UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

SISTEMA BASEADO EM LOCALIZAÇÃO DE SERVIÇOS DE TÁXI

ARTHUR HENRIQUE KIENOLT

BLUMENAU
2013

2013/1-08

ARTHUR HENRIQUE KIENOLT

SISTEMA BASEADO EM LOCALIZAÇÃO DE SERVIÇOS DE TÁXI

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Mauro Marcelo Mattos , Dr - Orientador

BLUMENAU
2013

2013/1-08

SISTEMA BASEADO EM LOCALIZAÇÃO DE SERVIÇOS DE TÁXI

Por

ARTHUR HENRIQUE KIENOLT

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Nome do professor Orientador, Dr. – Orientador, FURB
Membro:	Prof. Paulo Fernando da Silva, Mestre – FURB
Membro:	Prof. Francisco Adell Péricas, Mestre – FURB

Blumenau, 08 de julho de 2013

Dedico este trabalho à minha família, amigos, namorada e principalmente aos meus pais pela cooperação e apoio durante a realização deste.

AGRADECIMENTOS

À minha família, que sempre me apoiou em todas as etapas da minha vida.

Aos meus pais, pelo auxílio e dedicação em garantir a minha educação e meu futuro.

À minha namorada, pela compreensão e carinho em todo o curso.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, Mauro Marcelo Mattos, pela atenção e ajuda em todas as necessidades.

Nunca ande pelo caminho traçado, pois ele conduz somente aonde outros já foram.

Graham Bell

RESUMO

Foi desenvolvido um aplicativo baseado em localização chamado “Quero Táxi”. Tal aplicativo possibilita ao usuário requisitar um serviço de táxi, e através de um algoritmo baseado em coordenadas geográficas encaminhar o taxista mais próximo, desde que o mesmo aceite o serviço, ao usuário final. O aplicativo foi desenvolvido para plataforma Android, integrado a um servidor GSM rodando sobre a biblioteca SMSLib, e um WebService com acesso ao banco de dados Oracle.

Palavras-chave: Georreferenciamento. GSM. Android. SMSLib. WebService. Táxi. Coordenada.

ABSTRACT

It was developed the location-based application called "Quero Táxi". This application allows the user to order a taxi service, and through an algorithm based on geographical coordinates, direct the nearest taxi driver, since this accepts the service, to the final user. The application was developed for Android platform, integrated with a GSM server running on SMSLib library and a Web Service with access to Oracle database.

Key-words: Georeferencing. GSM. Android. SMSLib. WebService. Taxi. Coordinated.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Programming Interface*

FIFO – *First In First Out*

GPS – *Global Positioning System*

GSM – *Global System for Mobiles*

HTTP – *HyperText Transfer Protocolo Secure*

IPEA – Instituto de Pesquisa Econômica Aplicada

SDK – *Software Development Kit*

SMS – *Short Message Service*

SMTP – *Simple Mail Transfer Protocol*

SOAP – *Simple Object Access Protocol*

TCP – *Transmission Control Protocol*

URL – *Uniform Resource Location*

UML – *Unified Modeling Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Conforme Gasparini, Campos e D'agosto (2010, p. 57), atualmente existe um movimento no sentido de tornar as cidades sustentáveis e com melhor qualidade de vida. Um dos aspectos para se atingir este objetivo está associado ao desempenho do trânsito urbano, e, neste contexto, a cada momento surgem propostas de redução de circulação de veículos particulares e o incentivo ao uso de transporte público e do transporte não motorizado, como caminhadas e bicicletas.

De acordo com o IPEA (2011, p. 3), o conceito de mobilidade pode ser entendido como a facilidade de deslocamento, o qual, por vezes, é vinculado aqueles que são transportados ou se transportam e por outras vezes relacionado à cidade ou ao local onde o deslocamento pode acontecer. Um aspecto importante a destacar é que a mobilidade depende do nível de adequação entre as características da pessoa ou objeto que deseja ou se quer transportar com a capacidade do sistema de transporte e infraestrutura, aqui incluídas todas as formas de deslocamento possíveis (IPEA 2011, p. 3). Por exemplo, a percepção de baixa frequência de congestionamento por aqueles que se deslocam por moto (seja própria, mototáxi, ou outro tipo de uso) pode ratificar o motivo mais alegado por esses usuários para a escolha de seu modo de transporte (IPEA 2011, p. 12).

Conforme Vuchic (1981 apud DA COSTA e DA COSTA, 2010, p. 2) o táxi é um sistema de transporte público caracterizado por ser um serviço adaptável as necessidades do indivíduo. O serviço pode ser oferecido de três formas distintas:

a) através da circulação pela cidade com o indicativo de que está livre, devendo o usuário indicar corporalmente a necessidade de utilização do serviço;
b) através de áreas designadas especialmente para paragem de veículos (pontos de táxi, ou *stand*), geralmente nas proximidades de grandes pólos geradores de viagens de táxi (hotéis, estações de metrô, aeroportos, teatros, etc.), onde o usuário deve se aproximar e requisitar o serviço. É caracterizado pelo sistema *First In First Out* (FIFO);
c) através do serviço oferecido por cooperativas de táxi onde o cliente entra em contato via telefone ou internet e o veículo mais próximo do usuário é encaminhado através do contato via rádio (rádio-táxi).

Comparativamente, os táxis provocam os maiores custos sociais em relação a congestionamentos, poluição do ar e barulho do que os automóveis particulares, já que

realizam mais veículos-km por passageiro-km de viagem. Normalmente causam distúrbios e atrasos no tráfego durante o ato de embarcar e desembarcar um passageiro, além de serem restritos a uma clientela reduzida, devido ao alto custo, e por serem de natureza individual. No entanto, têm uma significativa vantagem no quesito estacionamento, já que precisam de virtualmente nenhum, além de oferecer, de forma geral, maior agilidade, conforto e acessabilidade do que os transportes coletivos (DA COSTA, DA COSTA; 2010, p. 3).

Diante do exposto, este trabalho apresenta o desenvolvimento de um sistema para dispositivos móveis, que através do envio de uma *Short Message Service* (SMS), com o apoio de bibliotecas do sistema Android e recursos do *Global Positioning System* (GPS), permite requisitar os serviços de táxi, buscando o taxista disponível mais próximo do passageiro e o encaminhando ao usuário final.

1.1 OBJETIVOS DO TRABALHO

O objetivo do trabalho é o desenvolvimento de um sistema para possibilitar o envio de requisições de serviços de táxi georreferenciados a partir de dispositivos móveis.

Os objetivos específicos do trabalho são:

a) possibilitar o cadastro e consulta de novos usuários;
b) permitir o envio de mensagens georeferenciadas;
c) definir a menor distância entre taxista e passageiro através do cálculo de coordenadas;
d) possibilitar a visualização da posição do usuário através do Google Maps;
e) validar a aplicação através de estudo de caso.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em quatro capítulos, sendo neste capítulo apresentados a introdução e os objetivos do trabalho. No segundo capítulo é apresentada a fundamentação teórica, iniciando com uma introdução a mensagens SMS e a biblioteca SMSLib. Em seguida

é descrita a utilização de *web services*, e sua importância para a aplicação. Ainda no segundo capítulo é exposto o funcionamento do GPS. Após, é apresentado o funcionamento do Google Maps. O capítulo é finalizado com a descrição de trabalhos correlatos.

O terceiro capítulo relata o desenvolvimento do sistema, iniciando com os requisitos e seguido da especificação, implementação e operacionalidade. Por fim, há os resultados obtidos.

O quarto e último capítulo refere-se as conclusões do trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nas seções seguintes são apresentadas informações sobre SMS e SMSLib, WebServices no Android, GPS no Android, Google Maps no Android e trabalhos correlatos.

2.1 SMS

Short Message Service (SMS) é um serviço *wireless* aceito globalmente que possibilita a transmissão de mensagens alfanuméricas entre usuários móveis e sistemas externos tais como *eletronic mail, paging e voicemail* (BAULER, 2011).

Ainda segundo Bauler (2011), O SMS foi criado com o objetivo de prover a troca de mensagens de texto de até 160 caracteres entre dispositivos móveis. Posteriormente foi aperfeiçoado e passou a permitir a troca de mensagens com computadores, aplicativos e outros serviços IP, utilizando *gateways* protocoladores.

Este serviço possui diversas vantagens (HORD, 2011):

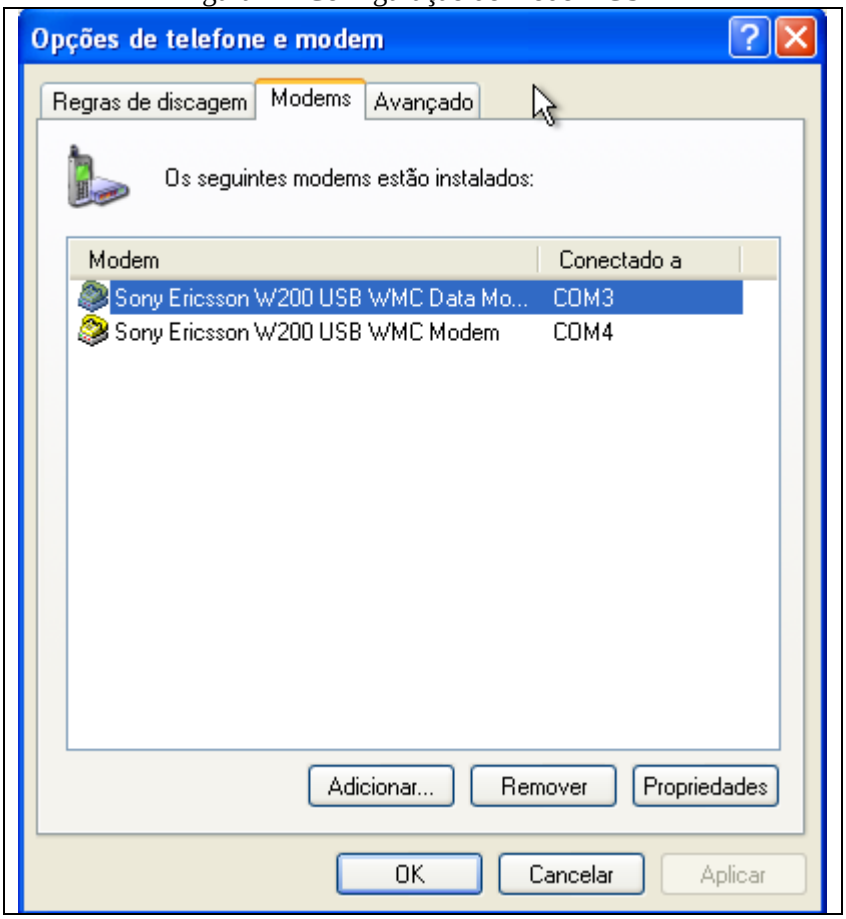
a) ele é mais discreto, o que o torna a forma ideal de comunicação quando não se pode falar publicamente;
b) normalmente, se gasta menos tempo para enviar uma mensagem de texto do que para fazer uma ligação telefônica ou enviar um e-mail;
c) deficientes auditivos de todos os graus podem utilizar deste serviço para se comunicar.

2.1.1 SMSLib

Conforme Manguni, Navarro e Rosario(2010), SMSLib é um pacote de código aberto que permite que um programa em Java envie e receba mensagens SMS através de um modem *Global System for Mobile* (GSM) ou comandos manuais. Através desta biblioteca e juntamente com um modem GSM compatível, é possível criar uma aplicação *desktop* que realiza o envio e recebebimento de mensagens SMS. As mensagens são acessadas diretamente

na memória do modem GSM, que precisa estar instalado e disponibilizado em uma porta serial, conforme Figura 1.

Figura 1 – Configuração do modem GSM



Para iniciar o modem na aplicação é necessário utilizar classe `SerialModemGateway`, passando cinco parâmetros diretamente na instância da classe. O primeiro parâmetro é o id que irá identificar o modem, podendo ser qualquer informação do tipo texto. O segundo parâmetro determina a porta serial em que o modem está configurado. O terceiro refere-se ao *baudrate*, ou seja, a velocidade em que o modem consegue transmitir dados. O quarto parâmetro é o fabricante do modem. O quinto e último parâmetro é para informação do modelo do modem. Uma vez instanciado o modem GSM na aplicação e iniciado o servidor, as mensagens podem ser obtidas e enviadas através das intruções `Service.getInstance().readMessages` e `Service.getInstance().sendMessages`, respectivamente. As mensagens obtidas podem ser manipuladas e excluídas, sendo o acesso realizado diretamente na memória do modem.

O quadro 1 apresenta um trecho de código que exemplifica a utilização da classe `SerialModemGateway`.

Quadro 1 – Exemplo de utilização da classe `SerialModemGateway`

```
try
{
	System.out.println("Example: Read messages from a serial gsm
modem.");
	System.out.println(Library.getLibraryDescription());
	System.out.println("Version: " + Library.getLibraryVersion());
	// Create the Gateway representing the serial GSM modem.
	SerialModemGateway gateway = new SerialModemGateway("modem.com4",
		"COM4", 115200, "Huawei", "E160");
	gateway.setProtocol(Protocols.PDU);
	gateway.setInbound(true);
	gateway.setOutbound(true);
	gateway.setSimPin("0000");
	Service.getInstance().addGateway(gateway);
	Service.getInstance().startService();
	msgList = new ArrayList<InboundMessage>();
	Service.getInstance().readMessages(msgList, MessageClasses.ALL);
	for (InboundMessage msg : msgList)
		System.out.println(msg);
	System.in.read();
	System.in.read();
}
catch (Exception e)
{
	e.printStackTrace();
}
finally
{
	Service.getInstance().stopService();
}
```

Fonte: SMSLib (2013).

2.2 WEB SERVICES NO ANDROID

Um *web service* fornece uma interface que permite clientes e servidores interagirem de uma forma mais geral que os navegadores. Os clientes acessam as operações na interface de um serviço *web* através de pedidos e respostas formatados em *eXtensible Markup Language* (XML) e geralmente transmitidos através do protocolo *HyperText Transfer Protocolo Secure* (HTTPS). A idéia central é que uma aplicação cliente, executando dentro de um ambiente de uma determinada organização, possa interagir com um servidor em outra organização sem a necessidade de uma supervisão humana (COULORIS, 2005).

Os *web services* realizam o empacotamento das mensagens trocadas entre clientes e

servidores utilizando XML. O conjunto de regras para o uso do XML é definido no protocolo *Simple Object Access Protocol* (SOAP) que é utilizado para encapsular as mensagens e transmití-las através da web por protocolos como HTTP, *Simple Mail Transfer Protocol* (SMTP) ou *Transmission Control Protocol* (TCP) (COULORIS, 2005).

É possível utilizar a plataforma Android para consumir informações através de um *web service*, porém, o *Software Development Kit* (SDK) do Android não oferece uma solução simples embutida para consumir *web services* (USHISIMA, 2011). Primeiramente é necessário utilizar uma biblioteca de apoio, chamada KSOAP2. Esta biblioteca encapsula a geração e leitura das instruções XML, bem como a comunicação com o serviço, tratando, inclusive, erros levantados pelo servidor e lançando exceções quando apropriado (VINICIUS, 2011). De acordo com Ushisima (2011), a biblioteca permite ainda o consumo de web services sem a necessidade de geração de código ou uso de *proxies* dinâmicos (USHISIMA, 2011). Para realizar a requisição, conforme Ushisima (2011), deve-se utilizar uma instância da classe `SoapObject`, e cada parâmetro necessário na chamada é uma propriedade de `SoapObject`. Para realizar a chamada ao serviço de aplicações é necessário utilizar a instrução *call*, da classe `HttpTransportSE`, passando na instância do objeto a *Uniform Resource Location* (URL) em que o serviço está disponível. Como retorno, tem-se um objeto também do tipo `SoapObject`, sendo cada propriedade deste objeto da consulta ao *web service*.

O quadro 2 apresenta o método `convert`, exemplificando a utilização das classes `SoapObject` e `HttpTransportSE`, citadas anteriormente.

Quadro 2 – Método `convert`

```
    public String Convert(String fromCurrency, String toCurrency, String
amount) {
        SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
        request.addProperty("from", fromCurrency);
        request.addProperty("to", toCurrency);
        request.addProperty("value", amount);

        SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
        envelope.dotNet = true;
        envelope.setOutputSoapObject(request);
        try {
            HttpTransportSE androidHttpTransport = new
HttpTransportSE(URL);
            androidHttpTransport.call(SOAP_ACTION, envelope);
            SoapPrimitive result = (SoapPrimitive)
envelope.getResponse();
            return result.toString();
        } catch (Exception e) {
            return e.getMessage();
        }
    }
```

Fonte: Ushisima (2011).

2.3 GPS NO ANDROID

O GPS é um sistema de radionavegação desenvolvido pelo Departamento de Defesa dos Estados Unidos da América com o intuito de ser o principal sistema de navegação das forças armadas americanas. Ele resultou da fusão de dois programas financiados pelo governo norte-americano para desenvolver um sistema de navegação de abrangência global: *Timation* e *System 621B*, sob responsabilidade da Marinha e da Força Aérea, respectivamente. Em razão da alta acurácia proporcionada pelo sistema e do grande desenvolvimento da tecnologia envolvida nos receptores GPS, uma grande comunidade usuária emergiu dos mais variados segmentos da comunidade civil (navegação, posicionamento geodésico, agricultura, controle de frotas etc.) (MONICO, 2000).

A concepção do sistema GPS permite que um usuário em qualquer local da superfície terrestre tenha à sua disposição no mínimo quatro satélites para serem rastreados (MONICO, 2000).

O princípio básico de navegação pelo GPS consiste na medida de distâncias entre o usuário e quatro satélites. Conhecendo as coordenadas dos satélites num sistema de referência

apropriado, é possível calcular as coordenadas da antena do usuário no mesmo sistema de referência dos satélites (MONICO, 2000).

A plataforma Android fornece recursos para o desenvolvimento de aplicativos baseados em localização. A principal classe para utilização destes serviços é a `LocationManager`. Como outros recursos disponibilizados pela Google, não é possível instanciar diretamente um objeto `LocationManager`. Para tal, é necessário realizar uma chamda para o método `getSystemService` passando como parâmetro uma *String*, que representa o tipo de serviço necessário, que neste caso é `LOCATION_SERVICE`. A partir desta instância, é possível acessar uma lista de provedores para obter a localização do usuário, sendo esta lista representada pela classe LocationProvider (ANDROID DEVELOPERS, 2012a).

A solicitação de coordenadas através de um dispositivo pode ser realizada através do GPS ou da internet. A obtenção das coordenadas através da internet é realizada com o auxílio das torres de celular do sinal Wi-Fi. Desta forma, obtem-se a posição tanto em locais fechados quanto em locais aberto, com um menor consumo de energia e reposta mais rápida, porém com uma menor precisão. Por sua vez, o GPS tem uma precisão muito superior em comparação a internet, mas com maior consumo de bateria e reposta lenta. Há também a necessidade de estar em campo aberto (ANDROID DEVELOPERS, 2012a).

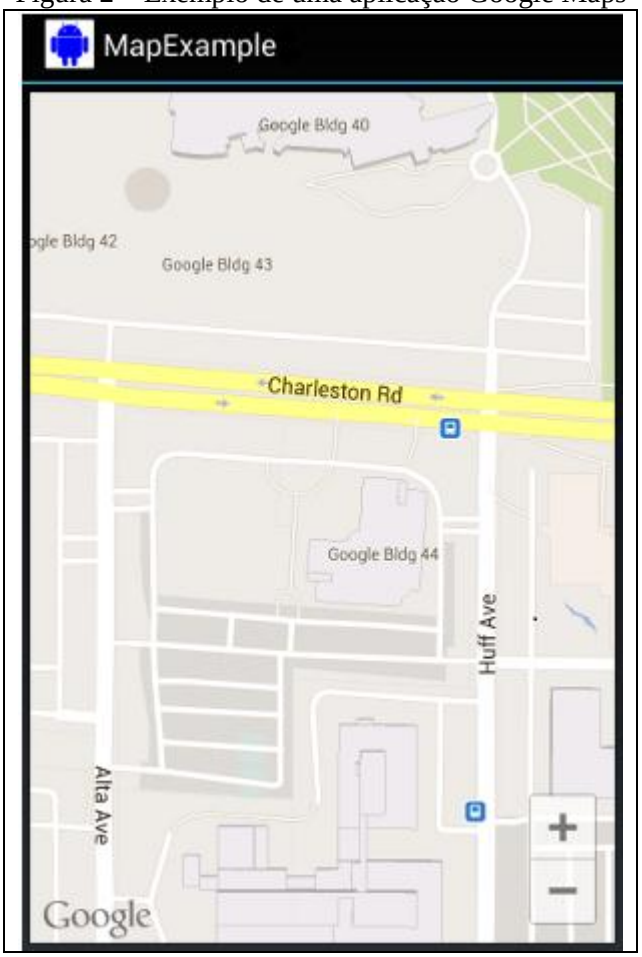
Para obter a posição do dispositivo no Android, é necessária a utilização do método `requestLocationUpdate`, da classe `LocationManager`. São quatro os parâmetros necessários para esta chamada. O primeiro parâmetro passado deve ser o tipo de provedor para busca da localização, que pode variar entre a internet, definido pela instrução `LocationManager.NETWORK_PROVIDER`, e GPS, definido pela instrução `LocationManager.GPS_PROVIDER`. O segundo parâmetro diz respeito à frequência em que deve ser atualizada a posição, em termos de tempo. O terceiro determina a frequência de atualização com base na distância percorrida. O quarto e último parâmetro deve ser o objeto que implementa a interface `LocationListener` (ANDROID DEVELOPERS, 2012b).

2.4 GOOGLE MAPS

Google Maps é uma aplicação para visualiação e navegação em mapas interativos (ANDROID DEVELOPERS, 2012c). Esta ferramenta fornece diversas funcionalidades, como

solicitar rotas, demonstrar um ponto específico e demarcar as principais localizações da preferência do usuário. Um exemplo desta aplicação para dispositivos móveis pode ser visto na Figura 2.

Figura 2 – Exemplo de uma aplicação Google Maps



Fonte: Android Developers (2012c).

O Google Maps oferece uma interface intuitiva e mapeamento altamente reponsivo, com ruas detalhadas e imagens aéreas. Além disso os mapas permitem ao usuário controle total de navegação.

2.4.1 GOOGLE MAPS API V2

Desde 3 de dezembro de 2012 a versão 1 da *Application Programming Interface* (API) Google Maps do Android foi descontinuada, e desde 18 de março de 2013 não é mais possível requisitar novas licenças de uso para esta versão, sendo os usuários encorajados a utilizar a versão 2 da API (ANDROID DEVELOPERS, 2013d).

Através do Google Maps API, é possível adicionar mapas com a base de dados da Google em sua aplicação. A API garante automaticamente o acesso aos servidores da Google, o *download* dos dados e a visualização do mapa. Podem ser adicionados ainda polígonos, marcadores e sobreposições no mapa, bem como alterar a forma de visualização do mesmo (ANDROID DEVELOPERS, 2013d).

2.4.2 GOOGLE MAPS NO ANDROID

A plataforma Android está fortemente ligada com os serviços da Google. Desta forma, a Google disponibilizou a Google Maps API v2. Diferentemente da versão anterior, onde a mesma funcionava como uma biblioteca externa, a nova API da Google está integrada diretamente no Android SDK, facilitando a utilização da ferramenta.

Outra mundança referente à versão anterior da API diz respeito a obtenção da licença de uso. Para esta nova versão, existem dois tipos de licenças de uso (ANDROID DEVELOPERS, 2013d):

a) *debug certificate*: as ferramentas de compilação do Android fornecem um modo de assinatura de depuração, que torna mais fácil para o desenvolvedor criar e testar sua aplicação. Ao utilizar este certificado, as ferramentas do Android SDK invocam o programa *keytool*, nativo do java, para criar automaticamente um certificado para testes. Para usuários do Eclipse, esta configuração é padrão, sendo a chave gerada ao depurar o projeto. Este tipo de certificado somente é válido para uso no aplicativo que está sendo testado e não é possível publicar o aplicativo com esta chave;
b) *release certificate*: este tipo de certificado é específico para aplicativos que serão publicados. Este certificado é gerado quando é utilizada a opção *release* no Eclipse, juntamente com o programa *keytool*, nativo do java. Este certificado permite que seu aplicativo seja publicado.

Para adicionar o mapa na aplicação Android, conforme Android Developers (2013d), é necessário utilizar a classe `GoogleMap`. Esta classe é responsável por todas as ações realizadas sobre o mapa, como conectar com o *database* da Google, realizar o *download* dos mapas, apresentar os mapas no dispositivo, controlar as opções de *zoom*, entre outros. Não é possível instanciar diretamente uma classe `GoogleMap`. Desta forma, deve-se utilizar um objeto da classe `Fragment` para obter acesso ao objeto `GoogleMap`. A classe `Fragment` funciona como

um container para o objeto `GoogleMap`. Para versão 3.1 do Android em diante deve ser utilizada a classe `MapFragment`. Para as versões anteriores deve ser utilizada a classe `SupportMapFragment`.

O quadro 3 demonsta um exemplo de utilização a classe `SupportMapFragment`.

Quadro 3 – Utilização da classe `SupportMapFragment`

```
private void setUpMapIfNeeded() {
    // Do a null check to confirm that we have not already instantiated
the map.
    if (mMap == null) {
        mMap = ((MapFragment)
getFragmentManager().findFragmentById(R.id.map))
                            .getMap();
        // Check if we were successful in obtaining the map.
        if (mMap != null) {
            // The Map is verified. It is now safe to manipulate the map.

        }
    }
}
```

Fonte: Android developers (2013d).

2.5 TRABALHOS CORRELATOS

Existem aplicações que possuem características semelhantes ao proposto neste trabalho, tais como o ResolveAí (RESOLVEAÍ, 2012), TaxiMov (TAXIMOV, 2013) e EasyTaxi (LEVOVC, 2013).

2.5.1 ResolveAí

O ResolveAí é um aplicativo de transporte e utilidade com o qual é possível agendar uma corrida de táxi de forma rápida e fácil, e está atualmente presente em 20 cidades (RESOLVEAI, 2012a).

Para utilizar o aplicativo, primeiramente é necessário realizar um cadastro, com informações básicas sobre o usuário. Uma vez realizada esta ação e logado no sistema, é possível realizar a requisição do serviço. Ao buscar a posição do dispostivo, por algumas vezes a precisão pode não ser exata. Porém, o aplicativo permite definir através de um mapa a

posição manualmente. Após definida a posição do dipostivo, deve-se informar o endereço de destino, para então o aplicativo destinar um taxista (RESOLVEAI, 2012b).

Os pedidos podem ser acompanhados em tempo real para a maioria dos carros da frota, sendo que os carros são equipados com dispositivos GPS.

Na figura 3 pode ser visualizado um pedido em andamento no aplicativo ResolveAí.

Figura 3 – Aplicativo ResolveAí

Fonte: ResolveAi(2012).

2.5.2 TaxiMov

Criado e desenvolvido pela e-flows, empresa de serviços e produtos na área de mobilidade, o Taximov é um sistema de solicitação de táxi por telefone ou internet, seja computador, *notebook* ou *smartphones*. Ao solicitar um táxi, o software localiza o motorista que está mais próximo do endereço do cliente e que melhor atenda aos requisitos solicitados (REVISTA APOLICE, 2013).

Segundo a revista Apólice (2013), a solicitação pode ser realizada por qualquer dispositivo com acesso à Internet. O cliente informa os seus dados, o endereço de partida e as características desejadas do táxi ou do taxista, como porta-malas grande, se aceita cartão de crédito, se tem ar-condicionado etc. Automaticamente, o Taximov localiza o automóvel livre

mais próximo e que melhor atenda aos requisitos solicitados e envia uma mensagem convidando o motorista a realizar a corrida. Basta que o cliente aceite o convite para que ele seja informado de todos os dados sobre o táxi, além do tempo estimado de chegada.

O TaxiMov não permite a requisição de serviço por parte do usuário diretamente de um aplicativo pelo dispositivo. A requisição deve ser realizada através do site, via browser.

Abaixo, na figura 4, pode-se observar a solicitação de serviço, visualizada por parte do taxista (GRANJANEWS, 2012).

Figura 4 – Aplicativo TaxiMov

Fonte: TaxiMov (2013).

2.5.3 EasyTaxi

O EasyTaxi é um aplicativo de smartphone que conecta o passageiro ao táxi mais próximo, assim viabilizando uma corrida rápida, prática e segura (EASYTAXI, 2013).

A segurança do usuário e do taxista é o diferencial do serviço da Easy Taxi. Quem pede um veículo pelo aplicativo tem acesso a informações sobre o motorista (nome e carro) e pode acompanhá-lo em tempo real via GPS. Já o taxista tem a segurança de saber quem será

seu passageiro e onde ele está, já que seu endereço é preenchido automaticamente por meio de uma tecnologia de geolocalização (SEGS, 2013).

Oferecido em quatro idiomas, Português, Inglês, Espanhol e Coreano, o aplicativo pode ser baixado de forma gratuita em smartphones com sistema Android ou iOS. Caso o usuário não tenha um celular com acesso à internet, ele pode solicitar um táxi do serviço por meio do site oficial. Com usabilidade simples, o processo de pedido do táxi é concluído em apenas alguns cliques e em menos de 10 minutos o veículo estará à disposição do passageiro (SEGS, 2013).

Na figura 5 pode ser visualizada a tela para requisição de um serviço no aplicativo EasyTaxi.

Figura 5 – Aplicativo EasyTaxi

Fonte: EasyTaxi (2013).

3 DESENVOLVIMENTO

Neste capítulo são abordadas as etapas de desenvolvimento do projeto. Primeiramente são descritos os requisitos do trabalho. Em seguida são apresentadas a especificação e implementação, destacando e explicando as principais classes e funções do aplicativo. Por fim, são apresentados os resultados obtidos.

3.1 REQUISITOS DO PROJETO

A aplicação deverá:

a) possibilitar o envio de mensagem SMS, contemplando na mesma as informações de georreferenciamento (Requisito Funcional – RF);
b) disponibilizar uma opção para envio automático da mensagem (RF);
c) definir a menor distância entre o passageiro e o taxista, através do cálculo de coordenadas (RF);
d) enviar uma mensagem SMS para o taxista selecionado com o endereço do passageiro (RF);
e) apresentar para o taxista através do Google Maps a localização do passageiro (RF);
f) possibilitar o cadastro/consulta de taxistas vinculados ao serviço, bem como suas informações básicas (RF);
g) utilizar o ambiente Eclipse para o desenvolvimento (Requisito Não-Funcional – RNF);
h) utilizar a linguagem de programação Java (RNF);
i) utilizar o sistema operacional Android (RNF).

3.2 ESPECIFICAÇÃO

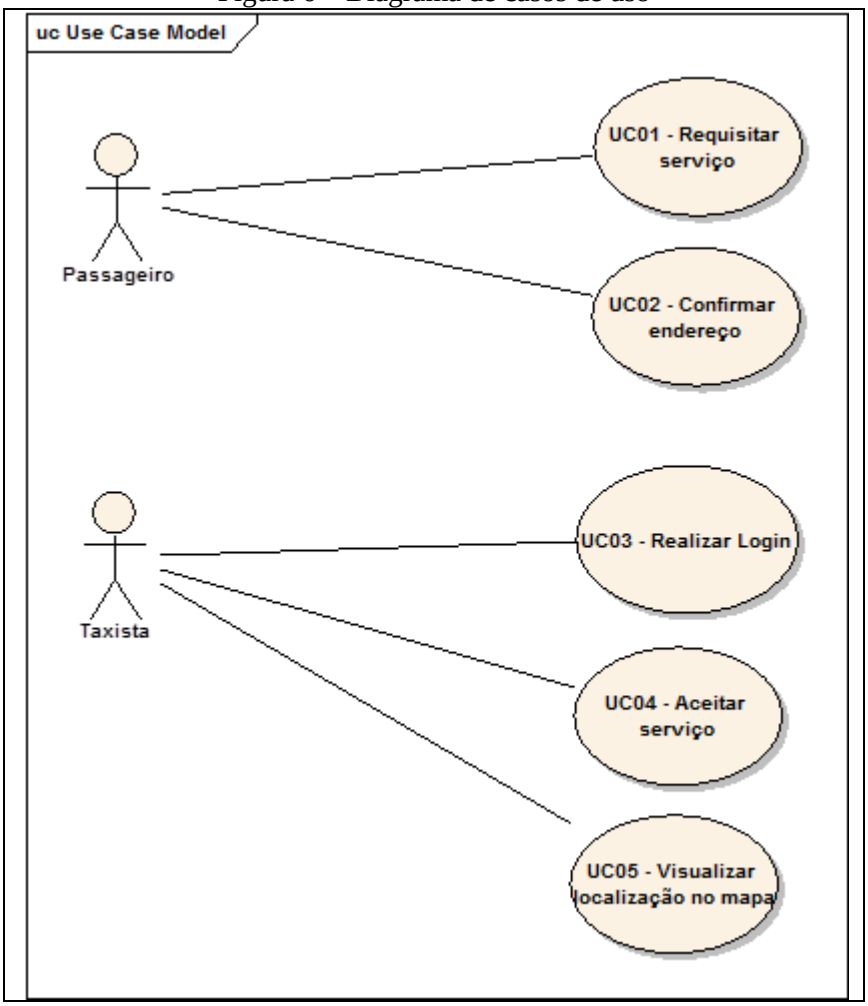
Nesta seção são descritos os diagramas de casos de uso, de classes, de distribuição e de sequência. Para a elaboração dos diagramas foi utilizada a ferramenta Enterprise Architect versão 6.5, seguindo a notação *Unified Modeling Language* (UML).

3.2.1 Diagrama de casos de uso

O trabalho disponibiliza um aplicativo para dispositivos móveis para requisição de serviços de táxi a partir de mensagens georreferenciadas. Nos casos de uso são apresentados dois usuários: passageiro, que envia a mensagem de requisição e o taxista, que recebe.

Na figura 6 é apresentado o diagrama de casos de uso da aplicação.

Figura 6 – Diagrama de casos de uso



3.2.2 UC01 – Requisitar serviço

A ação inicial do usuário deve ser a requisição do serviço, conforme descrito no Quadro 1.

Quadro 4 – Caso de uso UC01

UC01 – Requisitar serviço	
Pré-condição	O dispositivo deve estar conectado à internet e com o GPS ativo.
Cenário principal	01) O passageiro aperta o botão “Requisitar serviço”. 02) O aplicativo apresenta tela com o endereço atual
Exceção	Se o GPS e a internet não estiverem ativos, será apresentada tela para habilitar ambos.
Pós-condição	É apresentada tela para confirmação do endereço.

3.2.2.1 UC02 – Confirmar endereço

O segundo caso de uso apresenta o endereço atual do passageiro, permitindo o mesmo confirmar ou cancelar a requisição.

Quadro 5 – Caso de uso UC02

UC02 – Confirmar endereço	
Pré-condição	O passageiro deve ter requisitado o serviço
Cenário principal	01) O passageiro aperta o botão “OK”, confirmando o endereço. 02) O aplicativo inicia o serviço, requisitando taxistas
Cenário alternativo	Passageiro aperta o botão “Cancelar”, cancelando a requisição.
Pós-condição	O serviço é iniciado por parte do passageiro.

3.2.2.2 UC03 – Realizar login

Neste caso de uso, o taxista deve logar no sistema, para começar a receber as requisições.

Quadro 6 – Caso de uso UC03

UC03 – Confirmar endereço	
Pré-condição	O taxista deve possuir cadastro
Cenário principal	01) O taxista informa login; 02) O taxista informa senha; 03) O taxista aperta o botão “Login”.
Cenário alternativo	01.1) O taxista não possui cadastro 01.2) O taxista seleciona a opção “Realizar cadastro” 01.3) O taxista realiza o cadastro. 01.4) O taxista realiza o login.
Pós-condição	O serviço é iniciado por parte do taxista.

3.2.2.3 UC04 – Aceitar serviço

À partir de uma requisição feita por um passageiro, o sistema realiza o cálculo de menor distância. Caso o taxista seja selecionado é apresentada para o mesmo uma opcão com o endereço do passageiro e a possibilidade de aceitar ou recusar o serviço. O quarto caso de uso representa esta ação.

Quadro 7 – Caso de uso UC04

UC04 – Aceitar serviço	
Pré-condição	O serviço deve ter sido iniciado por parte do passageiro e do taxista.
Cenário principal	01) O aplicativo apresenta tela com o endereço do passageiro. 02) O taxista aceita o serviço, apertando o botão “OK”. 03) O aplicativo apresentada na tela o endereço, oferecendo opção para visualização no mapa.
Cenário alternativo	02.1) O taxista não aceita o serviço, apertando o botão “Cancelar”. 02.2) O aplicativo seleciona outro taxista para atender a requisição.
Pós-condição	O serviço é aceito por parte do taxista, sendo apresentado em sua tela o endereço do passageiro e a possibilidade de visualização no mapa.

3.2.2.4 UC05 – Visualizar localização no mapa

Para facilitar a busca do taxista pelo passageiro, foi disponibilizada uma opção para visualização do endereço de busca diretamente no Google Maps. Esta ação é descrita no quinto caso de uso.

Quadro 8 – Caso de uso UC05

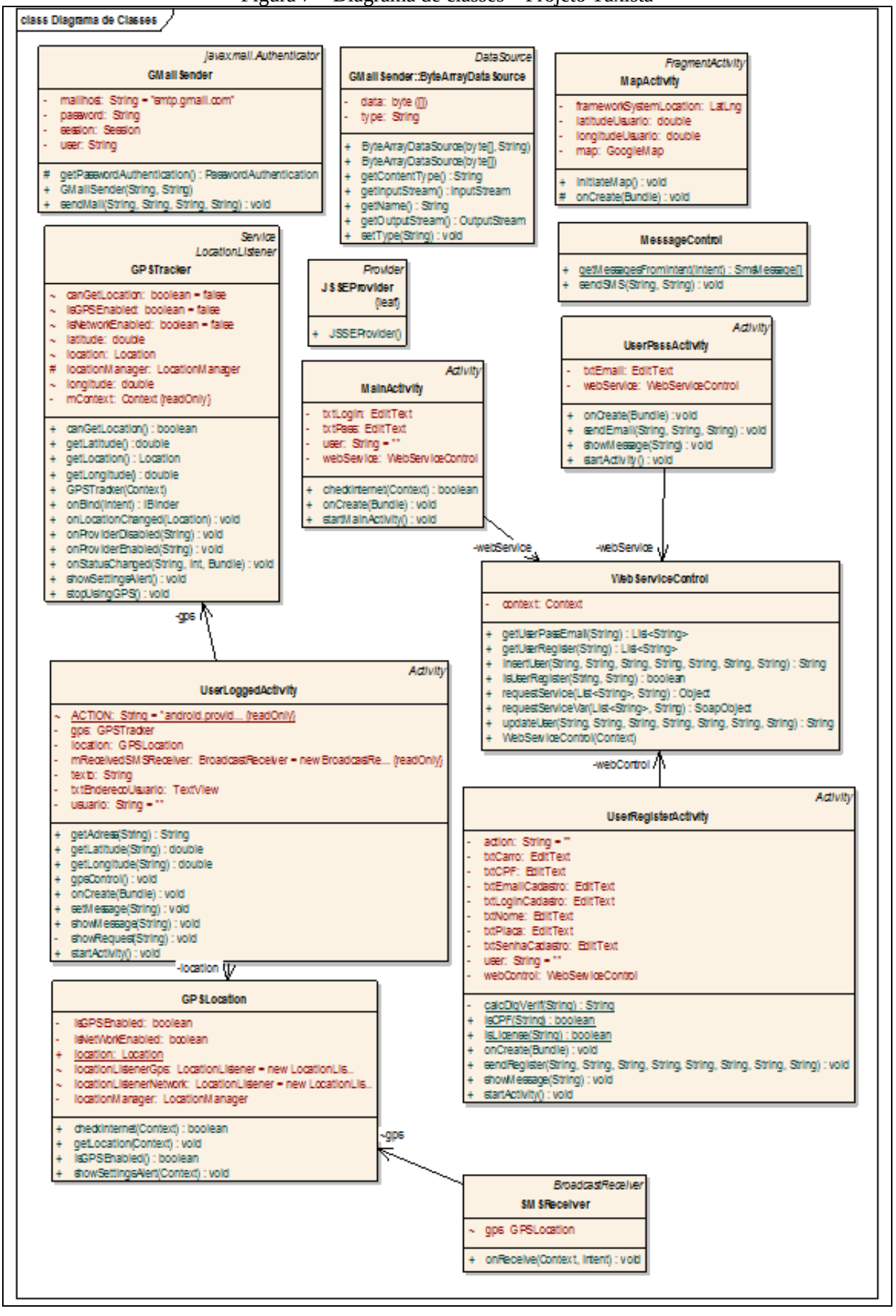
UC05 – Visualizar localização no mapa	
Pré-condição	O serviço deve ter sido aceite por parte do taxista e o endereço de busca disponibilizado.
Cenário principal	01) O taxista aperta o botão “Visualizar mapa” 02) O aplicativo apresenta o mapa com o endereço de destino.
Pós-condição	É apresentado o mapa com marcador sobre o endereço de destino.

3.2.3 Diagrama de classes

No desenvolvimento do sistema, foi necessária a criação de quatro aplicações. A primeira, representada no diagrama de classes da figura 7, trata da aplicação utilizada pelo taxista para gerenciar suas requisições, sendo a `MainActivity` a *activity* principal do aplicativo. A partir desta que as demais *activities* são iniciadas.

A classe `MainActivity` possui 2 métodos, sendo que o método `onCreate` é executado ao iniciar a classe. Este chama o método `startMainActivity` que é responsável por atribuir os *listeners* para cada componente de tela, definidos no *layout* definido como `tela_inicial.xml`. A partir desta classe é possível acessar 3 novas activities, sendo elas a `UserLoggedActivity`, executada ao realizar o login, a `UserPassActivity`, executada caso o usuário tenha esquecido sua senha e deseja recuperá-la, e a `UserRegisterActiviy`, para realizar novos cadastros.

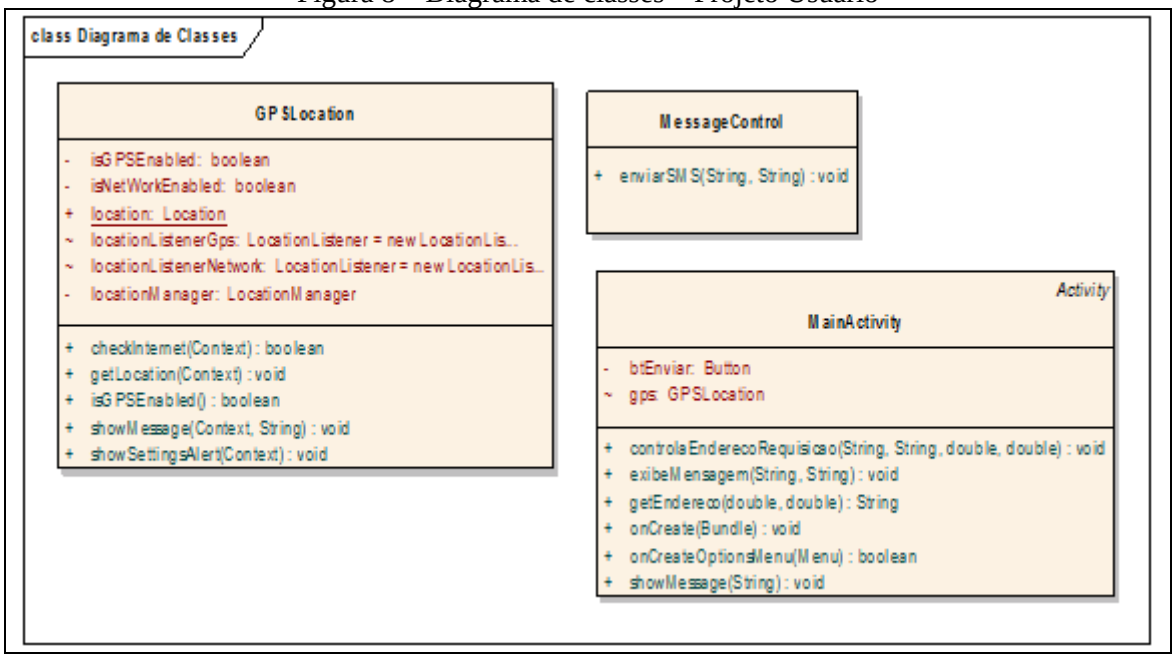
Figura 7 – Diagrama de classes – Projeto Taxista



A classe `WebServiceControl` é responsável por garantir e realizar todas as operações diretamente no banco de dados, sendo o seu acesso garantido através de um servidor *web*.

Na segunda aplicação, utilizada pelo usuário requisitante, o diagrama de classes está demonstrado na figura 8, onde a principal *activity* é a `MainActivity`.

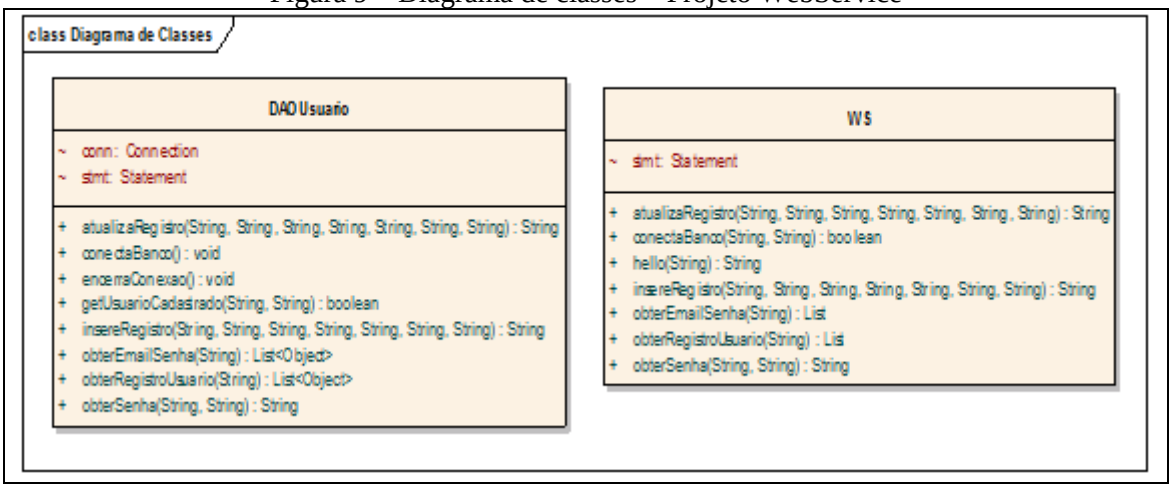
Figura 8 – Diagrama de classes – Projeto Usuário



Por tratar da requisição da informação, nesta aplicação há apenas 3 classes. A classe `MainActivity` é responsável por apresentar a tela da aplicação e atribuir o *listener* para o único componente de tela existente. `GPSTracker` é a classe responsável por obter a localização do usuário, e ela implementa a interface `LocationListener`. Por fim, a classe `MessageControl` realiza o controle sobre as mensagens enviadas para requisição.

A terceira aplicação é o servidor web, definido no diagrama apresentado na figura 9.

Figura 9 – Diagrama de classes – Projeto WebService



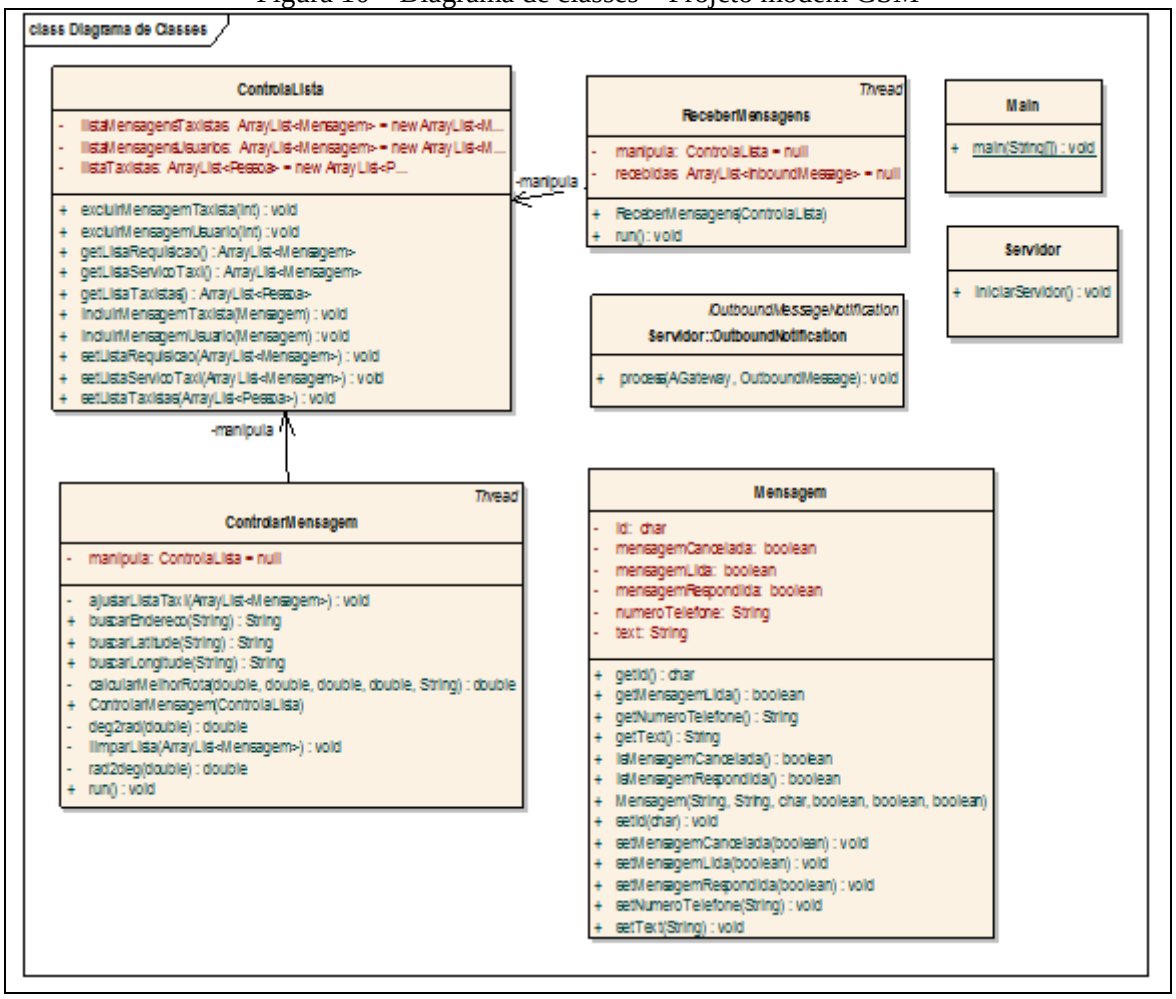
A principal classe deste projeto é a `WS`. Nesta classe são definidas as operações que serão utilizadas através do *web service*. Nas operações por sua vez há apenas uma chamada para os métodos da classe `DAOUsuario`, que é responsável por realizar a conexão com o banco e realizar as leituras e gravações necessárias.

A quarta e última aplicação refere-se ao servidor GSM, visualizada no diagrama da figura 10, responsável por ser o intermediário entre as mensagens do usuário e dos taxistas. Nesta aplicação a principal classe é a `Main`, sendo esta responsável por iniciar o servidor, previamente definido na classe `Servidor`.

As mensagens são gerenciadas através das classes `ControlarMensagem` e `ReceberMensagem`. A classe `ReceberMensagem` manipula as mensagens, criando instâncias do tipo `Mensagem` para cada mensagem recebidas. Esta classe define e separa mensagens de usuários e taxistas, armazenando-as em listas, controladas por sua vez pela classe `ControlarMensagem`.

A classe `EnviarMensagem` realiza o tratamento das requisições do usuário, enviando as mensagens de requisição de serviço aos taxistas. Nesta mesma classe é definido o algoritmo de menor distância.

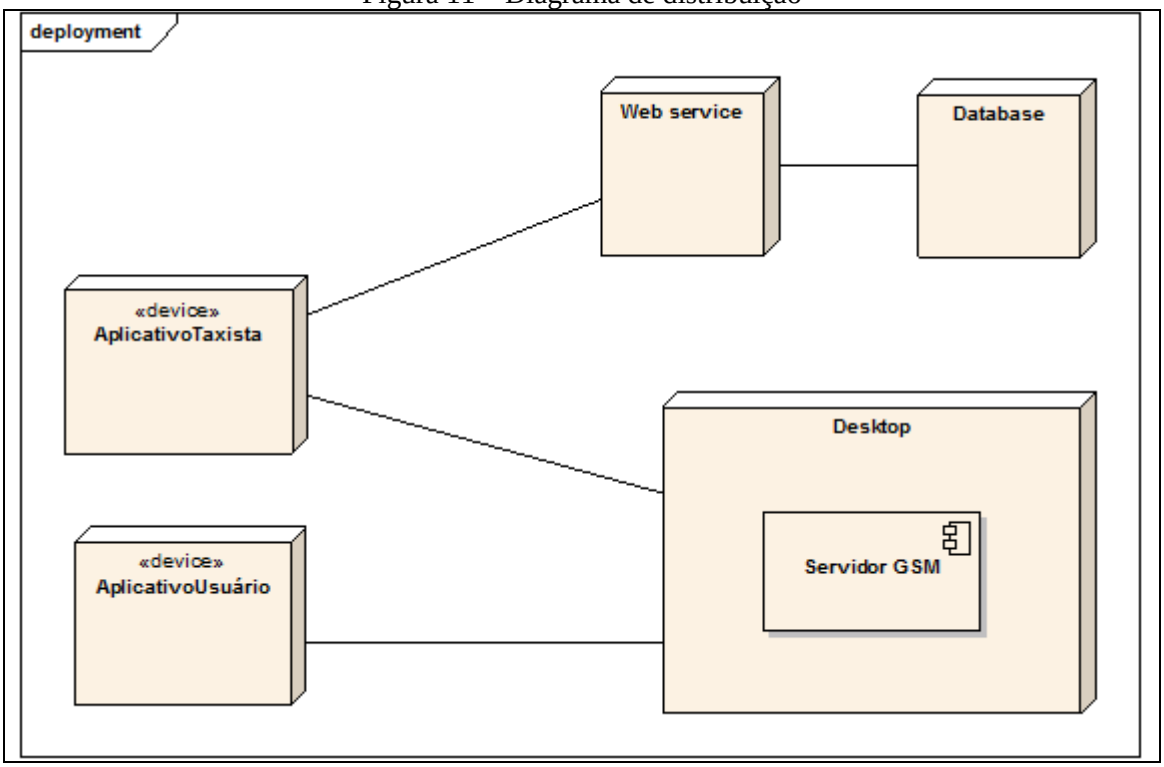
Figura 10 – Diagrama de classes – Projeto modem GSM



3.2.4 Diagrama de distribuição

O diagrama de distribuição apresentado na figura 11 apresenta a forma como as tecnologias se integram.

Figura 11 – Diagrama de distribuição



3.2.5 Diagrama de sequência

Os diagramas de sequência representados nas figuras 12 e 13 mostram respectivamente a interação do usuário e do taxista com o aplicativo.

Figura 12 – Diagrama de sequência - Usuário

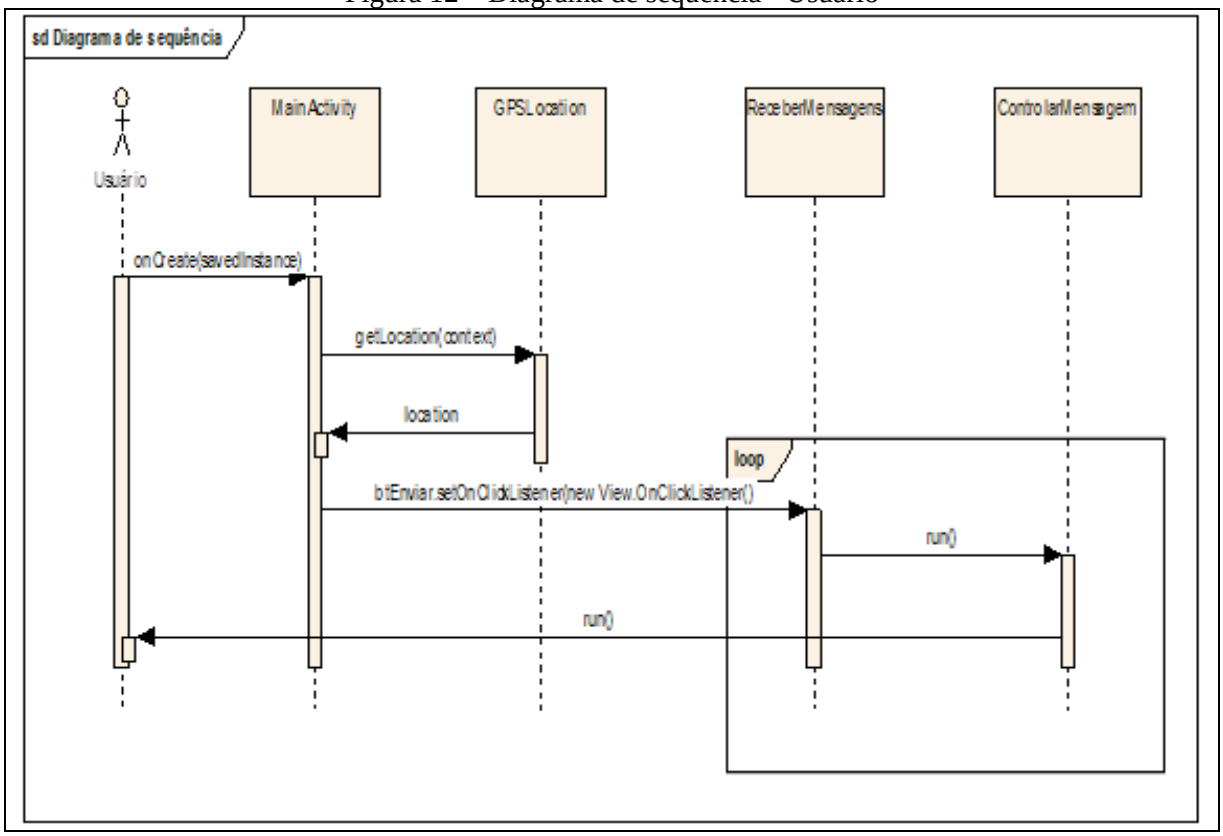
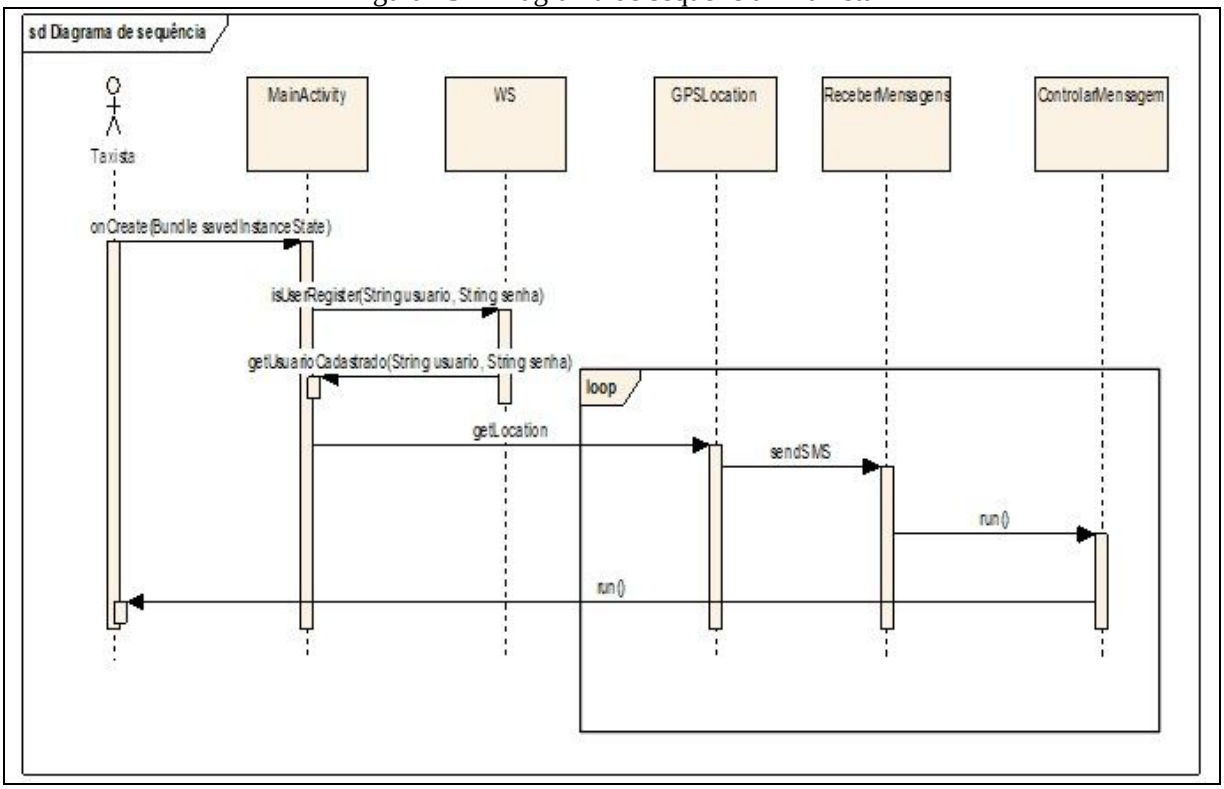


Figura 13 – Diagrama de sequência - Taxista



3.3 IMPLEMENTAÇÃO

Esta seção descreve técnicas e ferramentas utilizadas para o desenvolvimento do sistema. São detalhadas as principais classes, apresentando trechos de código para facilitar o entendimento. Por fim, é apresentada a operacionalidade dos aplicativos.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento de todos os aplicativos do sistema foi utilizada a linguagem de programação Java em conjunto com o Android SDK, que disponibiliza as APIs e ferramentas necessárias para desenvolvimento para a plataforma Android. Como *Integrated Development Environment* (IDE), foi utilizado o Eclipse, juntamente com o *plugin Android Development Tools* (ADT).

O Android SDK disponibiliza várias bibliotecas e ferramentes essencias para o desenvolvimento de aplicativos para o sistema operacional Android, incluindo um emulador e um *debbuger* para facilitar os testes.

O desenvolvimento da aplicação foi realizado utilizando dois modelos de dispositivos móveis, sendo eles um *smartphone* Samsung Galaxy Ace Plus, da fabricante Samsung com o sistema operacional Android Gingerbread, e um celular Sony Ericsson W200i, da fabricante Sony.

No processo de tratamento das mensagens recebidas foi utilizada a biblioteca SMSLib. Foi escolhida esta bibilioteca pois a mesma é *open-source* e possui melhor especificação e documentação para uso.

3.3.2 Configuração da aplicação

Nas aplicações para Android, é comum haver um arquivo de *manifest*, normalmente nomeado de `AndroidManifest.xml`, onde são determinadas todas as configurações da aplicação, como por exemplo as *activities*, *layouts*, permissões, entre outros. O quadro 9 apresenta o conteúdo do `AndroidManifest.xml` da aplicação do taxista, por ser o mais completo entre os desenvolvidos neste projeto.

Neste exemplo pode-se identificar as permissões necessárias para o correto funcionamento da aplicação, como por exemplo as permissões de acesso a internet e de utilização dos serviços de mapa da Google. Pode-se ressaltar ainda como uma das configurações principais da aplicação a *key* de depuração, utilizada para realizar o *download* dos mapas do servidor da Google, e presente no fim do arquivo *manifest* da aplicação.

Quadro 9 – AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.aplicativotaxista"
    android:versionCode="1"
    android:versionName="1.0" >
    <!-- Permissões de uso da aplicação -->
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
      <permission
              android:name="com.example.mapdemo.permission.MAPS_RECEIVE"
              android:protectionLevel="signature"/>
      <uses-permission
android:name="com.where.common.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name=
        "com.google.android.providers.gsf.permission.READ_GSERVICES" />
      <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppThemeBlack" >
        <!-- Activity inicial -->
        <activity
```

```
            android:name="com.example.aplicativotaxista.MainActivity"
            android:label="@string/title_activity_aplicativo_usuario" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category    android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <!-- Demais activities -->
        <activity android:name="UserLoggedActivity"></activity>
        <activity android:name="UserPassActivity"></activity>
        <activity android:name="UserRegisterActivity"></activity>
        <activity android:name="MapActivity"></activity>
        <!-- Chave necessária para utilização do Google maps API v2 -->
        <meta-data android:name=
          "com.google.android.maps.v2.API_KEY"
          android:value="
AIzaSyD352AfNJ9O6NmNZ-igch7nNgqv-1aPd9s"/>
    </application>
</manifest>
```

3.3.3 Google Maps API v2

A integração do Google Maps com uma aplicação Android requer uma chave de autenticação. Esta chave é necessária para que seja possível acessar o banco de dados da Google e obter as informações dos mapas.

Desde 3 de dezembro de 2012 a v1 da Google Maps API foi descontinuada. As chaves obtidas para a primeira versão da API não podem ser utilizadas nesta nova versão. Desta forma, é necessário gerar uma chave específica para esta versão da API.

Para esta nova versão podem ser geradas chaves do tipo *debug* e do tipo *release*. As chaves do tipo *debug* são designadas para apliações em testes, que não serão publicadas na loja da Google. As chaves do tipo release são específicas para disponibilizar aplicações na loja. Para as aplicações desenvolvidas, foram utilizadas chaves do tipo *debug*.

O método para obter as chaves de testes requerem alguns passos. Ao executar a aplicação Android através do Eclipse, é gerado um arquivo *debug.keystore,* utilizado para extrair através da ferramenta *keytool* do Java o “SHA-1 fingerprint”. É necessário também

uma conta Google para cadastrar o projeto na Google APIs Console, informar a SHA-1 *fingerprint* obtida e obter a chave de utilização do Google Maps. Esta chave deve ser inserida no `AndroidManifest`, conforme demonstrado anteriormente.

3.3.4 Acesso *web service*

Para realizar a leitura e gravação de dados em banco de dados através da aplicação Android foi necessária a utilização de um web service. O método `requestServiceVar`, disponibilizar na classe `WebServiceControl` é responsável pela utilização destas operações, e está definido no quadro 10.

Quadro 10 – Método `requestServiceVar`

```
	public SoapObject requestServiceVar(List<String> parametros, String
metodo) {
	SoapObject resultSOAP = null;
	try {
		SoapObject request = new SoapObject("http://src/", metodo);
		for (int i = 0; i < parametros.size(); i++) {
			request.addProperty("arg"+i, parametros.get(i));
		}
		SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
		envelope.setOutputSoapObject(request);
		HttpTransportSE androidHttpTransport = new
HttpTransportSE("http://189.35.192.158:8080/WebServiceNovo/WS?wsdl/");
		androidHttpTransport.call(metodo, envelope);
		resultSOAP= (SoapObject) envelope.bodyIn;
	} catch (Exception e) {
		e.printStackTrace();
	}
	return resultSOAP;
}
```

Os parâmetros passados neste método são uma lista de *Strings*, que representa os parâmetros para as operações acessadas no *web service*, e uma *String* que representa o nome da operação a ser acessada. O objeto `SoapObject` representa o *namespace* do *web service* e o método que é necessário chamar. Este mesmo objeto é responsável ainda por armazenar os parâmetros para utilizar no método chamado. A chamada ao servidor é realizada utilizando o

objeto `HttpTransportSE` passando um envelope `SOAP` e o próprio `SoapObject`. Como resultado, temos um objeto do próprio tipo `SoapObject`, que contém o retorno das operações realizadas no *web service*.

3.3.5 Servidor GSM

Para intermediar as mensagens recebidas por usuários e taxistas e posteriormente realizar a melhor escolha através da localização dos mesmos, foi necessário construir um servidor GSM, capaz de filtrar as mensagens recebidas e enviar novas mensagens de requisição. A principal classe deste servidor é a `Servidor`, que através do método `iniciarServidor` configura um modem GSM para realizar as operações necessários. Este método está definido no quadro 11.

Quadro 11 – Método iniciarServidor

```
      public void iniciarServidor() throws Exception {
            OutboundNotification outboundNotification = new
OutboundNotification();
            SerialModemGateway gateway = new
SerialModemGateway("modem.com4", "COM4", 115200, "Sony Ericsson", "W200i");
            gateway.setProtocol(Protocols.PDU);
            gateway.setInbound(true);
            gateway.setOutbound(true);
            gateway.setSimPin("0000");

      Service.getInstance().setOutboundMessageNotification(outboundNotifica
tion);
            Service.getInstance().addGateway(gateway);
            Service.getInstance().startService();
      }
```

Primeiramente é necessário instanciar um objeto do tipo `SerialModemGateaway`, passando como parâmetro as configurações do modem GSM, com a porta em que o mesmo está conectado, modelo e *baudrate*. Em seguida é necessário "ativar" os métodos `setInbound` e `setOutbound`. Estes métodos da classe *gateway* definem respectivamente se será possível tratar as mensagens recebidas e enviadas. Por fim, deve-se adicionar o objeto `SerialModemGateway` ao servidor e iniciar o mesmo através do método `startService`.

As mensagens recebidas são armazenadas em listas com objetos do tipo `InbondMessage` e podem ser lidas a qualquer momento através da instrução `readMessages`. O objeto `InboundMessage` possui todas as informações da mensagem recebida, como texto, número do celular emitente, horário, entre outras características, sendo possível manipular estes dados e utilizá-los na aplicação.

3.3.6 GPS

O localização do usuário é obtida através do dispositivo GPS em conjunto com o provedor WiFi. A classe `GPSLocation`, referenciada no quadro 12, é responsável por atualizar a posição do usuário.

Quadro 12 – Método `getLocation`

```
public void getLocation(Context context) {
      if(locationManager == null)
           locationManager =
           (LocationManager)
           context.getSystemService(Context.LOCATION_SERVICE);
      try {
      isGPSEnabled =
      locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
      } catch(Exception ex) {}
      try {
      isNetWorkEnabled =
      locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
      } catch(Exception ex) {}
      if (isGPSEnabled) {
           locationManager.requestLocationUpdates
           (LocationManager.GPS_PROVIDER, 40000, 0,
           locationListenerGps);
      }
      if (isNetWorkEnabled) {
           locationManager.requestLocationUpdates
 (LocationManager.NETWORK_PROVIDER, 40000, 0,  locationListenerNetwork);

      }
      if (!checkInternet(context) && !isGPSEnabled) {
            showSettingsAlert(context);
      }
}
```

O método `getLocation`, disponível na classe `GPSLocation` é responsável por atualizar a posição do usuário. Esta classe possui uma instância dos objetos `Location` e `LocationManager`. Através do método `requestLocationUpdates` é configurado que sejam realizadas atualizações de localização através do GPS e da WiFi (primeiro parâmetro), a cada quarenta segundos (segundo parâmetro), não levando em consideração a distância percorrida (terceiro parâmetro). Por fim, é informado o *listener* para atualização da posição. A própria

classe `GPSLocation` implementa a interface `LocationListener`. O método `OnLocationChanged` é chamado toda vez que é realizada a atualização de posição, recebendo como parâmetro a nova posição do dispostivo.

3.3.7 Operacionalidade da implementação

A operacionalidade da aplicação é apresentada com base nos casos de uso descritos, sendo apresentadas *screenshots* da aplicação em execução e descrição dos passos para melhor endendimento.

A interface do protótipo é simples e foi desenvolvida com o objetivo de facilitar a utilização para usuários não habituados a dispositivos móveis. As imagens apresentadas foram registradas com celulares Samsung Galaxy Ace Plus, realizando uma solicitação de serviço do endereço `Rua Olavo Bilac, Velha – Blumenau`, para o endereço `Rua João Pessoa, Velha – Blumenau`.

3.3.7.1 Aplicativo usuário

A aplicação do usuário possui basicamente apenas uma tela e todas as ações são geradas a partir de um botão.

Ao iniciar a aplicação (figura 14) o usuário é orientado através do texto disponível em tela a realizar a requisição do serviço selecionando o botão “Requisitar serviço”.

Neste caso, é necessária a utilização do GPS para obter a posição do dispostivo e de internet para obter a descrição do endereço do usuário. Caso o usuário não esteja com estes serviços habilitados, é apresentada opção para ativar os mesmos (figura 15). Ao confirmar a ativação, a aplicação conduz ao menu de configurações.

Figura 14 – Tela inicial do aplicativo usuário

Figura 15 – Ativação do GPS

Para os casos em que o serviços necessários foram ativados ou estavam previamente ativados, ao requisitar o serviço é apresentada tela para confirmação do endereço da requisição, conforme pode ser visto na figura 16.

Figura 16 – Tela inicial do aplicativo usuário

Figura 17 – Ativação do GPS

Ao confirmar o serviço, a requisição entra na lista de requisições do servidor, e é apresentada a mensagem “Serviço iniciado. Aguarde” (figura 17).

3.3.7.2 Aplicativo taxista

Ao abrir o aplicativo devem ser informados o login e a senha, conforme demonstrado na figura 18. Para qualquer ação na tela inicial do aplicativo, é necessária conexão com a internet. Caso a opção de internet não esteja habilitada no dispositivo, é apresentada a mensagem “É necessário acessar a internet para utilização deste aplicativo. Deseja ir para o menu de configurações?”, ao acessar o aplicativo, conforme visualização da figura 19.

Figura 18 – Tela inicial do aplicativo taxista

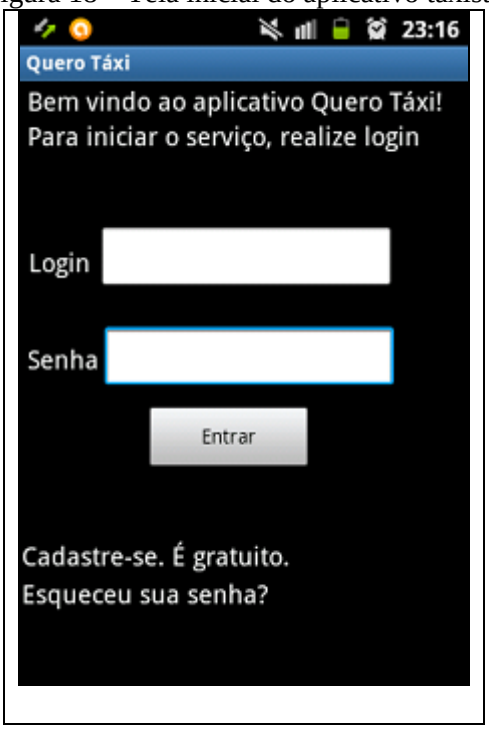


Figura 19 – Ativação da internet

Quando for iniciado o aplicativo do taxista pela primeira vez, o usuário não poderá acessar o serviço pois não possui um login para a aplicação. Desta forma, torna-se necessário realizar um cadastro, passando informações básicas sobre o usuário. Esta tela é demonstrada na figura 20.

Figura 20 – Cadastro de taxistas

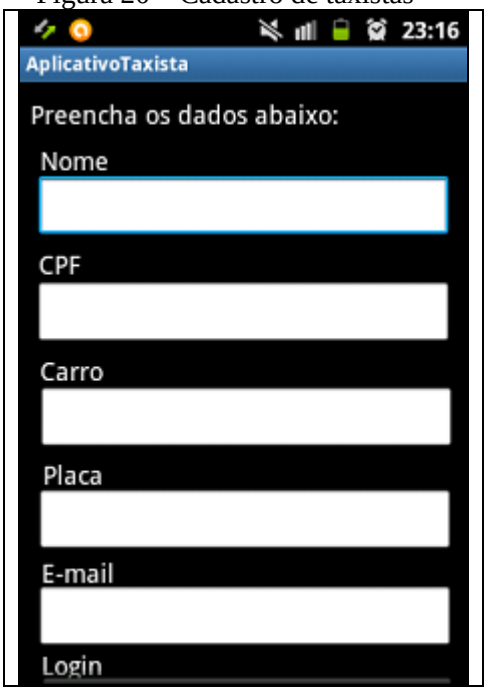


Figura 21– Reenvio de senha

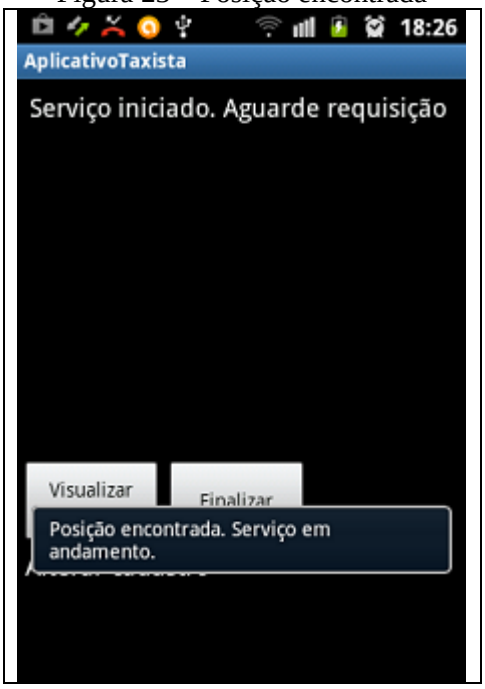
Ainda na tela inicial, caso o usuário tenha esquecido sua senha, é possível recuperá-la acessando a opcão “Esqueceu sua senha?” (figura 21). Será solicitado o e-mail do cadastro, e em seguida enviada as informações ao solicitante.

Para usuários que já possuam cadastro, ao informar o usuário e a senha, é aberta a tela para controle de serviços. Neste ponto da aplicação é necessária a utilização do GPS, juntamente com a internet.

O próximo passo consiste em aguardar a aplicação determinar a posição do dispostivo e posteriormente adicionar o mesmo na lista de taxistas disponíveis. A mensagem “Obtendo posição” (figura 22) é apresentada na tela, seguida de outra mensagem “Posição encontrada. Serviço em andamento” em caso de sucesso na obtenção da localização (figura 23).

Figura 22 – Obtendo posição

Figura 23 – Posição encontrada

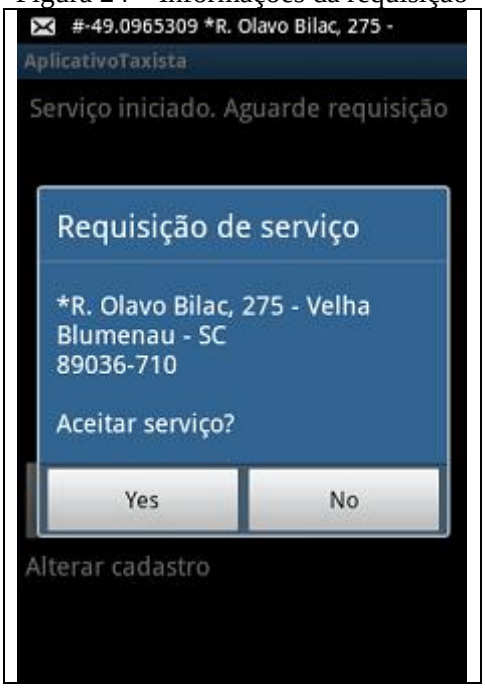


Uma vez definida a posicão, é necessário aguardar uma requisição. Nos casos em que não é apresentada nenhuma requisição de serviço para o taxista, a posição do dispositivo é atualizada, através do mesmo sistema de determinar a posição e adicionar na lista de taxistas.

Após o surgimento de uma requisição, é apresentada em tela uma opção para determinar se o serviço será ou não aceito. A mensagem apresenta informações do endereço de usuário requisitante, conforme pode ser verificado na figura 24.

Para os casos em que o serviço não for aceito, o sistema segue com a obtenção de posição e atualização do serviço.

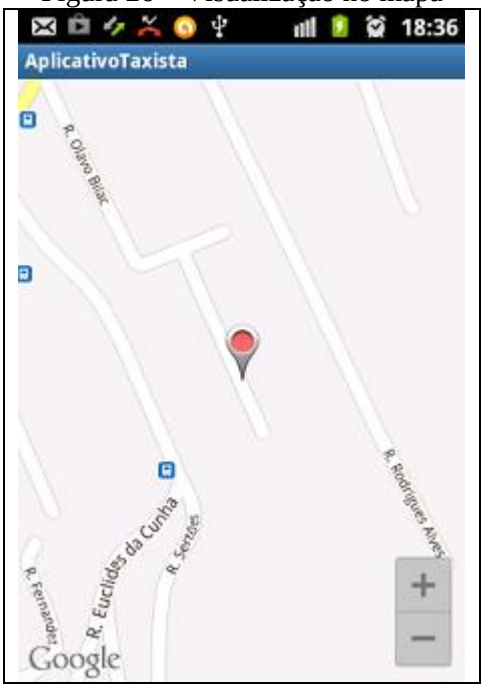
Figura 24 – Informações da requisição



Caso o serviço seja aceito, é gravado em tela o endereço da requisição para posterior consulta (figura 25). Se necessário o usuário pode consultar o endereço no mapa, através da opção “Visualizar mapa”. É exibida então a tela com o mapa da cidade, com um marcador sobre o endereço da requisicão (figura 26).

Figura 25 – Endereço da requisição

Figura 26 – Visualização no mapa



3.4 RESULTADOS E DISCUSSÃO

O presente trabalho apresentou o desenvolvimento de um sistema baseada em localização para realizar a requisição de serviços de táxi.

Neste sistema foi necessária a utilização de um servidor GSM, para controlar as mensagens SMS enviadas tanto pelo lado do taxista quanto do usuário requisistante. Para tal, foi utilizada a bibilioteca SMSLib, que permite o controle de envio e recepção de mensagens através de um modem GSM via aplicação *desktop*. No entanto, foram encontradas algumas dificuldades relativas a compatibilidade de dispositivos GSM com a biblioteca. Foram testados diversos modelos que possuem a dupla funcionalidade de modem GSM e celular, como por exemplo o Nokia 7230, Motorola Q11 e Nokia N95. Porém somente após os testes com o celular Sony Ericsson W200i foi possível utilizar plenamente o servidor GSM.

Para obter a atualização da localização dos usuários, primeiramente foi utilizado apenas o dispositivo GPS. Porém, nos testes realizados, foi verificado que este dispositvo apresenta certa demora e por muitas vezes não consegue definir a posição do dispositivo em ambientes fechados. Por sua vez a localização obtida através da WiFi mostrou-se rápida, porém sem a precisão do dispositivo GPS. Desta forma, optou-se por criar uma solução que abrangesse as duas situações. Os dois dispositivos devem estar ativos, sendo o dispotivo GPS prioritário em comparação a WiFi. Caso o dispositivo GPS não determine a posição, então é a utilizada a localização determinada pela rede WiFi.

Outro ponto analisado nos testes realizados diz respeito a performance do sistema. Infelizmente o principal empecilho referente a esta questão foi referente as redes GSM. Houve demasiada demora para realizar o envio, recebimento e processamento das solicitações. Por exemplo, por muitas vezes nos testes era enviada uma SMS de requisição e a mesma não era notificada no servidor GSM. Após enviar nova requisição e aguardar alguns minutos ambas as solicitações eram notificadas no servidor ao mesmo tempo, como se tivessem sido enviadas simultaneamente. Esta situação não pôde ser contornada, visto que não se trata de um problema na aplicação em si mas da infraestrutura da operadora utilizada nos testes.

Os principais testes realizados foram efetuados utilizando quatro dispositivos. Um dispositivo Sony Ericsson W200i para utilização como modem GSM e outros três dispositivos Samsung Galaxy Ace Plus para simular requisições. Os endereços utilizados para requisições foram da `Rua Iguaçu, Itoupava Seca – Blumenau` para a rua `São Paulo, Itoupava Seca – Blumenau` e rua `Olavo Bilac, Velha – Blumenau` para a rua `João Pessoa, Velha – Blumenau`. Todas as funcionalidades obtiveram um resultado positivo, sendo que sempre foi designado um taxista para o usuário requisitante.

O quadro 13 apresenta um comparativo entre as funcionalidades implementadas em relação aos trabalhos correlatos.

Quadro 13 – Comparação de resultados

funcionalidade	QueroTáxi	ResolveAí	TáxiMov	EasyTáxi
requisitar serviço por SMS	sim	não	não	não
requisitar serviço pelo dispositivo	sim	sim	não	sim
acompanhar pedido em tempo real	não	sim	sim	sim
visualizar localização do usuário no mapa	sim	sim	sim	sim
usuário escolher taxista	não	não	não	sim

4 CONCLUSÕES

O sistema desenvolvido permite que usuários realizem requisições de serviços de táxi através de *smartphones* e designa o taxista mais próximo à estes usuários, sendo que a localização do usuário requisitante pode ser visualizada pelo taxista nos mapas disponíveis pelo Google Maps.

O sistema atendeu a todos os objetivos específicos definidos, permitindo, por exemplo, o cadastro de novos usuários e o envio de mensagens georeferenciadas.

A plataforma Android mostrou-se bastante completa para o desenvolvimento de aplicações baseadas em localização, visto que a mesma possui diversas bibliotecas internas que auxiliam nesta questão. Outro ponto destacado refere-se a IDE utilizada para o desenvolvimento. O Eclipse, em conjunto com o Android SDK, se mostrou o mais apropriado, visto que possui *plugins* e emuladores capazes de simular uma posição e o uso de GPS.

Em relação aos trabalhos correlatos, pôde-se observar que o aplicativo desenvolvido não possui uma interface e funcionalidades tão robustas quanto os aplicativos comerciais, porém, certamente possui as principais características necessárias para a requisição do serviço.

A pesquisa e implementação proporcionaram ainda a chance de estudar e entender uma nova tecnologia em expansão, visto que a plataforma Android cada vez mais é utilizada em *smartphones* e *tablets*.

4.1 EXTENSÕES

Apesar dos resultados obtidos, o sistema ainda apresenta limitações e pontos que podem ser melhorados. Sugere-se:

a) desenvolver um módulo onde os usuários também devem realizar um cadastro, para consultas futuras e controle de requisições impróprias;
b) permitir que neste cadastro sejam adicionadas fotos dos passageiros;

c) desenvolver uma funcionalidade onde o usuário possa acompanhar o deslocamento do taxista em tempo real;
d) disponibilizar a aplicação para outros smartphones que possuam outro sistema operacional.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID DEVELOPERS. **LocationManager**. [S.l.], 2012a. Disponível em: <http://developer.android.com/reference/android/location/LocationManager.html>. Acesso em: 10 abr. 2013.

_____. **Obtaining user location**. [S.l.], 2012b. Disponível em: <http://developer.android.com/guide/topics/location/obtaining-user-location.html>. Acesso em: 10 abr. 2013.

_____. **Introduction to Google Maps**. [S.l.], 2013c. Disponível em: <https://developers.google.com/maps/documentation/android/intro>. Acesso em: 25 abr. 2013.

_____. **Map objects**. [S.l.], 2013d. Disponível em: <https://developers.google.com/maps/documentation/android/map >. Acesso em: 26 abr. 2013.

_____. **Google Maps API v2 key**. [S.l.], 2013d. Disponível em: <https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2>. Acesso em: 26 abr. 2013.

APOLICE. **Sistema TaxiMov**. [S.l.], 2013. Disponível em: < http://revistaapolice.com.br/2013/04/sistema-taximov-permite-solicitacao-online-de-taxi/>. Acesso em: 10 abr. 2013.

BAULER, André G. **Sistema de controle para empresa prestadora de serviço utilizando envio/recebimento de SMS**. 2011. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.inf.furb.br/~pericas/orientacoes/SMS2011.pdf>. Acesso em: 05 jun. 2013.

COULORIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Systems:** concepts and design. 4. ed. Londres, 2005.

DA COSTA, Bruno L. de C.; DA COSTA, Fabiene C. de C. **O sistema de táxis:** mobilidade urbana e redução nas emissões de gases de efeito estufa no rio de janeiro**.** In: PLURIS: 2010 – The Challenges of Planning in a Web Wide World, 2010, Faro – Portugal. Analis do Pluris 2010. Faro – Portugal, 2010.

GASPARINI, André; CAMPOS, Vânia B. G.; D'AGOSTO, Márcio de A. Modelos para estimativa da demanda de viagens de veículos de carga para supermercados e shopping-centers. **Transportes**, v. XVIII, n. 1, p. 57-64, mar. 2010.

HORD. Jennifer. **Como funciona o SMS**. [S.l.], 2010. Disponível em: <http://marcustaboza.blogspot.com.br/2010/02/como-funciona-o-sms.html>. Acesso em: 05 jun. 2013.

IPEA. **Mobilidade urbana**. [S.l.], 2011. Disponível em: <http://www.ipea.gov.br/portal/images/stories/PDFs/SIPS/110504_sips_mobilidadeurbana.pdf>. Acesso em: 3 set. 2011.

MAGUNI, Reynaldo L.; NAVARRO, Mel L. P.; ROSARIO, Kathleen R. **chitSMS:** community health information tracking system using short message service**.** 2010. 61 f. Department of Computer Science - College of Engineering. University of the Philippines.

MONICO, João F. G. **Posicionamento pelo NAVSTAR-GPS –** Descrição, Fundamentos e aplicações. 1. ed. São Paulo: São Paulo, 2000.

RESOLVEAI. **Aplicativo ResolveAí**. [S.l.], 2012a. Disponível em: < http://www.resolveai.com.br/#!/about-us>. Acesso em: 12 maio 2013.

_____. **Aplicativo ResolveAí**. [S.l.], 2012b. Disponível em: < http://www.resolveai.com.br/#!/help>. Acesso em: 12 maio 2013.

SEGS. **Aplicativo EasyTaxi.** [S.l.], 2013. Disponível em: <http://www.segs.com.br/index.php?option=com_content&view=article&id=117287:-easy-taxi-leva-servico-de-taxi-pelo-celular-para-porto-alegre&catid=71:categoria-veiculos&Itemid=367>. Acesso em: 10 maio 2013.

SMSLib. **About SMSLib**. [S.l.], 2013. Disponível em: <http://smslib.org/doc/about/>. Acesso em: 5 maio 2013.

USHISIMA, Ricardo. **Consumindo web servisse em aplicações Android**. [S.l.], 2011. Disponível em: < http://zbra.com.br/2011/03/30/consumindo-web-service-em-aplicacoes-android/>. Acesso em: 3 abr. 2013.

VINICIUS, Bruno. **Transcrevendo SoapObjects através de annotations & reflection**. [S.l.], 2011. Disponível em: <http://zbra.com.br/2011/07/06/androidksoap-pt-1-transcrevendo-soapobjects-atraves-de-annotations-reflection/>. Acesso em: 3 abr. 2013.