MARCELO CABRAL DE SOUZA

APLICATIVO MÓVEL PARA MONITORAMENTO DA EXECUÇÃO DE MANDADOS POR OFICIAIS DE JUSTIÇA

FLORIANÓPOLIS, 2013

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

MARCELO CABRAL DE SOUZA

APLICATIVO MÓVEL PARA MONITORAMENTO DA EXECUÇÃO DE MANDADOS POR OFICIAIS DE JUSTIÇA

Trabalho de conclusão de curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de especialista.

Professor Orientador: Luis Carlos Martinhago Schlichting, Doutor.

FLORIANÓPOLIS, 2013

CDD 629.895

S729a

Souza, Marcelo Cabral de

Aplicativo móvel para monitoramento da execução de mandados por oficiais de justiça [MP] / Marcelo Cabral de Souza; orientação de Luis Carlos Martinhago Schlichting– Florianópolis, 2013.

1 v.: il.

Monografia de Pós-Graduação (Desenvolvimento de Produtos Eletrônicos) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Inclui referências.

1. Monitoramento. 2. Android. 3. Poder judiciário de Santa Catarina. I. Schlichting, Luis Carlos Martinhago. II. Título.

Sistema de Bibliotecas Integradas do IFSC
Biblioteca Dr. Hercílio Luz – Campus Florianópolis
Catalogado por: Ana Paula F. Rodrigues Pacheco CRB 14/1117

APLICATIVO MÓVEL PARA MONITORAMENTO DA EXECUÇÃO DE MANDADOS POR OFICIAIS DE JUSTIÇA

MARCELO CABRAL DE SOUZA

Este trabalho foi julgado adequado para obtenção do Título de Especialista e aprovado na sua forma final pela banca examinadora do Curso de Especialização em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 18 de dezembro de 2013.

Banca Examinadora:

_____
Luis Carlos Martinhago Schlichting, Doutor

_____
Fernando Santana Pacheco, Doutor

_____
Hugo Marcondes, Mestre

AGRADECIMENTOS

Agradeço ao apoio incondicional prestado por minha esposa Michele, principalmente pelo presente maravilhoso que recebi durante o decorrer do curso, minha filha Antônia.

Agradeço aos meus pais, Tolentino e Maria, pelos incentivos e palavras de apoio.

Agradeço as minhas irmãs, Fernanda e Fabiana, e seus maridos, Murilo e Jairo, e também minha sobrinha Jackelyne.

Agradeço ao Professor Doutor Luis Carlos Martinhago Schlichting, pela paciência e por ter se disponibilizado em ser meu orientador.

Essa não é mais uma carta de amor
São pensamentos soltos
Traduzidos em palavras
Pra que você possa entender
O que eu também não entendo
Jota Quest

RESUMO

Este trabalho de conclusão de curso tem como objetivo desenvolver um aplicativo móvel para auxiliar o Poder Judiciário em sua principal atividade, a prestação jurisdicional. O aplicativo foi desenvolvido utilizando a plataforma Android e permite acompanhar as atividades previstas para os Oficiais de Justiça. Além de possibilitar realizar a coleta de horário e coordenadas geográficas, no momento em que o Oficial de Justiça cumprir um mandado judicial, o aplicativo também oferece a possibilidade de efetuar a aquisição de dados relativos ao deslocamento do Oficial de Justiça, com o intuito de avaliar o trajeto feito durante a execução de suas atividades. Os resultados obtidos, a partir dos dados coletados nos testes realizados simulando as atividades de um Oficial de Justiça, confirmaram que a solução proposta atingiu seus objetivos.

Palavras-Chave: Poder Judiciário. Oficial de Justiça. Monitoramento. Aplicativos Móveis. Android.

ABSTRACT

This coursework aims to develop a mobile application to assist the Judicial Branch in its principal activity, the jurisdictional provision. The application was developed using the Android platform and allows you to track the activities planned for the Bailiffs. Besides enabling to collect the time and geographic coordinates, the time that the Bailiff serve a court order, the application also offers the possibility of making the acquisition of data on the displacement of the bailiff, in order to assess path made during the execution of their activities. The results obtained from the data collected in tests simulating the activities of a Bailiff, confirmed that the proposed solution reached their goals.

Keywords: Judicial Branch. Bailiffs. Monitoring. Mobile Applications. Android.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AJAX - Asynchronous JavaScript and XML
API - Application Programming Interface
CNJ - Conselho Nacional de Justiça
DCOM - Distributed Component Object Model
EJB - Enterprise JavaBeans
HTML - HyperText Markup Language
HTTP - Hypertext Transfer Protocol
JPA - Java Persistence API
JSF - JavaServer Faces
JSON - JavaScript Object Notation
JSP - JavaServer Pages
LBS - Location-Based Services
OHA - Open Handset Alliance
RPC - Remote Procedure Calls
SDK - Software Development Kit
SGBD - Sistema Gerenciador de Banco de Dados
SOAP - Simple Object Access Protocol
SQL - Structured Query Language
UDDI - Universal Description, Discovery and Integration
WSDL - Web Service Description Language
XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A inclusão de produtos eletrônicos para auxílio em nossas tarefas diárias, sejam domésticas ou profissionais, tem permitido uma revolução em nossos costumes. A evolução tecnológica apresentada nos dias atuais permite a possibilidade de desenvolvermos equipamentos cada vez menores, onde a necessidade de estar em posse do mesmo não se traduza em incômodo. A miniaturização e a combinação de diversos módulos em um único hardware, além da criação de um software que permita o acesso as diversas funcionalidades fornecidas por este dispositivo facilitaram por demais a vida de um desenvolvedor.

A criação de um dispositivo de monitoramento necessita de módulos GPS e GPRS, além do desenvolvimento de um *software* para controle e integração destes módulos. Porém, da evolução tecnológica comentada anteriormente temos o advento dos *smartphones* e dispositivos similares, como *phoblets* e *tablets*, que já trazem integrados todos estes módulos, além de possuir um sistema operacional padrão rodando em diversos dispositivos de diversos fabricantes.

Segundo MEGDA, 2012, “nos últimos anos o mercado de rastreamento e monitoramento de veículos tem crescido exponencialmente”. O autor esclarece que para o acompanhamento em tempo real da situação de um objeto a ser monitorado a dupla GPS / GPRS é a mais comumente utilizada na área de monitoramento. É de conhecimento que diversas soluções estão disponíveis no mercado e a criação de mais uma deve ser bem planejada. No escopo deste trabalho de conclusão de curso estaremos focados um pouco mais além das soluções disponíveis no mercado, pleitearemos também a confirmação do cumprimento de mandados judiciais e sua validação por meio da confirmação *in loco* através da utilização de um dispositivo móvel.

1.1 Justificativa

Devido a adoção de metas niveladoras por parte do Conselho Nacional de Justiça (CNJ) visando o atendimento mais célere por parte dos tribunais, uma vez que só tende a aumentar as demandas da população na busca por seus direitos por meio do Poder Judiciário, assim como a constante necessidade por uma justiça mais transparente, é sabido que ações que busquem melhorar a prestação jurisdicional são sempre bem-vindas.

Um ator intimamente ligado a estas questões é o Oficial de Justiça, responsável por cumprir mandados relativos a diligências fora de cartório, entre outras atividades. É certo que o desempenho de suas funções exige muita dedicação e esmero, recebendo assim um enorme respaldo junto ao Poder Judiciário. Porém, por exercerem atividade de extrema importância no âmbito judicial, devem ter sua atuação em relação a prestação jurisdicional fiscalizada, assim como sua produtividade deve ser avaliada rotineiramente.

Dadas as necessidades acima descritas, a proposta de um aplicativo móvel que efetue o monitoramento da execução de mandados pelos Oficiais de Justiça pode ser justificada. Permitir realizar de forma simples e objetiva, ou seja, simplesmente portando o dispositivo móvel enquanto este realiza a captura das informações de deslocamento. Assim como, quando do cumprimento de mandados o Oficial de Justiça acionar e indicar no dispositivo qual mandado está cumprindo, permitem a obtenção de informações valiosas que poderão ser utilizadas futuramente para diversos tipos de avaliação.

1.2 Definição do Problema

Este projeto nasceu da necessidade do acompanhamento do cumprimento de mandados judiciais realizado por Oficiais de Justiça no Poder Judiciário de Santa Catarina de forma célere, eficiente e efetiva. Visando a confirmação do deslocamento e a tentativa de cumprimento

de um mandado pelo Oficial de Justiça, buscamos desenvolver um produto capaz de realizar estas confirmações de forma transparente e que necessite interação humana mínima, por parte do Oficial de Justiça, apenas nos momentos em que o mesmo cumpra o que foi determinado pelo mandado que esteja portando.

A confirmação do deslocamento e cumprimento é feita mediante o monitoramento da localização de um dispositivo a ser portado pelo Oficial de Justiça. Este dispositivo deve possuir mecanismos que permitam adquirir o posicionamento geográfico do equipamento, compulsoriamente um receptor GPS, além de possuir mecanismos de comunicação com um servidor web, para envio e recebimento de informações. Neste caso existem diversos meios para realizar a comunicação com o dispositivo, como bluetooth, redes ethernet, redes wireless ou pacotes de dados via telefonia celular.

A necessidade do monitoramento, através do rastreio das coordenadas geográficas e horário, advém de duas situações hipotéticas a serem implantadas no cotidiano do Oficial de Justiça. Em uma primeira situação, planeja-se a possibilidade da cessão de um automóvel ao Oficial de Justiça para realizar o cumprimento dos mandados judiciais. Atualmente cada Oficial de Justiça faz uso do seu próprio veículo e percebe uma indenização em compensação ao desgaste sofrido pelo seu veículo durante este deslocamento. Com a cessão de um veículo pelo Poder Judiciário, o monitoramento do deslocamento permitiria a verificação do real uso do veículo e se a utilização do mesmo estaria sendo realizada de forma vinculada ao cumprimento dos mandados judiciais.

Em uma segunda situação, os valores pagos a título de condução ao Oficial de Justiça são realizados independente do deslocamento realizado e quantos forem os mandados cumpridos, de posse das reais distâncias percorridas pelo Oficial de Justiça seria possível realizar o pagamento proporcional a distância percorrida similarmente ao definido pela resolução 380 (TRIBUNAL PLENO DO TRIBUNAL DE JUSTIÇA DO ESTADO DE MATO GROSSO DO SUL, 2002). O monitoramento permitiria a coleta dos dados necessários para a criação de uma sistemática que permitisse a mudança da forma de cálculo do pagamento da condução tornando mais justo o pagamento destes valores. Essas questões estão relacionadas à economia e à melhoria da qualidade da prestação do serviço público ao jurisdicionado, dada a

possibilidade da transparência da real situação do cumprimento dos mandados judiciais e o esforço envolvido para a realização do mesmo.

1.3 Objetivo Geral

Desenvolver um aplicativo móvel para monitoramento da execução de mandados por Oficiais de Justiça.

1.4 Objetivos Específicos

Dentro do contexto discutido até o momento, elencamos os seguintes objetivos específicos a serem atingidos:

- Revisar a bibliografia sobre o tema;
- Conhecer técnicas para desenvolvimento de aplicativos móveis Android;
- Desenvolver um aplicativo móvel Android;
- Criar um *web service* para recepção e envio de informações de monitoramento;
- Simular o rastreio do deslocamento de um Oficial de Justiça utilizando as coordenadas geográficas obtidas através do aplicativo móvel e disponibilizando estes dados em um mapa;
- Simular a localização dos mandados a cumprir e cumpridos utilizando as coordenadas geográficas obtidas através do aplicativo móvel e disponibilizando estes dados em um mapa.

2 DESENVOLVIMENTO

Este projeto busca o desenvolvimento de um aplicativo para dispositivos móveis, a fim de auxiliar o Poder Judiciário no intuito da obtenção de informações precisas acerca do cumprimento de mandados judiciais por meio dos Oficias de Justiça, além de informações sobre o uso, em caso de cessão, dos veículos conduzidos por estes.

Para atingir tal objetivo, no capítulo 2.1 Revisão da Literatura, será realizada uma pesquisa bibliográfica sobre as características que compõem as atividades de um Oficial de Justiça e o como se dá o cumprimento de um mandado, partindo do princípio da prestação jurisdicional sob cargo do Poder Judiciário. Posteriormente serão introduzidos os tipos de dispositivos móveis existentes e que poderiam auxiliar na consecução das tarefas a serem realizadas pelo Oficial de Justiça. Adiante, seguiremos aprofundando nosso conhecimento acerca da plataforma Java, que apresenta uma linguagem de programação compartilhada com a plataforma Android. Demonstraremos também como esta tecnologia foi desenvolvida, assim como outras tecnologias envolvidas no desenvolvimento de um aplicativo para um dispositivo móvel rodando Android. Posteriormente abordaremos o desenvolvimento de aplicações através de banco de dados, *web service* e tecnologias relacionadas.

No capítulo 2.2 Metodologia demonstramos como o aplicativo proposto foi desenvolvido utilizando o conhecimento adquirido através da revisão da literatura. Além disso, é exibido um módulo para recepção dos dados capturados, incluindo uma ferramenta simples para visualização dos dados coletados em um mapa acessível a partir de um navegador *web*.

Por fim, no capítulo 2.3 Apresentação dos Resultados serão apresentados os testes realizados a partir do aplicativo móvel, exibindo os dados coletados por meio da captura das coordenadas geográficas durante uma simulação de deslocamento de um Oficial de Justiça, assim como a simulação do cumprimento de mandados realizando checagens em locais pré-determinados.

2.1 Revisão da Literatura

Para que este projeto possa ser desenvolvido com clareza e simplicidade, diversos conceitos devem ser compreendidos tornando-se necessário analisar os seguintes tópicos:

- Poder Judiciário;
- Dispositivos móveis;
- Java;
- Android;
- Banco de dados;
- Web service.

2.1.1 Poder Judiciário

Segundo SERAFIM, 2012, o Poder Judiciário tem como função básica a prestação jurisdicional através da figura do magistrado, que no caso do direito processual civil, decide sobre uma determinada pretensão que há entre as partes, ou resolve uma lide que existe entre estas. Porém para que a função jurisdicional seja realizada, além do juiz são necessários, conforme definido no Código de Processo Civil, a atuação dos auxiliares da justiça quais sejam, o escrivão, o oficial de justiça, o perito, o depositário, o administrador e o intérprete, conforme disposto pelo Código de Processo Civil em seus artigos de 139 a 153.

2.1.1.1 Oficial de Justiça

As atividades do oficial de justiça são detalhadas por JÚNIOR, 2009, p. 211, que diz que o oficial "é o antigo meirinho, o funcionário do juízo que se encarrega de cumprir os mandados relativos a diligências fora de cartório, como citações, intimações, notificações, penhoras, sequestros, busca e apreensão, imissão na posse, condução de

testemunhas e etc. Sua função é subalterna e consiste apenas em cumprir ordens dos juízes, as quais, ordinariamente, se expressam em documentos escritos que recebem a denominação de mandados.”.

O mesmo JÚNIOR, 2009 resume as funções do oficial de justiça como as de um “mensageiro e executor das ordens judiciais” sendo incumbido ao mesmo o que dispõe no art.143 do Código de Processo Civil, “fazer pessoalmente as citações, prisões, penhoras, arrestos e mais diligências próprias do seu ofício, certificando no mandado o ocorrido, com menção de lugar, dia e hora” e “entregar, em cartório, o mandado, logo depois de cumprido”.

Podemos encontrar outras atividades atribuídas ao oficial de justiça definidas na Constituição Federal, no Código de Processo Civil, Código de Processo Penal e demais leis esparsas. Além disso encontramos nos códigos de normas editados pelas Corregedorias da Justiça, que tendem a regular situações peculiares, com relação à forma pela qual as normas legais devem ser observadas.

De acordo com o inciso V do art. 143 do Código de Processo Civil alguns tribunais passaram a denominar o oficial de justiça pelo termo Oficial de Justiça e Avaliador. Atentando-se ao ato normativo 0007097-66.2009.2.00.0000 do CONSELHO NACIONAL DE JUSTIÇA, 2009, que baseado na resolução 48/07 desse mesmo conselho analisou as atividades do Oficial de Justiça.

Alguns conceitos relacionados as atividades dos oficiais de justiça estão disponíveis no artigo 2º da resolução 380 do TRIBUNAL PLENO DO TRIBUNAL DE JUSTIÇA DO ESTADO DE MATO GROSSO DO SUL, 2002:

> [...]
> II - mandado é a determinação imperativa, escrita, emanada de autoridade judiciária, para cumprimento de decisões ou de atos judiciais. O mandado pode conter um ou mais atos judiciais, um ou mais destinatários, um ou mais endereços e, para sua execução plena, pode ser necessário um ou mais deslocamentos;
> III - ato judicial é aquele ato externo, praticado pelo oficial de justiça e avaliador, em cumprimento de mandados ou de decisões judiciais, pelo qual se produzem os efeitos legais

> de ordem judicial, servindo, ao mesmo tempo, de instrumento e de prova material de sua existência;
> IV - destinatário é todo aquele a quem se destina a ordem judicial a ser cumprida;
> V - diligência é a execução de certos serviços judiciais, emanados por escrito de autoridades superiores, para serem cumpridos fora dos cartórios ou do Tribunal de Justiça;
> VI - deslocamento é o ato ou o efeito de deslocar-se. Para a consecução plena de um mandado, pode haver um ou mais deslocamentos, consoante hajam um ou mais endereços a serem localizados.
> [...]

2.1.1.2 Mandados

Buscando entender melhor o conceito de mandado destacam-se os arts. 225 a 227 do Código de Processo Civil:

> Art. 225. O mandado, que o oficial de justiça tiver de cumprir, deverá conter: (Redação dada pela Lei nº 5.925, de 1º.10.1973)
> I - os nomes do autor e do réu, bem como os respectivos domicílios ou residências;(Redação dada pela Lei nº 5.925, de 1º.10.1973)
> [...]
> IV - o dia, hora e lugar do comparecimento; (Redação dada pela Lei nº 5.925, de 1º.10.1973)
> [...]
> Art. 226. Incumbe ao oficial de justiça procurar o réu e, onde o encontrar, citá-lo:
> I - lendo-lhe o mandado e entregando-lhe a contrafé;
> II - portando por fé se recebeu ou recusou a contrafé;
> III - obtendo a nota de ciente, ou certificando que o réu não a apôs no mandado.
> Art. 227. Quando, por três vezes, o oficial de justiça houver procurado o réu em seu domicílio

> ou residência, sem o encontrar, deverá, havendo suspeita de ocultação, intimar a qualquer pessoa da família, ou em sua falta a qualquer vizinho, que, no dia imediato, voltará, a fim de efetuar a citação, na hora que designar.
> [...]

Adicionalmente o Comunicado Eletrônico 50 de autoria da CORREGEDORIA-GERAL DA JUSTIÇA, 2013 define que "o Oficial de Justiça, ao devolver os mandados ao Cartório ou à Central de Mandados, deverá atualizar no sistema a situação do mandado.[...]". As situações foram definidas com os seguintes entendimentos:

> a) Cumprido – Ato positivo: aquele cuja ordem foi executada na íntegra, ou que, contendo ordens sucessivas e, uma delas tendo sido cumprida, tenha esgotado o objeto do mandado. Exemplo: mandado de citação – citação efetuada = mandado cumprido; mandado de citação e penhora – citação efetuada e parcelamento do débito = cumprido.
> b) Cumprido – Ato positivo parcial: o que, contendo mais de uma ordem, tenha sido devolvido com uma ou mais ordens não executadas. Exemplo: mandado de citação, penhora e avaliação – realizadas a citação e a penhora, mas não a avaliação = mandado parcialmente cumprido.
> c) Cumprido – Ato negativo: aquele em que nenhuma ordem foi executada, porém houve diligência.
> d) Devolvido sem cumprimento: aquele em que nenhuma ordem foi executada e não houve diligência. Exemplo: mandado encaminhado para o oficial errado, ou faltando informações, dentre outras. Neste caso, o oficial certifica que não houve o cumprimento do mandado e devolve para a central.

2.1.2 Dispositivos Móveis

Esta seção do trabalho é dedicada a apresentação dos tipos de dispositivos móveis encontrados atualmente no mercado. A definição de como cada dispositivo móvel se classifica entre os tipos apresentados, *smartphone*, *tablet* ou *phoblet*, é considerada abstrata e baseada em como as fabricantes de dispositivo desejam vender seus produtos. Segundo CNET, 2013, existe um certo consenso de que a classificação do dispositivo estaria ligada ao tamanho de sua tela, de 3 a 5 polegadas para *smartphones*, 5 a 7 polegadas para *phoblets* e acima de 7 polegadas para *tablets*.

Smartphone é uma categoria de dispositivo móvel que oferece recursos avançados que extrapolam as características de um telefone celular típico. Em comparação aos telefones convencionais, *smartphones* geralmente têm telas maiores e processadores mais potentes. São caracterizados por executar um sistema operacional que fornece uma interface padronizada e uma plataforma para desenvolvimento de aplicativos. Os aplicativos escritos para uma determinada plataforma de *smartphone* normalmente podem ser executados em qualquer *smartphone* com a mesma plataforma, independentemente do fabricante (PHONESCOOP, 2013).

Segundo PC MAGAZINE, 2013, *tablet* é um computador de uso geral, contido em uma única peça na forma de uma prancheta, tendo como característica a presença de uma tela sensível ao toque, como o dispositivo de entrada. Além disso, pode fazer uso de botões físicos ou sensores, como acelerômetros, presentes no dispositivo para suplementar a falta de mecanismos de entrada. Teclados virtuais podem ser projetados na tela em complementação à falta de um teclado físico, além disso é possível realizar a conexão de dispositivos externos, como teclados, em substituição aos métodos de entrada disponibilizados pelo *tablet*, porém prejudicando a facilidade de uso e transporte.

Phoblet ou *Phablet* é uma amálgama entre as palavras *Phone* e *Tablet*, utilizadas para designar um dispositivo *crossover* com tela *touch screen* maiores que entre 5 e 7 polegadas. As telas deste dispositivo híbrido situam-se numa posição intermediária, geralmente maiores que as telas de *smartphones*, não possuindo dimensões superiores aos de um *mini-tablet*. Em relação aos recursos disponíveis são os mesmos

pressentes nos dispositivos coirmãos (FORBES, 2013).

2.1.3 Java

A tecnologia Java é tanto uma linguagem de programação quanto uma plataforma. A linguagem de programação Java é caracterizada por ser simples, orientada a objetos, portável, dinâmica, entre outras características. Um programa desenvolvido nesta linguagem é considerado portável por ser capaz de rodar em diversos sistemas operacionais, como Windows, Solaris, Mac OS e Linux, a partir de uma máquina virtual Java (ORACLE, 2013a).

Uma plataforma é um ambiente de hardware ou software em que uma aplicação é executada. A plataforma Java é caracterizada por possuir dois componentes: uma máquina virtual e uma interface de programação de aplicativos (*Application Programming Interface* - API). A máquina virtual é a base da plataforma Java, trata-se de um programa, para uma determinada plataforma de hardware ou software, em que uma aplicação desenvolvida na linguagem de programação Java é executada. Já uma API é uma coleção de componentes de software que disponibilizam recursos úteis e podem ser utilizadas para criar outros componentes ou aplicativos de *software* (ORACLE, 2013a).

A plataforma Java pode ser dividida em quatro outras plataformas conforme o segmento da aplicação a ser desenvolvida, são elas:

- Java Platform, Standard Edition (Java SE): provê todas as funcionalidades básicas para a linguagem de programação Java, nela é definida dos tipos básicos da linguagem até classes de alto nível para manipulação de banco de dados, acesso a rede, segurança e interface gráfica;
- Java Platform, Enterprise Edition (Java EE): além de fornecer todas funcionalidades da plataforma Java SE, também permite o desenvolvimento de aplicações distribuídas e com requisitos de segurança, escalabilidade e confiabilidade, em suma, é voltado para aplicações corporativas e para internet;
- Java Platform, Micro Edition (Java ME): esta plataforma possui apenas uma parte das funcionalidades da Java SE, é

focada no desenvolvimento para dispositivos móveis e embarcados, possuindo adicionalmente alguns recursos extras para o seu segmento;

- JavaFX: é uma plataforma para a criação de aplicações visualmente mais elaboradas, voltadas para internet, usando uma API reduzida. Aplicativos desenvolvidos nesta plataforma podem ser clientes de serviços da plataforma Java EE.

2.1.3.1 Java EE

Podemos denominar as aplicações desenvolvidas na plataforma Java EE por aplicações corporativas ou empresariais, uma vez que esta plataforma foi projetada para resolver os problemas enfrentados em sua maioria por empresas. Esta plataforma busca atingir grandes corporações, agências e governos, assim como desenvolvedores individuais e pequenas organizações. Ela foi projetada para reduzir a complexidade do desenvolvimento de aplicações, oferecendo um modelo de desenvolvimento e diversas APIs (ORACLE, 2013b).

O modelo de desenvolvimento do Java EE é baseado em camadas, ou seja, a funcionalidade do aplicativo é dividida em áreas funcionais isoladas. A plataforma Java EE possui quatro camadas, quais sejam, camada cliente, camada *web*, camada de negócios e camada de dados. Na camada cliente podemos ter um navegador *web*, uma aplicação autônoma ou outros servidores realizando solicitações ao servidor de aplicações Java EE. Este servidor processa os pedidos e retorna uma resposta de volta ao cliente. A camada *web* é responsável por gerar conteúdo dinâmico em vários formatos para a camada cliente, coletar entrada de dados dos usuários e retornar resultados apropriados remetidos pela camada de negócios, além de controlar o fluxo de telas ou páginas no cliente. Diversas tecnologias são utilizadas na camada *web*, como Servlets e JavaServer Pages (JSP). A camada de negócios fornece a lógica de negócios para um aplicativo, ou seja, um código que fornece a funcionalidade de um domínio de negócio em particular. Nesta camada utilizam-se tecnologias como Enterprise JavaBeans (EJB), Java Persistence API (JPA) e *web service*. Por fim, a camada de dados, composta por servidores de banco de dados, sistemas de planejamento

de recursos empresariais e outras fontes de dados legados, como mainframes, normalmente localizados em uma máquina separada do servidor de aplicações Java EE (ORACLE, 2013b).

2.1.3.2 Servlet

Servlet é uma classe da linguagem Java utilizada pelos desenvolvedores que necessitam responder requisições *Hypertext Transfer Protocol* (HTTP) e gerar conteúdo dinâmico. Apesar de uma servlet poder responder a qualquer tipo de requisição elas são comumente utilizadas para estender os aplicativos hospedados em servidores web. Ela é uma das tecnologias Java EE mais antigas e serve de base para outras tecnologias como JSP e *frameworks* como Struts e JavaServer Faces (JSF) (ORACLE,2013c).

2.1.3.3 JavaServer Pages

JSP é um documento de texto que contém dois tipos de conteúdo: dados estáticos, que podem ser expressos em qualquer formato baseado em texto e elementos JSP, que permitem construir conteúdo dinâmico. O diferencial apresentado pelas páginas JSP é a separação da interface do usuário da geração de conteúdo, a partir da lógica de negócio. Sendo JSP uma extensão da tecnologia servlet, toda página JSP ao ser compilada no servidor que a hospeda gera um código Java, visto que a mesma é uma abstração de alto nível desta tecnologia. Baseado nisso, utilizando JSP é possível produzir aplicações que acessem banco de dados, manipulem arquivos, capturem informações a partir de formulários, além de dados dos usuários, dos navegadores utilizados para acessar a página e do servidor que está executando a mesma (ORACLE,2013d).

2.1.3.4 Enterprise JavaBeans

EJB é uma arquitetura de componentes para servidores de aplicações Java EE que permite o desenvolvimento rápido e simplificado para aplicações distribuídas, transacionais, seguras e portáveis. A especificação 2.1 disponibilizou o suporte a *web service*, permitindo a criar facilmente aplicativos voltados para esta tecnologia, porém apresentava como ponto fraco sua tecnologia de persistência de dados, conhecida por *entity beans*. Persistência significa manter o estado de um objeto além do período em que um aplicativo ou processo está ativo em um servidor de aplicações. Persiste-se a informação basicamente através de um banco de dados, apesar de existir muitas outras formas de se persistir dados. Na especificação 3.0 agregou-se a tecnologia JPA, destinada a gestão de persistência e mapeamento objeto / relacional, com a intenção de isolar completamente os desenvolvedores da tarefa de lidar diretamente com persistência (ORACLE,2013e).

ORACLE, 2013f esclarece que devemos considerar o uso de EJB quando uma aplicação possuir um dos seguintes requisitos:

- A aplicação deve ser escalável, ou seja, permite acomodar um número crescente de usuários, sendo necessário distribuir componentes do aplicativo em várias máquinas além de garantir que a localização física destas máquinas ficará transparente para os usuários;
- A aplicação deve ser transacional, ou seja, garantir a integridade dos dados em operações com acesso simultâneo a objetos compartilhados;
- A aplicação deve permitir acesso a uma variedade de clientes, ou seja, com apenas algumas linhas de código os clientes remotos podem facilmente localizar os EJBs.

A princípio, não teremos necessidade de utilizar EJB uma vez que nenhum dos requisitos acima seriam necessários para a consecução das atividades deste projeto. Em relação ao primeiro requisito, apenas um servidor de aplicações será utilizado para executar a aplicação a ser desenvolvida, trata-se de um *web service* para acesso ao banco de dados.

O segundo requisito está relacionado ao acesso simultâneo a objetos compartilhados, o que não está no escopo deste trabalho, já que acessos ao servidor disponibilizarão apenas informações individualizadas. E por fim, nossos clientes serão apenas dispositivos móveis, de forma que não haveria necessidade de atendermos a este último requisito.

2.1.3.5 Servidor de aplicação Java EE

Um servidor Java EE é um servidor de aplicações que implementa as APIs da plataforma Java EE e fornece seus serviços de forma padronizada. Vários tipos de componentes podem ser hospedados nestes servidores, cada um deles correspondendo aos diversos níveis de uma aplicação multicamadas. Os componentes serão atendidos pelo servidor Java EE através de um *container*, sendo este uma interface entre o componente e a funcionalidade de mais baixo nível fornecida pela plataforma. O *container web* é a interface entre os componentes *web* e o servidor *web*, estes componentes podem ser uma servlet ou uma página JSP. Já o *container* EJB é uma interface entre os EJBs e o servidor Java EE. Entre os servidores de aplicação compatíveis com alguma versão da especificação do Java EE, implementando-a total ou parcialmente, temos: WebSphere, WebLogic, GlassFish, Jboss, Apache Tomcat, entre outros (ORACLE,2013g).

É importante salientar que o servidor de aplicações Apache Tomcat é uma implementação parcial da especificação Java EE. Apenas o *container web* foi desenvolvido, dessa forma apenas aplicações baseadas em servlet e JSP rodam neste servidor, não sendo considerado portanto um servidor de aplicações Java EE que implemente a especificação completa da tecnologia (APACHE, 2013).

2.1.4 Android

Android é um sistema operacional e uma plataforma para desenvolvimento de *software* para dispositivos móveis. Ela foi desenvolvida pela Android Inc. e o responsável pela criação da mesma é

Andy Rubin. A empresa foi adquirida pela Google em 2005 e em conjunto com a *Open Handset Alliance* (OHA), um consórcio formado por empresas como LG, Nextel, TIM, Telefonica, Vodafone, Asus, Dell, Samsung, Intel, Texas, eBay, entre outras, mantém a plataforma em constante desenvolvimento. A OHA é uma aliança de negócios composta por empresas da área de telefonia móvel, fabricantes de semicondutores e dispositivos móveis, desenvolvedores de *software* e provedores de serviço, cobrindo toda a cadeia de suprimentos para o mundo móvel (DARCEY e CONDER, 2009).

Na figura 1 podemos verificar a atuação das diversas plataformas em relação a suas vendas globais na área de *smartphones*. É possível verificar a consolidação do Android como principal plataforma para *smartphone* com cerca de 75% de participação. As vendas de dispositivos rodando o sistema operacional da Apple vem perdendo espaço continuamente e atingem cerca de 18% do mercado. As outras plataformas possuem cerca de 7% de participação (TECH-THOUGHTS, 2013).

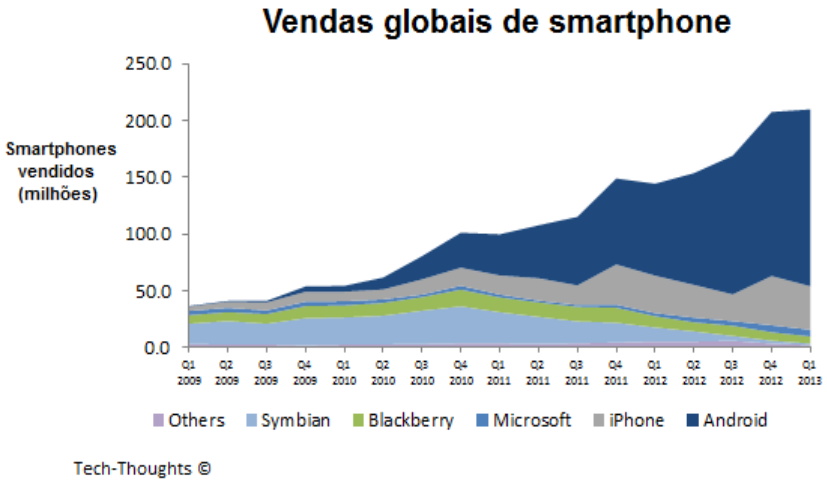


Figura 1 – Vendas globais de smartphone por sistema operacional
Fonte: TECH-THOUGHTS, 2013

Segundo ABLESON et al., 2012, pg.4, "[...] Android inclui um Sistema Operacional (OS) baseado em um *kernel* Linux, uma rica Interface de Usuário (IU), aplicativos de usuário, bibliotecas de código, *frameworks* de aplicativos, suporte a multimídia[...]" entre outros. Basicamente todas as tarefas mais comuns a serem programadas pelo desenvolvedor já estão definidas nas bibliotecas presentes nos pacotes fornecidos junto ao *Software Development Kit* (SDK) do Android.

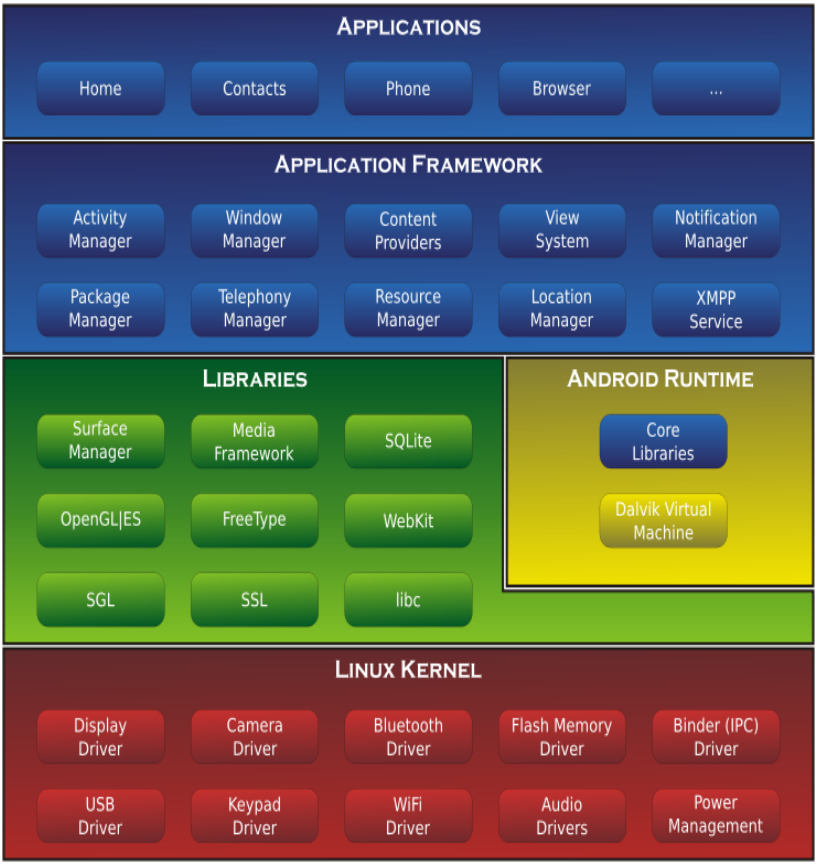


Figura 2 – Diagrama da arquitetura do sistema operacional Android
Fonte: WIKIPEDIA, 2014

Na figura 2, podemos observar o diagrama da arquitetura do sistema operacional Android. A arquitetura apresenta as seguintes camadas:

- Linux Kernel: disponibiliza um *kernel* Linux responsável por fornecer os serviços essenciais da plataforma, nele são disponibilizados *drivers* de *hardware*, gerenciador de processo e memória, segurança, rede, gerenciamento de energia, entre outros. Serve também como camada de abstração entre o *hardware* do dispositivo e o restante da arquitetura;
- Libraries: disponibiliza as bibliotecas nativas suportadas pelo Android. Estas bibliotecas garantem suporte a multimídia e gráficos, banco de dados, segurança para Internet, navegador *web*, entre outros;
- Android Runtime: é o responsável por permitir que um dispositivo Android não seja apenas um dispositivo móvel executando uma versão Linux reduzida:
    - Core Libraries: são as bibliotecas fundamentais do Android, além de fornecer a maioria das funcionalidades Java, estão presentes funcionalidades específicas da plataforma Android;
    - Dalvik Virtual Machine: é uma máquina virtual otimizada para execução de múltiplas instâncias de aplicativos Android. Por questões de licenciamento não é utilizada uma máquina virtual Java pura, uma vez os aplicativos Android são desenvolvidos utilizando a linguagem de programação Java.
- Application Framework: fornece as classes usadas para criar aplicativos Android. Ele também fornece uma abstração genérica para acesso ao hardware e gerencia a interface do usuário e recursos dos aplicativos;
- Applications: abrange todos os aplicativos instalados no dispositivo, tanto os nativos quanto os de terceiros.

ANDROID, 2013, apresenta como componentes do ciclo de vida de uma aplicação Android os seguintes elementos:

- Activities: uma *activity* é um componente da aplicação que fornece uma interface com a qual os usuários podem interagir, a fim de realizar alguma atividade, como realizar uma ligação, tirar uma foto, enviar um e-mail, ou visualizar um mapa. Uma aplicação geralmente possui uma ou mais *activities*, que possuem ligação entre si permitindo realizar um fluxo de atividades. Quando uma *activity* é iniciada a *activity* anterior é interrompida e encaminhada para uma pilha. Ao ativar o botão de voltar ou quando a *activity* em execução é encerrada o sistema recupera a última *activity* depositada na pilha e a exibe novamente;
- Views: uma *view* define o layout da aplicação onde a estrutura visual para a interface com o usuário é declarada. É responsável pela manipulação dos eventos da interface, como um toque em determinada área da tela, exibição ou entrada em segundo plano, e a vinculação deste com algum método da *activity*;
- Intents: uma *intent* é um objeto de mensagens que pode ser utilizado para solicitar uma ação de outro componente do próprio aplicativo em execução ou outro aplicativo. A *intent* pode solicitar a execução de uma *activity* ou um *service*, assim como enviar uma mensagem em *broadcast*;
- Services: um *service* é um componente de aplicação que pode executar operações de longa duração em segundo plano sem a interação do usuário, para tal não precisa fornecer uma interface para o usuário. O *service* pode ser iniciado por componente do aplicativo e manter-se executando em segundo plano, mesmo se o usuário trocar de aplicativo. Um *service* pode lidar com transações de rede, comunicar-se com um *web service*, tocar música, executar gravação e leitura de arquivo ou interagir com um provedor de conteúdo;
- Notifications: uma *notification* permite que o aplicativo alerte o usuário quando algum evento ocorre, como novas

> mensagens ou um evento de calendário. Estas notificações ficarão disponíveis na área de notificação ou através do LED de notificação do dispositivo.

As aplicações Android podem interagir com o sistema operacional e hardware adjacente através de uma coleção de gerenciadores. Cada gerenciador é responsável por armazenar o estado de algum serviço do sistema. O *LocationManager* pode ser utilizado para acessar os serviços baseados em localização de um dispositivo móvel. Para gerenciar o estado das interfaces com o usuário são utilizados *ViewManager* e *WindowManager*. *ContentProvider* atua na interação entre aplicações, elas podem utilizá-la para acessar o conteúdo de uma aplicação, assim como uma aplicação pode atuar como *ContentProvider* e disponibilizar acesso ao seu conteúdo. Um exemplo desta situação é uma aplicação de contatos telefônicos, elas são provedoras de conteúdo, permitindo que aplicações de terceiros acessem seus dados de contato e os utilizem de diversas formas (DARCEY e CONDER, 2009).

2.1.4.1 Localização

Conforme Darcey e Conder, 2009, pg.315, a possibilidade de utilizar meios que forneçam a localização de um dispositivo móvel é possível através do uso da API *Location-Based Services* (LBS). Segundo WIKIPEDIA, 2013, este tipo de API torna mais fácil a construção de aplicações *location-aware*, ou seja, aplicações que podem passivamente ou ativamente determinar sua localização, sem a necessidade de se concentrar nos detalhes da tecnologia de localização subjacente, permitindo também a minimização do consumo de energia usando todos os recursos de hardware do dispositivo.

Caso existam vários fornecedores de localização no dispositivo é possível restringir através de critérios quais deles estarão disponíveis. Podemos definir critérios para permitir o uso de fornecedores baseados em precisão utilizando *setAccuracy* ou economia de energia a partir de *setPowerRequirement*. Os critérios de precisão podem variar entre *NO_REQUIREMENT*, *ACCURACY_COARSE*, *ACCURACY_FINE*,

ACCURACY_LOW, ACCURACY_MEDIUM e *ACCURACY_HIGH*, ou seja, estas são constantes que indicam que não haverá requerimento algum quanto a precisão ou exigir a mais precisa informação possível. Estes requisitos podem variar em relação ao uso de energia, informações sobre altitude e velocidade e até sobre a possibilidade da ocorrência de custos para utilização do fornecedor de localização.

Caso os recursos de um *hardware* GPS não existam, ou estejam indisponíveis, e não haja outro recurso de localização compatível com LBS, o SDK do Android disponibiliza serviços de localização alternativos, sendo que estes possuem vantagens e desvantagens em termos de uso de energia, velocidade e precisão. Pelo fato da existência de LBS e GPS não ser obrigatória nos dispositivos móveis, torna-se interessante informar a necessidade da presença destes recursos nos aplicativos que necessitem destas características. O método indicado para se informar esta necessidade é utilizando a tag *<uses-feature android:name="android.hardware.location">* no arquivo de manifesto da aplicação. No caso da exigência de localização precisa sugere-se utilizar em substituição a tag *<uses-feature android:name="android.hardware.location.gps">*.

A determinação da localização do dispositivo, requer apenas cinco passos:

1. Obter uma instância do gerenciador de localização;
2. Adicionar uma permissão apropriada, no arquivo de manifesto, dependendo do tipo de informação de localização será utilizada;
3. Escolher um fornecedor de localização através dos métodos *getAllProviders*() ou *getBestProvider*();
4. Implementar uma classe *LocationListener*;
5. Invocar o método *requestLocationUpdates*() para iniciar o recebimento dos dados de localização.

Podemos observar na Figura 3 o diagrama de sequência para realizarmos os passos citados anteriormente.

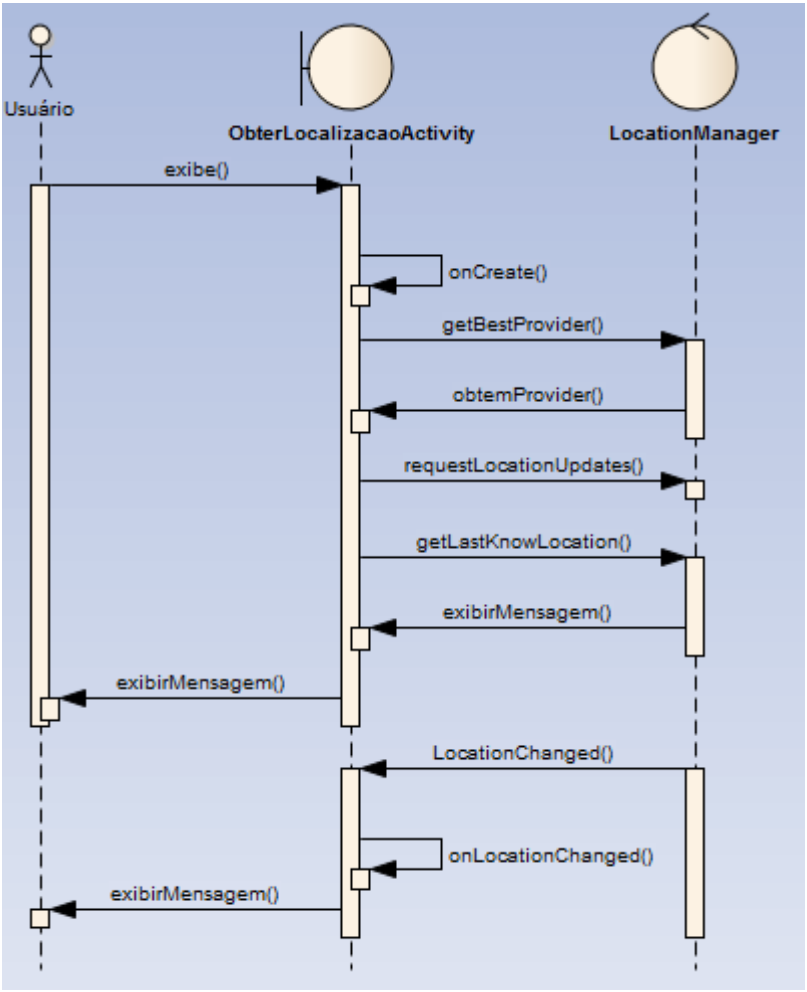


Figura 3 – Diagrama de sequência para obtenção da localização de um dispositivo móvel

Os dados recebidos com a localização do dispositivo retornam na forma de coordenadas com latitude e longitude, que apesar de serem úteis para o cálculo da distância ou para exibir uma posição no mapa, invariavelmente, o endereço da localização é mais útil. Para obter um endereço para uma determinada latitude e longitude, utiliza-se a

chamada *Geocoder.getFromLocation()*, responsável por retornar uma lista de endereços. Esta chamada é síncrona e utiliza a Internet para obter os dados, podendo levar um longo tempo para realizar esta atividade, desta forma para evitar congelamento e travamentos no aplicativo e não oferecer uma boa experiência ao usuário, considera-se o uso desta chamada em segundo plano. Enquanto o aplicativo está recebendo o endereço, exibe-se um indicador de atividade em processamento para mostrar que o aplicativo ainda está funcionando.

A utilização da API de localização é restritiva no quesito de usabilidade, ela fornece apenas informações das coordenadas geográficas, não sendo visualmente atraente ao usuário. Uma possibilidade para melhorar a interface de um aplicativo baseado em localização seria através do uso de mapas, já que estes poderiam facilitar a compreensão quanto aos dados obtidos e isto se daria por meio da utilização da API do Google Maps para Android. Esta API permite que mapas sejam exibidos a partir da base de dados do Google Maps no dispositivo móvel. Adicionalmente, a própria API lida com o acesso aos servidores do Google Maps, o download dos dados, a visualização do mapa e pela resposta quando o usuário interage com o mapa. Além disso é possível adicionar marcadores, círculos e conjuntos de segmentos de linhas, e até mesmo mudar a visão do usuário para uma determinada área do mapa (ANDROID, 2014).

2.1.5 Banco de Dados

Um sistema gerenciador de banco de dados (SGBD) permite aos usuários armazenar, acessar e modificar dados de forma eficiente e organizada. Originalmente, os usuários de SGBDs eram programadores e utilizavam linguagens de programação para acessar estes dados, visando criar uma interface amigável para um usuário não-técnico, não havia uma forma prática para um acesso casual aos dados armazenados. Buscando um acesso mais simplificado, foi criado em 1970 pela IBM a *Structured Query Language* (SQL), uma linguagem de programação de propósito específico. Futuramente tornou-se padrão pela ANSI em 1986 e pela ISO em 1987, e vem sendo utilizada desde então por grande parte dos SGBDs disponíveis no mercado (MICROSOFT, 2013).

São exemplos de bancos de dados SQL as ferramentas SQLite e MySQL, estas duas opções foram escolhidas para fazerem parte deste projeto principalmente pelo fato de serem gratuitas, além de podermos citar o fato de que a primeira possui suporte nativo oferecido pela plataforma Android. Com relação a MySQL sua escolha deve-se principalmente a critérios de familiaridade, facilidade de uso e instalação oferecidas pela ferramenta.

SQLite é uma biblioteca de software gratuita que implementa um banco de dados SQL e que possui como características não ter necessidade de configurações, não instanciar um servidor de banco de dados em separado e ser capaz de rodar localmente em um dispositivo móvel. Além disso ele é transacional, ou seja, permite que um conjunto de procedimentos executados sejam vistos pelo usuário como uma única ação, garantindo atomicidade, consistência, isolamento e durabilidade, ou seja, todas alterações em uma transação devem ocorrer completamente ou não (SQLite, 2013).

	Android	SQL
Consulta	String[] coluna = {“latitude”,“longitude”,“datahora”}; String ordem = “datahora”; Cursor cursor = db.query(“rastreio”, coluna, null, null, null, null, ordem);	select latitude, longitude, datahora from rastreio order by datahora
Inclusão	ContentValues valor = new ContentValues(); valor.put(“id”,0); db.insert(“rastreio”, null, valor);	insert into rastreio (id) values (0)
Atualização	ContentValues valor = new ContentValues(); valor.put(“id”,0); String selecao = = “id > ?”; String[] argumento = {String.valueOf(1)}; db.update(“rastreio”, valor, selecao, argumento);	update rastreio set id = 0 where id = 1
Exclusão	String selecao = “id > ?”; String[] argumento = {String.valueOf(10)}; db.delete(“rastreio”, selecao, argumento);	delete from rastreio where id > 10

Tabela 1 – Demonstração do uso da biblioteca SQLite

Segundo ABLESON et al., 2012, pg.148, para utilizar SQLite na plataforma Android não é preciso muito conhecimento em SQL, e quase não é necessário escrever nesta linguagem. Uma série de chamadas a biblioteca SQLite é disponibilizada junto ao Android para manipular os dados disponíveis no banco de dados. Uma demonstração do uso da biblioteca SQLite pode ser observada na tabela 1.

2.1.6 Web service

Segundo W3SCHOOLS, 2014a, *web service* é um componente de uma aplicação web e é utilizado para integração entre sistemas distintos ou desenvolvidos em plataformas diferentes, sendo esta uma das características mais marcantes, a independência de linguagem e plataforma. A utilização de tecnologias e protocolos padronizados é o que garante a interoperabilidade entre os sistemas a serem integrados. Fazem parte da especificação *web service* as seguintes tecnologias e protocolos: WSDL, SOAP e UDDI.

2.1.6.1 XML

Devemos abordar *Extensible Markup Language* (XML) visto que ela é utilizada massivamente em *web service*. XML é uma linguagem de marcação baseada em texto puro, que assim como *HyperText Markup Language* (HTML) identifica os dados utilizando *tags* (identificadores entre colchetes angulares: <...>), sendo que coletivamente estas *tags* são conhecidas como marcação. Ao contrário da HTML, as *tags* XML identificam os dados, ao invés de especificar a forma de apresentá-lo. Há uma série de razões para a aceitação crescente do XML, entre elas o fato de não ser disponibilizado formato binário, sendo possível criar e editar arquivos utilizando qualquer editor de texto, facilitando a depuração das informações do arquivo. Estes arquivos tanto servem em pequenas proporções para uma simples configuração de um aplicativo, assim como para transferência de grandes quantidades de dados ao ser disponibilizado em arquivos maiores. Como já citado,

XML informa que tipo de dados serão armazenados e não como exibi-lo, já que suas *tags* identificam as informações e quebram os dados em partes podendo ser interpretado facilmente a partir de um parser (analisador) XML (ORACLE, 2014).

2.1.6.2 WSDL

Web Service Description Language (WSDL) é um documento escrito em XML que descreve um *web service*. Ele especifica o local para acesso ao *web service*, as operações ou métodos disponibilizados e os meios para comunicar-se com este serviço. O documento pode ser disponibilizado em um arquivo ou publicado na *web*, ou ambos (W3SCHOOLS, 2014a).

2.1.6.3 SOAP

As tecnologias utilizadas para integração entre sistemas, *Distributed Component Object Model* (DCOM) e Corba, eram baseadas em *Remote Procedure Calls* (RPC) e não estavam habilitadas a utilizar a Internet para comunicarem-se, já que *firewalls* e servidores *proxy* normalmente bloqueiam este tipo de tráfego. A melhor maneira para realizar a comunicação entre aplicações sem este tipo de contratempo seria utilizando HTTP, já que o mesmo é suportado por todos os navegadores e servidores *web*, dessa forma *Simple Object Access Protocol* (SOAP) foi criado para alcançar este objetivo. Ele fornece uma forma de comunicação simples entre aplicações rodando em diferentes sistemas operacionais, com diferentes tecnologias e linguagens de programação baseado em XML (W3SCHOOLS, 2014a).

2.1.6.4 UDDI

Universal Description, Discovery and Integration (UDDI) é um

framework para descrição, descoberta e integração de serviços utilizando a Internet. Sua função básica é baseada nos arquivos WSDL, que descrevem a interface de um *web service* e permitem que estas informações possam ser facilmente acessadas (W3SCHOOLS, 2014a).

2.1.6.5 Bibliotecas para acesso a web service

Neste projeto faremos uso de duas bibliotecas para acesso a *web service*. Uma delas é conhecida por METRO, que será incorporada ao aplicativo disponibilizado no servidor de aplicação, e a outra chama-se ksoap, destinada a ser incluída no aplicativo móvel.

METRO é uma biblioteca utilizada para implementar *web service* em uma aplicação Java. A partir dela é possível criar *web services* simples até os mais elaborados utilizando técnicas de segurança, confiabilidade, além de transações. Seu uso em nosso projeto se faz necessário visto que a implementação do Apache Tomcat está restrita apenas ao *container web*, devendo fazer uso de implementações de bibliotecas externas para suportar esta tecnologia (METRO, 2014).

ksoap2-android fornece uma biblioteca cliente SOAP reduzida para a plataforma Android. Apesar de anunciar ser voltada para o desenvolvimento Android esta biblioteca segue a especificação Java ME, funcionando também em dispositivos que rodem nesta plataforma (KSOAP, 2014). Seu uso se faz necessário já que a implementação do aplicativo Android fará uso de um *web service* baseado em SOAP, salientando que a plataforma Android já possui nativamente suporte a REST, uma outra tecnologia para implementação de *web service*.

2.1.7 AJAX

AJAX é o acrônimo para *Asynchronous JavaScript and XML*, sendo importante salientar que não se trata de uma nova linguagem de programação e sim um novo jeito de utilizar os padrões existentes, entre eles HTML, JavaScript e XML. Não é obrigatório o uso em conjunto destas tecnologias, assim como é possível em alguns casos substituir

XML por *JavaScript Object Notation* (JSON). Basicamente, AJAX é um método para criar página web dinamicamente e de forma ágil, porém assíncrona, realizando a troca de dados entre navegador e servidor web a fim de atualizar determinadas partes de uma página web sem ter que recarregar toda página (W3SCHOOLS, 2014b).

2.1.7.1 JSON

JSON é uma sintaxe utilizada para troca de dados muito similar ao XML. Esta sintaxe é sintaticamente idêntica ao código utilizado para a criação de objetos JavaScript. Dada esta semelhança, não precisamos utilizar uma bibilioteca de parser, já que utilizando um comando JavaScript é possível analisar os dados JSON e produzir objetos JavaScript nativos. A sintaxe JSON apresenta apenas duas estruturas: uma coleção de pares nome/valor denominada objeto e uma lista de valores conhecida por array. Na página oficial da JSON são disponibilizadas diversas bibliotecas, em diversas linguagens de programação, que implementam a notação. Neste projeto utilizaremos uma biblioteca implementada para a linguagem de programação Java chamada org.json (JSON, 2014).

2.2 Metodologia

Neste capítulo será demonstrado o processo de desenvolvimento para criação do ambiente integrado entre aplicativo móvel e servidor *web*. Dada as tecnologias abordadas no decorrer deste projeto, concebemos a arquitetura para integração dos módulos componentes deste projeto, conforme disposto na figura 4. Além dos módulos citados, em possíveis trabalhos futuros é possível incluir uma estação de trabalho para acesso aos dados capturados e exportados ao servidor *web*. Estes módulos estão conectados por um meio nebuloso conhecido por nuvem, podendo esta ser qualquer forma de comunicação que possibilite a troca de informações entre os módulos. Como citado na definição do

problema, poderíamos trabalhar com uma conexão via bluetooth, rede ethernet, redes wireless ou pacotes de dados via telefonia celular.

Preferencialmente, adotamos redes wireless para comunicação dispositivo móvel / servidor web por questões de segurança, visto que não implementamos criptografia para os dados transmitidos nas comunicações entre os dois módulos. Dessa forma, utilizaremos apenas a rede local para transmissão de dados, visando evitar o acesso de usuários externos a algum dado que pudesse ser disponibilizado pelo servidor *web*. Adicionalmente, visando adicionar um pouco mais de confiabilidade aos dados transmitidos, implementamos uma autenticação via matrícula e senha nas conexões com o *web service*, ou seja, apenas com a validação destas dados alguma informação poderá ser manipulada ou transitar do servidor *web* para o dispositivo móvel.

Baseando-nos no dispositivo móvel como ferramenta de interação do Oficial de Justiça com a arquitetura, ele é responsável por receber e enviar dados exclusivamente ao servidor web. Por meio do dispositivo, em sua interface, o Oficial de Justiça poderá receber mandados a serem cumpridos e ao término de suas atividades enviar as informações relativas aos mandados cumpridos e o rastreio de toda sua movimentação durante o dia de trabalho.

A estação de trabalho servirá a quem for designado para realizar o monitoramento e avaliação dos dados enviados pelo Oficial de Justiça. Basicamente será composta por um computador *desktop*, podendo ser expandida para qualquer dispositivo que possua um navegador *web*, para fins de visualização e acompanhamento do cumprimento de mandados e a movimentação realizada pelo Oficial de Justiça.

Por fim, o servidor *web*, elemento responsável por receber e fornecer as informações manipuladas pelo Oficial de Justiça. Este equipamento processará os mandados a serem enviados ao Oficial de Justiça e será responsável por receber estes dados atualizados, além das informações do rastreio. Quando da necessidade de acompanhamento fornecerá, a quem desejar visualizar, subsídios para consulta da situação dos mandados a cumprir e cumpridos, além do rastreio.

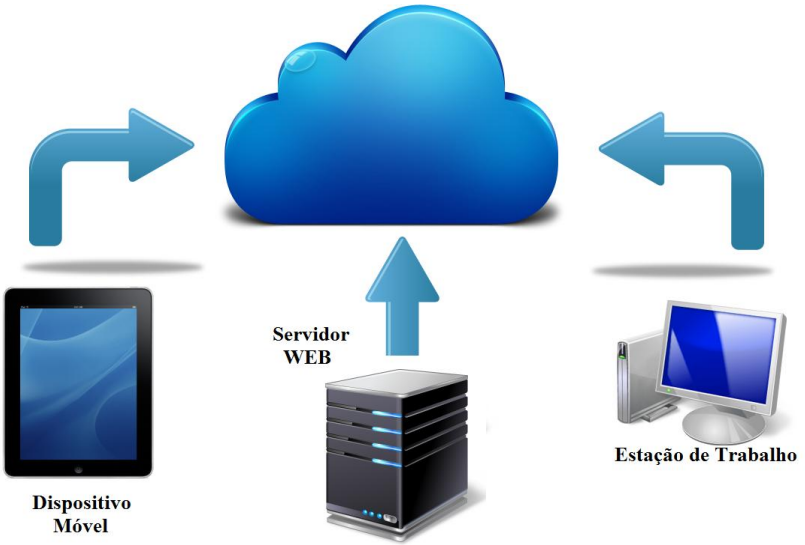


Figura 4 – Arquitetura para integração

2.2.1 Módulo do servidor web

A definição do *hardware* para a configuração do servidor *web* para o desenvolvimento do projeto está estreitamente ligada ao *software* em execução no equipamento. Por questões de afinidade, o *software* responsável pelo envio e recebimento dos dados é desenvolvido em Java e uma das plataformas para execução da aplicação é o Apache Tomcat, mais especificamente em sua versão 7, já que foram utilizados servlets e JSP. Conforme avaliado no capítulo 2.1.3.4, não utilizaremos EJB no desenvolvimento da aplicação, desta forma, um servidor web Apache Tomcat é suficiente para servir aos propósitos deste projeto.

Outro *software* necessário para este projeto é o SGBD, nele encontram-se armazenados os dados utilizados neste projeto. O *software* escolhido foi o MySQL, por ser gratuito e de fácil instalação, sendo que utilizamos em nosso projeto a versão 5.5.

As escolhas dos *softwares* apresentados foram baseadas em sua gratuidade, facilidade de uso, larga adoção no desenvolvimento de

aplicações e pelo fato de serem multiplataforma, possibilitando sua instalação em equipamentos que utilizem sistema operacional Windows ou Linux.

2.2.1.1 Modelo do banco de dados no servidor web

As informações armazenadas no banco de dados são dispostas em tabelas conforme o diagrama entidade-relacionamento apresentado na figura 5. Considerando que o escopo deste trabalho traduz-se na interação com dispositivos móveis temos que as tabelas *situacao* e *oficial* estão previamente carregadas com dados provenientes do Comunicado Eletrônico 50 e com os Oficiais de Justiça autorizados a interagir com o aplicativo.

A tabela *mandado* possui uma carga parcial de dados com os mandados a serem cumpridos, tendo apenas os campos *numero*, *situacao*, *oficial*, *latitude*, *longitude*, *nome* e *endereco* preenchidos. Após o cumprimento e envio dos mandados pelo Oficial de Justiça através do dispositivo móvel, os campos *latitude_cumprimento*, *longitude_cumprimento* e *datahora_cumprimento* passam a ter algum conteúdo.

Por fim, a tabela *rastreio* é exclusivamente preenchida com dados provenientes da captura do posicionamento em deslocamento do Oficial de Justiça. Nela o campo *id* assume um valor arbitrário auto incremental visando facilitar sua manipulação, enquanto os campos *oficial*, *latitude*, *longitude* e *datahora* recebem os dados originários da captura automática das coordenadas geográficas assumidas pelo dispositivo móvel.

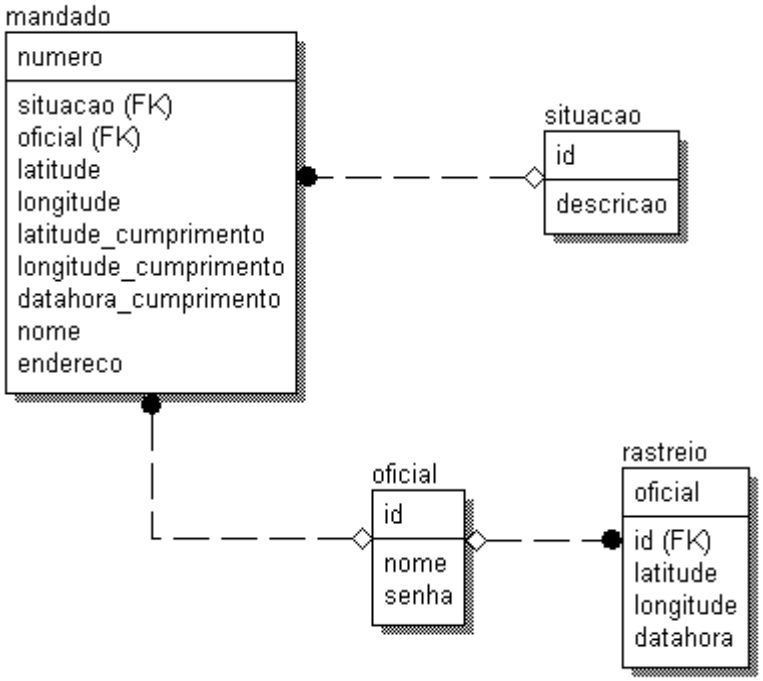


Figura 5 – Modelo ER do banco de dados no servidor web

O script SQL, disposto no quadro 1, é proveniente de um *dump* do banco de dados, demonstrando a criação e prévio preenchimento de alguns dados conforme comentado anteriormente.

```
CREATE TABLE mandado (
  numero varchar(10) COLLATE utf8_bin NOT NULL,
  situacao int(11) NOT NULL,
  oficial int(11) NOT NULL,
  latitude varchar(50) COLLATE utf8_bin NOT NULL,
  longitude varchar(50) COLLATE utf8_bin NOT NULL,
  latitude_cumprimento varchar(50) COLLATE utf8_bin DEFAULT NULL,
  longitude_cumprimento varchar(50) COLLATE utf8_bin DEFAULT NULL,
  datahora_cumprimento varchar(20) COLLATE utf8_bin DEFAULT NULL,
  PRIMARY KEY (numero)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
```

```
AUTO_INCREMENT=0;

INSERT INTO mandado VALUES('0230000001', 0, 1, '-27.539400', '-
48.626760', '', '', '');
INSERT INTO mandado VALUES('0230000002', 0, 1, '-27.605720', '-
48.584711', '', '', '');
INSERT INTO mandado VALUES('0230000003', 0, 1, '-27.601790', '-
48.548250', '', '', '');
INSERT INTO mandado VALUES('0230000004', 0, 1, '-27.594000', '-
48.543359', '', '', '');
INSERT INTO mandado VALUES('0230000005', 0, 1, '-27.553469', '-
48.621736', '', '', '');

CREATE TABLE oficial (
  id int(11) NOT NULL AUTO_INCREMENT,
  nome varchar(50) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  senha varchar(50) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
AUTO_INCREMENT=0;

INSERT INTO oficial VALUES(1, 'OFICIAL DE JUSTIÇA','Ofcl_Jst');

CREATE TABLE rastreio (
  id int(11) NOT NULL AUTO_INCREMENT,
  oficial varchar(10) NOT NULL,
  latitude varchar(50) NOT NULL,
  longitude varchar(50) NOT NULL,
  datahora varchar(50) NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
AUTO_INCREMENT=0;

CREATE TABLE situacao (
  id int(11) NOT NULL AUTO_INCREMENT,
  descricao varchar(50) COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
AUTO_INCREMENT=0;

INSERT INTO situacao VALUES(0, 'Não cumprido');
INSERT INTO situacao VALUES(1, 'Cumprido – Ato positivo');
```

```
INSERT INTO situacao VALUES(2, 'Cumprido – Ato positivo parcial');
INSERT INTO situacao VALUES(3, 'Cumprido – Ato negativo');
INSERT INTO situacao VALUES(4, 'Devolvido sem cumprimento');
```

Quadro 1 – Script SQL do banco de dados no servidor web

2.2.1.2 Serviços ao dispositivo móvel

O aplicativo no servidor *web* foi concebido para atuar como fornecedor e receptor de dados em um único módulo, de forma a possibilitar tratar requisições provenientes dos dispositivos móveis portados pelos Oficiais de Justiça e das solicitações de dados oriundas de estações de trabalho que visam monitorar os dados fornecidos.

O módulo que interage com o dispositivo móvel foi desenvolvido utilizando um *web service* para intermediar o acesso entre o cliente e o banco de dados. Cada requisição de envio ou recebimento de dados encaminha uma cadeia de caracteres aglutinados tendo como separador um caractere #, conforme quadro 2. Em cada lado da aplicação, sabendo qual método do *web service* foi requisitado, faz-se a separação dos dados e preenche-se um determinado objeto com estas informações através de seu construtor. Posteriormente as informações contidas neste objeto são armazenadas no banco de dados.

```
...
  public Mandado(String msg) {
    String[] split = msg.split("#");
    this.numero = split[0];
    this.situacao = split[1];
    this.latitude_cumprimento = split[2];
    this.longitude_cumprimento = split[3];
    this.horario_cumprimento = split[4];
  }

  @Override
  public String toString() {
    return numero + "#" + oficial + "#" + latitude + "#" + longitude + "#" +
situacao + "#" + nome + "#" + endereco + "#";
  }
...
```

Quadro 2 – Código Java para envio e recebimento de mandados via web service

Os métodos implementados no *web service* focam nas requisições do dispositivo móvel e respondidos pelo servidor web. O método *cumprimento* é responsável por enviar ao servidor web os mandados cumpridos pelo Oficial de Justiça, que em caso de cadastro com sucesso retorno uma resposta *OK*, conforme quadro 3. O método *rastreio* é similar ao anterior, recebe uma sequência de caracteres com os dados de rastreamento do deslocamento do Oficial de Justiça emitindo uma resposta *OK* em caso de sucesso no processamento da requisição.

```
...
public class WS {
  public String mandado(String msg) {
    String retorno = "";
    Oficial oficial = new Oficial(msg);
    if
(oficial.getSenha().equals(Storage.getInstance().getOficial(oficial.getId()).getSenh
a())) {
      List<Mandado> mandados =
Storage.getInstance().getMandado(oficial.getId());
      for (Mandado mandado : mandados) {
        retorno += mandado.toString();
      }
    } else {
      retorno = "WRONG_PASS";
    }
    return retorno;
  }

  public String rastreio(String msg) {
    Rastreio rastreio = new Rastreio(msg);
    if
(rastreio.getOficial().getSenha().equals(Storage.getInstance().getOficial(rastreio.g
etOficial().getId()).getSenha())) {
      return Storage.getInstance().setRastreio(rastreio) == true ? "OK" : "NOK";
    } else {
      return "WRONG_PASS";
    }
  }

  public String cumprimento(String msg) {
    Mandado mandado = new Mandado(msg);
    if
(mandado.getOficial().getSenha().equals(Storage.getInstance().getOficial(mandad
```

```
o.getOficial().getId()).getSenha())) {
        return Storage.getInstance().updateMandado(mandado) == true ? "OK" :
"NOK";
    } else {
        return "WRONG_PASS";
    }
  }
}
...
```

Quadro 3 – Código Java da implementação do web service

O método *mandado* tem funcionamento inverso, ao ser requisitado pelo dispositivo móvel recebe apenas o código do Oficial de Justiça portador do equipamento. O *web service* trata de requisitar ao banco de dados via a classe *Storage* quais mandados foram encaminhados para serem cumpridos pelo oficial identificado na requisição. Uma lista de mandados é retornada e seus dados passam a ser concatenados um a um, objeto a objeto, de forma a ser transmitida em uma única sequência de caracteres ao dispositivo móvel. Encaminhada ao requisitante e encerrada a conexão não há retorno da situação, o próprio requisitante deve tratar os dados recebidos e em caso de erro solicitar a retransmissão.

Podemos visualizar nos apêndices A, B e C os fluxogramas utilizados para guiar o desenvolvimento dos métodos do *web service*. Adicionalmente, no apêndice J observa-se o diagrama de classes da aplicação em execução no servidor *web*.

2.2.1.3 Serviços à estação de trabalho

O serviço fornecido à estação de trabalho trata-se do encaminhamento de uma página HTML contendo um mapa gerado pela API Google Maps. Este mapa pode ser preenchido com dados relativos às informações de rastreio e mandados cumpridos e a serem cumpridos. Os dados inseridos no mapa são coordenadas geográficas fornecidas pelo dispositivo móvel e obtidas através de uma requisição AJAX realizada a partir da página HTML. Esta requisição aciona uma página JSP, código 4, responsável por realizar uma consulta ao banco de dados

resgatando uma lista de coordenadas geográficas gravadas. Estes dados são retornados no formato JSON, um formato utilizado no intercâmbio de dados computacionais, e caracterizado por ser facilmente analisado por qualquer tipo de linguagem de programação.

```
...
  JSONArray list = new JSONArray();
  JSONObject json;
  List<Rastreio> rastreio = Storage.getInstance().getRastreio("1");
  for (Rastreio r : rastreio) {
    json = new JSONObject();
    json.put("Latitude", r.getLatitude());
    json.put("Longitude", r.getLongitude());
    json.put("Horario", r.getHorario());
    json.put("Icone", r.getHorario().substring(11, 13).concat(".png"));
    json.put("ID", r.getId());
    list.put(json);
  }
  out.println(list.toString());
...
```

Quadro 4 – Código gerador de massa de dados no formato JSON

O resultado do quadro 4 pode ser observado no quadro 5, no formato JSON, sendo este um pequeno trecho de uma sequência de coordenadas geográficas incluindo horário de captura, ícone identificador no mapa e código identificador da captura.

```
[
...
{"Latitude":-27.551871,"Longitude":-48.619555,"Horario":"23/11/2013
09:09:10","Icone":"09.png","ID":1689},
{"Latitude":-27.554021,"Longitude":-48.619423,"Horario":"23/11/2013
09:09:21","Icone":"09.png","ID":1690},
...
]
```

Quadro 5 – Código JSON contendo coordenadas geográficas

2.2.2 Módulo do dispositivo móvel

Uma das características primordiais no desenvolvimento de nosso aplicativo é a obtenção das coordenadas geográficas através do dispositivo móvel. A invocação para acesso a estas informações é realizada através do gerenciador de localização *LocationManager*. Essa classe fornece acesso aos serviços de localização do sistema, estes serviços permitem que o aplicativo obtenha atualizações periódicas de localização geográfica através do dispositivo móvel. Existem duas constantes que definem o nome do provedor de coordenadas geográficas. O provedor *NETWORK_PROVIDER* é utilizado quando não dispomos de um módulo GPS em nosso dispositivo móvel. Este provedor determina a localização com base na disponibilidade de uma torre de celular ou um ponto de acesso Wi-Fi, os resultados são obtidos através de uma pesquisa de rede. O provedor *GPS_PROVIDER* determina a localização por meio de satélites GPS, dependendo das condições, este provedor pode demorar um pouco para retornar uma localização, porém fornece uma boa precisão nas coordenadas geográficas retornadas.

```
import android.location.*;

public class ObterLocalizacaoActivity extends Activity implements
LocationListener {
  private String bestProvider;
  private String latitude;
  private String longitude;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cumprir_mandado);

                  LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

    Criteria criteira = new Criteria();
    criteria.setAccuracy(Criteria.NO_REQUIREMENT);
    criteria.setPowerRequirement(Criteria.NO_REQUIREMENT);
```

```
    String bestProvider = locationManager.getBestProvider(criteria, true);

    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, tempo * 1000, 1, this);
    locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, tempo * 1000, 1, this);

    Location location = locationManager.getLastKnownLocation(bestProvider);
    latitude = String.valueOf(location.getLatitude());
    longitude = String.valueOf(location.getLongitude());
      Toast.makeText(this, "Latitude: " + latitude + " - Longitude: " + longitude, Toast.LENGTH_LONG).show();
  }

  @Override
  public void onLocationChanged(Location location) {
    latitude = String.valueOf(location.getLatitude());
    longitude = String.valueOf(location.getLongitude());
       Toast.makeText(this, "Latitude: " + latitude + " - Longitude: " + longitude, Toast.LENGTH_LONG).show();
  }

  @Override
  public void onProviderDisabled(String arg0) {
    Toast.makeText(this, "Habilite o GPS", Toast.LENGTH_LONG).show();
  }

  @Override
  public void onProviderEnabled(String arg0) {
    Toast.makeText(this, "GPS habilitado", Toast.LENGTH_LONG).show();
  }
  @Override
  public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
  }
}
```

Quadro 6 – Código instanciando o gerenciador de localização

No quadro 6 exemplificamos a instanciação de um gerenciador de localização e a escolha dos provedores a serem utilizados. A escolha do provedor é feita através do método *requestLocationUpdates* o qual exige que informemos quatro parâmetros. O primeiro é a constante que identifica o tipo do provedor a ser utilizado, *NETWORK_PROVIDER* ou *GPS_PROVIDER*. O segundo parâmetro define o intervalo de tempo em

milisegundos, enquanto o terceiro parâmetro define a distância em metros, que definem condições que são verificadas por um *listener*, ou seja, uma classe que notifica um objeto quando uma determinada ação ocorre. Ao ser atingida a condição estabelecida pelos parâmetros anteriores, o *listener* aciona o método *onLocationChanged*() da classe definida no quarto parâmetro.

2.2.2.1 Modelo do banco de dados no dispositivo móvel

Na plataforma Android o acesso a banco de dados torna-se mais simples caso a opção escolhida para tal seja o SQLite. No capítulo 2.1.5, apresentamos a tabela 1 onde é demonstrado como podemos utilizar o banco de dados SQLite em uma aplicação Android.

Nosso aplicativo utiliza o modelo apresentado na figura 6. Basicamente são três tabelas responsáveis por armazenar os dados a gerados pelo aplicativo móvel. A tabela *mandado* recebe os dados dos mandados a serem cumpridos pelo Oficial de Justiça e será atualizada com os dados do cumprimento coletados pelo GPS e informados pelo usuário. A tabela *rastreio* é carregada periodicamente com os dados de localização do dispositivo móvel, não existe interação do usuário com estes dados e a coleta ocorre de forma imperceptível, uma vez que a cada notificação do *listener* do gerenciador de localização inserimos os dados no banco. A tabela *config* receberá os dados de configuração do aplicativo da seguinte forma:

- ip: identifica o endereço ip para conexão com o *web service*;
- porta: identifica a porta a ser utilizada na conexão com o *web service*;
- rastreio: informa se haverá ou não o rastreio do dispositivo móvel;
- tempo: informa o intervalo de tempo a ser considerado na gravação das atualizações de localização;
- oficial: identifica o oficial de justiça que está portando o dispositivo móvel;
- senha: identifica a senha do oficial de justiça para interação com o *web service*.

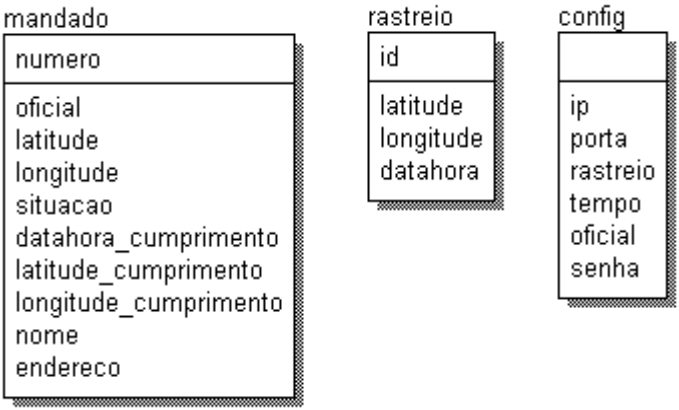


Figura 6 – Modelo do banco de dados no aplicativo móvel

2.2.2.2 Interfaces do aplicativo móvel

O aplicativo móvel desenvolvido foi nomeado com o termo em latim para Oficial de Justiça, qual seja *Longa Manus*. Para aprofundar o entendimento sobre o aplicativo foram disponibilizados nos apêndices D, E, F, G, H e I alguns fluxogramas contendo explicações acerca do aplicativo desenvolvido. Adicionalmente, no apêndice K é exibido um diagrama de classes do aplicativo móvel demonstrando a estrutura e as relações das classes que servem de modelo aos objetos.

A aplicação é composta por sete *activities*, cada qual representando uma tela do dispositivo. A tela principal, ou *MainActivity*, disposta na figura 7, possui quatro botões responsáveis por acessar outras *activities* quando um destes botões é pressionado. Automaticamente, ao ser instanciado, o aplicativo passa a realizar a aquisição dos dados de localização geográfica do dispositivo móvel. Porém, caso esta opção tenha sido desabilitada nas configurações o rastreio não é realizado.

Figura 7 – Tela inicial do aplicativo móvel

Ao acionarmos o botão Obter mandados o aplicativo abre uma conexão de rede com o endereço e porta, registrados na tabela *config*, para acesso ao *web service* que é disponibilizado em nosso servidor *web*. Em caso de sucesso ao conectarmos no *web service*, receberemos os mandados a serem cumpridos pelo Oficial de Justiça. A figura 8 retrata uma tela de atividades em suspenso informando que estamos obtendo os mandados. Enquanto estamos em comunicação com o *web service*, nossa aplicação ficará sombreada indicando de forma clara que a interação com o aplicativo está em suspensão e devemos aguardar o término da atividade. Ao término da comunicação com o web service e recebimento dos mandados a serem cumpridos, os dados recebidos são gravados na tabela *mandado*. Os dados são recebidos em uma sequência de caracteres separados por um caractere #, ao receber estes dados uma lista de objetos *Mandado* será criada e gravada sequencialmente na respectiva tabela. Em seguida, na figura 9 podemos observar a mensagem de retorno quando recebemos com sucesso os mandados a serem cumpridos. Após a exibição desta mensagem, o usuário pode retornar para a tela principal da aplicação e realizar outras atividades. É possível verificar o fluxograma completo desta atividade no apêndice E.

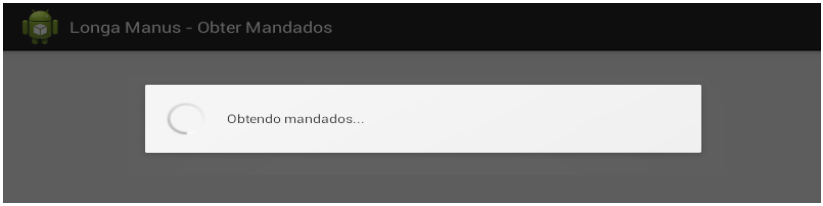


Figura 8 – Tela Obter Mandados

Figura 9 – Tela de resultado da obtenção dos mandados

A próxima atividade a ser realizada após obter os mandados junto ao servidor *web* é cumprir os mandados. Esta ação pode ser realizada selecionando a opção *Cumprir mandado*. Esta tela listará os mandados recebidos pelo dispositivo móvel e que ainda estão na situação de mandados a serem cumpridos, conforme podemos observar figura 10. Estes mandados poderão ser acessados para informarmos o cumprimento dos mesmos. Nota-se que esta situação não permite retorno, ou seja, ao registrar o cumprimento de um mandado não há possibilidade de desfazer a informação do cumprimento. Em caso de cadastro errôneo este mandado poderá ser baixado novamente, alterando-se no banco de dados do servidor *web* a situação do mandado para a situação a ser cumprido.

Figura 10 – Tela Cumprir mandado

Ao selecionarmos um mandado são oferecidas duas opções, conforme figura 11, *Traçar rota* e *Cumprir mandado*.

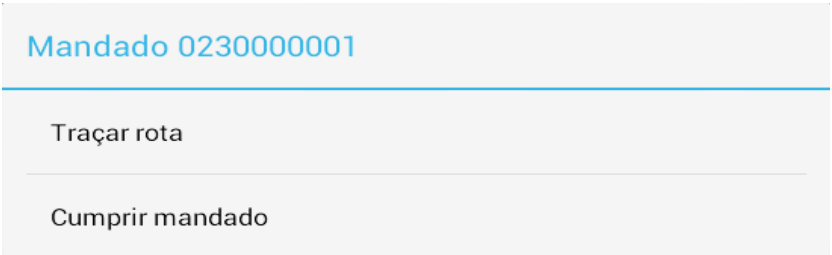


Figura 11 – Opções apresentadas após selecionar um mandado

Ao selecionarmos a opção *Traçar rota* os dados relativos a posição atual do dispositivo móvel e do mandado a ser cumprido são enviados aos servidores do Google. Com estas informações é gerada uma rota entre estes dois pontos, esta rota pode ser adicionada ao mapa como observado na figura 12. O marcador azul indica a origem do roteiro, enquanto o destino é indicado pelo marcador vermelho. Pressionando este marcador, algumas informações sobre o mandado são exibidas como número, nome e endereço de cumprimento. O mapa fornecido pela API do Google Maps permite que o usuário interaja com a interface, permitindo realizar zoom, reposicionar o mapa, entre outras atividades. O fluxograma desta atividade está disposto no apêndice F.

Caso a opção selecionada seja *Cumprir mandado*, um mapa é exibido e dois marcadores são adicionados. Um marcador azul identifica a posição atual do dispositivo móvel, e um marcador vermelho identifica a posição do mandado selecionado. Ao redor do marcador azul traçamos um círculo para permitir, futuramente, a implantação de um mecanismo de tolerância, visto que existem alguns problemas de precisão quanto aos mecanismos de localização em determinadas situações. Este mapa também permite que alguns dados do mandado a ser cumprido sejam exibidos ao pressionar o marcador vermelho. Pressionando novamente o mapa, é disponibilizado ao usuário do aplicativo a possibilidade de dar cumprimento ao mandado.

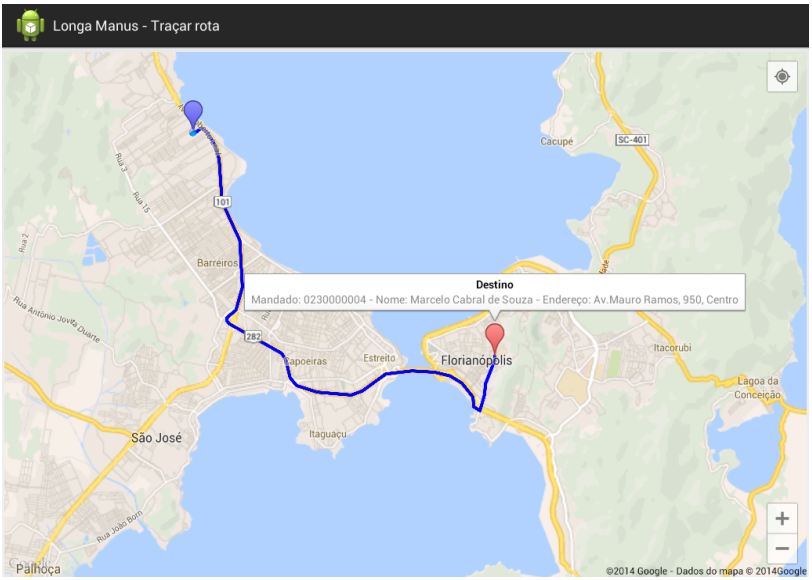


Figura 12 – Tela Traçar rota

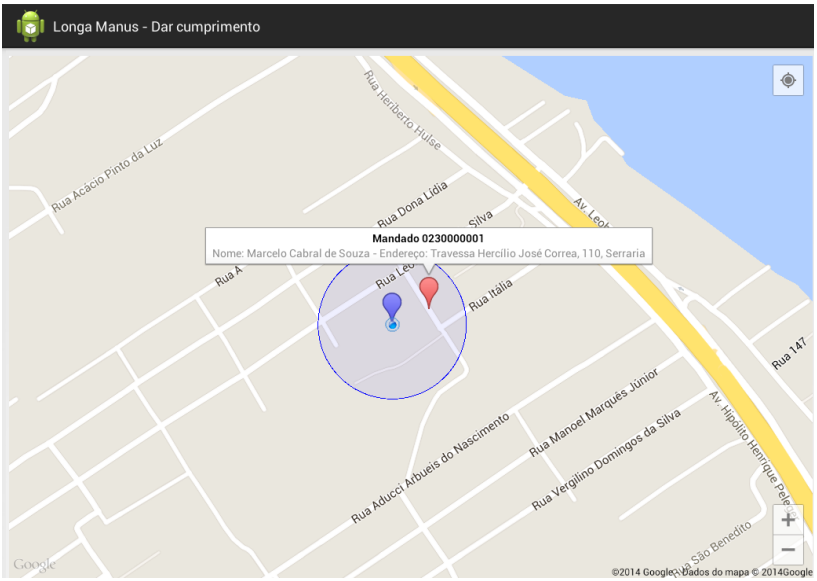


Figura 13 – Tela Dar cumprimento

As figuras 14, 15 e 16 apresentam as mensagens apresentadas pelo aplicativo quando selecionado um mandado para informar-se o cumprimento do mesmo. A figura 14 solicita informação de confirmação do cumprimento do mandado.Uma pergunta simples com duas opções, uma afirmativa e outra negativa. Em caso positivo o aplicativo encaminha uma nova mensagem ao usuário, em caso negativo o usuário retorna a tela anterior. A figura 15 apresenta as situações a serem assumidas pelo mandado após o cumprimento do mesmo, de acordo com o Comunicado Eletrônico 50, disponível no anexo A. Informando-se a nova situação, o usuário é encaminhado para a tela apresenta na figura 16. Esta tela apresenta os dados coletados pelo receptor GPS do dispositivo móvel, entre eles latitude, longitude e horário, além da situação selecionada na tela anterior. Estas informações serão enviados para armazenamento da informação do cumprimento do mandado no banco de dados da aplicação. A tela solicitará a confirmação da gravação dos dados e realizará a gravação no banco de dados em caso positivo finalizando o procedimento. Esta atividade pode ser visualizada no apêndice G.

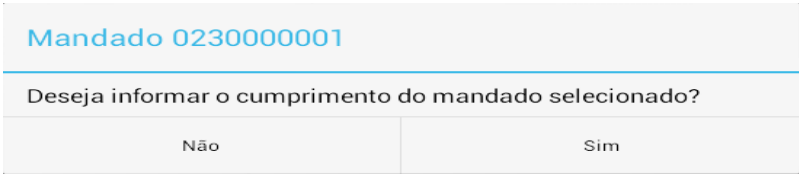


Figura 14 – Mensagem de confirmação de cumprimento do mandado

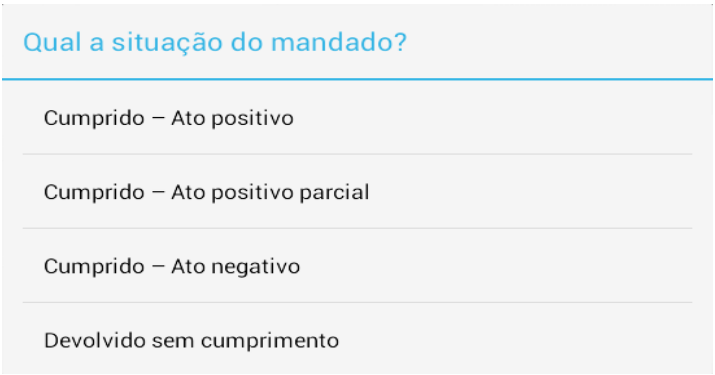


Figura 15 – Mensagem de informação da nova situação do mandado

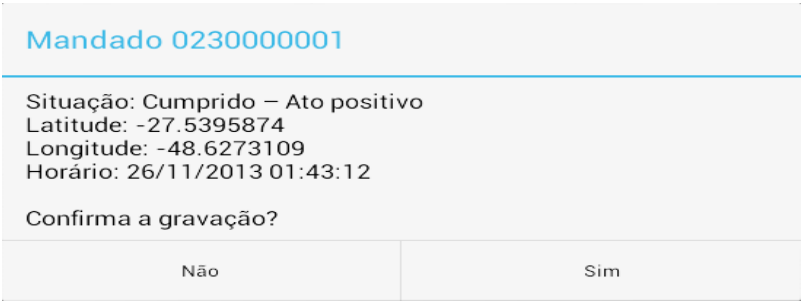


Figura 16 – Mensagem de confirmação para gravação da nova situação do mandado

Na figura 17 é apresentada a tela *Exportar Mandados*, nela verificamos a presença de dois botões. Um deles responsável por exportar os mandados cumpridos, conforme informação registrada pelo Oficial de Justiça, e a outra os dados de rastreio, obtidos durante o deslocamento do usuário.

Figura 17 – Tela Exportar mandados

Nas figuras 18 e 19 observamos a mensagem de envio dos mandados cumpridos e a tela de êxito na operação. Ao selecionar o botão *Mandados Cumpridos* o método *cumprimento* é acionado via *web service*. Uma sequência de caracteres com os dados obtidos na execução da atividade da tela *Cumprir mandado* é enviada ao servidor e o aplicativo recebe uma mensagem de sucesso confirmando o recebimento. No apêndice H podemos verificar o fluxograma desta atividade.

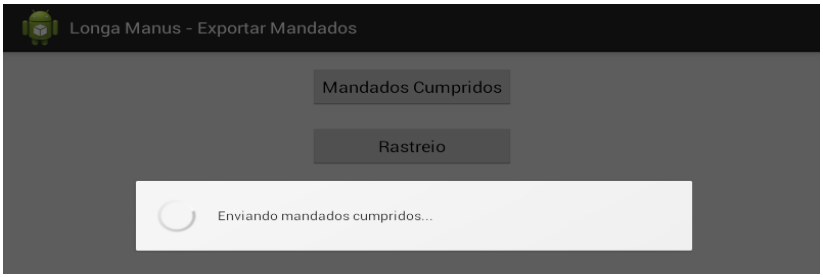


Figura 18 – Exportando mandados cumpridos

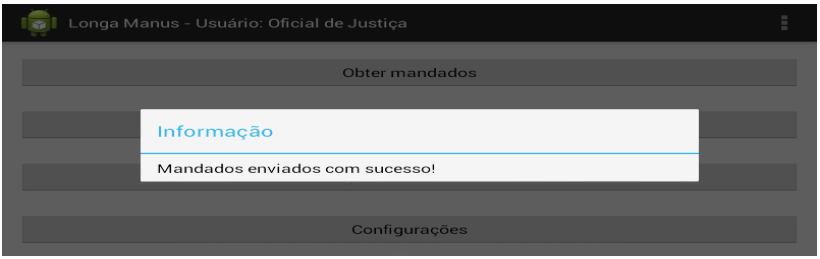


Figura 19 – Mensagem resultante da exportação de mandados cumpridos

As figuras 20 e 21 apresentam as telas resultantes da seleção do botão Rastreio. A primeira figura exibe um contador indicando o progresso no envio dos registros obtidos durante o deslocamento do Oficial de Justiça enquanto portava o dispositivo móvel, ao término da exportação a figura 21 é exibida. Similarmente aos envio dos mandados cumpridos, o envio do rastreio é feito invocando-se o método *rastreio* do *web service*. As atividades de exportação das informações de rastreio podem ser verificadas no fluxograma disponível no apêndice I.

Figura 20 – Exportando coordenadas geográficas de rastreamento

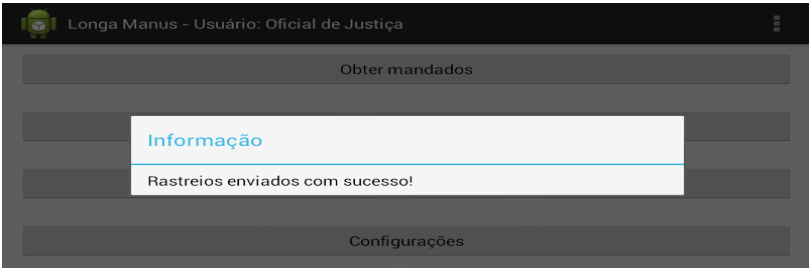


Figura 21 – Mensagem resultante da exportação de rastreios

Por fim, temos a figura 22 contendo a tela de configurações do aplicativo. Nela teremos os dados de conexão com o *web service*, endereço *IP* e *porta*, que receberá as requisições do aplicativo para receber os mandados a serem cumpridos e enviar os dados dos mandados cumpridos e rastreio.

Adicionalmente, temos as informações de tempo de rastreio, que possibilitará a configuração do intervalo de tempo entre a captura das informações de rastreio, permitindo configurar mais ou menos detalhamento das informações registradas. Uma caixa de checagem chamada *Coletar dados de rastreio* é responsável por permitir habilitar e desabilitar o rastreio do dispositivo móvel. Os últimos campos são *matrícula* e *senha* do oficial, estas informações são enviadas em toda conexão do dispositivo móvel com o *web service*, visando aumentar a segurança na troca de informações com o servidor *web*.

Figura 22 – Tela Configurações

2.3 Apresentação dos Resultados

Foram utilizados dois dispositivos móveis para testar o aplicativo móvel desenvolvido: um *smartphone* Mototrola Razr XT918 e um *tablet* Samsung GT-N8000. A escolha dos dispositivos para rodar a aplicação desenvolvida para uso do Oficial de Justiça foi definida baseada nos seguintes critérios:

- econômicos, como gratuidade para desenvolvimento na plataforma;
- baixo valor para aquisição dos dispositivos móveis;
- facilidade de uso;
- acesso via *web service*;
- banco de dados;
- presença de módulo GPS;
- acesso a rede.

Os testes realizados consistiram em deslocar-se portando o dispositivo por trechos pré-determinados e ao final do mesmo realizar a transmissão do rastreio obtido via *web service* ao servidor *web*. Posteriormente, foi realizada a verificação do trajeto percorrido adicionando as coordenadas obtidas em um mapa gerado pela API Google Maps. As verificações realizadas estavam condizentes com o trajeto realizado. É possível verificar na figura 23 uma linha vermelha disposta sobre o mapa, esta linha representa um deslocamento realizado entre os municípios de São José e Florianópolis para fins de teste do mecanismo de rastreio.

Na figura 24, temos os mesmos dados de rastreio, porém visualizando marcadores que registraram pontos específicos em que foram realizadas as coletas de dados. O número 13 indica o horário em que o dispositivo estava circulando, 13:45:48, conforme mensagem exibida sobre o marcador ao posicionar o mouse sobre o mesmo. É possível ver um exemplo desta mensagem no canto esquerdo inferior da figura.

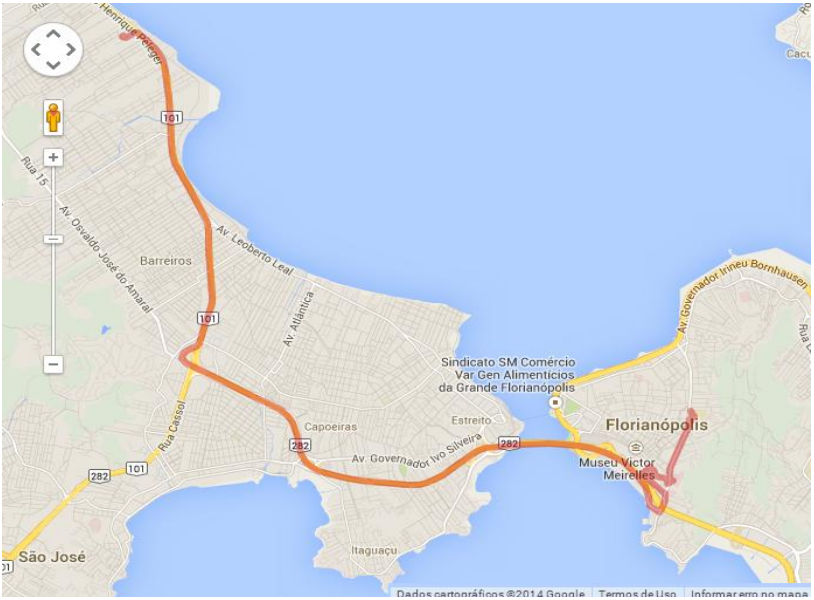


Figura 23 – Demonstração do teste de deslocamento utilizando linhas

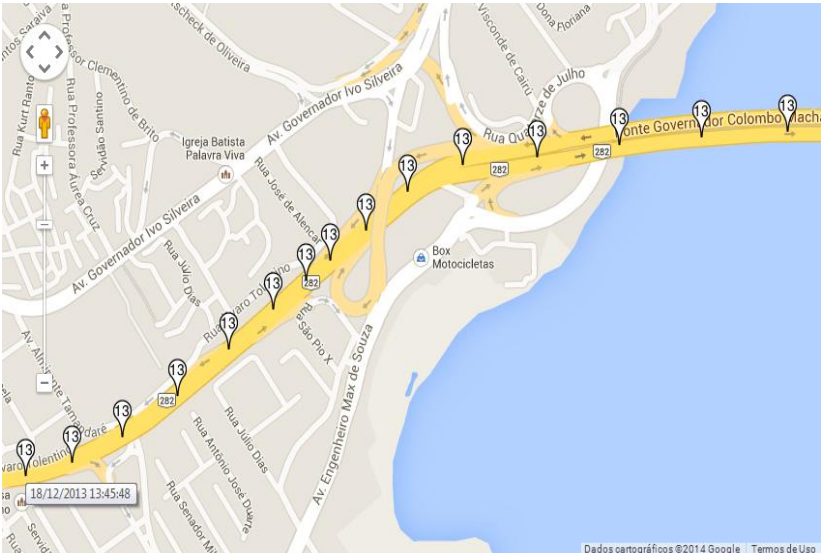


Figura 24 – Demonstração do teste de deslocamento utilizando pontos

A Figura 25 apresenta pontos contendo mandados cumpridos e a cumprir. Os mandados cumpridos estão coloridos em azul e os mandados a cumprir estão coloridos em vermelho. Esta ferramenta visual permite verificar o ponto em que o Oficial de Justiça confirmou o cumprimento de um mandado em relação à coordenada geográfica designada para o cumprimento da atividade. Na figura apresentada temos dois mandados em vermelho ainda não cumpridos, números 2 e 4, e um mandado cumprido, número 3. Podemos notar que o mandado 3 foi cumprido porém está um pouco distante do ponto definido para a entrega do mandado. Neste caso a coordenada geográfica do endereço cadastrado aponta para a entrada do local de cumprimento do mandado judicial enquanto o cumprimento do mandado foi realizado no estacionamento do local indicado. Apesar deste caso ser plausível, para esta situação, podem ocorrer casos em que haja a ocorrência de um registro indevido de visita ao local de cumprimento do mandado. Por outro viés é interessante propôr-se um raio de tolerância que permita lidar com possíveis margens de erro na captura do módulo GPS do dispositivo móvel utilizado pelo Oficial de Justiça.

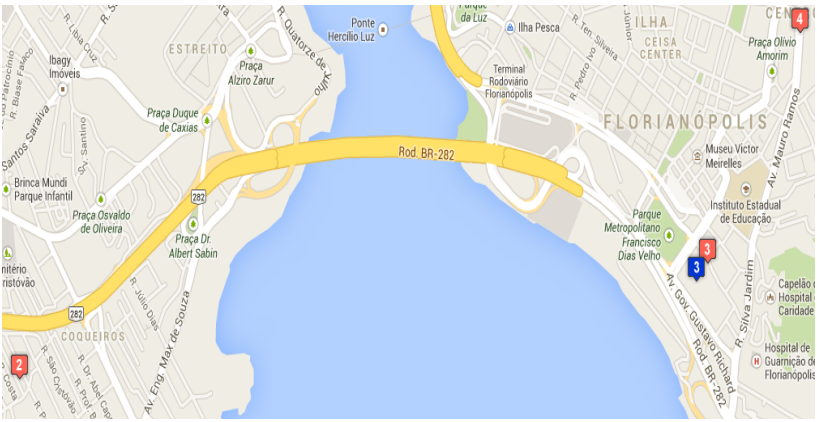


Figura 25 – Página HTML contendo mapa com coordenadas geográficas dos mandados

3 CONCLUSÃO

Apesar de não termos a possibilidade de realizarmos testes do aplicativo com os usuários finais foram realizadas capturas de dados de forma a simular um usuário portando o dispositivo móvel. As figuras contendo pontos de coordenadas geográficas inseridas sobre mapas foram obtidas a partir do aplicativo móvel *Longa Manus*.

Entre os aspectos verificados está a configuração do campo *Tempo de Rastreio* na tela de configurações. Em distâncias percorridas rapidamente, um intervalo para tempo de rastreio alto gera grandes espaçamentos, que acabam por comprometer a inteligibilidade do mapa com coordenadas geográficas apresentadas. Porém, intervalos pequenos acabam gerando pontos em excesso, o que acaba por deixar a execução da página web para verificação do rastreio do Oficial de Justiça muito lenta e visualmente poluída. Uma sugestão seria trabalhar com a velocidade de deslocamento do dispositivo, permitindo gerar menos pontos em velocidades mais baixas e aumentar a quantidade de pontos em deslocamentos com velocidades mais elevadas.

Um outro ponto a ser observado é a precisão do GPS. Em alguns casos a utilização do equipamento em ambientes fechados ou urbanos gera distorções grotescas de localização no dispositivo. Diversos artigos sobre cânions urbanos estão disponíveis na internet, além de existir um comparativo disponível em SPARKFUN, 2013, sobre o assunto. A definição de uma área para validação do cumprimento do mandado deveria ser aprofundada para não gerar inconsistências na validação do cumprimento do mandado.

Entre os objetivos específicos levantados cumprimos a revisão da bibliografia acerca do tema trabalhado e conhecemos as técnicas para desenvolvimento de aplicativos Android. Na parte de aplicação prática, a partir dos conhecimentos adquiridos, desenvolvemos o aplicativo móvel Android para aquisição dos dados relativos as atividades do Oficial de Justiça, assim como criamos o *web service* para recepção e envio de informações de monitoramento. Por fim, conseguimos inserir as informações capturadas e pudemos simular visualmente o rastreio do deslocamento e o cumprimento de mandados de um Oficial de Justiça, utilizando as coordenadas geográficas obtidas através do aplicativo móvel e visualizando estes dados em um mapa utilizando API do

Google Maps.

Dessa forma concluímos que o projeto apresentado alcançou os objetivos pretendidos e poderia ser de extrema utilidade caso sejam concretizadas as hipóteses levantadas no escopo deste trabalho. O rastreio do veículo automotor através de um dispositivo móvel para verificação da correta utilização de um veículo cedido ao Oficial de Justiça, assim como o pagamento de condução por quilometragem percorrida teriam o auxílio necessário utilizando-se a solução apresentada neste trabalho de conclusão de curso. Não podemos esquecer das possibilidades apresentadas pela solução de verificação do cumprimento dos mandados judiciais ao oferecermos mecanismos para a obtenção das coordenadas geográficas, registro das mesmas vinculadas aos dados de um mandado e envio das informações para um *web service* em um servidor *web*.

Podemos oferecer algumas possibilidades para trabalhos futuros a serem desenvolvidos, especializando ainda mais este projeto, como:

- ferramenta para análise das coordenadas geográficas de cumprimento dos mandados a partir dos dados coletados pelo dispositivo móvel;
- ferramenta para análise das coordenadas geográficas de cumprimento dos mandados e o trajeto realizado pelo Oficial de Justiça para cumprimento dos mandados;
- alerta de proximidades visual e sonora dos pontos disponibilizados para cumprimento dos mandados;
- realizar o incremento de segurança na aplicação e no web service, principalmente no que se refere à integridade dos dados.

REFERÊNCIAS

MEGDA, Ronaldo. **Monitoramento versus Rastreamento**. Disponível em http://infogps.uol.com.br/blog/2012/02/23/monitoramento-versus-rastreamento/. Acesso em: 14.jun.2013.

SERAFIM, Jhonata Goulart. **Os auxiliares da justiça no âmbito do direito processual civil analisado pela doutrina**. Amicus Curiae, Criciúma, Volume 9, Número 9, 2012.

BRASIL. **Lei n. 5869, de 11 de janeiro de 1973**. Institui o Código de Processo Civil. Vade Mecum Saraiva, São Paulo, p. 387, 2009. Legislação Federal.

JÚNIOR, Humberto Theodoro. **Curso de direito processual civil**. 50. ed. Rio de Janeiro: Forense, 2009.

CONSELHO NACIONAL DE JUSTIÇA. **Ato normativo 0007097-66.2009.2.00.0000, de 2 de dezembro de 2009**. Resolução 48/07 do CNJ – Exigência de nível superior para investidura no cargo de oficial de justiça - Acréscimo de dispositivo à resolução – Regra de transição para as legislações estaduais em sentido contrário ao ato normativo do Conselho. Disponível em https://www.cnj.jus.br/ecnj/consulta_processo.php?num_processo_consulta=00070976620092000000&consulta=s. Acesso em: 10.jun.2013.

TRIBUNAL PLENO DO TRIBUNAL DE JUSTIÇA DO ESTADO DE MATO GROSSO DO SUL. **Resolução 380, de 24 de abril de 2002**. Regulamenta a Lei n. 2.388, de 26 de dezembro de 2001, que dispôs sobre a indenização de transporte aos oficiais de justiça e avaliadores do Poder Judiciário do Estado de Mato Grosso do Sul. Campo Grande: 2002. Disponível em http://www.tjms.jus.br/sistemas/biblioteca/legislacao_comp.php?lei=17465. Acesso em: 9.jun.2013.

CORREGEDORIA-GERAL DA JUSTIÇA. **Comunicado Eletrônico n. 50**. Situação dos mandados Sistema SAJ 3.0 e 5.0. Florianópolis: 2013. 1 p.

SANTA CATARINA. **Lei Complementar n. 156, de 15 de maio de 1997**. Dispõe sobre o Regimento de Custas e Emolumentos e adota outras providências. Disponível em http://cgj.tj.sc.gov.br/consultas/liberada/regcustas_emolumentos.pdf. Acesso em: 10.jun.2013. Legislação Estadual.

CNET. **What makes a tablet a tablet?**. Disponível em http://news.cnet.com/8301-31021_3-20006077-260.html?tag=newsLeadStoriesArea.1. Acesso em: 26.dez.2013.

PHONE SCOOP. **Smartphone definition**. Disponível em http://www.phonescoop.com/glossary/term.php?gid=131. Acesso em: 26.dez.2013.

PC MAGAZINE. **Tablet computer definition from PC Magazine Encyclopedia**. Disponível em http://www.pcmag.com.br/us/encyclopedia/term/52520/tablet-computer. Acesso em: 26.dez.2013.

FORBES. **Why get a tablet when you can have a phablet?**. Disponível em http://www.forbes.com/sites/parmyolson/2012/02/28/why-get-a-tablet-when-you-can-have-a-phablet. Acesso em: 26.dez.2013.

ORACLE. **About the Java Technology**. Disponível em http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html. Acesso em: 29.dez.2013.

_____. **Overview of Enterprise Applications**. Disponível em http://docs.oracle.com/javaee/6/firstcup/doc/gcrky.html. Acesso em: 29.dez.2013.

_____. **What Is a Servlet?**. Disponível em http://docs.oracle.com/javaee/1.4/tutorial/doc/Servlets2.html. Acesso

em: 29.dez.2013.

_____. **What Is a JSP Page?**. Disponível em http://docs.oracle.com/javaee/1.4/tutorial/doc/JSPIntro2.html. Acesso em: 29.dez.2013.

_____. **Enterprise JavaBeans Technology**. Disponível em http://www.oracle.com/technetwork/java/javaee/ejb/index.html. Acesso em: 29.dez.2013.

_____. **What Is an Enterprise Bean?**. Disponível em http://docs.oracle.com/javaee/1.4/tutorial/doc/EJBConcepts2.html. Acesso em: 29.dez.2013.

_____. **Java EE Servers**. Disponível em http://docs.oracle.com/javaee/6/firstcup/doc/gcrkq.html. Acesso em: 29.dez.2013.

APACHE. **Apache Tomcat**. Disponível em http://tomcat.apache.org/. Acesso em: 29.dez.2013.

MICROSFOT. **Structured Query Language (SQL)**. Disponível em http://msdn.microsoft.com/en-gb/library/windows/desktop/ms714670(v=vs.85).aspx. Acesso em: 31.dez.2013.

SQLite. **SQLite Home Page**. Disponível em http://www.sqlite.org/. Acesso em: 31.dez.2013.

_____. **SQLite is Transactional**. Disponível em http://www.sqlite.org/transactional.html. Acesso em: 31.dez.2013.

MYSQL. **Top 10 Reasons to Choose MySQL for Next Generation Web Applications**. Disponível em http://www.mysql.com/why-mysql/white-papers/top-10-reasons-to-choose-mysql-for-next-generation-web-applications/. Acesso em: 31.dez.2013.

W3SCHOOLS. **Web Services Tutorial**. Disponível em

http://www.w3schools.com/webservices/default.asp. Acesso em: 01.jan.2014.

_____. **Ajax Tutorial**. Disponível em http://www.w3schools.com/ajax/. Acesso em: 03.jan.2014.

ORACLE. **Introduction to XML**. http://docs.oracle.com/javaee/1.4/tutorial/doc/IntroXML2.html. Acesso em: 01.jan.2014.

METRO. **Metro**. Disponível em https://metro.java.net/. Acesso em: 01.jan.2014.

KSOAP. **ksoap2-android - A lightweight and efficient SOAP library for the Android platform**. Disponível em https://code.google.com/p/ksoap2-android/. Acesso em: 01.jan.2014.

JSON. **JSON**. Disponível em http://www.json.org/json-pt.html. Acesso em: 03.jan.2014.

DARCEY, Lauren; CONDER, Shane. **Android: Wireless Application Development**. 2. ed. Boston: Addison-Wesley, 2009. 761 p.

TECH-THOUGHTS. **Global Smartphone Market Share Trends - Q1 2013**. Disponível em http://www.tech-thoughts.net/2013/05/global-smartphone-market-share-trends-android-iphone-windows-phone.html. Acesso em: 20.jul.2013.

ABLESON, W. Frank; SEN, Robi; KING, Chris; ORTIZ, C. Enrique. **Android em ação**. 3. ed. Tradução Eduardo Kraszczuck e Edson Furmankiewicz. Rio de Janeiro: Elsevier, 2012. 622 p.

OPEN HANDSET ALLIANCE. **Members**. Disponível em http://www.openhandsetalliance.com/oha_members.html. Acesso em: 5.ago.2013.

ANDROID. **Android Developers**. Disponível em http://developer.android.com/index.html. Acesso em: 22.ago.2013.

ANDROID. **Google Maps Android API v2**. Disponível em https://developers.google.com/maps/documentation/android/. Acesso em: 6.jan.2014.

WIKIPEDIA. **Location awareness**. Disponível em http://en.wikipedia.org/wiki/Location-aware. Acesso em: 06.out.2013.

_____. **Android System Architecture**. Disponível em http://en.wikipedia.org/wiki/File:Android-System-Architecture.svg. Acesso em 10.jan.2014.

SPARKFUN. **GPS Tracking Comparisons**. Disponível em https://www.sparkfun.com/tutorials/169. Acesso em: 1.dez.2013.

ANEXO A - COMUNICADO ELETRÔNICO N. 50

Comunicado Eletrônico CGJ n. 50

Data: 06/09/2013
Destino: Chefe de Cartório, Chefe da Central de Mandado e Oficiais de Justiça
Assunto: Situação dos mandados SAJ 3.0 e 5.0.
Autos n.º: 0010782-52.2012.8.24.0600

O Oficial de Justiça, ao devolver os mandados ao Cartório ou à Central de Mandados, deverá atualizar no sistema a situação do mandado.
Cumpre ressaltar que há situações diferentes para o **SAJ/PG3** e o **SAJ/PG5**, quais sejam:

1) As seguintes situações no **SAJ/PG3** são passíveis de escolha pelos oficiais no momento da certificação:

a) **Cumprido** (ato positivo), aquele cuja ordem foi executada na íntegra, ou que, contendo ordens sucessivas e, uma delas tendo sido cumprida, tenha esgotado o objeto do mandado. Ex.: mandado de citação – citação efetuada = mandado cumprido; mandado de citação e penhora – citação efetuada e parcelamento do débito = cumprido.

b) **Parcialmente cumprido** (ato positivo parcial), o que, contendo mais de uma ordem, tenha sido devolvido com uma ou mais ordens não executadas. Ex.: mandado de citação, penhora e avaliação – realizadas a citação e a penhora, mas não a avaliação = mandado parcialmente cumprido.

c) **Não cumprido** (sem cumprimento – ato negativo), o que não teve executada qualquer das ordens nele contidas.

2) As seguintes situações no **SAJ/PG5** são passíveis de escolha pelos oficiais no momento da certificação:

a) **Cumprido – Ato positivo**, aquele cuja ordem foi executada na íntegra, ou que, contendo ordens sucessivas e, uma delas tendo sido cumprida, tenha esgotado o objeto do mandado. Ex.: mandado de citação – citação efetuada = mandado cumprido; mandado de citação e penhora – citação efetuada e parcelamento do débito = cumprido.

b) **Cumprido - Ato positivo parcial**, o que, contendo mais de uma ordem, tenha sido devolvido com uma ou mais ordens não executadas. Ex.: mandado de citação, penhora e avaliação – realizadas a citação e a penhora, mas não a avaliação = mandado

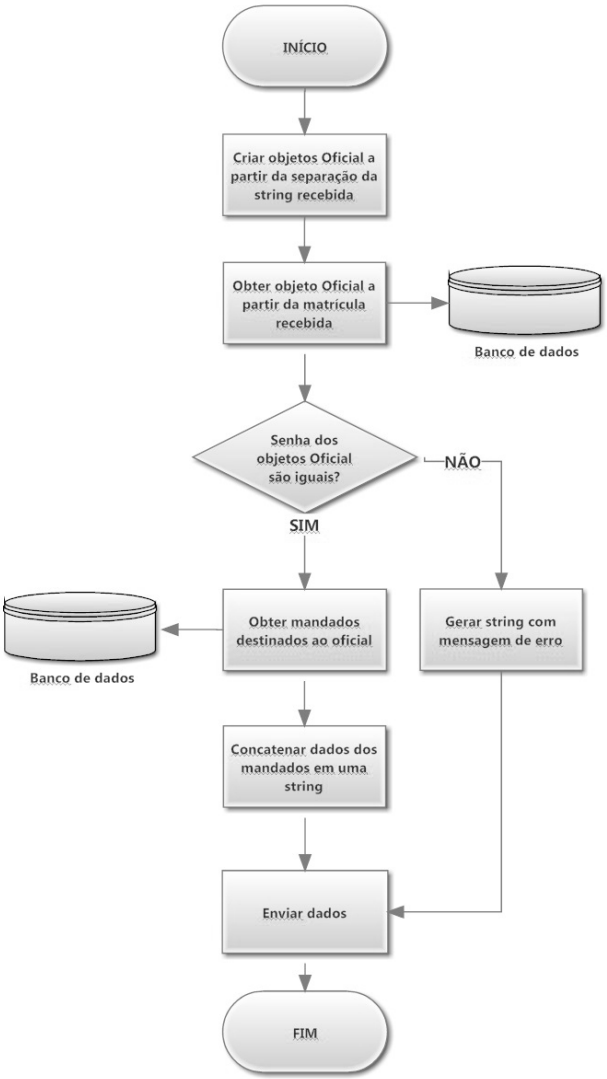
parcialmente cumprido.
c) **Cumprido – Ato negativo**, aquele em que nenhuma ordem foi executada, porém houve diligência.
d) **Devolvido sem cumprimento**, aquele em que nenhuma ordem foi executada e não houve diligência. Ex.: mandado encaminhado para o oficial errado, ou faltando informações, dentre outras. Neste caso, o oficial certifica que não houve o cumprimento do mandado e devolve para a central.

Atenciosamente,

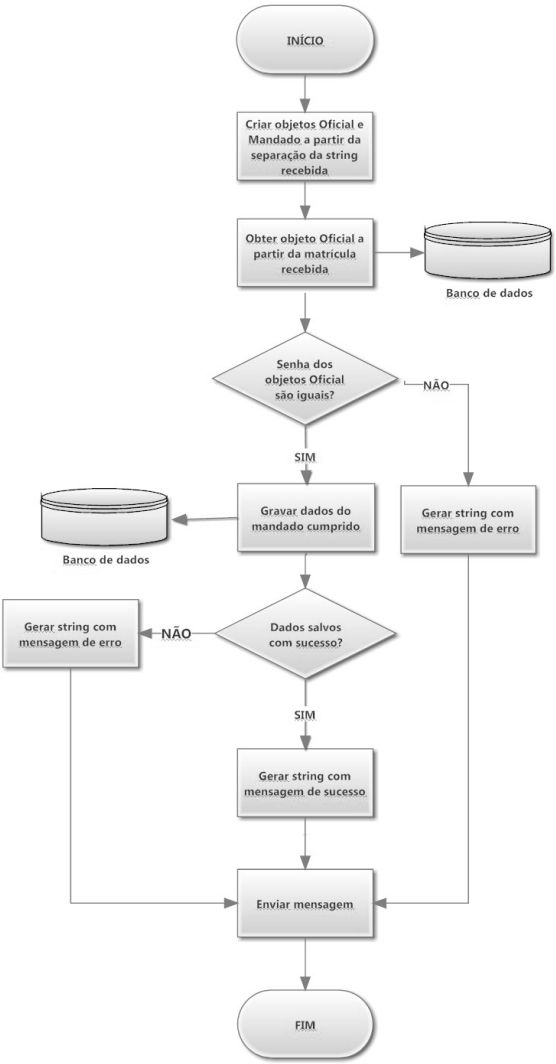
Corregedoria-Geral da Justiça

APÊNDICES

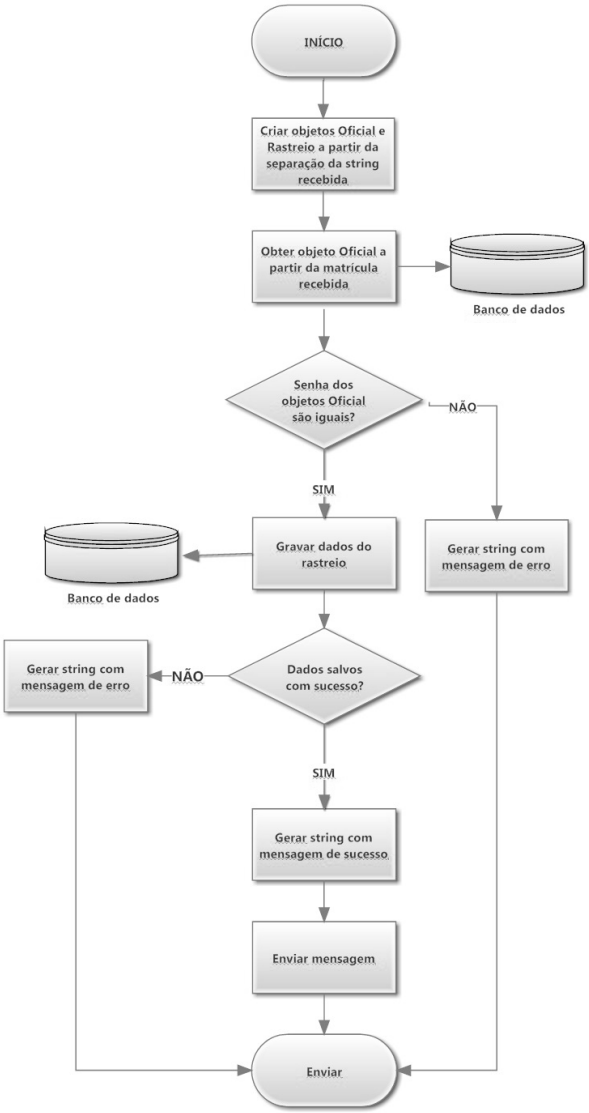
APÊNDICE A – FLUXOGRAMA DO WEB SERVICE MANDADO



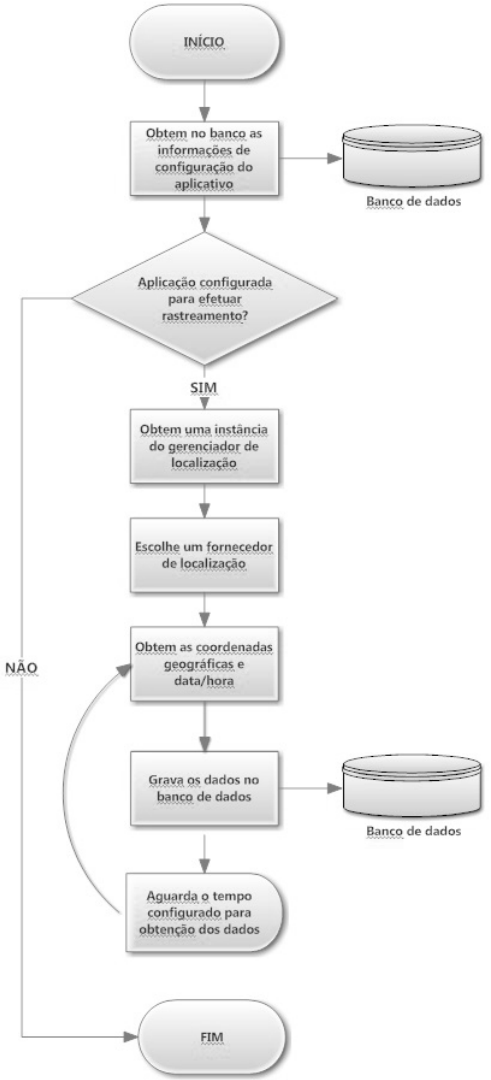
APÊNDICE B - FLUXOGRAMA DO WEB SERVICE CUMPRIMENTO



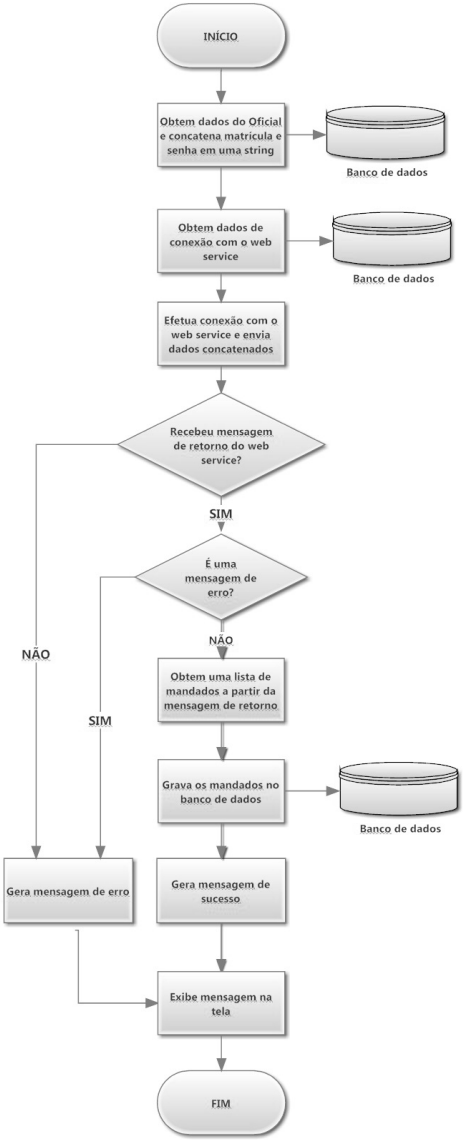
APÊNDICE C – FLUXOGRAMA DO WEB SERVICE RASTREIO



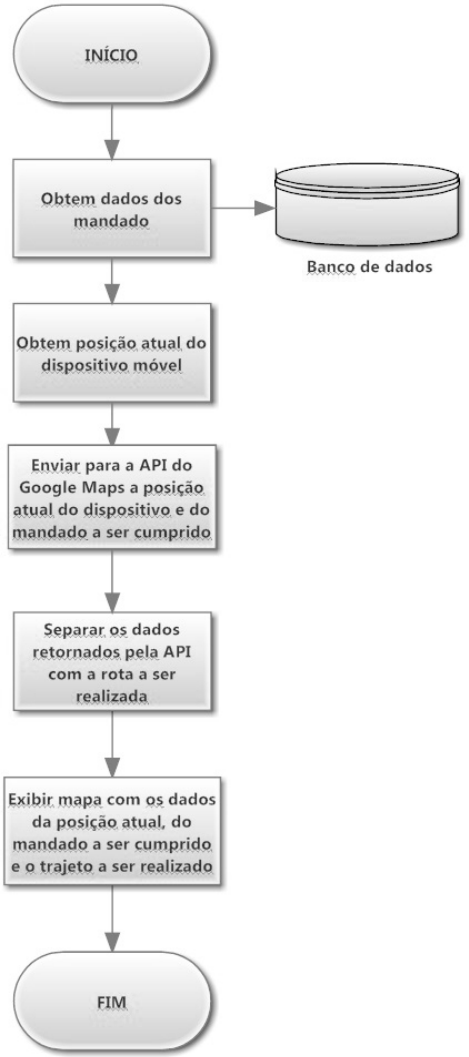
APÊNDICE D – FLUXOGRAMA DA ATIVIDADE REALIZAR RASTREIO



APÊNDICE E – FLUXOGRAMA DA ATIVIDADE OBTER MANDADO

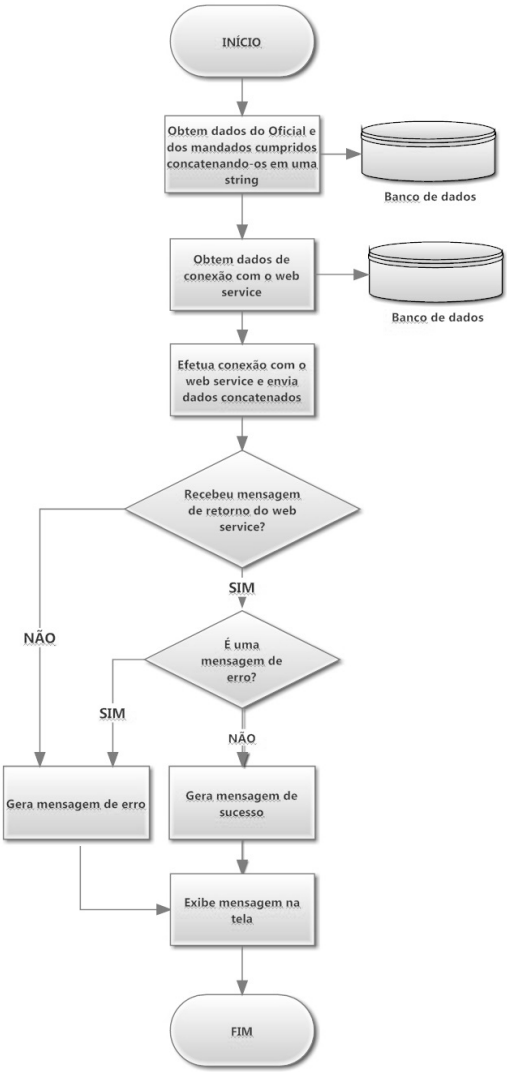


APÊNDICE F – FLUXOGRAMA DA ATIVIDADE TRAÇAR ROTA

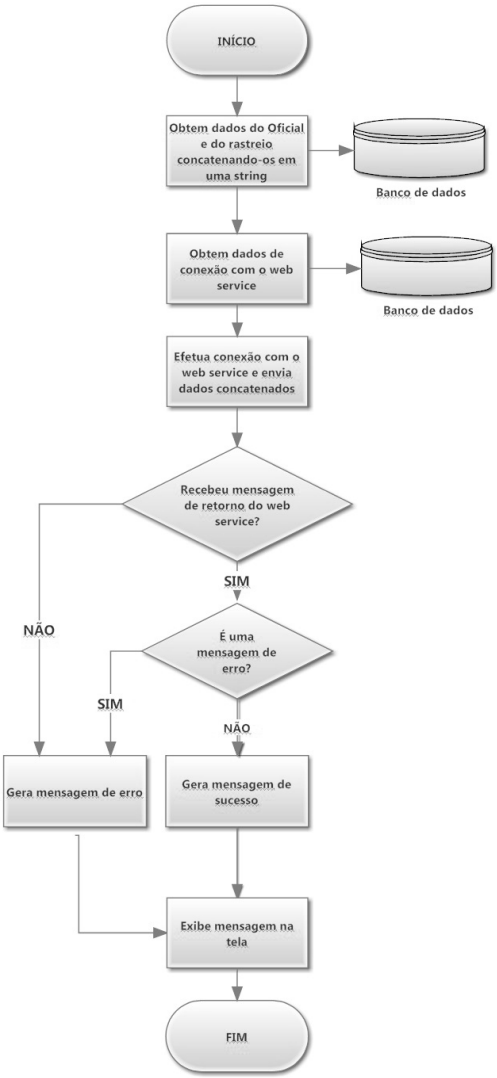


APÊNDICE G – FLUXOGRAMA DA ATIVIDADE CUMPRIR MANDADO

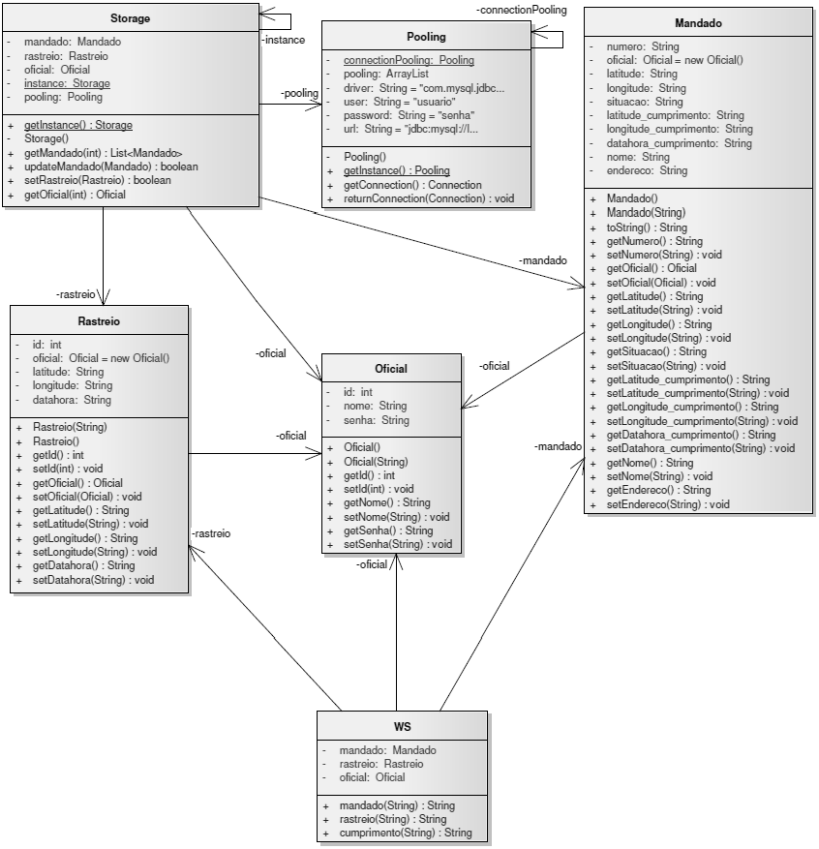
APÊNDICE H – FLUXOGRAMA DA ATIVIDADE ENVIAR MANDADOS CUMPRIDOS



APÊNDICE I – FLUXOGRAMA DA ATIVIDADE ENVIAR RASTREIO



APÊNDICE J – DIAGRAMA DE CLASSES DO WEB SERVICE



APÊNDICE K – DIAGRAMA DE CLASSES DA APLICAÇÃO MÓVEL