antes de enviá-lo à um dispositivo real. A terceira opção é utilizar uma IDE alternativa chamada *Android Studio*, que atualmente encontra-se em uma versão Beta.

Por experiências dos integrantes com o referido *bundle*, em projetos anteriores, optou-se por utilizar o mesmo no desenvolvimento deste projeto.

Para testes de software será utilizada a emulação do dispositivo Android provida pelo *bundle*, até onde for possível. É dito isso pois o projeto envolve o uso de GPS, Google Maps e compartilhamento de informações entre usuários (*crowdsourcing*), o que não é possível de ser feito apenas com a utilização de um emulador. Dessa forma, será necessário testar a aplicação em um dispositivo real, como um *smartphone* ou um *tablet*, com sistema operacional Android.

[1] O site pode ser acessado em http://developer.android.com/

3.2.2 LINGUAGEM DE PROGRAMAÇÃO

Em relação à linguagem de programação, não há a necessidade de fazer um levantamento dentre as linguagens de programação existentes atualmente e escolher qual será a mais adequada ao projeto. Uma vez que o projeto tem como foco o desenvolvimento de uma aplicação Android, é mandatório que a mesma seja desenvolvida utilizando a linguagem de programação Java juntamente com a API (que contém bibliotecas e ferramentas) incluída no Android SDK. A princípio, não será utilizada nenhuma biblioteca de terceiros (*third-party*).

3.2.3 CONTROLE DE VERSÃO

Será utilizado um sistema de controle de versão (CVS, *Control Version System*) para o versionamento do código-fonte do aplicativo Android. Vários são os sistemas de versionamento de arquivos, dentre os quais os mais comuns são o SVN e Git. Para este último, existem repositórios de código-fonte gratuitos, como o GitHub (https://github.com) e o BitBucket (https://bitbucket.org).

O ideal seria manter um servidor próprio com o SVN ou o Git instalados. No entanto, isso poderia envolver custos adicionais, pois seria necessário deixar uma máquina dedicada para o controle de versão e acessível para todos os integrantes do projeto, além de realizar *backups* periódicos.

O GitHub permite que qualquer pessoa crie uma quantidade ilimitada de repositórios gratuitamente e compartilhe-o com quantos colaboradores for necessário. No entanto, o código-fonte fica disponível publicamente, para qualquer um acessar. Para projetos acadêmicos (como TCCs, teses de mestrado e doutorado), isso pode não ser indicado. O GitHub também oferece repositórios privados, com acesso restrito, mas apenas a partir da assinatura de um plano.

Em contrapartida, o BitBucket oferece uma quantidade ilimitada de repositórios privados gratuitamente, permitindo até cinco colaboradores em um projeto. Como o projeto é constituído de dois integrantes e levantou-se a necessidade de versionar o código-fonte sem disponibilizá-lo publicamente, optou-se por utilizar o BitBucket como repositório de código-fonte e o Git como como sistema de controle de versão.

3.2.4 PLATAFORMA DE *SOFTWARE*

Em relação ao sistema operacional Android, levanta-se outra questão. O Android, como sistema operacional, possui diversas versões desde o seu lançamento. Cada uma dessas versões acaba por incluir novas funcionalidades não apenas para o usuário final, mas também, para os desenvolvedores. É certo que existe a chamada retrocompatibilidade, que permite que a grande maioria das aplicações desenvolvidas para uma versão mais recente do S.O., execute sem maiores problemas em uma versão mais antiga. No entanto, para que isso ocorra, muitas vezes a aplicação fica limitada, pois o desenvolvedor é forçado à desabilitar certos recursos que não são compatíveis, de forma alguma, com versões mais antigas da plataforma.

No momento da escrita deste documento, a versão mais recente do Android é a versão 4.4 (KitKat). A maioria dos dispositivos disponíveis no mercado – *smartphones* e *tablets* – possuem uma versão do Android 4.0+ instalada. A mais recente versão do S.O., a versão 5.0 (Lollipop), está prestes a ser lançada, e apenas alguns poucos dispositivos já existentes no mercado receberão a atualização para o novo sistema.

A Tabela 1 fornece informações referentes ao *market share* das versões do Android, desde a 2.2 (Froyo) até a 4.4 (KitKat). Os dados foram coletados através da aplicação Google Play Store em um período de sete dias, finalizando no dia 03 de novembro de 2014. É possível verificar o percentual de quantos dispositivos executam uma determinada versão do sistema operacional. Essas informações são úteis para que o desenvolvedor determine qual versão da plataforma terá como "alvo".

Tabela 1: *Market Share* das versões do Android. Fonte: (Android Developers, 2014b)

Versão	**Codinome**	**API**	**Distribuição**
2.2	Froyo	8	0,6%
2.3.3 - 2.3.7	Gingerbread	10	9,7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	8,5%
4.1.x	Jelly Bean	16	22,8%
4.2.x	Jelly Bean	17	20,8%
4.3	Jelly Bean	18	7,3%
4.4	KitKat	19	30,2%

Observa-se na tabela que cerca de 10,4% usuários usa um dispositivo com uma versão antiga da plataforma, de codinome Froyo (2.2) ou Gingerbread (2.3.3 - 2.3.7). Uma parcela ainda menor utiliza as versões Ice Cream Sandwich (4.0.3 - 4.0.4), 8,5%. Grande parte dos usuários – cerca de 50% – utiliza o Android Jelly Bean (4.1.x - 4.3) e 30,2%

utiliza a última versão lançada, KitKat (4.4).

Portanto, devido à popularidade da versão Jelly Bean (4.1.x - 4.3) do sistema operacional Android, o desenvolvimento do projeto terá como foco dispositivos que executam, no mínimo, esta versão da plataforma.

3.2.5 PLATAFORMA DE *HARDWARE*

Para o projeto em questão não será necessária a utilização de uma plataforma de *hardware*, pois a aplicação executará diretamente sobre um dispositivo móvel, tal como um *smartphone* ou um *tablet*.

3.3 ETAPAS DE DESENVOLVIMENTO

Inicialmente será feito um estudo das APIs disponíveis no Android SDK, e um levantamento para detectar se há a necessidade de utilizar APIs de terceiros (*third-party*).

Com os diagramas UML já projetados, será preparado o ambiente de desenvolvimento, realizando-se o *download* do *bundle*, instalação de *drivers* para depuração em um dispositivo móvel real via cabo USB e configurações necessárias. Na sequência, dar-se-á início à implementação da aplicação.

Inicialmente, os testes serão realizados no emulador Android, disponibilizado pelo *bundle*, conforme citado na seção anterior. Testes mais complexos serão feitos em um dispositivo real (*smartphone* e/ou *tablet*) que execute o sistema operacional Android com versão 4.1.x, pelo menos.

4 RECURSOS DE *HARDWARE* E *SOFTWARE*

O presente capítulo menciona os recursos de *hardware* e *software* necessários para o desenvolvimento do projeto.

4.1 RECURSOS DE *HARDWARE*

Uma vez que o projeto diz respeito à uma aplicação Android que executará sobre um dispositivo móvel (*smartphones* e *tablets*), não é necessário a compra e/ou o uso de componentes de *hardware* (microcontroladores, circuitos integrados, fontes de alimentação, bateria, entre outros) para o desenvolvimento do projeto.

4.2 RECURSOS DE *SOFTWARE*

Os recursos de *software* englobam ambientes de desenvolvimento, linguagem de programação e plataformas de software.

Conforme citado no capítulo anterior, para a programação e teste da aplicação Android será utilizado o ambiente de desenvolvimento **Eclipse ADT**, que nada mais é que um *bundle* (pacote) que inclui diversos componentes necessários para a concepção do aplicação. Segundo o site oficial para desenvolvimento de aplicações Android, Android Developers (2014a), os componentes inclusos são os seguintes:

- Ambiente de desenvolvimento Eclipse + *plugin* ADT (*Android Developer Tools*);
- Android SDK (*Software Development Kit*);
- Ferramentas adicionais para a plataforma Android;
- Uma versão da plataforma Android (geralmente a mais recente);
- Uma versão da imagem do sistema Android para o emulador.

É possível realizar o *download* do *bundle* gratuitamente no Android Developers, no seguinte link: <http://developer.android.com/sdk/index.html>. Há versões disponíveis para Windows e Linux, tanto 32-bit quanto 64-bit. Para Mac OS X, apenas a versão 64-bit encontra-se disponível.

Segundo Sims (2014), a linguagem de programação "oficial" para o desenvolvimento de aplicações Android é Java. Grande parte das aplicações são escritas em Java e suas API (*Application Programming Interface*) são projetadas para serem chamadas primeiramente por Java. É possível desenvolver utilizando linguagens como C e C++, mas isso é algo que a própria Google, desenvolvedora do Android, não incentiva (SIMS, 2014).

Pode-se fazer o *download* de várias versões do JDK (*Java Development Kit*) gratuitamente a partir do site oficial da Oracle: <http://oracle.com>. O JDK é necessário para a compilação de aplicativos na linguagem Java e por consequência, o Eclipse ADT irá utilizá-lo juntamente com o Android SDK para compilação da aplicação Android.

Durante todo o desenvolvimento do projeto, será utilizado o Git como sistema de controle de versão, conforme citado no capítulo anterior. Será criado um repositório privado no BitBucket (<https://bitbucket.org>) para código-fonte da aplicação Android. Outro repositório foi criado no GitHub (<https://github.com>) para o "versionamento" deste documento, que está sendo escrita em LaTeX. A criação dos repositórios, tanto no BitBucket quanto no GitHub, é gratuita.

Por fim, a plataforma de *software* sobre a qual executará a aplicação Android, é o próprio sistema operacional Android, disponível em diversos *smartphones* e *tablets* existentes no mercado. Conforme estudo apresentado no capítulo anterior, o desenvolvimento da aplicação Android terá como foco sistemas operacionais que executam, no mínimo, a versão 4.1.x (Jelly Bean).

5 VIABILIDADE E CRONOGRAMA PRELIMINAR

Neste capítulo serão apresentadas a viabilidade técnica e financeira do projeto, bem como o cronograma preliminar para o desenvolvimento do mesmo.

5.1 VIABILIDADE

Uma vez que o projeto envolve programação em Java, é altamente viável tecnicamente, pois ambos os integrantes são familiarizados com esta linguagem de programação. Além disso, a aplicação derivada do projeto executará em *smartphones* e *tablets* com sistema operacional Android, muito popular e disseminado entre a população nos dias de hoje. Os próprios integrantes do projeto possuem tais dispositivos, o que facilita ainda mais o teste e a constatação da aplicação prática do projeto.

Além disso, o projeto é viável financeiramente. Não será necessária a compra de equipamentos e, uma vez que o projeto envolverá apenas *software*, com a utilização de ferramentas gratuitas ou livres – conforme já exposto neste documento – os custos financeiros serão mínimos ou nulos.

5.2 CRONOGRAMA PRELIMINAR

A Tabela 2 fornece informações acerca das etapas necessárias para o desenvolvimento do projeto, bem como os seus tempos de duração estimados.

Tabela 2: Cronograma preliminar. Fonte: Autoria própria.

Número	**Etapa**	**Dias**
1	Estudo de APIs e do Android SDK	10
2	Projeto de software (modelagem com UML)	10
3	Implementação do projeto de software	30
4	Testes do protótipo utilizando um dispositivo móvel – emulado – com sistema operacional Android	7
5	Testes do protótipo utilizando um dispositivo móvel – real – com sistema operacional Android	7
6	Finalização do artefato	14
7	Finalização da documentação	14
	Total	92

6 CONTEXTO

O Contexto do projeto se aplica à cidade de Curitiba, em especial aos usuários de transporte público da cidade, haja vista que o aplicativo visa informar, com a maior precisão possível, a localização dos automóveis que operam em uma determinada linha de ônibus.

Apesar do caráter altruísta do projeto, o mesmo desenvolver-se-á sem qualquer ligação externa aos elementos do grupo ou fora do ambiente da UTFPR. Logo, é um aplicativo de propósito independente, mas com foco voltado à população em geral.

Como consequência, a alocação de recurso destinados ao progresso do projeto será de responsabilidade de seus desenvolvedores, exclusivamente, assim como a responsabilidade por sua construção.

7 ESPECIFICAÇÃO: PROJETO DE SOFTWARE

O presente capítulo apresenta os requisitos funcionais, não-funcionais e os diagramas de caso de uso para o projeto em questão.

7.1 REQUISITOS FUNCIONAIS

Nesta seção serão apresentados os requisitos funcionais do aplicativo móvel.

RF1 O aplicativo móvel deverá ser executado em um *smartphone* ou *tablet* com sistema operacional Android.

RF2 O aplicativo móvel deverá coletar as coordenadas do dispositivo no qual está sendo executado.

RF3 O aplicativo móvel deverá apresentar ao usuário uma tela com as opções para acompanhamento de ônibus.

RF4 O aplicativo móvel deverá oferecer um recurso para que o usuário submeta *feedbacks* sobre uma determinada linha de ônibus.

RF4.1 Os *feedbacks* deverão estar pré-definidos no aplicativo.

RF4.2 Os *feedbacks* devem conter informações relevantes sobre um determinado ônibus, como por exemplo, se o mesmo está lotado, se atrasou, ou se houve algum acidente no seu percurso.

RF5 O aplicativo móvel deverá armazenar as informações sobre uma determinada linha de ônibus.

RF5.1 As informações deverão ficar armazenadas por, no máximo, 10 minutos.

RF5.2 As informações podem ser provenientes do próprio usuário/dispositivo móvel ou de outros usuários/dispositivos móveis.

RF6 O aplicativo móvel deverá procurar por usuários/dispositivos próximos que estejam utilizando o aplicativo.

RF6.1 Os dispositivos móveis devem estar com Wi-Fi ligado e executando o aplicativo móvel.

RF7 O aplicativo móvel deverá encaminhar suas informações armazenadas para quaisquer outros dispositivos móveis (*smartphones* ou *tablets*) encontrados.

RF7.1 Os dispositivos móveis devem estar com Wi-Fi ligado e executando o aplicativo móvel.

RF8 O aplicativo móvel deverá oferecer ao usuário um recurso para mostrar um mapa informando a posição de um ônibus.

7.2 REQUISITOS NÃO-FUNCIONAIS

Nesta seção serão apresentados os requisitos não-funcionais do aplicativo móvel.

RNF1 O aplicativo móvel deverá executar sobre o sistema operacional Android versão 4.1 (Jelly Bean), no mínimo.

RNF2 O aplicativo móvel deverá utilizar a tecnologia Wi-Fi para seu correto funcionamento.

RNF3 O aplicativo móvel deverá utilizar uma arquitetura de rede descentralizada, integrando conceitos de DTN e P2P.

RNF4 O aplicativo móvel deverá ser desenvolvido na linguagem Java.

RNF5 O aplicativo móvel deverá ser desenvolvido utilizando o conjunto de ferramentas do Android SDK.

RNF6 O aplicativo móvel deverá ser desenvolvido utilizando o ambiente de desenvolvimento Android Studio.

7.3 DIAGRAMAS DE CASOS DE USO

Esta seção apresenta o Diagrama de Casos de Uso para o aplicativo móvel a ser desenvolvido, bem como a descrição detalhada de cada caso. O diagrama pode ser observado na Figura 3.

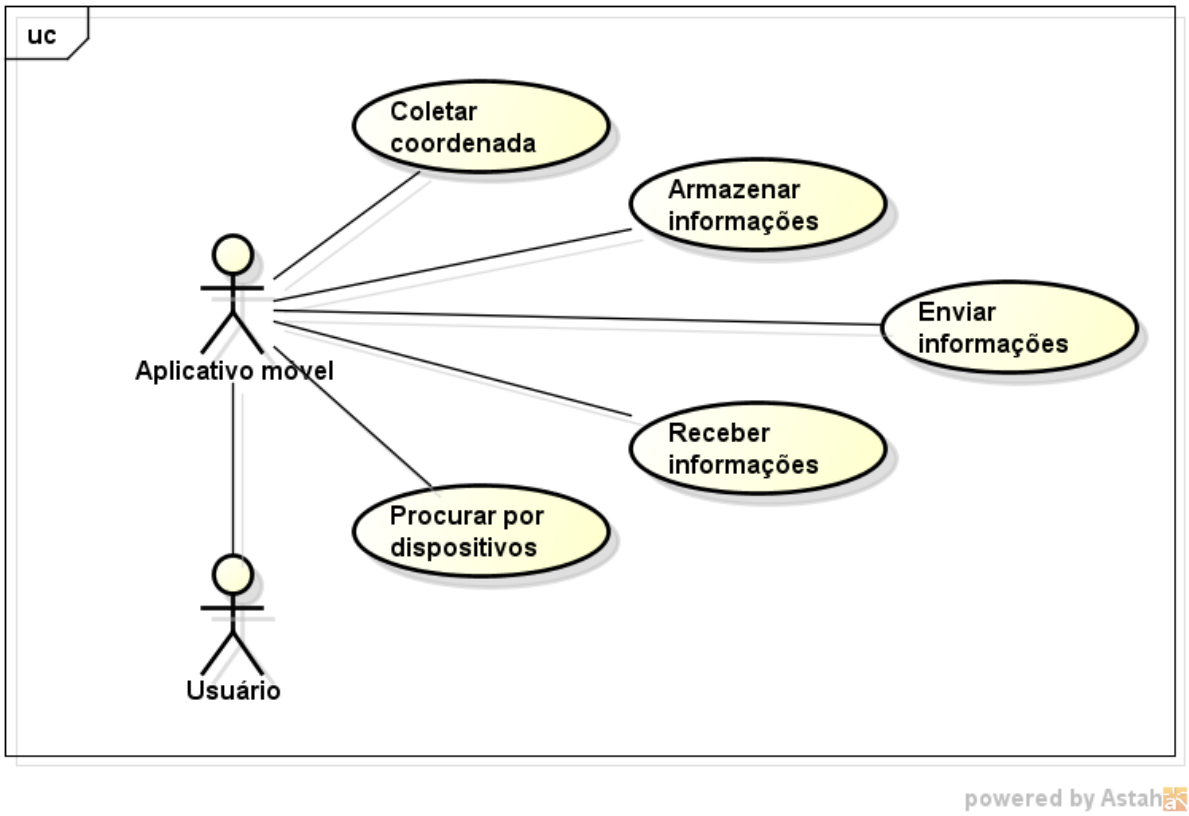

Figura 3: Diagrama de Casos de Uso.
Fonte: Autoria própria.

7.3.1 COLETAR COORDENADA

Nome: Coletar coordenada
Ator principal:Aplicativo móvel
Ator de bastidor: N/A
Descrição: quando se desejar conhecer a localização em que se encontra o dispositivo móvel, o GPS embutido no *smartphone* será acionado para que as coordenadas sejam coletadas.

Pré-condições:

- O dispositivo móvel possui um sistema de GPS.

Pós-condições:

- O GPS retorna a localização em que se encontra o dispositivo.

Fluxo Básico:

1. O Aplicativo móvel coleta a localização atual do dispositivo móvel, através do GPS integrado.
2. As coordenadas obtidas pelo GPS são armazenadas na memória do *smartphone*.

Fluxo Alternativo:

- O Aplicativo móvel não consegue acionar o dispositivo de GPS: uma mensagem de erro é apresentada na tela do aparelho.
- O GPS é acionado, mas não é possível obter as coordenadas em que se encontra o dispositivo: uma mensagem de erro é apresentada na tela do aparelho.
- Não há espaço suficiente para armazenar as novas coordenadas obtidas do GPS: uma mensagem de erro é apresentada na tela do aparelho.

Regras de Negócio: nenhuma.

7.3.2 ARMAZENAR INFORMAÇÕES

Nome: Armazenar informações
Ator principal: Aplicativo móvel
Ator de bastidor: Usuário
Descrição: este caso de uso ocorre quando o usuário submete alguma informação referente à algum ônibus, quando as coordenadas são coletadas ou quando informações são recebidas de outros dispositivos.

Pré-condições:

- Necessário haver informações para armazenamento, sejam elas inseridas pelo próprio usuário, coletadas pelo GPS integrado ou recebidas de outros dispositivos.

Pós-condições:

- Informação é armazenada na memória do dispositivo, por 10 minutos.

Fluxo Básico:

1. Após o Usuário submeter alguma informação, o aplicativo móvel armazena-a em memória por 10 minutos.

Fluxo Alternativo I:

1. Informações são recebidas de outros dispositivos.
2. Essas informações são armazenadas em memória por 10 minutos.

Fluxo Alternativo II:

1. A coordenada do dispositivo foi coletada.
2. Armazena a coordenada em memória, por 10 minutos.

Regras de Negócio: nenhuma.

7.3.3 ENVIAR INFORMAÇÕES

Nome: Enviar informações
Ator principal: Aplicativo móvel
Ator de bastidor: Usuário
Descrição: este caso de uso ocorre quando há informações armazenadas no dispositivo, prontas para serem enviadas.

Pré-condições:

- Necessário haver informações para envio, sejam elas inseridas pelo próprio usuário, coletadas pelo GPS integrado ou recebidas de outros dispositivos.

- Outros dispositivos móveis nas proximidades devem estar conectados e executando a aplicação.

Pós-condições:

- Informações são enviadas para outros dispositivos móveis próximos que estejam conectados.

Fluxo Básico:

1. O Usuário decide notificar demais usuários sobre alguma informação relevante.
2. O Usuário seleciona a informação no menu com opções pré-definidas.
3. O Usuário confirma determinada informação para o ônibus em específico.
4. O Aplicativo móvel envia a informação para quaisquer outros dispositivos móveis próximos que estejam conectados.
5. A informação fica armazenada em memória por 10 minutos.

Fluxo Alternativo I:

- Informações são recebidas de outros dispositivos móveis.
- As informações são repassadas para demais dispositivos móveis próximos que estejam conectados.
- Essas informações ficam armazenadas em memória por 10 minutos.

Fluxo Alternativo II:

1. A coordenada do dispositivo foi coletada.
2. A coordenada é repassada para demais dispositivos móveis próximos que estejam conectados.
3. A coordenada fica armazenada em memória por 10 minutos.

Regras de Negócio: nenhuma.

7.3.4 RECEBER INFORMAÇÕES

Nome: Receber informações
Ator principal: Aplicativo móvel
Ator de bastidor: N/A
Descrição: este caso de uso ocorre quando o aplicativo móvel recebe informações/mensagens de outros dispositivos.

Pré-condições:

- As mensagens recebidas devem estar construídas de acordo com um protocolo pré-configurado no dispositivo.

Pós-condições:

- As informações/mensagens são armazenadas na memória por 10 minutos.

Fluxo Básico:

1. O aplicativo móvel recebe uma informação/mensagem de outro dispositivo móvel.
2. O aplicativo móvel verifica se reconhece o padrão da mensagem recebida.
3. O aplicativo móvel armazena a nova mensagem na memória de mensagens recebidas (para consulta local).
4. O aplicativo móvel armazena a nova mensagem na memória de mensagens a enviar (para propagar a mensagem a outros dispositivos).

Fluxo Alternativo:

- A mensagem recebida não possui os padrões esperados e é automaticamente descartada.
- A memória destinada a alocação de mensagens recebidas está cheia: uma mensagem de notificação é mostrada na tela do dispositivo.

Regras de Negócio: nenhuma.

7.3.5 PROCURAR POR DISPOSITIVOS

Nome: Procurar por dispositivos
Ator principal: Aplicativo móvel
Ator de bastidor: Usuário
Descrição: este caso de uso ocorre periodicamente durante a execução do aplicativo móvel.

Pré-condições:

- O aplicativo móvel deve estar executando.
- O Wi-Fi no dispositivo móvel deve estar ligado.

Pós-condições:

- Uma conexão é estabelecida com outros dispositivos móveis que estejam próximos e executado o aplicativo.

Fluxo Básico:

1. O Usuário liga a conexão Wi-Fi no dispositivo móvel.
2. O Usuário inicia o Aplicativo móvel.
3. O Aplicativo móvel procura por outros dispositivos móveis nas proximidades.
4. Se um dispositivo é encontrado, inclui o mesmo na lista de dispositivos encontrados e estabelece conexão.
5. O Aplicativo móvel aguarda dois minutos.
6. Realiza nova busca, retornando ao passo 3.

Fluxo Alternativo:

- O usuário pressiona o botão para procurar por dispositivos imediatamente.
- Vai para o passo 3 do Fluxo Básico.

Regras de Negócio: nenhuma.

8 GESTÃO E PLANEJAMENTO

O total de horas estimadas para desenvolvimento do projeto foi calculada através do método de *Use Case Points*, que baseia-se em informações extraídas diretamente dos diagramas de Casos de Uso especificados.

O presente capítulo mostra todos os passos desenvolvimentos para o cálculo dos *Use Case Points*, bem como a utilização dos métodos PERT (*Program Evaluation and Review Technique*) e CPM (*Critical Path Method*) para elaboração da Rede PERT-CPM, pela qual é possível observar o caminho crítico para o desenvolvimento do projeto, usando como base as tarefas e suas horas estipuladas.

8.1 *USE CASE POINTS*

A estimativa por *Use Case Points* consiste em alguns passos. Clemmons (2006) exemplificou os passos para o cálculo do *Use Case Points*, os quais serão seguidos neste capítulo.

O primeiro passo é calcular o UAW (*Unadjusted Actor Weight*), o peso total dos atores do sistema. Os atores foram classificados como: ator complexo (Usuário), com peso 3 e ator médio (Aplicativo móvel), com peso 2.

$$\text{UAW} = 3\text{x}1 + 2\text{x}1 = 5$$

Em seguida, calcula-se o peso não-ajustado de cada Caso de Uso do sistema. Os casos de uso foram classificados da seguinte forma:

- Coletar coordenada: caso de uso médio (peso 10).
- Armazenar informações: caso de uso médio (peso 10).
- Enviar informações: caso de uso complexo (peso 15).

- Receber informações: caso de uso médio (peso 10).
- Procurar por dispositivos: caso de uso médio (peso 10).

O UUCW (*Unadjusted Use Case Weight*) é então calculado:

$$\text{UUCW} = 10\text{x}3 + 15\text{x}2 = 60$$

No terceiro passo, calcula-se o UUCP (*Unadjusted Use Case Point*), que consiste na soma de UAW com UUCW:

$$\text{UUCP} = \text{UAW} + \text{UUCW} = 65$$

Prossegue-se então para o cálculo de fatores técnicos, que cobre requisitos funcionais do sistema, e fatores de ambiente, que cobre requisitos não-funcionais associados ao processo de desenvolvimento.

A variável de fatores técnicos, TFactor, é obtida através do somatório dos níves F1 a F13, multiplicados pelo seu peso:

Tabela 3: Fatores técnicos. Fonte: Autoria própria.

Fator	**Fatores que contribuem para a complexidade**	**Peso**	**Valor**	**Total**
F1	Sistemas distribuídos	2	5	10
F2	Tempo de resposta	1	5	5
F3	Eficiência para o usuário final (on-line)	1	5	5
F4	Processamento interno complexo	1	4	4
F5	Código reusável	1	2	2
F6	Facilidade de instalação	0,5	1	0,5
F7	Facilidade de uso (facilidade operacional)	0,5	4	2
F8	Portabilidade	2	3	6
F9	Facilidade de mudança	1	3	3
F10	Concorrência (acesso simultâneo à aplicação)	1	4	4
F11	Recursos de segurança	1	3	3
F12	Fornece acesso direto para terceiros	1	0	0
F13	Requer treinamento especial para o usuário	1	0	0
TFactor				44,5

O TCF (*Technical Complexity Factor*) é então calculado:

$$\text{TCF} = 0{,}6 + (0{,}01 \text{ x } \text{TFactor}) = 0{,}6 + (0{,}01 \text{ x } 44{,}5) = 1{,}045$$

A variável de fatores de ambiente, EFactor, é calculada através do somatório dos níveis F1 a F8, multiplicados pelo seu peso:

Tabela 4: Fatores de ambiente. Fonte: Autoria própria.

Fator	**Fatores que contribuem para a complexidade**	**Peso**	**Valor**	**Total**
F1	Familiaridade da equipe com o processo formal de desenvolvimento adotado	1,5	4	6
F2	Colaboradores de meio período	-1	2	-2
F3	Capacidade do líder de projeto em análise de requisitos e modelagem	0,5	4	2
F4	Experiência da equipe em desenvolvimento de aplicações do gênero em questão	0,5	5	2,5
F5	Experiência em orientação a objetos	1	5	5
F6	Motivação da equipe	1	5	5
F7	Dificuldades com a linguagem de programação	-1	3	-3
F8	Requisitos estáveis	2	3	6
EFactor				21,5

O EF (*Environmental Factor*) é então calculado:

$$\text{EF} = 1{,}4 + (-0{,}03 \times \text{EFactor}) = 1{,}4 + (-0{,}03 \times 21{,}5) = 0{,}755$$

Com os resultados obtidos nos passos anteriores, O valor de UCP (*Use Case Points*) pode ser então calculado:

$$\text{UCP} = \text{UUCP} \times \text{TCF} \times \text{EF} = 51{,}28$$

O tempo de trabalho estimado pode ser obtido multiplicando-se UCP por 20. Portanto, estimou-se 1025,6 horas de trabalho.

8.2 GERENCIAMENTO DE TEMPO

Para a elaboração da Rede PERT-CPM foi necessário, primeiramente, levantar as tarefas do projeto em questão. As tarefas levantadas, bem como a duração de cada uma delas podem ser observadas na Tabela 5:

Tabela 5: Tarefas do projeto. Fonte: Autoria própria.

Número	Tarefa	Duração
1	Estudo de APIs e do Android SDK	100
2	Projeto de software (modelagem UML)	100
3	Implementação do projeto de software	280
4	Testes do protótipo utilizando um dispositivo móvel (emulado) com sistema operacional Android	60
5	Testes do protótipo utilizando um dispositivo móvel (real) com sistema operacional Android	100
6	Finalização do artefato	192
7	Testes e ajustes finais no software	104
8	Finalização da documentação	80
9	Preparação da apresentação para banca	8
Total		1024

Utilizando-se dos resultados expostos, a Rede PERT-CPM para o projeto é mostrada na Figura 4.

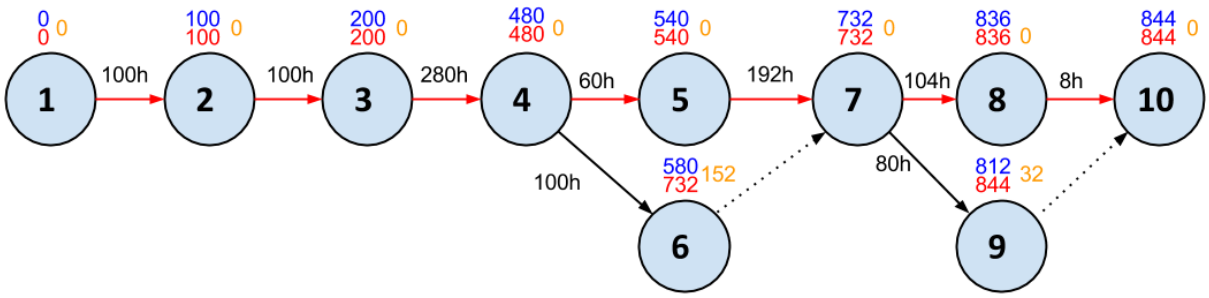

Figura 4: Rede PERT-CPM.
Fonte: Autoria própria.

Na figura, o "cedo"de cada evento é evidenciado na cor azul, ao passo que o "tarde"de cada evento é evidenciado na cor vermelha. A folga, diferença entre tarde e cedo, é apresentada na cor laranja/amarelo.

O caminho crítico fica destacado pelas setas em vermelho, e perpassa pelos eventos 1, 2, 3, 4, 5, 7, 8 e 10 e, por consequência, pelas tarefas 1, 2, 3, 4, 6, 7 e 9.

9 ANÁLISE DE RISCOS

O presente capítulo apresenta a análise de riscos elaborada para o projeto. A análise consiste, basicamente, em seis riscos, que são:

1. Escolha de tecnologias que não atendam às necessidades do projeto.
2. Dificuldades no desenvolvimento do software.
3. Descumprimento do prazo estabelecido para a realização de alguma etapa prevista no cronograma do projeto.
4. Definição imprecisa do escopo do projeto.
5. Desistência de um membro da equipe.
6. Inexistência de tempo para a conclusão do projeto.

Na sequência serão apresentadas a descrição, avaliação do risco, impacto, probabilidade e estratégia de ação correspondentes a cada um dos riscos levantados. A avaliação do risco e a probabilidade são classificadas na seguinte escala: baixa, baixa-média, média, média-alta e alta.

Tabela 6: Análise do risco 1. Fonte: Autoria própria.

Risco	Escolha de tecnologias que não atendam às necessidades do projeto.
Descrição	O projeto envolve o uso de várias tecnologias, portanto, há a possibilidade de escolha equivocada de alguma delas o que pode vir a não satisfazer os requisitos mínimos necessários.
Avaliação do risco	Médio-Alto.
Impacto	Caso o emprego de uma tecnologia seja incapaz de suprir as necessidades mínimas requeridas, a etapa que faz uso deste não poderá ser finalizada.
Probabilidade	Média.
Estratégia de ação	As tecnologias que serão utilizados terão seus requisitos detalhadamente avaliados para que o grupo possa eliminar escolhas equivocadas.

Tabela 7: Análise do risco 2. Fonte: Autoria própria.

Risco	Dificuldades no desenvolvimento do software.
Descrição	Apesar de um médio conhecimento das tecnologias, corre-se o risco de empecilhos aparecem mesmo com esse conhecimento prévio. Deve-se observar ainda que o trabalho propõem-se a desenvolver uma nova arquitetura de rede que poderá se mostrar impraticável de implementação.
Avaliação do risco	Alto.
Impacto	Sendo o desenvolvimento do aplicativo o cerne do projeto, problemas com o mesmo terão que ser cuidadosamente analisados e solucionados para não inviabilizarem o projeto.
Probabilidade	Média-Alta.
Estratégia de ação	Antes de se iniciar a codificação, a equipe fará a modelagem do software para que se tenha claramente definido o comportamento do mesmo, facilitando testes e depuração de código.

Tabela 8: Análise do risco 3. Fonte: Autoria própria.

Risco	Descumprimento do prazo estabelecido para a realização de alguma etapa prevista no cronograma do projeto.
Descrição	O projeto será dividido em etapas e estas serão temporizadas de acordo com o seu grau de dificuldade para então designar-se um membro da equipe para realizá-la. Caso este membro não a conclua no prazo estipulado, etapas dependentes desta terão que ser adiadas, prejudicando o projeto como um todo.
Avaliação do risco	Alto.
Impacto	Embora algumas tarefas possam ser realizadas paralelamente, existem aquelas que são dependentes de outras e, caso ocorra atraso em uma, a tarefa dependente terá que esperar até o cumprimento da outra, o que por sua vez pode atrasar outras tarefas, ocasionando um efeito em cadeia que pode prejudicar fortemente o desenvolvimento do projeto.
Probabilidade	Média.
Estratégia de ação	Os membros da equipe terão reuniões semanais para verificar o progresso de cada etapa definida, pretende-se realizar as tarefas em grupo e dessa forma, facilitar o entendimento de uma tarefa que não esteja com um progresso adequado.

Tabela 9: Análise do risco 4. Fonte: Autoria própria.

Risco	Definição imprecisa do escopo do projeto.
Descrição	Uma definição equivocada do escopo da proposta pode trazer grandes dificuldades no desenvolvimento do projeto, culminando na entrega incompleta do mesmo.
Avaliação do risco	Médio-Alto.
Impacto	Como em todo planejamento de projeto, procura-se deixar o escopo do mesmo o mais claro possível para evitar o não cumprimento dos objetivos. Dessa forma, o cumprimento das metas é de fundamental importância seja em ambiente acadêmico ou empresarial. Logo, a descoberta de uma indefinição nesse aspecto colocaria em risco todo o planejamento e execução das tarefas.
Probabilidade	Média.
Estratégia de ação	Durante o planejamento do projeto, buscou-se definir seu escopo de tal forma que os membros da equipe já identificassem as principais áreas técnicas para o seu desenvolvimento. Assim, evita-se dispêndio de tempo para adquirir conhecimentos não pertinentes ao projeto.

Tabela 10: Análise do risco 5. Fonte: Autoria própria.

Risco	Desistência de um membro da equipe.
Descrição	Ao longo do projeto, há a possibilidade de que algum membro da equipe não se identifique mais com o projeto e queira abandoná-lo ou que seja obrigado, por motivos de força maior, a se afastar do projeto.
Avaliação do risco	Alto.
Impacto	Com a redução de integrantes da equipe, haveria uma sobrecarga sobre o membro remanescente.
Probabilidade	Baixa-Média.
Estratégia de ação	Na fase inicial do projeto todos os membros foram instigados à expor suas ideias de tal forma que o projeto como um todo tivesse a participação da dupla que o compõe, desta forma, aumentando a motivação individual dos integrantes para a completa realização do projeto. Todavia, se de fato essa situação ocorrer haverá a necessidade de uma reestruturação do plano de projeto.

Tabela 11: Análise do risco 6. Fonte: Autoria própria.

Risco	Inexistência de tempo para a conclusão do projeto.
Descrição	Como os integrantes da equipe tem que fazer outas atividades em paralelo, pode faltar tempo para terminar o projeto em sua plenitude.
Avaliação do risco	Alto.
Impacto	Caso haja escassez de tempo será inviável o término do projeto dentro do prazo estipulado, ocasionado atraso no dia de sua apresentação.
Probabilidade	Baixa-Média.
Estratégia de ação	Estimar prazos para cada tarefa, e se necessário, reduzir os requisitos finais, diminuindo assim o tempo do projeto.

10 CONCLUSÃO

Sabe-se que transporte coletivo é um tema em voga na maioria das cidades brasileiras, e também é uma área na qual aloca-se grande quantidade de recursos. Em especial, na cidade de Curitiba, tida por muitos anos como possuidora do melhor transporte público do país, há uma preocupação grande nesse aspecto.

Ao longo do ano é possível visualizar por toda cidade, campanhas publicitárias voltadas para esse propósito, tendo em vista conscientizar o usuário do transporte coletivo acerca de alguns comportamentos, e também, tentando melhorar a qualidade do serviço prestado. Verifica-se também que as tentativas de melhora do transporte coletivo não partem apenas da prefeitura da cidade, como documentou-se nesse exposto, um grupo de desenvolvedores buscou materializar um aplicativo que mostrasse, em tempo real, a localização de cada ônibus em uma(s) determinada(s) linha(s) de interesse. No entanto, a aplicação apresentava um funcionamento um tanto instável, já que dependia dos serviços de um servidor da URBS para seu funcionamento correto.

Numa tratativa que visa resolver esse problema, elaborou-se um projeto com intuito de elaborar um aplicativo que funcione sob uma arquitetura *ad-hoc* e utilize recursos disponíveis em um *smartphone*, e portanto, presente no dia a dia da maioria dos usuários de ônibus na cidade. Entretanto, na acepção da proposta de projeto, encontrou-se certa dificuldade em definir o que o aplicativo ia fazer, quais seriam os recursos existentes no programa e principalmente, a forma de propagar a informação que faria o sistema, como um todo, funcionar.

A escolha para a arquitetura de rede foi pensada de modo a aliar conceitos de DTN e P2P. A priori, a ideia apresentou-se confusa, já que as duas abstrações parecem paradoxais, todavia, existem registros na literatura de algoritmos que funcionam com essas características. Passados esses questionamentos iniciais o projeto progrediu de forma tranquila. A equipe trabalhou em conjunto quase na totalidade do tempo, o que facilitou a tomada de decisões e resolução de conflitos.

No que tange projetos futuros, um enfoque interessante que poderia ser discutido apoia-se num mecanismo que garanta algum tipo de redundância na rede de comunicação. De certa forma, isso suportaria um prisma egrégio do projeto que é a atualização das informações em tempo real, ou seja, se uma abordagem falhar, existiria outra para garantir a propagação dos dados em um tempo exequível. Outra singularidade consiste em tornar acessíveis aos usuários, todas as linhas de ônibus cadastradas na URBS, sejam elas metropolitanas ou urbanas, de forma que haja cobertura total de informações em toda a cidade.

Por fim, busca-se com esse projeto fornecer um mecanismo de fácil manipulação e que traga benefícios à população em geral.

REFERÊNCIAS

ALVES, D.; MARTINEZ, L. M.; VIEGAS, J. M. Retrieving real-time information to users in public transport networks: An application to the lisbon bus system. *Procedia - Social and Behavioral Sciences*, v. 54, n. 0, p. 470 – 482, 2012. ISSN 1877-0428. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877042812042279>.

Android Developers. *Android SDK*. 2014. Disponível em: <http://developer.android.com/sdk/index.html>. Acesso em: 21 de novembro de 2014.

Android Developers. *Dashboards*. 2014. Disponível em: <https://developer.android.com/about/dashboards/index.html>. Acesso em: 09 de novembro de 2014.

CACCIAPUOTI, A. S.; CALEFFI, M.; PAURA, L. Mobile p2p: peer-to-peer systems over delay tolerant networks. *Delay Tolerant Networks: Protocols and Applications*, p. 1–35, 2011.

CLEMMONS, R. K. *Project Estimation with Use Case Points*. 2006. Disponível em: <http://www.crosstalkonline.org/storage/issue-archives/2006/200602/200602-Clemmons.pdf>. Acesso em: 03 de março de 2015.

HOWE, J. The Rise of Crowdsourcing. *Wired Magazine*, v. 14, n. 6, 06 2006. Disponível em: <http://www.wired.com/wired/archive/14.06/crowds.html>.

KANHERE, S. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In: *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*. [S.l.: s.n.], 2011. v. 2, p. 3–6.

PAN, G. et al. Trace analysis and mining for smart cities: issues, methods, and applications. *Communications Magazine, IEEE*, v. 51, n. 6, p. 120–126, 2013. ISSN 0163-6804.

PARADELLS, J. et al. Infrastructureless smart cities. use cases and performance. In: *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*. [S.l.: s.n.], 2014. p. 1–6.

SCHUURMAN, D. et al. Smart Ideas for Smart Cities: Investigating Crowdsourcing for Generating and Selecting Ideas for ICT Innovation in a City Context. *JTAER*, v. 7, n. 3, 2012. Disponível em: <http://dblp.uni-trier.de/db/journals/jtaer/jtaer7.html\#SchuurmanBMM12>.

SIMS, G. *I want to develop Android Apps - What language should I learn?* 2014. Disponível em: <http://www.androidauthority.com/want-develop-android-apps-languages-learn-391008/>. Acesso em: 21 de novembro de 2014.

SUJATHA, K. et al. Design and Development of Android Mobile based Bus Tracking System. *First International Conference on Networks & Soft Computing (ICNSC)*, p. 231–235, 2014.

VENDRAMIN, A. C. B. K. *Cultural GrAnt: um Protocolo de Roteamento baseado em Inteligência Coletiva para Redes Tolerantes a Atrasos*. Tese de Doutorado — Universidade Tecnológica Federal do Paraná, Curitiba, PR, Brasil, 2012.

WALRAVENS, N. Mobile Business and the Smart City: Developing a Business Model Framework to Include Public Design Parameters for Mobile City Services. *Journal of theoretical and applied electronic commerce research*, scielocl, v. 7, p. 121 – 135, 12 2012. ISSN 0718-1876. Disponível em: <http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-18762012000300011&nrm=iso>.

WARTHMAN, F. *Delay – and Disruption – Tolerant Networks – A Tutorial*. 2012. Disponível em: <http://ipnsig.org/wp-content/uploads/2012/07/DTN_Tutorial_v2.05.pdf>. Acesso em: 02 de dezembro de 2014.