UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA E DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

VALMIR KIELTYKA

SIRASS – SISTEMA DE RASTREAMENTO DE SMARTPHONE

CURITIBA
2013

VALMIR KIELTYKA

SIRASS – SISTEMA DE RASTREAMENTO DE SMARTPHONE

Monografia de especialização apresentada ao Curso de Especialização em Tecnologia Java e Desenvolvimento para Dispositivos Móveis da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de especialista.

Orientador: Professor Dr. João Alberto Fabro

CURITIBA
2013

AGRADECIMENTOS

Primeiramente a DEUS e a minha família. Aos professores dedicados e pacientes do Curso de Especialização Java e Desenvolvimento para Dispositivos Móveis que transmitiram com muita motivação o conhecimento de qualidade. Ao professor Fabro pela paciência, ajuda, conhecimento e conselhos vitais na confecção do documento. A todos que contribuíram, ajudaram e aconselharam de forma direta e indireta no desenvolvimento desta monografia.

RESUMO

KIELTYKA, Valmir. **Sistema de Rastreamento de Smartphone**. 2013. 73 páginas. Monografia de Especialização em Tecnologia Java e Desenvolvimento para sistemas móveis – Universidade Tecnológica Federal do Paraná, 2013.

O surgimento de novas tecnologias e a melhoria das já existentes apoia o desenvolvimento de novos sistemas que contribuem para o desenvolvimento da tecnológica. A necessidade de manter-se disponível e conectado a web faz com que os aparatos tecnológicos estejam em nosso bolso de dia e a noite, para estarmos acessíveis a qualquer momento. Este projeto tem por objetivo a criação de um sistema para localização em tempo real de aparelhos celulares Smartphones, utilizando a plataforma Android para desenvolvimento do aplicativo com características dos sistemas de geoprocessamento para dispositivos móveis.

Palavras-chave: Android, WebService, GPS, Geoprocessamento.

ABSTRACT

Kiełtyka, Valmir. **Smartphone Tracking System.** In 2013. 73 pages. Specialization in Technology Development for Java and Mobile Systems - Federal University of Technology - Paraná, 2013.

The emergence of new technologies allows the development of new systems that contribute to the technological advance. The need to keep available and connected to the web all time makes us to walk with technological devices in the pocket day and night, thus being accessible at any time. This work aims to create a system for locating mobile Smartphones, using the Android platform to develop an application with geoprocessing capabilities, applied to locate and track mobile devices.

Keywords: Android, WebServices, GPS, Geoprocessing.

LISTA DE SIGLAS

ADT - Android Development Tools
API - Application Programming Interface
DNS - Domain Name Service
EJB - Enterprise Java Beans
GPS – Global Positioning System - Sistema de Posicionamento Global
HTTP - Hyper Text Transfer Protocol
IDE - Integrated Development Environment
IP - Internet Protocol
J2EE - Java 2 Enterprise Edition
JEE - Java Enterprise Edition
JAAS - Java Authentication and Authorization Service
JAX-B – Java API for XML Binding
JAX-RPC - Java API for XML - based RPC
JAX-WS - Java API for Xml – Java para Web Services XML
JDBC - Java Database Connectivity
JDK - Java Development Kit
JMS - Java Message Service
JMX - Java Management Extensions
JNDI - Java Naming and Directory Interface
JPA - Java Persistence API
JPQL - Java Persistence Query Language
JRE - Java Runtime Environment
JSF - Java Server Faces
JSP - Java Server Pages
JTA - Java Transaction API
LAN - Local Area Network
ORM - Object Relational Mapping
PC - Personal Computer
REST - Representational State Transfer
SEI - Service Endpoint Interface

SDK - Software Development Kit

SGDB - Sistema Gerenciador de Banco de Dados

SIB - Service Implementation Bean

SMS - Short Message Service

SOA - Service Oriented Architecture

SOAP - Simple Object Access Protocol

SQL - Structured Query Language

UML - Unified Modeling Language

URL - Uniform Resource Locator

UTFPR - Universidade Tecnológica Federal do Paraná

VPN - Virtual Private Network - Rede Privada Virtual

XML - Extensible Markup Language

WSDL - Web Service Definition Language

WSS - Web Services Security

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRO

1 INTRODUÇÃO

Diariamente, a internet é acessada por milhões de pessoas em todo o mundo, sem mencionar os inúmeros negócios fechados simultaneamente, fazendo a movimentação de bilhões e bilhões de dólares. Nas duas últimas décadas as transformações e as evoluções tecnológicas têm acompanhado a vida das pessoas, sem que estas se deem conta desse processo.

E, quando o assunto é informação, ou melhor, transmitir informação, nada melhor do que utilizar a tecnologia. A cada dia, mais e mais serviços são disponibilizados por empresas para que as pessoas interessadas tenham acesso a informações de maneira rápida e eficaz.

Para o acesso as informações que são atualizadas frequentemente possuímos a rede de conexão, internet, acessada principalmente de computadores, porem há uma crescente nestes últimos anos para área dos celulares, principalmente os *smartphones*.

O uso de *smartphones* para acesso e transmissão de dados vem crescendo assustadoramente, tanto no Brasil como no mundo, conforme a matéria divulgada pelo site da Folha de São Paulo[1],

> De 2011 para 2012, a quantidade de dados consumidos por cada *smartphone* praticamente dobrou, pois a média global passou de 189 Mbytes por mês para 342 Mbytes por mês, um aumento de 81%. As informações são do estudo Visual Networking Index, da Cisco, que também registrou um aumento de 70% no tráfego global de dados móveis no mesmo período de 520 petabytes por mês para 885 petabytes mensais. Um petabytes equivale a mil terabytes, ou 250 mil DVDs. No Brasil, o crescimento foi ligeiramente menor do que a média global --de 11,8 petabytes, em 2011,

[1] FOLHA DE S.PAULO. **Uso de dados móveis em smartphones quase dobra em um ano**. Disponível em:<http://www1.folha.uol.com.br/tec/1239448-uso-de-dados-moveis-em-smartphone-quase-dobra-em-um-ano.shtml>. Acessado em 20/04/ 2013.

> para 19,8 petabytes, em 2012, um aumento de cerca de 68%.

Sendo assim, ao carregar um *smartphones* para suprir as necessidades de trabalho, assim como para o entretenimento, o usuário está mais sujeito a esquecer, perder ou até mesmo sofrer roubo ou furto do aparelho.

Segundo dados da *Federal Communications Commission* - FCC, órgão regulador das telecomunicações nos Estados Unidos, quase um a cada três roubos do país envolve telefones celulares. "A violência epidêmica de crimes de roubo e de revenda de aparelhos furtados é real e o crescimento desses crimes ameaça as comunidades americanas", disse Schneiderman em um comunicado à imprensa. De acordo com pesquisas, cerca de 113 *smartphones* são roubados ou perdidos a cada minutos nos Estados Unidos, e muitas vezes essas ações de roubo terminam em violência"[2].

Temos hoje no mercado alguns softwares específicos para localização de celular, porém uma grande parte deles com recursos limitados, de difícil instalação, com custos, necessidade de cadastros e repasse de informações confidenciais e de fácil interrupção do serviço.

A implementação e a modelagem do processo de negócio, proposto neste trabalho serão aplicados para usuários do sistema Android. Com base no estudo realizado sobre os *softwares* existentes e mais procurados no mercado, e os problemas apontados pelos usuários de *smartphones* que tiveram seu aparelho roubado ou perdido sem a possibilidade de rastreamento ou com limitações no rastreamento, este trabalho apresenta o desenvolvimento do Sistema de Rastreamento de *Smarthphone* - SIRASS - objetivando suprir uma alternativa de simples instalação e configuração, permitindo a localização remota.

[2] Disponível em <http://tecnologia.uol.com.br/noticias/redacao/2013/06/13/para-reduzir-roubo-de-aparelhos-promotores-americanos-sugerem-opcao-de-matar-smartphones.htm>. Acessado em 09/04/13.

1.1 CONTEXTUALIZAÇÃO

Empresas tanto da iniciativa pública quanto da iniciativa privada necessitam que seus funcionários estejam conectados a internet ou estejam com seus celulares ligados ininterruptamente para assim ficarem disponíveis para eventuais necessidades. Os *Smartphones* possibilitam que o usuário esteja conectado à rede e/ou disponível para ligações, mesmo não estando em sua sala ou local de trabalho, e podem ser levados para todos os lugares, por isso há uma maior possibilidade de perda por esquecimento em qualquer lugar visitado, ou ainda por roubo do aparelho em qualquer lugar.

1.2 PROBLEMA

Devido ao grande montante de aparelhos *Smartphones* vendidos no Brasil, temos uma ideia da quantidade de usuários. Foi realizada uma pesquisa global conduzida pela empresa F-SECURE que disponibilizou a informação “em entrevista a usuários de banda larga de 14 países, no caso dos brasileiros, 25% afirmaram que já tiveram o dispositivo móvel roubado ou perdido, índice considerado alto se comparado à média mundial de 11%” [3].

Possuímos um índice bem alto no Brasil em relação a roubo ou perda de dispositivos móveis, sendo muito difícil localizar os aparelhos, pois não instalamos aplicativos de localização ou contratamos serviços de busca, com estes fatos acabamos perdendo importantes dados nele embarcado, sejam fotos, vídeos, mensagens entre outros documentos, resumindo as informações contidas em um dispositivo móvel podem até ser mais valiosas do que o próprio aparelho.

[3] TECMUNDO. **25% dos brasileiros já tiveram seu celular roubado ou perdido**. Disponível em: <http://www.tecmundo.com.br/celular/31075-25-dos-brasileiros-ja-tiveram-seu-celular-roubado-ou-perdido.htm>. Acessado em 20/04/2013.

1.3 OBJETIVO

1.3.1 Objetivo Geral

Desenvolver um aplicativo para o sistema operacional Android, que possibilite a monitoração em tempo real da localização do aparelho, utilizando um serviço de Web Service.

1.3.2 Objetivos Específicos

Os objetivos específicos estão compostos pelas atividades realizadas abaixo, para viabilizar o desenvolvimento do software:

- Identificação das funcionalidades necessárias e elaboração da documentação para análise técnica, as quais auxiliaram no desenvolvimento do software.
- Pesquisa de softwares similares existentes e trabalhos correlatos.
- Desenvolvimento de um sistema que monitore um *smartphone*, a fim de receber e enviar o ponto exato da sua localização no mapa, por meio de um aplicativo para o sistema operacional Android.
- Implantar uma aplicação Web Service para transmitir e obter os dados da localização do dispositivo móvel.

1.4 JUSTIFICATIVA

Devido ao grande crescimento da tecnologia e telecomunicações, a venda de *smartphones* com o sistema operacional Android aumentou significadamente. Com o uso ininterrupto do aparato tecnológico, estamos sujeitos à perda ou roubo do *smartphone*.

Com algum conhecimento uma pessoa, que veio a achar ou furtar um dispositivo móvel, pode retirar o chip do celular assim deduzindo que desabilitou serviços para localização do aparelho. Hoje no mercado há alguns softwares específicos para localização de celular, porém uma grande parte deles com recursos limitados, de difícil instalação e operação, com custos altos e ainda de fácil interrupção do serviço.

As melhorias de um sistema são objetivos essenciais para uma gama de entidades e pessoas, bem como o desenvolvimento de inovações que visem um maior desempenho, podendo resultar numa contribuição para a sociedade em geral.

Este trabalho apresenta como proposta o desenvolvimento do SIRASS, uma aplicação cliente-servidor utilizando-se de linguagem Java, arquitetura JEE e agente e cliente desenvolvidos para o sistema operacional Android, com o objetivo de repassar informações da localização geográfica de um *smartphone* e visualizá-las em um site da internet.

O presente projeto visa uma solução para reaver *smartphones* perdidos ou roubados, pois com as ferramentas práticas e simples do SIRASS é possível localizar o aparelho, tendo sua posição apontada no mapa.

1.5 ESTRUTURAS DA MONOGRAFIA

O trabalho em questão é composto de estudos a aplicações de várias tecnologias de mercado, tendo como resultado um sistemas para localização de *smartphones*.

O capítulo 1 foi uma abordagem geral da monografia, o cenário atual, seus objetivos e problemas envolvidos.

O capítulo 2 apresenta um referencial teórico abordando alguns aplicativos semelhantes ao SIRASS e descrevendo os conceitos das tecnologias utilizadas para o desenvolvimento do trabalho.

O capítulo 3 (UML) mostra o detalhamento do trabalho. Neste capítulo são apresentados todos os componentes desenvolvidos e integrados através da modelagem UML.

O capítulo 4 apresenta as funcionalidades do sistema e as telas das aplicações.

O capítulo 5 expõe as conclusões e considerações finais sobre o trabalho, bem como sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo tem como objetivo descrever a pesquisa efetuada sobre sistemas de informação e as tecnologias disponíveis no mercado, para dar a fundamentação teórica ao projeto.

2.1 TECNOLOGIAS

A seguir estão especificadas as tecnologias que serão utilizadas para o desenvolvimento do projeto.

2.1.1 GPS

O GPS (*Global Positioning System*) que significa sistema de posicionamento global, em português, é um sistema de navegação por satélite com um aparelho móvel que envia informações sobre posição de algo em qualquer horário e em qualquer condição climática.

O GPS foi desenvolvido pelo departamento de defesa dos Estados Unidos no início da década de 1970 (ELRABBANY, 2002, p.1)[4]. A tecnologia inicialmente era utilizada somente pelos militares, posteriormente os civis também passaram a ter acesso à tecnologia com uma precisão inferior à militar. Há pouco tempo era utilizado na aviação geral e comercial e na navegação marítima, porém, atualmente é utilizado por pessoas em automóveis e em dispositivos moveis, com um sistema de mapas, para localizar sua posição ou rotas para viajar.

Os satélites do GPS (apresentados na figura 1), segundo French (1996):

> “estão distribuídos em órbitas com 60º de espaçamento. Em cada órbita há aproximadamente 3 ou 4 satélites,

[4] EL-RABBANY, Ahmed. **Introduction to GPS: The Global Positioning System.** Boston: Artech House, 2000.

compondo um total de 24 satélites, considerando todas as órbitas. A estação central de controle pode movimentar os satélites dentro da órbita, porém não é possível a movimentação entre as órbitas”[5].

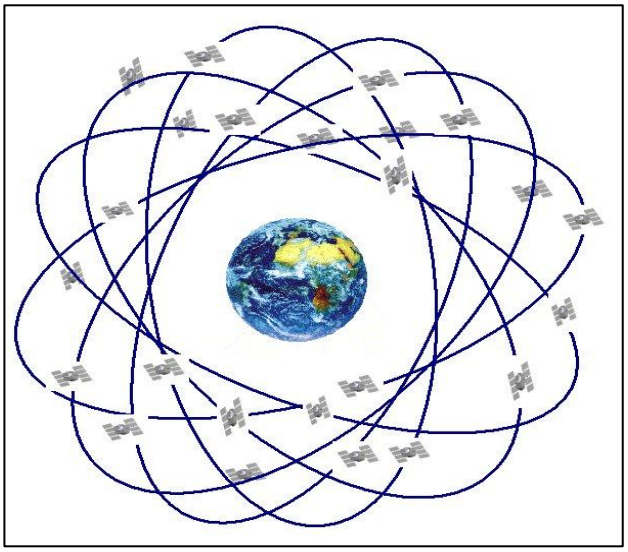

Figura 1 - Satélites do GPS
Fonte: http://www.fc.up.pt/lic_eg/

A figura 2 ilustra os segmentos do GPS. Conforme El-Rabbany (2002)[6] o GPS é formado por três segmentos espaço, controle e usuários. O espaço é formado pelos 24 satélites nas órbitas, o controle são as estações localizadas em vários lugares do mundo com o objetivo de monitorar os satélites e o funcionamento do sistema e ainda fazer ajustes da rota se necessário, e o segmento dos usuários compreende todos os que utilizam o sistema de posicionamento.

[5] FRENCH, Gregory T. **GPS An Introduction to the Global Positioning System**: **What It Is and How It Works**. Bethesda: GeoResearch, 1996.

[6] EL-RABBANY, Ahmed. **Introduction to GPS: The Global Positioning System.** Boston: Artech House, 2000, p. 02.

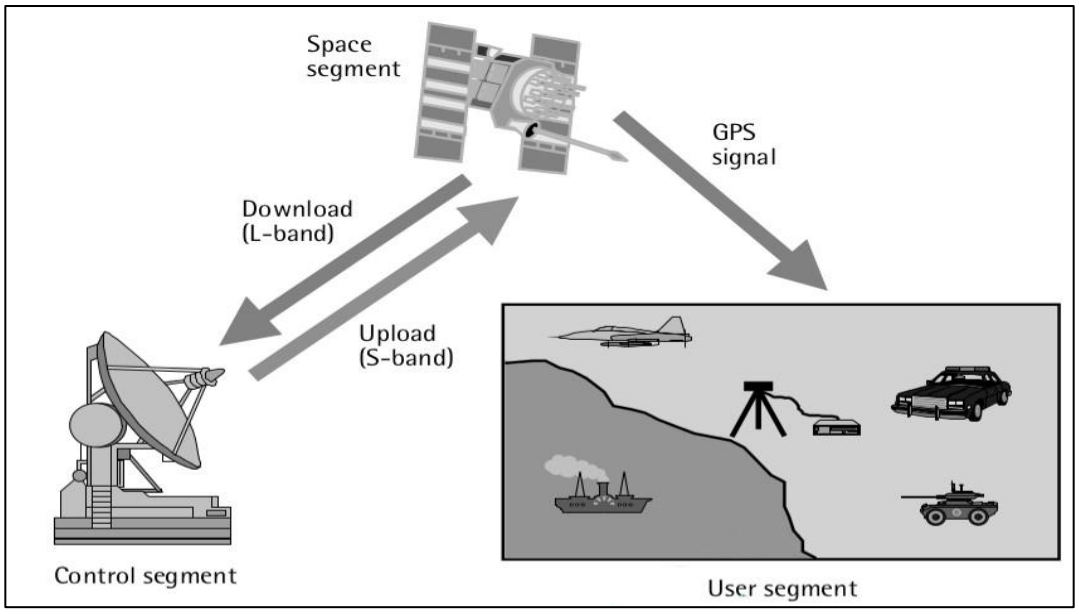

Figura 2 - Seguimento do GPS
Fonte: El-Rabbany (2002, p.1)

2.1.2 Linguagem de programação

Para Fischer e Grodzinsky *(1993)* a linguagem de programação é um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador [7].

Segundo Ana Cristina e Flávio Soares uma das principais metas das linguagens de programação é permitir que programadores tenham uma maior produtividade, permitindo expressar suas intenções mais facilmente do que quando comparado com a linguagem que um computador entende nativamente (código de máquina)[8].

Hoje contamos com várias linguagens de programação, entre elas destacam-se PHP e Java. A Linguagem Java foi utilizada neste projeto, por ser a com melhor suporte para o desenvolvimento em *smartphones* com o sistema operacional *Android*.

---

[7] FISCHER, Alice E.; GRODZINSKY, Frances. ***The Anatomy of Programming Languages.*** Englewood Cliffs, New Jersey: Prentice Hall, 1993. p. 3.

[8] MELO, Ana Cristina Vieira de; Silva, Flávio Soares Corrêa da. **Princípios de Linguagens de Programação.** São Paulo: Edgard Blücher Ltda, 2003. p. 7-11.

2.1.2.1 Java

A linguagem de programação Java foi desenvolvida em 1991 por um grupo de pesquisadores da Sun Microsystems, denominado *Green*. Seu objetivo era a utilização em dispositivos eletrônicos inteligentes, como set-top boxes, micro-ondas e controles remotos, porém não obtiveram o resultado esperado. Então, aproveitando-se da expansão da internet, em 1995 a Sun Microsystems oficialmente lançou a linguagem Java, que foi utilizada para geração de conteúdos dinâmicos nas páginas na Internet (DEITEL E DEITEL, 2003)[9].

A linguagem Java tem constantemente evoluído, estando presente cada vez mais no mercado e na área acadêmica. Com suas funcionalidades para as aplicações desktops, web e mobile, sua comunidade vem crescendo e ajudando cada vez mais no desenvolvimento da linguagem, tornando-se uma ferramenta de sucesso na solução de muitos problemas enfrentado pelos desenvolvedores atuais.

A linguagem de programação Java utiliza o paradigma de programação orientado a objetos é uma linguagem do tipo interpretada, ou seja, no processo de compilação o código fonte É transformado em bytecodes e para a interpretação destes é necessário um Máquina Virtual Java (JVM).

Segundo Schildt (2007)[10], uma das vantagens da existência do JVM é a de proporcionar a execução de programas Java em vários ambientes. Devido à presença da JVM na plataforma Java, as aplicações desenvolvidas são portáveis, ou seja, após o desenvolvimento do trabalho sua execução pode ser feita em qualquer Sistema Operacional que possua uma implementação da JVM, atualmente os principais Sistemas Operacionais como Windows, Linux e Mac OS possuem o JVM.

Ainda estão presentes dois serviços importantes na plataforma Java que são o *Garbage Collector*, utilizado para varrer os objetos criados e desnecessários assim liberando a memória, e a otimização realizada nos *bytecodes* gerados.

---

[9] A. DEITEL, Paul J.; DEITEL, H. M. **Java: how to program**. EUA: Prentice Hall. 2012. p. 09.

[10] SCHILDT, Herbert. **Java The Complete Reference**. Nova York: McGraw-Hill, 2007, p. 09.

2.1.3 Java Enterprise Edition – JAVA EE

Segundo Jendrock, Evans, Gollapudi, Haase e Srivathsa (2010):

> "O modelo de aplicação Java EE define uma arquitetura para implementar serviços como aplicações multicamadas, proporcionando a escalabilidade, acessibilidade e o gerenciamento, necessários para as aplicações corporativas[11]."

Sriganesh, Brose e Silverman (2006)[12] definem o Java EE como uma conglomeração de conceitos, padrões de programação e inovações, todos escritos na linguagem de programação Java.

2.1.4 Web Service

Kopack (2003)[13] define *Web Service* como um sistema que pode ser acessado remotamente, e semelhante a sites *Web* ele é identificado na rede por uma *URL* e fornece ao cliente uma resposta a uma requisição.

Web Service é um tipo de aplicação para a web sendo possível executar uma rotina externa através de algum protocolo de rede, é uma referência para integração de sistemas distribuídos, cujos componentes podem ser aplicados e executados em dispositivos distintos.

Kalin (2010)[14] esclarece que, a forma de implementação de um *Web Service* poderia ser feita com uma única classe *Java*, mas, seguindo as

---

[11] JENDROCK Eric; EVANS, Ian; GOLLAPUDI, Devika; HAASE, Kim; CHINMAYEE, Srivathsa. ***The Java EE Tutorial****:*Basic Concepts. 4 ed. *Boston: Addison-Wesley, 2010. p.5.*

[12] SRIGANESH, Rima P.; BROSE, Gerald; Silverman, Micah. **Mastering Enterprise JavaBeans 3.0**. 1 ed. Indianópolis: Wiley Publishing, 2006. p. 1.

[13] POTTS, Stephen. KOPACK, Mike. **Aprenda Web Services em 24 horas**. Tradução de Marcos Vieira. Ed. Campus, 2003.

[14] KALIN, Martin. **Java Web Services: Implementando.** Rio de Janeiro, RJ. Ed. Altabooks,2010.

melhores práticas, deve haver uma interface que declare os métodos chamada de SEI (*Service Endpoint Interface*), e uma implementação, que defina os métodos declarados na interface, chamada de SIB (*Service Implementation Bean*).

Na figura 3 temos uma ilustração de algumas tecnologias que podem ser usadas para projetar, construir e implantar Web Services.

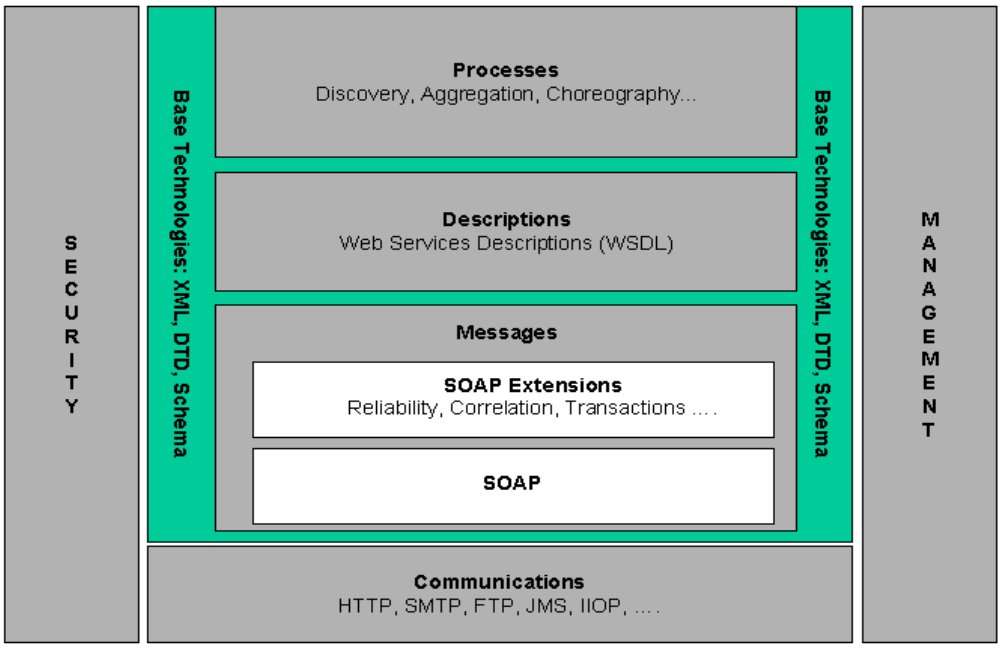

Figura 3 - Web Service arquitetura de pilha
Fonte: http://www.w3.org/TR/ws-arch/

Neste projeto, a aplicação cliente, que acessa o *Web Service*, foi implementada para a plataforma *Android* utilizando o *plug-in* para a *IDE Eclipse* chamado de *Android SDK* e a biblioteca *KSOAP2*. E a biblioteca *KSOAP2* tem a característica de ser mais flexível quanto à geração do código de suporte, necessário ao cliente para acessar o *Web Services*.

2.1.4.1 Soap

DEITEL; DEITEL (2012)[15] define SOAP Web Service como um protocolo de plataforma-independente que usa XML para interagir com Web Services.

O protocolo de acesso simples a objetos (SOAP) é um protocolo de comunicação baseado em XML, o SOAP é um tipo de Web Service e sua arquitetura é orientada a serviço SOA, que permite o transporte de informações sobre HTTP. Ele é utilizado para fazer a comunicação entre sistemas que não estejam escritos na mesma linguagem de programação.

Pelo fato de SOAP utilizar XML e HTTP, a comunicação entre sistemas é facilitada, pois qualquer linguagem que permite a comunicação com HTTP pode utilizar SOAP para comunicação com outros sistemas.

Uma mensagem SOAP é um documento XML que contém os elementos que definem a mensagem, como o envelope responsável por identificar um mensagem SOAP, o header que contém as informações de cabeçalho, o *body* contém as informações principais de uma mensagem, e *fault* que quando acontece um erro no Web Service as informações e detalhes são enviados por ele.

A figura 4 representa o modelo relativo ao processo geral para estabelecimento de um Web Service baseado em SOAP.

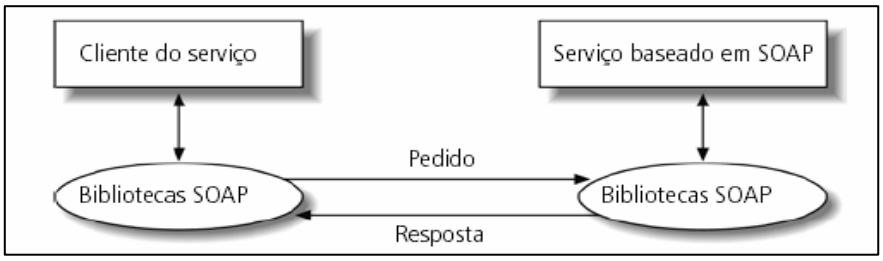

Figura 4 - Arquitetura de um típico Web Service baseado em SOAP
Fonte: Kalin (2010)

[15] A. DEITEL, Paul J.; DEITEL, H. M. **Java: how to program.** EUA: Prentice Hall. 2012. 9 ed. P.1302.

2.1.4.2 Rest

DEITEL; DEITEL (2012)[16] define REST como uma arquitetura de rede que usa o mecanismo tradicional de requisição/resposta da web como *GET* e *POST*.

REST e SOAP são diferentes, REST é um modelo de arquitetura de software para sistemas de hipermídias distribuídos, a WEB é um exemplo deste sistema, já o SOAP é um protocolo de mensagem.

O modelo REST utiliza um conjunto de interfaces genéricas para promover Interações sem estado através da transferência de representações de recursos, em vez de operar diretamente sobre esses recursos.

2.1.4.3 Jax-ws

O JAX-WS é uma API Java para Web Services, utilizada para a criação deste que utiliza XML/SOAP como forma de comunicação e também faz parte da JEE. Com a JAX-WS é possível transformar uma classe com diversas operações em um *WebService* e expô-lo na rede para que outras aplicações possam utilizar através de RPC.

Vohra (2011)[17] explica que um serviço da *Web* em *JAX-WS* consiste, basicamente, em uma classe *Java* com a declaração da anotação *javax.jws.WebService*.

2.1.4.4 Ksoap

O Ksoap é uma biblioteca desenvolvida em Java que permite que aplicativos móveis se comuniquem com WebService. O Ksoap foi inicialmente

[16] A. DEITEL, Paul J.; DEITEL, H. M. **Java: how to program.** EUA: Prentice Hall. 2012. 9 ed. P.1302.

[17] VOHRA, Deepak. **Acessando um Serviço da Web em JAX-WS a partir do Android**. Publicado em 01/07/2011. Disponível em <http://www.ibm.com/developerworks/br/library/ws-android/index.html?ca=drs>. Acessado em 17/05/2013.

desenvolvido para ser utilizado com JME, porém recentemente foi adaptado para ser usado em aplicativos Android (ksoap2-android)[18].

2.1.4.5 EJB

Enterprise Java Beans (EJB) e uma arquitetura de componentes para desenvolvimento de aplicações JAVA Server-side, muiti-tier, distribuídas, seguras, portáteis, escaláveis e orientadas a objetos. Esse permite o desenvolvimento mais rápido e simplificado de aplicações distribuídas, pois ele possui controle de transação e persistência deixando o desenvolvedor mais focado em pensar nas regras de negocio[19].

Existem dois tipos de escopos de acesso ao EJB: Remoto e Local. No tipo Local, o acesso é local na aplicação e exige que o cliente esteja na mesma JVM que o EJB. No tipo Remoto, a interface do EJB pode ser exposta e utilizada por um cliente que esteja em outra JVM e até em outra rede.

2.1.4.6 Json

O *JSON* foi projetado como um formato de troca de “dados”, e o *XML* foi projetado primeiramente como um formato de intercâmbio de documentos. A maioria das linguagens de programação não mapei**a**m diretamente as estruturas de dados contidas nas construções em *XML* – já que a distinção entre elementos e atributos não está presente nas estruturas de dados de grande parte das linguagens de programação (DEITEL; DEITEL, 2012, A, p. 1320)[20].

---

[18] KSOAP. **Biblioteca para comunicação com Web Services em Android.** Disponível em: <http://code.google.com/p/ksoap2-android/>. Acesso em 10/05/2013.

[19]ORACLE. **Enterprise JavaBeans Technology.** Disponível em <http://www.oracle.com/technetwork/java/ejb-141389.html>. Acessado em 15/05/2013.

[20] A. DEITEL, Paul J.; DEITEL, H. M. **Java: how to program.** EUA: Prentice Hall. 2012. 9 ed. P.1320.

O JSON é utilizado para troca de informações em requisições AJAX e recentemente tem sido muito utilizado em arquiteturas do tipo REST como formato de envio e recebimento de mensagens. A principal vantagem do JSON em relação ao XML é o formato reduzido que ele utiliza.

2.1.5 Servidores de Aplicação

Borrielo (2006)[21] explica que servidores de aplicação que utilizam à plataforma J2EE estão se tornando cada vez mais frequentes e recomendados para empresas, destacando as características de alta disponibilidade, alta capacidade para acesso simultâneo nos *containers* WEB e EJB, facilidades de configuração e utilização de banco de dados, facilidade de alteração, publicação e disponibilização de sistemas em tempo real, segurança, balanceamento de carga e tolerância a falhas, como alguns dos recursos que estes servidores oferecem.

Gomes (2000)[22] explica que J2EE fornece ao programador recursos para desenvolver softwares exatamente para ambientes corporativos. Formado por um conjunto de recursos padronizados, que fornecem os principais serviços necessários para desenvolver um sistema multicamada de qualidade.

Caelum (2013)[23] também explica que existem diversas dessas implementações de *Java EE*. A partir do *Java EE 6*, existe o termo "*application server web profile*", para poder se referenciar a servidores que não oferecem tudo, mas um grupo menor de especificações, consideradas essenciais para o desenvolvimento *Web*.

---

[21] BORRIELO, Daniela. GOMES, Rafael V. A.. JUNIOR, Antonio P. Castro. SANTOS, Roberto F. T. **Análise comparativa entre servidores de aplicação livres que seguem a plataforma J2EE**.Disponível <http://www.sbpcnet.org.br/livro/58ra/SENIOR/RESUMOS/resumo_394.html>. Acessado em 10/05/2013.

[22] GOMES, Handerson Ferreira. **A plataforma J2EE**. Webinsider, 2000. Disponível em <http://webinsider.uol.com.br/2000/12/05/a-plataforma-j2ee/>. Acessado em 10/05/2013.

[23] CAELUM. **FJ-21: Java para Desenvolvimento Web**. Apostilas gratuitas. Disponível em <http://www.caelum.com.br/download/caelum-java-web-fj21.pdf>. Acessado em 11/05/2013.

2.1.6 Segurança

Segundo Moreira (2001) a segurança da informação é um dos mais vastos, complexos e recorrentes temas dentro da tecnologia da informação por envolver diversos componentes e preocupações diferentes durante sua implantação e produção em um ambiente computacional[24].

A segurança em WebService é muito complexa e ampla. Para uma melhor utilização de segurança pode-se utilizar o Web Services Security que é uma coleção de protocolos que especificam como diferentes níveis de segurança podem ser reforçados em WebService baseados em SOAP ou REST. Por exemplo, o WSS especifica como assinaturas digitais e informações de codificação podem ser inseridas em cabeçalhos.

Para Hartman (2003) os *Web Services* expõem suas funcionalidades através de uma arquitetura orientada a serviços que é bem mais aberta em virtude de sua característica distribuída e de sua natureza heterogênea quanto a plataformas de execução. Por prover essas facilidades de acesso e integração, a arquitetura de Web Services é frágil em relação à segurança, que deve ser então considerada o foco principal durante o projeto de um sistema baseado em Web Services. O desafio é manter a eficiência dessas funcionalidades e ainda proporcionar um ambiente seguro[25].

O objetivo da segurança, quando envolve informações protegidas, é ter um mecanismo preciso para manter um alto nível de segurança e ainda fazê-lo o máximo desobstruído e transparente possível.

2.1.7 Sistema Operacional ANDROID

O sistema operacional Android foi utilizado como plataforma para desenvolvimento do aplicativo cliente para o *WebService*.

---

[24] MOREIRA, Nilton Stringasci. **Segurança Mínima: uma visão corporativa da segurança de informações.** 1. ed. Rio de Janeiro: Axcel, 2001.

[25] HARTMAN, Bret et al. **Mastering Web Services Security.** 1. ed. Indianapolis: Wiley, 2003.

Segundo GOOGLE (2013)[26] o sistema operacional Android oferece ferramentas para a criação de aplicativos interessantes a fim de tirar proveito dos recursos de hardware disponíveis em cada dispositivo. Ele se adapta automaticamente a interface do usuário para olhar o seu melhor em cada dispositivo, dando-lhe tanto controle quanto você quiser sobre a interface do usuário em diferentes tipos de dispositivos. O sistema operacional é baseado em Linux e as aplicações são desenvolvidas na linguagem Java e executadas em uma máquina virtual chamada Dalvik.

O sistema operacional *Android, para LECHETA(2010):*

> "Inclui um sistema operacional baseado no *Linux* e diversas aplicações, com uma rica interface gráfica, um *browser* para navegar na Internet, integração com o *Google Maps*, suporte a multimídia, GPS, banco de dados integrado, jogos em 3D e muito mais"[27].

Vascolcelos (2011 b)[28] explica que o *Android* tem uma estrutura especial para aproveitar o uso de recursos como imagens, vídeos, etc. E também possibilita a configuração de recursos alternativos para suportar configurações especificas como o tamanho da tela, orientação, etc.

---

[26] GOOGLE Android. **Estrutura de desenvolvimento poderoso**. Disponível em <http://developer.android.com/about/index.html />. Acessado em 20/05/2013.

[27] LECHETA, Ricardo R. . **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** São Paulo, SP . Novatec, 2010.

[28] VASCOLCELOS b, Marcos Antonio. **Android – Estrutura e organização da aplicação**. Mark Vasconcelos Disponível em <http://markytechs.wordpress.com/2011/01/24/android-estrutura-e-organizacao-daaplicacao/>. Acessado em 29/05/2013.

2.1.7.1 Estrutura do projeto Android

Segundo LECHETA (2010)[29] a estrutura do projeto Android esta descrita na Tabela 1:

PASTA	DESCRIÇÃO
src	Pasta do projeto que contém as classes Java. Contém a classe que foi criada pelo wizard.
gen	Contém a classe R.java que é gerada automaticamente permite que a aplicação acesse qualquer recurso como arquivos e imagens utilizando as constantes desta classe. Esta classe nunca deve ser alterada manualmente.
assets	Contém arquivos opcionais ao projeto, como por exemplo, uma fonte customizada.
res	Pasta que contém os recursos da aplicação, como imagens, layouts de telas e arquivos de internacionalização. Tem três subpastas: drawable, layout e values.
drawable	Pasta com as imagens da aplicação. Atualmente como existem diversos celulares Android com resolução de tela diferentes, é possível customizar as imagens para ficar com o tamanho exato em cada resolução automaticamente. Para isso existem 3 pastas para as imagens: drawable-lpdi, drawable-mdpi e drawable-hpdi.
layout	Contém os arquivos XML de layouts para construir as telas da aplicação.
values	Contém os arquivos XML utilizados para a internacionalização da aplicação e outras configurações. O XML é composto de uma layout simples com chave=valor.

Tabela 1- Estrutura do projeto do Android
Fonte: Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK

[29] LECHETA, Ricardo R. . **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** São Paulo, SP . Novatec, 2010. P. 51-52.

2.1.7.2 Activity

Segundo LECHETA(2010)[30] uma *Activity* deve herdar da classe android.app.Activity, tornando-se subclasse desta. Geralmente representa uma tela da aplicação e é responsável por tratar os eventos gerados nessa tela, como, por exemplo, quanto o usuário pressiona um botão ou quando um item de menu é escolhido.

SILVA (2013)[31] esclarece que todas as classes de uma aplicação de *Android* devem ser derivadas da classe *Activity* (Atividade) e possui, como método principal, o método *onCreate*. Dentro desse método ele invoca o método *onCreate* da super classe passando mesmo parâmetro (o *savedInstanceState*), logo após esse método, vem o método *setContentView*, responsável por exibir a tela da minha aplicação, baseado nos *layouts xml.* Por padrão ele chama o arquivo main.xml. A *Activity* é uma tela da aplicação, onde é possível adicionar componentes (*Views*) e programar eventos.

Cada *Activity* deve ser obrigatoriamente declarada no arquivo AndroidManifest.xml, da mesma forma que declaramos um *servlet* n arquivo web.xml de uma aplicação web. Isso é feito por meio da *tag* <*acrivity*>, assim demonstrado <Activity android:name=”MinhaClasseActivity”/>.

2.1.7.3 Arquivo AndroidManifest.xml

Segundo LECHETA (2010)[32] o arquivo AndroidManifest.xml é o arquivo principal do projeto e centraliza as configurações da aplicação. Este arquivo é a base de uma aplicação Android. Ele é obrigatório e deve ficar na pasta raiz do projeto, contendo todas as configurações necessárias para executar a aplicação,

---

[30] LECHETA, Ricardo R. . **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** São Paulo, SP . Novatec, 2010. P. 93.

[31] SILVA, Luciano Alves da. **Programando passo a passo**. Apostila de Android. Disponível em <http://www.portalandroid.org/comunidade/viewtopic.php?f=7&t=2528>. Acessado 20/04/2013.

[32] LECHETA, Ricardo R. . **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** São Paulo, SP . Novatec, 2010. p. 74.

como o nome do pacote utilizado, o nome das classes de cada Activity e várias outras configurações.

Podemos comparar o arquivo AndroidManifest.xml com o arquivo web.xml utilizado nas aplicações web Java. Em uma aplicação web cada classe de servlet deve ser declarada no arquivo web.xml e, da mesma forma, no Android cada Activity deve ser declarada no arquivo AndroidManifest.xml.

2.1.8 Google Directions

O Google Directions é uma API para cálculo de rotas e distâncias. Ele calcula as rotas disponíveis entre dois pontos. O *Google Directions* fornece a API em forma de serviços REST. A entrada do serviço é passada como parâmetros GET na URL e o retorno podem ser em formato XML ou JSON.

2.1.9 Banco de Dados

Um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e que possuem um significado implícito. As informações são uma coleção de dados com um significado. (ELMARSI 2005)[33].

A definição de um banco de dados implica em especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados em um banco de dados.

Segundo ELMARSI (2005) um sistema gerenciador de banco de dados é uma coleção de programas que permite aos usuários criar e manter um banco de dados, facilitar os processos de definição, construção, manipulação e compartilhamento de banco de dados entre vários usuários e aplicações.

Date (2000)[34] afirma que um SGBD é basicamente um sistema computadorizado de armazenamento de registros cujo propósito geral é o

[33] ELMASRI, Ramez. **Sistemas de banco de dados**. São Paulo: Pearson, 2005. p. 4.

[34] DATE, C. J. **Introdução a sistemas de bancos de dados.** 7. ed. Rio de Janeiro: Campus, 2000. p. 4.

armazenamento de informações e proporcionar a busca e atualização destas informações.

Um banco de dados é uma coleção integrada de dados onde o mesmo envolve os próprios dados, o hardware em que os dados residem, o software que controla o armazenamento e os usuários (DEITEL; DEITEL, 2003)[35].

2.1.9.1 PostgreSQL

Segundo PostgreSQL BR (2003), PostgreSQL é:

> "[...] um SGBD (Sistema Gerenciador de Banco de Dados) objeto-relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos" [36].

O PostgreSQL também tem como característica seguir as normas estabelecidas no SQL:2003, além de oferecer importantes recursos, tais como: queries complexas, chaves estrangeiras, triggers, views e integridade transacionais. Além de possuir tais recursos, ele é extensível, ou seja, é possível a criação de funções, operações, métodos de índices e 16 linguagens procedurais (POSTGRESQL, 2005)[37].

Entretanto, como o SGBD PostgreSQL é um produto aberto, não há uma equipe própria que possa fornecer o devido suporte técnico. O seu suporte é totalmente dependente de seus usuários, que formam grupos de discussões e fóruns. Por estes motivos, soluções para problemas que venham ocorrer podem não ser facilmente encontradas.

---

[35] DEITAL, H. M.; DEITEL, P.J. **Java como programar.** 3. ed. Porto Alegre: Bookman, 2001. p. 812.

[36]POSTGRESQL. **Introdução.** [S.l.], [2003]. Disponível em: <http://wiki.postgresql.org/wiki/Introdução_e_Histórico>. Acessado em 20/04/2013.

[37]POSTGRESQL. **Triggers.** [S.l.], [2005]. Disponível em: <http://www.postgresql.org/docs/8.0/interactive/triggers.html>. Acessado em 16/05/2013.

2.2 TRABALHOS CORRELATOS

Os aplicativos destinados para a localização de dispositivo móvel, a seguir, foram encontrados através de busca em site[38] específico que cita a empresa detentora da tecnologia empregada, sendo eles a base de desenvolvimento do SIRASS.

2.2.1 Buscar Meu IPhone (Apple)

Desenvolvido pela própria Apple, o programa permite que o usuário localize seu *smartphone* acessando sua localização através de um computador, entre suas vantagens estão que o programa é gratuito, permite exibir uma mensagem na tela do *smartphone*, emitir som, bloqueá-lo remotamente e até pagar os dados do dispositivo, porém temos alguns problemas como: primeiro, para este serviço estar disponível precisamos ter uma conta no ICloud (serviço de armazenamento de dados da Apple) ou no MobileMe; segundo, o programa precisa ser ativado nas configurações do aparelho.

2.2.2 SeekDroid (Google – Android)

Para os smartphone que possuem o sistema Android temos o SeekDroid, que possui as vantagens como habilitar o GPS do aparelho remotamente para assim localizá-lo, também permite que o seu usuário veja as ligações realizadas e também possibilita que o conteúdo do aparelho seja apagado, porém este aplicativo tem um custo de R$ 8,70 para ser adquirido, este aplicativo e de fácil desativação, os dados apagado por este aplicativo não podem ser recuperados.

[38] Disponível em <http://g1.globo.com/tecnologia/noticia/2011/10/aplicativos-ajudam-usuario-achar-celular-e-tablet-apos-roubo-ou-perda.html>. Acessado em 04/06/2013.

2.2.3 Lost (Google – Android)

O aplicativo Lost é para usuários do sistema Android, ele é gratuito, faz a localização do aparelho, apaga seus dados remotamente e possibilita a leitura de mensagens de texto que foram enviadas e recebidas pelo smartphone. Este aplicativo e de fácil desativação, tem pouca procura, os dados apagado por este aplicativo não podem ser recuperados.

2.2.4 Wheres my droid

Além de oferecer opções para encontrar o *smartphone*, o Wheres My Droid também permite emitir sons e enviar mensagens de texto para o celular perdido. Ele possui versão gratuita, porém muito limitada.

2.2.5 Protect (Blackberry - Research in Motion)

Para os donos de aparelhos Blackberry há um aplicativo chamado Protect, fornecido pela Research in Motion (RIM). O programa é gratuito, possui a vantagem de bloquear o celular remotamente, apagar as informações contidas no aparelho e ainda faz um backup das informações do smartphone.

2.2.6 Comparação

O ganho pela proposta está em comparação com alguns aplicativos específicos para Android. A Tabela 2 é o resumo de alguns itens que o SIRASS tem evidenciando em comparação com outros aplicativos. Usa-se a legenda da letra 'N' para identificar a inexistência de determinada ferramenta e a letra 'S' para existência.

APLICATIVOS / VANTAGENS	SeekDroid	Lost	Wheres my droid	SIRRAS
Gratuito	N	S	N	S
Difícil desativação	S	N	N	S
Fácil operação	N	N	S	S
Envio de sons ou mensagens ao celular	S	S	S	S
Apagam dados sem recuperação	S	S	N	N
Consulta da localização por internet	S	S	N	S

Tabela 2 - Comparação Aplicativos
Fonte: Autoria própria

Como podemos verificar na Tabela acima, quanto somatória final da presença de determinada ferramenta, o SIRASS possui algumas vantagens em comparação a outros aplicativos. Estas vantagens demonstram a escolha mais apropriada para o usuário.

3 DESENVOLVIMENTO

Todo o projeto, tanto WebService quanto o aplicativo móvel em Android foram desenvolvidos utilizando o Java SDK na versão 7. O Java SDK é composto por dois pacotes de software, sendo um o JDK que contém ferramentas de software para o desenvolvimento de aplicações utilizando a plataforma Java e o JRE que contém ferramentas de software necessárias à execução de aplicações Java.

Para desenvolver o WebService e o aplicativo móvel para Android foi utilizada a ferramenta de desenvolvimento IDE Eclipse SDK 4.2 versão Juno.

Esta monografia fez uso da UML para elaboração da documentação resumida de análise e projeto dos sistemas WebService e aplicativo Android.

3.1 DESCRIÇÕES DO PROJETO

Este aplicativo verifica a atual localização do *smartphone* através da latitude e longitude e constantemente envia os dados, de tempo em tempo para o WebService, que salva as informações em um banco de dados para consultas, da atual localização do dispositivo móvel bem como seu trajeto, consultas feitas pelo site projetowebvrk-com-br.webnode.com.

A Figura 5 apresenta a arquitetura geral do SIRASS, e seus principais componentes.

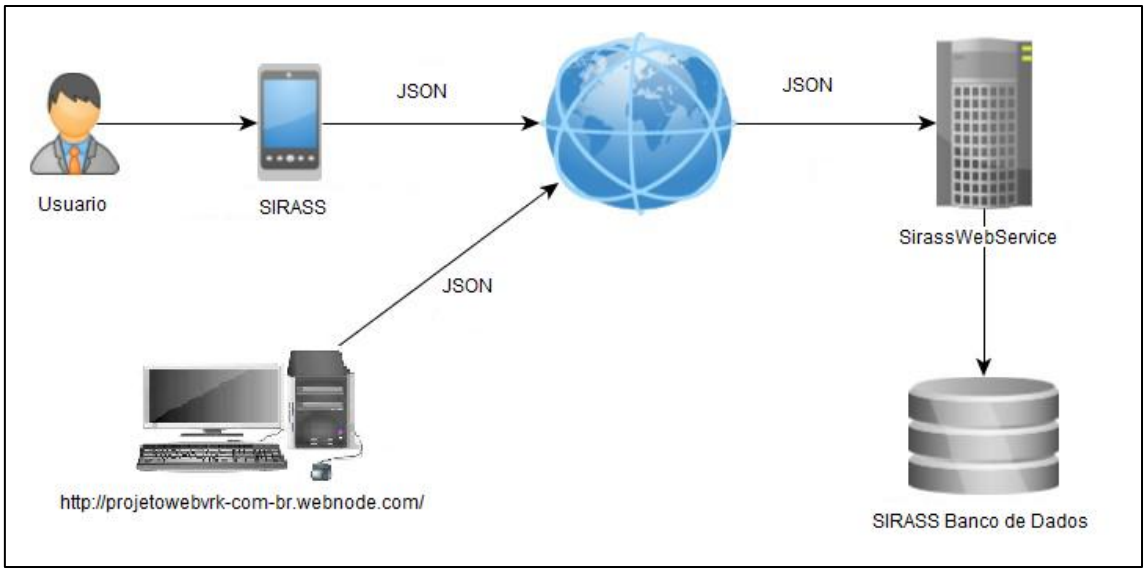

Figura 5 - Arquitetura geral do SIRASS
Fonte: Autoria própria

Na Tabela 3 esta descrito a funcionalidade de cada componente ilustrado na Figura 5, esta correspondente a arquitetura geral do SIRASS.

Componente	Descrição
SIRASS	Aplicativo móvel responsável pelo envio dos dados os usuário e a obtenção da localização do *smartphone*, através da latitude e longitude, para enviar constantemente esses dados ao WebService.
SirassWebService	Servidor Web que oferece todos os serviços utilizados pelo aplicativo e pelo site para busca. Serviços como verificar duplicidade de e-mail, enviar ao banco dados do cliente, localização do *smartphone* e buscar dados no banco.
SIRASS Banco de Dados	Base de dados que contém as ultimas informações da localização do *smartphone*, sua latitude e longitude, e dados do cliente com e-mail e senha.
Projetowebvrk-com-br.webnode.com	Permite consultar a localização de um *smartphone* Android em tempo real e em todo o mundo pela internet.

Tabela 3- Descrição da arquitetura SIRASS
Fonte: Autor próprio

3.1.1 Estrutura do aplicativo Android

O aplicativo tem a finalidade de enviar dados do usuário e do seu *smartphone*, este último sendo o foco. Iniciando a instalação do aplicativo, o cliente irá fornecer seu e-mail e senha para futuro acesso, através de um site na internet, a localização do seu dispositivo móvel em um mapa do Google. Esta localização vem das informações de latitude e longitude enviadas pelo *smartphone*.

Para desenvolver o aplicativo móvel SIRASS foi utilizado o Android SDK, juntamente com a ferramenta de desenvolvimento Eclipse SDK. O Google disponibiliza a instalação de uma ferramenta que integra o Eclipse ao Android SDK, um emulador de *smartphones*, a fim de executar e testar as aplicações desenvolvidas.

A figura 6 apresenta a estrutura do projeto Android.

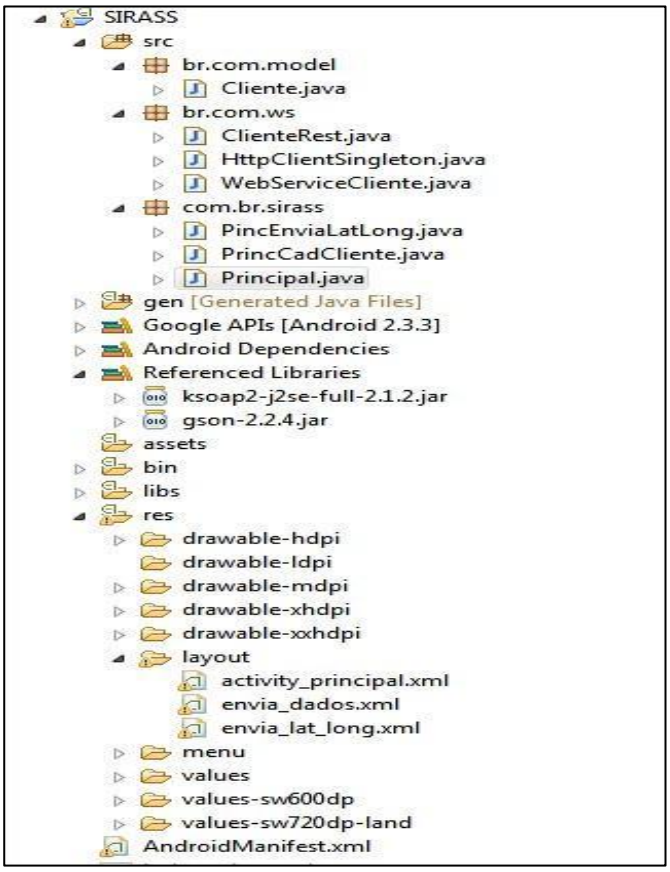

Figura 6 - Estrutura aplicativo Android
Fonte: Autoria própria

3.1.1.1 Biblioteca Ksoap 2 e Gson 2.2.4

Para que o aplicativo Android possa enviar dados para o WebService, desenvolvido com a tecnologia REST-Based JSON, foram utilizadas as bibliotecas “ksoap2-j2se-full-2.1.2.jar” e “gson-2.2.4.jar” que possuem as classes que geram as requisições e recebem as respostas necessárias para se comunicar com o WebService.

3.1.1.2 Classes do aplicativo Android

O aplicativo SIRASS contém as classes interligadas, responsáveis pelo funcionamento do aplicativo. A seguir são apresentadas as classes e suas funcionalidades.

A Figura 7 mostra a classe cliente, com os atributos necessários para o objetivo do aplicativo, foi usado o *implements Serializable* devido ao envio de objetos via socket (cliente/servidor).

```
package br.com.model;

import java.io.Serializable;


@SuppressWarnings("serial")
public class Cliente implements Serializable
{

    private String email;
    private String senha;
    private Double latitude;
    private Double logitude;


    //getters e setters
```

Figura 7 - Classe Android – Cliente
Fonte: Autoria própria

Na Figura 8 visualizamos a classe HttpClientSingleton, que está sendo chamada em cada transação com o WebService, sendo assim muito requisitada,

para não perder o controle de quantas instâncias desta classe estariam em memória num dado momento, foi decidido o uso de um padrão de projeto *singleton* para garantir a existência de apenas uma instância.

```
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;

public class HttpClientSingleton
{

    private static final int JSON_CONNECTION_TIMEOUT = 3000;
    private static final int JSON_SOCKET_TIMEOUT = 5000;
    private static HttpClientSingleton instance;
    private HttpParams httpParameters ;
    private DefaultHttpClient httpclient;

    private void setTimeOut(HttpParams params)
    {
      HttpConnectionParams.setConnectionTimeout(params, JSON_CONNECTION_TIMEOUT);
      HttpConnectionParams.setSoTimeout(params, JSON_SOCKET_TIMEOUT);
    }

        private HttpClientSingleton()
        {
            httpParameters = new BasicHttpParams();
            setTimeOut(httpParameters);
            httpclient = new DefaultHttpClient(httpParameters);
        }


            public static DefaultHttpClient getHttpClientInstace()
            {
                if(instance==null)
                    instance = new HttpClientSingleton();

                return instance.httpclient;
            }
}
```

Figura 8 - Classe android – HttpClientSingleton
Fonte: Autoria própria

A Figura 9 mostra a classe WebServiceCliente, responsável pelos GET's, POST's, PUT's e DELETE's, sendo esta classe a interface com o WebService.

Nesta classe foi implementado apenas os métodos GET (usando o objeto Httpget), e POST (usando o Httppost), pois para a funcionalidade do aplicativo SIRASS será necessário apenas uma consulta banco de dados, para verificar se há duplicidade de e-mail, e após envio de dados para serem gravados constantemente no bando de dados.

Conforme as boas práticas recomendam, uma classe nunca deve ter muitas funções, ou fazer mais do que o necessário, nem ser muito extensa, por isso foram desenvolvidas várias classes simples.

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URI;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import android.util.Log;

public class WebServiceCliente
{
    public final String[] get(String url){
        String[] result = new String[2];
        HttpGet httpget = new HttpGet(url);
        HttpResponse response;

        try{
            response = HttpClientSingleton.getHttpClientInstace().execute(httpget);
            HttpEntity entity = response.getEntity();

            if (entity != null){
                result[0] = String.valueOf(response.getStatusLine().getStatusCode());
                InputStream instream = entity.getContent();
                result[1] = toString(instream);
                instream.close();
                Log.i("get", "Result from post JsonPost : " + result[0] + " : " + result[1]);
            }
        }
        catch (Exception e){
            Log.e("NGVL", "Falha ao acessar Web service", e);
            result[0] = "0";
            result[1] = "Falha de rede!";
        }
        return result;
    }

        public final String[] post(String url, String json){
            String[] result = new String[2];

            try{
                HttpPost httpPost = new HttpPost(new URI(url));
                httpPost.setHeader("Content-type", "application/json");
                StringEntity sEntity = new StringEntity(json, "UTF-8");
                httpPost.setEntity(sEntity);
                HttpResponse response;
                response = HttpClientSingleton.getHttpClientInstace().execute(httpPost);
                HttpEntity entity = response.getEntity();

                if (entity != null){
                    result[0] = String.valueOf(response.getStatusLine().getStatusCode());
                    InputStream instream = entity.getContent();
                    result[1] = toString(instream);
                    instream.close();
                    Log.d("post", "Result from post JsonPost : " + result[0] + " : " + result[1]);
                }

            }
            catch (Exception e)
            {
                Log.e("NGVL", "Falha ao acessar Web service", e);
                result[0] = "0";
                result[1] = "Falha de rede!";
            }
            return result;
        }


            private String toString(InputStream is) throws IOException
            {
                byte[] bytes = new byte[1024];
                ByteArrayOutputStream baos = new ByteArrayOutputStream();

                int lidos;
                    while ((lidos = is.read(bytes)) > 0){baos.write(bytes, 0, lidos);}

                    return new String(baos.toByteArray());
            }
}
```

Figura 9 - Classe android – WebServiceCliente
Fonte: Autoria própria

Na figura 10 temos a classe ClienteRest, esta vai fazer o serviço intermediário, a conexão entre o WebService e a Activity.

```
import br.com.model.Cliente;
import com.google.gson.Gson;

public class ClienteRest {
    private static final String URL_WS = "http://192.168.157.101:8080/SirassWebService_Busca/cliente/";

    public Cliente getCliente(int id) throws Exception
    {
        String[] resposta = new WebServiceCliente().get(URL_WS + id);

        if (resposta[0].equals("200")){
            Gson gson = new Gson();
            Cliente cliente = gson.fromJson(resposta[1], Cliente.class);
            return cliente;
        }
        else {throw new Exception(resposta[1]);}
    }


        public String inserirCliente(Cliente cliente) throws Exception
        {
            Gson gson = new Gson();
            String clienteJSON = gson.toJson(cliente);
            String[] resposta = new WebServiceCliente().post(URL_WS + "incluirCliente", clienteJSON);

            if (resposta[0].equals("200")) {return resposta[1];}
            else {throw new Exception(resposta[1]);}
        }

        public String inserirDadosCliente(Cliente cliente) throws Exception
        {
            Gson gson = new Gson();
            String clienteJSON = gson.toJson(cliente);
            String[] resposta = new WebServiceCliente().post(URL_WS + "incluirDadosCliente", clienteJSON);

            if (resposta[0].equals("200")) {return resposta[1];}
            else {throw new Exception(resposta[1]);}
        }
}
```

Figura 10 - Classe android ClienteRest
Fonte: Autoria própria

A Figura 11 mostra um método que faz a leitura dos valores recebidos pelo GPS, nesta classe utiliza-se o LocationManager.

Em aplicativos que trabalham com informações de posição geográfica, é necessário utilizar algum dispositivo como GPS ou a rede para obter as coordenadas. No Android, a classe LocationManager oferece serviços de localização geográfica obtidos através do GPS e rede. Para utilizá-la é necessário se registrar para receber as atualizações de localização. Geralmente isso é feito no método *onResume* e deve ser informado o tipo de serviço de localização a ser usado: GPS_PROVIDER(GPS) ou NETWORK_PROVIDER(Rede WiFi).

```
public void startGPS(){
    LocationManager lManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

    LocationListener lListener = new LocationListener(){
        @Override
        public void onLocationChanged(Location locat) {updateView(locat);}
        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {}
        @Override
        public void onProviderEnabled(String provider) {}
        @Override
        public void onProviderDisabled(String provider) {}
    };
    lManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, lListener);
}
```

Figura 11 - Método android faz leitura dos valores recebidos do GPS
Fonte: Autoria própria

3.1.1.3 Arquivo AndroidManifest.xml

A Figura 12 apresenta o código do arquivo AndroidManifest.xml, contendo em suas linhas o caminho da Activity que será iniciada pelo aplicativo.

O aplicativo SIRASS comunica-se com serviços via Internet, devido a isso é necessário declarar a permissão INTERNET, assim como é usado o GPS (simulação) é necessário a permissão ACCESS_FINE_LOCATION.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.br.sirass"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="10" />

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION"/>
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.br.sirass.Principal"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="PincEnviaLatLong" android:label="@string/app_name"></activity>
        <activity android:label="@string/app_name" android:name="PrincCadCliente"></activity>
    </application>

</manifest>
```

Figura 12 - Arquivo android Manifest.xml
Fonte: Autoria própria

3.1.2 Estrutura do WebService

O WebService tem a finalidade de primeiramente verificar o se há duplicidade quanto ao e-mail informado pelo usuário, após salvar os dados do cliente, e-mail e senha, e enfim receber a latitude e longitude enviados por um *smartphone* de um determinado usuários, identificado pelo seu endereço eletrônico, os dados serão salvos no banco PostgreSQL.

Para o desenvolvimento do WebService foi utilizado a arquitetura RESTful baseado em SOAP, JAX-RS e Jersey para a comunicação com o aplicativo móvel, utilizado as ferramenta Eclipse SDK Java EE juntamente com o Tomcat 7 como servidor local.

A figura 13 apresenta a estrutura do projeto WebService.

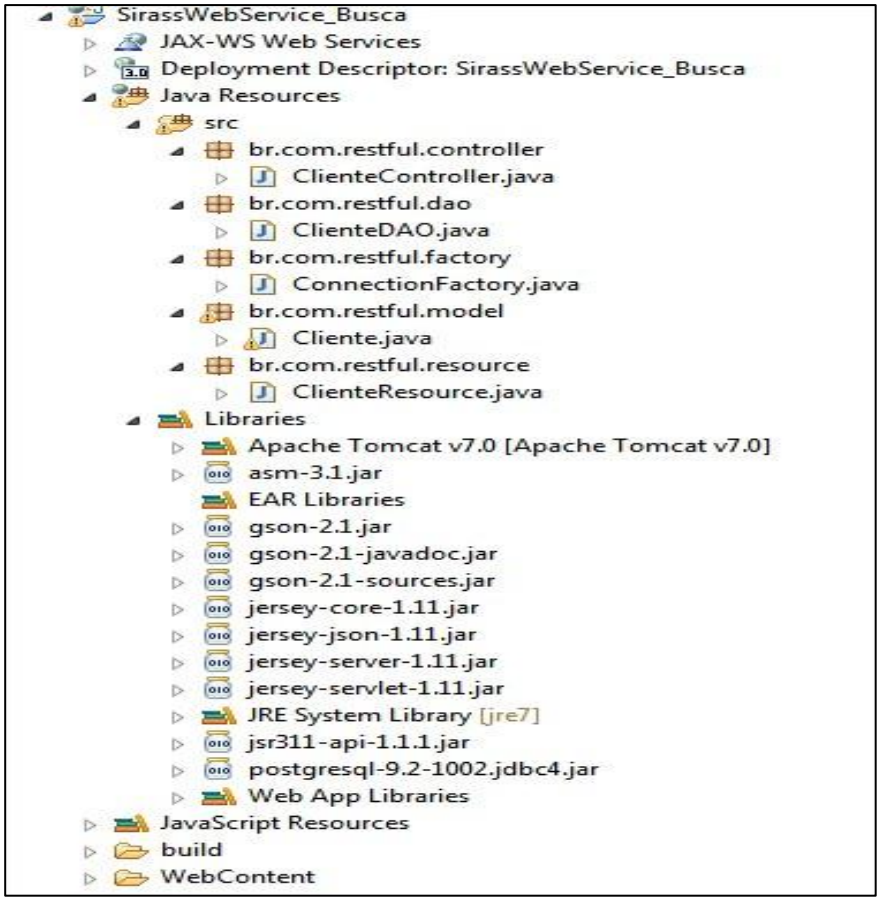

Figura 13 - Estrutura do WebService
Fonte: Autor próprio

3.1.2.1 Classes do projeto WebService

Para o desenvolvimento do WS foram criados 5(cinco) classes sendo elas Cliente.java, ClienteDAO.java, ClienteController.java, ConnectionFactory.java e ClienteResource.java.

Na Figura 14 temos a classe Cliente, que contém os atributos do usuário de um *smartphone*. Foi utilizada a anotação @XmlRootElement na definição da classe, ela é fundamental para que o Jersey conheça a representação do objeto cliente em JSON. Desse modo os atributos irão se tornar nós no momento em que o JSON for enviado para o aplicativo móvel Android, e do mesmo jeito quando chegar um JSON no corpo de uma requisição, o sistema vai saber converter esse JSON em um objeto do tipo Cliente.

```
import java.io.Serializable;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public final class Cliente implements Serializable {

    private Integer id;
    private String email;
    private String senha;
    private Double latitude;
    private Double longitude;

    public Integer getId() {return id;}

    public void setId(Integer id) {this.id = id;}

    public String getEmail() {return email;}

    public void setEmail(String email) {this.email = email;}

    public String getSenha() {return senha;}

    public void setSenha(String senha) {this.senha = senha;}

    public Double getLatitude() {return latitude;}

    public void setLatitude(Double latitude) {this.latitude = latitude;}

    public Double getLongitude() {return longitude;}

    public void setLongitude(Double longitude) {this.longitude = longitude;}

    public String toString(){

    public int hashCode() {

    public boolean equals(Object obj) {

}
```

Figura 14 - Classe cliente WS
Fonte: Autoria própria

A Figura 15 demonstra a classe ClienteDAO ele faz as operações básicas de Cadastramento, Remoção, Listagem e Atualização.

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import br.com.restful.factory.ConnectionFactory;
import br.com.restful.model.Cliente;

 //Classe responsável por conter os metodos do CRUD
public class ClienteDAO extends ConnectionFactory {

    private static ClienteDAO instance;

    //Método responsável por criar uma instancia da classe ClienteDAO (Singleton)
     public static ClienteDAO getInstance()
     {
        if(instance == null) {instance = new ClienteDAO();}
        return instance;
     }

     //Método responsável por verificar se ja existe email no banco
     public int verificarCadastro(String email)
     {
         int id = 0;
         Connection conexao = null;
         PreparedStatement pstmt = null;
         ResultSet rs = null;
         conexao = criarConexao();

         try
         {
             pstmt = conexao.prepareStatement("select id from cliente where email = " + "'" + email + "'");
             rs = pstmt.executeQuery();

             System.out.println(email);

             if(rs.next()){
                 id = rs.getInt("id");
             }
               System.out.println(id);

          }
          catch (Exception e){
                 System.out.println("Erro ao verificar existencia do cliente: " + e);
                 e.printStackTrace();
          }

          finally {fecharConexao(conexao, pstmt, rs);}

          return id;

      }
     //Método responsável por inserir dados do cliente no banco postgresql
      public int inserirCliente (String email, String senha){

     //Método responsável por inserir a latitude e longitude do cliente no banco postgresql
      public int inserirDadosClient (String email, Double latitude, Double longitude)


      //Método responsável por listar todos os clientes do banco
      public ArrayList<Cliente> listarTodos(){

}
```

Figura 15 - Classe ClienteDAO WS
Fonte: Autoria própria

Na Figura 16 é apresentada a classe ClienteController, ela vai tratar os erros e os procedimentos no objetos. Aqui foi criado um método para inserir dados um a um usando o método do ClienteDAO.

```
import java.util.ArrayList;
import br.com.restful.dao.ClienteDAO;
import br.com.restful.model.Cliente;

//Classe responsável por ser o controlador entre o resource e a camada DAO
public class ClienteController {

   public ArrayList<Cliente> listarTodos(){
       System.out.println("Enviando para o GIT");
       return ClienteDAO.getInstance().listarTodos();
   }

   public String verificarCadastro(String x){

        String text;

        if(ClienteDAO.getInstance().verificarCadastro(x) == 0){
            text = "Não temos este email em nosso banco de dados";
        }
        else {text = "Este email ja existe em nosso cadastro";}
       return text;
   }

   public String inserirCliente(String a, String b){

       ClienteDAO clienteDAO = new ClienteDAO();
       if ( clienteDAO.inserirCliente(a, b) != 0){return "Cliente inserido com seucesso";}
       else{return "Falha ao inserir cliente no banco de dados";}
   }

   public String inserirDados(String em, Double latitude, Double longitude){
       ClienteDAO clienteDAO = new ClienteDAO();
       if ( clienteDAO.inserirDadosClient(em, latitude, longitude) != 0){
           return "Dados do "+ em +" latitude:" +latitude+ " e Longitude: "+longitude +" inseridos com seucesso";
       }

           else{return "Falha ao inserir dados do "+em+" no banco de dados";}
   }
```

Figura 16 - Classe ClienteController WS
Fonte: Autor próprio

A figura 17 é apresentada a classe ConnectionFactory responsável pela conexão com o banco de dados PostgreSQL.

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

//Classe responsável por conter os metodos criar e fechar o banco de dados.
public class ConnectionFactory {

    // Caminho do banco de dados.
    private static final String DRIVER = "org.postgresql.Driver";
    private static final String URL = "jdbc:postgresql://localhost:5432/WebServiceTCC";
    private static final String USUARIO = "postgres";
    private static final String SENHA = "1234";

    // Método responsável por criar uma conexao com o banco
    public Connection criarConexao(){

        Connection conexao = null;
        try {
            Class.forName(DRIVER);
            conexao = DriverManager.getConnection(URL, USUARIO, SENHA);

        } catch (Exception e) {
            System.out.println("Erro ao criar conexão com o banco: " + URL);
            e.printStackTrace();
        }
        return conexao;
    }

        public void fecharConexao(Connection conexao, PreparedStatement pstmt, ResultSet rs){

        try {
            if(conexao != null){conexao.close();}
            if(pstmt != null){pstmt.close();}
            if(rs != null){rs.close();}

        } catch (Exception e) {System.out.println("Erro ao fechar conexão com o banco: " + URL);}
    }
}
```

Figura 17 - Classe ConnectionFactory WS
Fonte: Autoria própria

Na Figura 18 é apresentada a classe ClienteResource, que passa a chamar os métodos da camada de serviço.

```java
import java.util.ArrayList;

 //Classe responsável por conter os metodos REST de acesso ao webservice
 @Path("/cliente")
 public class ClienteResource {

     //Métodos responsáveis por fazer chamada ao controller
     //Verificando o email se esta cadastrado
     @GET
     @Path("/verificarCadastro")
     @Produces("application/json")
     public String verificarCadastro(String x){
         String a = x;
         return new ClienteController().verificarCadastro(a);}

     //Incluir cliente no banco de dados
     @POST
     @Path("/incluirCliente")
     @Produces("application/json")
     public String incluirCliente(Cliente cliente){
         String email, senha;
         email = cliente.getEmail();
         senha = cliente.getSenha();
         return new ClienteController().inserirCliente(email, senha);}

     @POST
     @Path("/incluirDadosCliente")
     @Produces("application/json")
     public String incluirDadosCliente(Cliente cliente){
         System.out.println(cliente.getLatitude());
         System.out.println(cliente.getLongitude());
         return new ClienteController().inserirDados(cliente.getEmail(), cliente.getLatitude(), cliente.getLongitude());}

     //buscar clientes no bando de dados
     @GET
     @Path("/listarTodos")
     @Produces("application/json")
     public ArrayList<Cliente> listarTodos(){
        return new ClienteController().listarTodos();}
 }
```

Figura 18 - Classe ClienteResource WS
Fonte: Autoria própria

3.1.2.2 Arquivo web.xml do WebService

A Figura 19 apresenta o código do arquivo web.xml do WebService, contendo em suas linhas o cadastro da servlet. Nesta classe é passado o parâmetro de inicialização ao qual corresponde ao pacote onde estão os resources.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <display-name>SirassWebService_Busca</display-name>
  <servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>com.sun.jersey.config.property.packages</param-name>
      <param-value>br.com.restful.resource</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Figura 19 - Arquivo web.xml WS
Fonte: Autoria própria

3.2 REQUISITOS

Na Tabela 4, são apresentados os requisitos funcionais do sistema SIRASS.

Requisito	Descrição
1. Realizar instalação	Deve-se poder instalar o aplicativo no *smartphone*, e para seu funcionamento o aparelho deve possuir acesso à internet ativa.
2. Ativar GPS e Wi-Fi	O sistema irá ativar o GPS ou Wi-Fi do dispositivo móvel.
3. Informar Login e senha	O sistema deve solicitar o e-mail do usuário e a senha, e depois de informados os dados e pressionado o botão OK, será feita a verificação de duplicidade de e-mail, se for necessário será solicitado o preenchimento de outro e-mail. O e-mail e senha serão solicitados para autenticação, quando for realizar a consulta da localização do *smartphone* no site projetowebvrk.
4. Conectar com servidor	O sistema estabelece conexão com o WebService.
5. Salvar informação	O sistema guardará no bando de dados as informações iniciais, e-mail e senha, do usuário. Em seguida o sistema envia a latitude e longitude, para serem salvos no banco PostgreSQL, para futura consulta. Será feito antes do envio da localização do smartphone a verificação dos serviços de GPS e WI-FI.
6. Consultar localização e trajeto	O sistema possibilitará a consulta, através do site projetowebvrk, da localização e trajeto no mapa de seu *smartphone*.

Tabela 4 - Requisitos do sistema
Fonte: Autoria própria

3.3 ARQUITETURA

3.3.1 Diagrama de casos de uso

Um caso de uso representa uma interação que um Ator realiza com o sistema. O objetivo do diagrama de Casos de Uso é mostrar as possibilidades de interação do usuário com o sistema. A figura 20 demonstra o diagrama de casos de uso referente ao sistema.

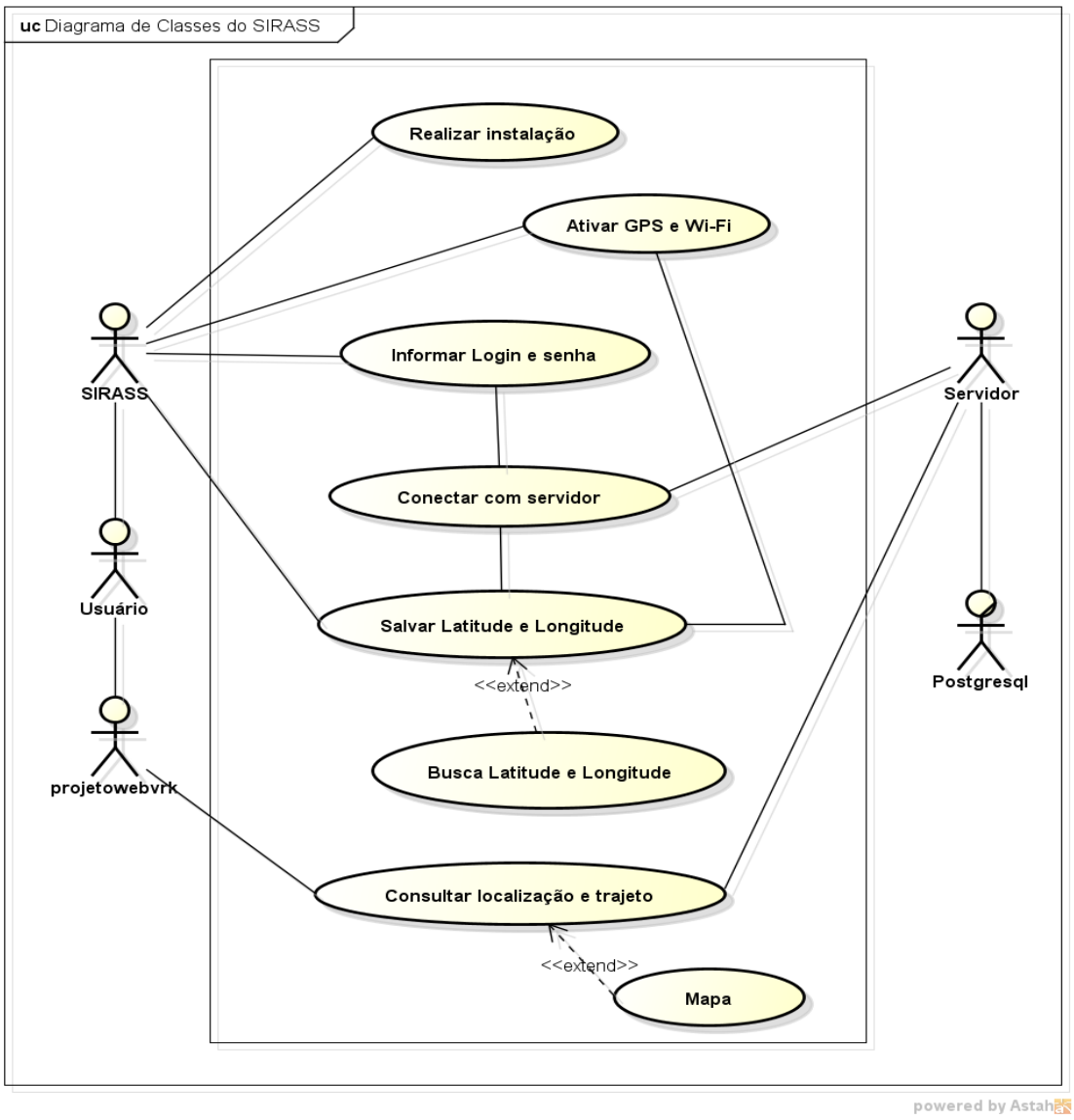

Figura 20 - Diagrama de casos de uso do sistema SIRASS
Fonte: Autoria própria

3.3.1.1 Descrição dos casos de uso

Descrição	**Instalação do aplicativo**
Fluxo	1. Usuário seleciona o aplicativo para a instalação. 2. Aplicativo inicia a instalação. 3. Ao final do processo abre uma nova tela, informando que para continuar é necessário ter acesso à internet ou conexão Wi-Fi. 4. Selecionar o botão OK para continuidade da instalação.
Pré-condições	Possuir um *smartphone* Android.
Pós-condições	

Descrição	**Ativar GPS e Wi-Fi**
Fluxo	1. O Aplicativo verifica se o GPS e o Wi-Fi estão habilitados se não estiverem os inicia. 2. É verificado a cada ciclo de tempo se os dispositivos GPS e Wi-Fi estão habilitados, se não estiverem os inicia.
Pré-condições	*Smartphone* deve possuir alguma conexão de rede.
Pós-condições	O aplicativo estará instalado.

Descrição	**Informar Login e senha**
Fluxo	1. Aplicativo abre uma nova tela solicitando e-mail e senha. 2. Usuário preenche os campos com as informações, o endereço de seu e-mail e a senha, após aperta o botão “OK”. 3. O sistema conecta-se com o servidor que faz uma varredura no bando de dados para verificar se não há duplicidade de e-mail. 4. Não havendo duplicidade o sistema informa que os dados foram aceitos com sucesso e o aplicativo encontra-se em funcionamento. 5. Caso tenha o e-mail já cadastrado o sistema solicitará um novo e-mail ao usuário.
Pré-condições	*Smartphone* deve possuir alguma conexão de rede.

Descrição	**Conectar com o servidor**
Fluxo	1. O aplicativo SIRASS abre uma conexão com WebService. 2. Envia dado para ser consultado pelo WS no banco PostgreSQL. 3. Envio de informações do usuário e localização do aparelho móvel, este último ininterruptamente em ciclos de tempo, para serem salvo no banco de dados.
Pré-condições	*Smartphone* deve possuir alguma conexão de rede.

Descrição	**Salvar Latitude e Longitude**
Fluxo	1. Ativar GPS e Wi-Fi. 2. Identificar o e-mail do usuário, já informado, buscar através do GPS a latitude e longitude do smartphone. 3. O sistema conecta com o servidor, enviando os dados do cliente ao WS para salvar estas informações no banco de dados.
Pré-condições	Smartphone deve possuir alguma conexão de rede.

Descrição	**Consultar localização e trajeto**
Fluxo	1. O usuário acessa o site projetoewbvrk-com-br.webnode.com. 2. Usuário seleciona o serviço “SIRASS”. 3. Na tela seguinte o cidadão preenche o e-mail e senha para autenticação e aperte o botão “Enviar”. 4. O usuário escolhe as opções “Localizar Smartphone” e “Trajeto do Smartphone”. 5. Será demonstrado a localização do dispositivo móvel ou o trajeto percorrido.
Pré-condições	1. Possuir um microcomputador com conexão a internet. 2. Ter acesso à internet.
Pós-condições	1. O aplicativo estar instalado. 2. Ter o e-mail e senha para acesso.

3.3.2 Diagrama de Sequência

Os diagramas de sequência demonstram o fluxo sistêmico dos processos ou funcionalidades da aplicação, o diagrama tem por objetivo demonstrar as interações que ocorrem entre os componentes de um sistema. Serão demonstrados nos diagramas a seguir os seguintes processos do SIRASS:

- Instalação e o serviço de envio das informações;
- Consultar a localização e trajetória do *smartphone*;

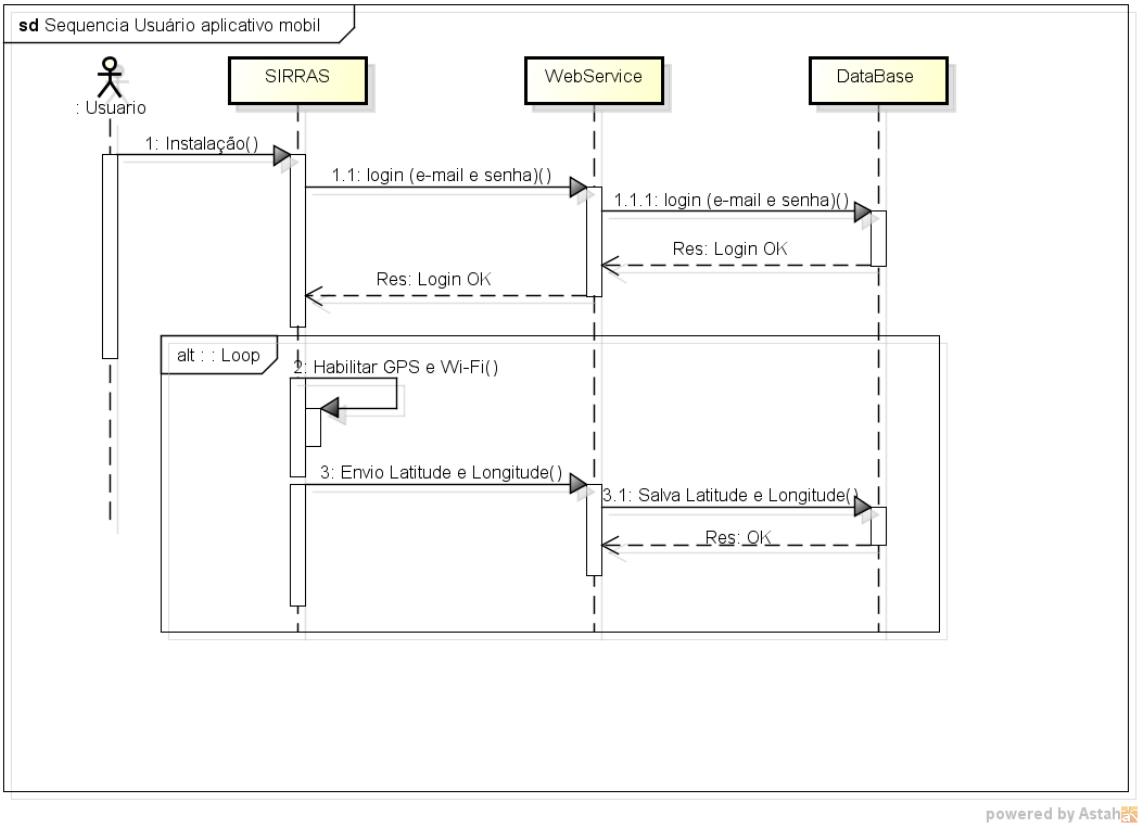

Figura 21 - Diagrama de sequência instalação do aplicativo
Fonte: Autoria própria

A figura 21 demonstra o processo de instalação do aplicativo, envio e registro do *login* e senha, sendo o *login* o e-mail do usuário. Continua o processo habilitando o GPS e o Wi-Fi, enviando os dados de localização latitude e longitude e após a gravação dos dados. O processo tem seu inicio com a instalação do aplicativo pelo usuário, ao qual tendo finalizado a instalação

solicita ao usuário um e-mail e senha para futura autenticação afim da consulta em um site da localização do *smartphone*, após as informações preenchidas e salvas o aplicativo inicia o processo, começando a habilitar o GPS e o Wi-Fi, se os mesmos estiverem desabilitados, após há captura da latitude e da longitude do dispositivo móvel e o envio dos dados para o servidor a fim de gravá-los no banco de dados, esse serviço de habilitar, capturar e enviar entra em loop, isto é constantemente feito de tempo em tempo.

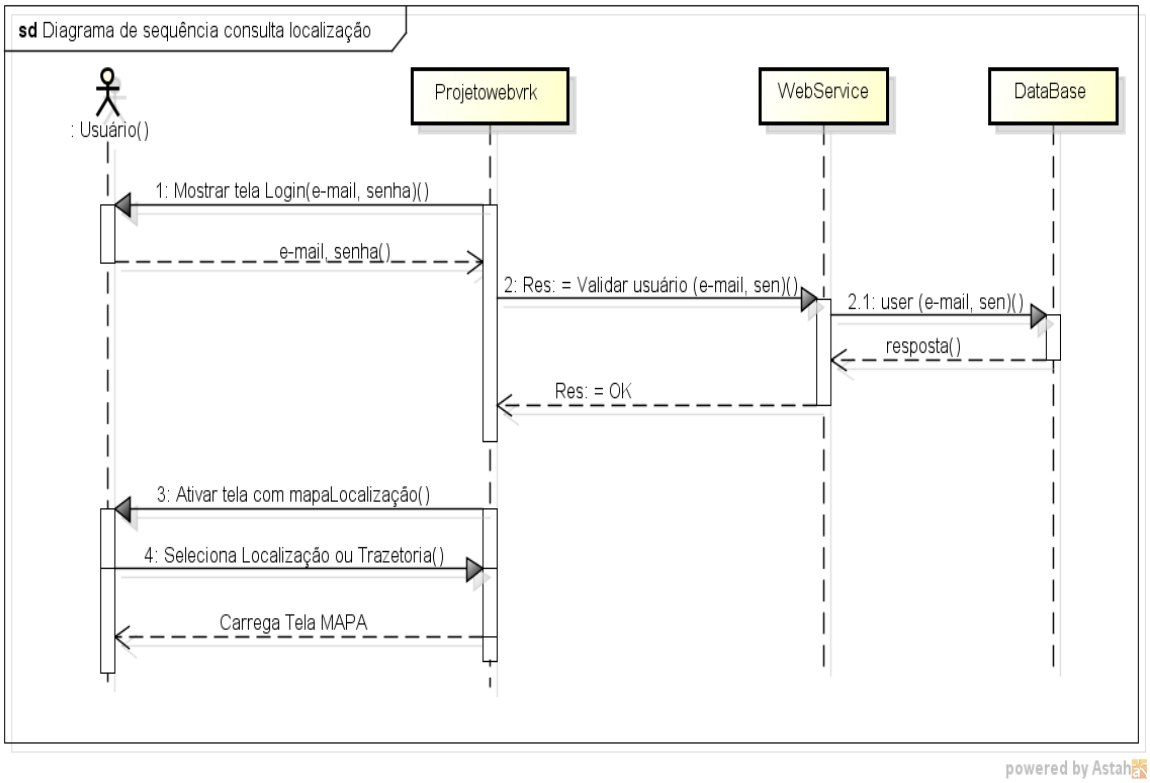

Figura 22 - Diagrama de sequência consulta localização
Fonte: Autoria própria

A figura 22 traz o processo para consulta da localização e o trajeto do dispositivo móvel. Ao usuário acessar o site projetowebvrk é providenciado ao usuário uma tela solicitando o e-mail e senha para autenticação das informações e após são habilitadas as opções de localização do *smartphone* e a trajetória do dispositivo móvel.

3.3.3 Diagrama de Classes

O Diagrama de classe tem por objetivo demonstrar as classes e as relações entre elas de um determinado aplicativo orientado a objeto.

3.3.3.1 Diagrama de Classes do Aplicativo Móvel para Android

Na Figura 23 é apresentado o diagrama de classes do aplicativo móvel.

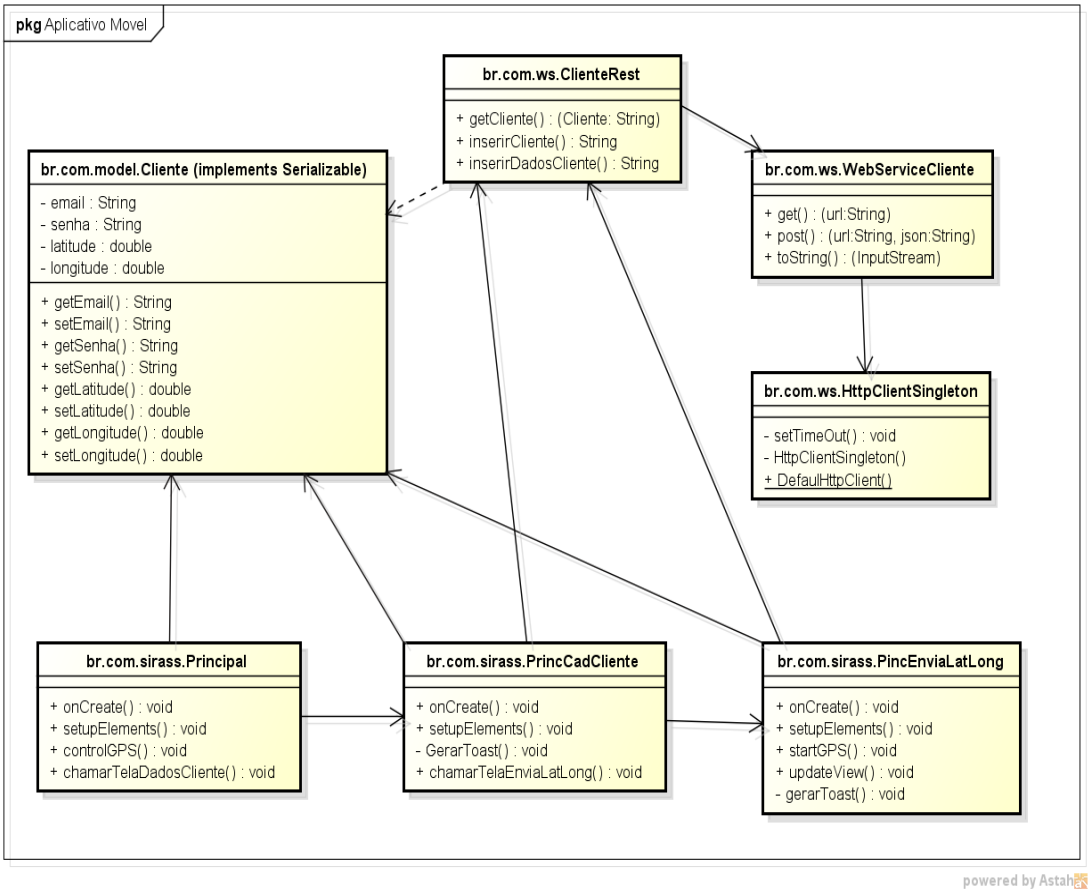

Figura 23 - Diagrama de Classe do Android
Fonte: Autoria própria

3.3.3.2 Diagrama de Classes do WebService

Na Figura 24 é demonstrado o diagrama de classes do WebService.

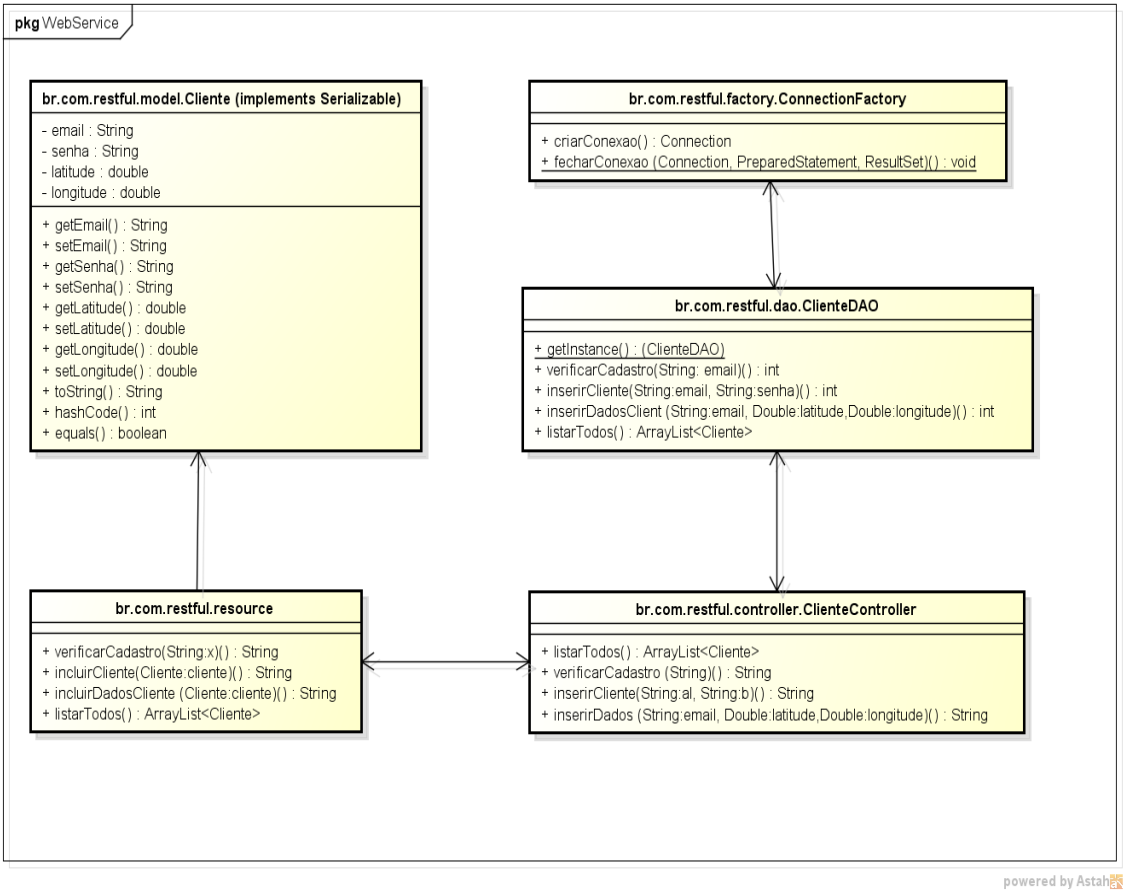

Figura 24 - Diagrama de Classe do WebService
Fonte: Autoria própria

3.3.4 Diagrama de Atividades

O diagrama de atividades mostra o fluxo entre atividades, seus paralelismos e desvios condicionais. O diagrama de atividades apresentado nas Figuras 25, diagrama de atividade do aplicativo móvel, e 26, diagrama de atividade do WebService, demonstram de forma ilustrativa as regras que envolvem a utilização do aplicativo móvel SIRASS.

3.3.4.1 Diagrama de atividades do aplicativo móvel

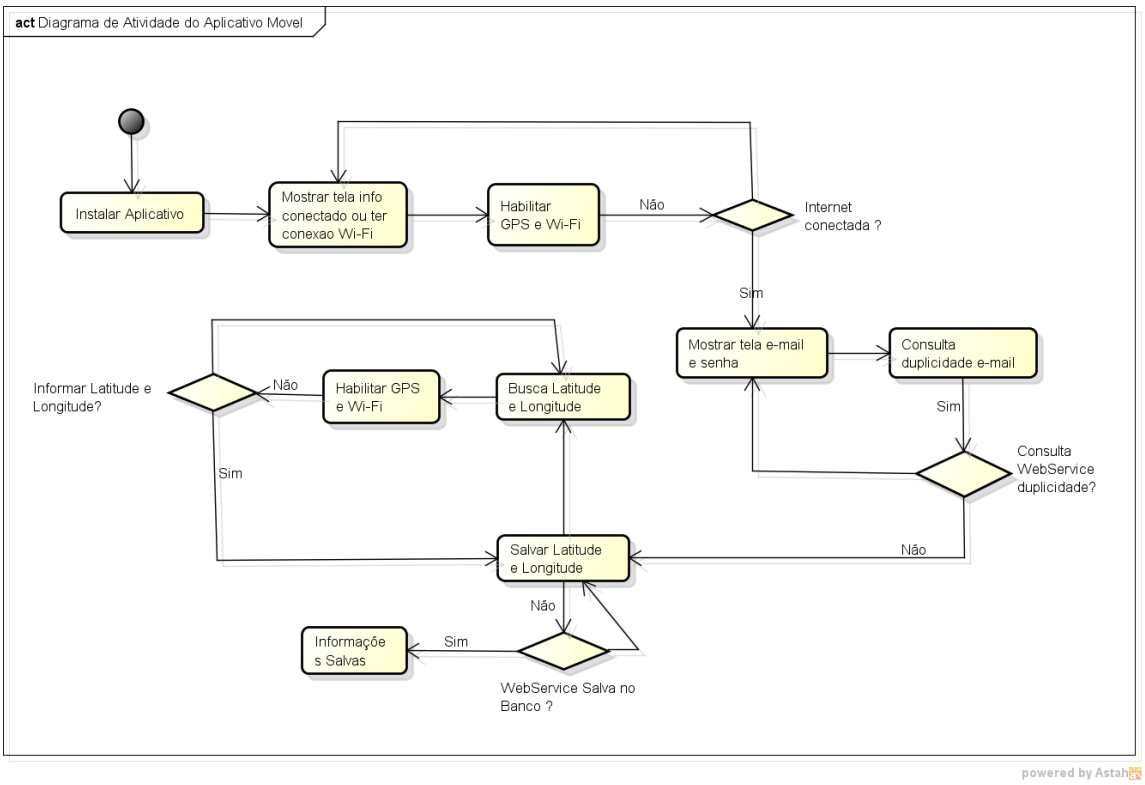

Figura 25 - Diagrama de Atividade do dispositivo móvel
Fonte: Autoria própria

3.3.4.2 Diagrama de atividades do WebService

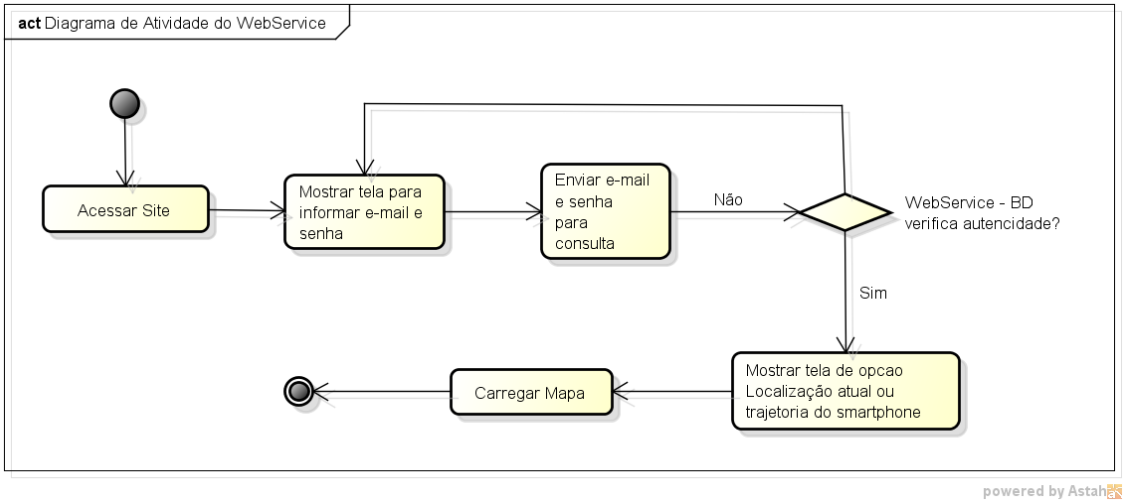

Figura 26 - Diagrama de Atividade do WebService
Fonte: Autor próprio

4 FUNCIONALIDADES E IMAGENS DAS APLICAÇÕES

Neste capitulo serão apresentadas as telas do sistema SIRASS, tanto do serviço de *Web Service* e do aplicativo para *Android*.

4.1 APLICATIVO MÓVEL DO SIRASS

Na Figura 27 temos a tela inicial do aplicativo móvel, após o início da instalação do SIRASS. Nesta tela o usuário verá o aplicativo buscando a conexão com a internet ou rede Wi-Fi para prosseguir no processo de instalação.

Após a conexão, será visualizado o botão OK e ao pressioná-lo iniciará o método responsável pela ativação do GPS no dispositivo móvel e o usuário terá outra tela disponível para informações de seus dados (figura 28).

Figura 27 - Primeira Tela do SIRASS

Fonte: Autoria própria

A Figura 28 demonstra a segunda tela do aplicativo móvel. Esta tela solicita ao usuário dois dados indispensáveis, o e-mail e a senha, que serão utilizados para consulta da localização ou trajeto do seu dispositivo móvel.

Ao preencher os dados nos respectivos campos e após ser acionado o botão Enviar, o aplicativo fará uma conexão com o WebService para verificar se existe duplicidade no e-mail informado, caso exista duplicidade de e-mail o sistema informará a necessidade de outro endereço eletrônico para continuar o processo, caso não exista duplicidade será informado ao usuário que seu aplicativo SIRASS foi instalado com sucesso e poderá visualizar a localização de seu dispositivo móvel pelo site *www.projetowebvrk-com-br.wennode.com*. Para o usuário essa é a última tela que será visualizada no *smartphone*, pois todos os processos que compreendem o funcionamento do SIRASS são realizados automaticamente sem a demonstração para o cliente, por isso as telas a seguir são para demonstrar o funcionamento do aplicativo como um todo.

Figura 28 - Segunda Tela do SIRASS
Fonte: Autoria própria

Na Figura 29 temos a terceira tela do aplicativo móvel, visualizada apenas para demonstrar o funcionamento do SIRASS.

Podemos visualizar que foram preenchidos os campos do e-mail e senha, após foi acionado o botão Enviar, os dados foram enviados ao WebService que fez a verificação de duplicidade no banco de dados, após o serviço verificou-se que não há outro endereço eletrônico igual ao informado fazendo assim a inserção dos dados no banco de dados PostgreSQL, concluindo este serviço envia a informação que o Cliente foi inserido com sucesso.

Com a inserção dos dados no banco temos o botão Ir_Para_Enviar_Lat_long que chama a tela seguinte, esta demonstra a latitude e longitude.

Figura 29 - Terceira Tela do SIRASS
Fonte: Autor próprio

A figura 30 demonstra a quarta tela do aplicativo SIRASS, com os campos Latitude e Longitude já preenchidos, porém para ter os valores informados na tela primeiramente deve ser acionado o botão Obter localização. Com o acionamento do primeiro botão é chamado o método que faz a localização inicial do usuário através da classe *LocationManager* do Android. A interface *LocationListener* provê o método *onLocationChanged* que é invocado de forma implícita pelo Android sempre que ocorre uma atualização na localização do dispositivo. Quando isso ocorre, as informações de latitude e longitude são atualizadas e a localização é atualizada na tela, então sempre que o primeiro botão for acionado, havendo um deslocamento do dispositivo móvel, os valores nos campos serão atualizados.

Para salvar os valores no banco de dados é necessário pressionar o botão Enviar, os dados só serão salvos se junto com os valores enviados for o e-mail do usuário para verificação pelo WebService.

O aplicativo faz automaticamente a obtenção da localização do dispositivo móvel, através da latitude e longitude, e o envio destes valores ao WebService para serem gravados constantemente em um intervalo de tempo, junto com este serviço será verificado se o GPS e o WI-FI estão ligados, se não estiverem acionados os dispositivos, o aplicativo os acionará.

Figura 30 - Quarta Tela do SIRASS
Fonte: Autoria própria

4.2 SERVIÇO WEB DO SIRASS

Na Figura 31 temos a primeira tela para o serviço do SIRASS web, nesta tela é solicitado o e-mail e senha do usuário que instalou o aplicativo móvel, após o preenchimento e o acionamento do botão Enviar será verificado junto ao WebService a validação.

Figura 31 - Tela dados cliente do serviço web SIRASS
Fonte: Autoria própria

A Figura 32 demonstra a segunda tela para o serviço do SIRASS web, podendo o cliente selecionar duas opções para localização do *smartphone* sendo a primeira a localização atual e a segunda a trajetória que o dispositivo móvel percorreu, exatamente nesta tela foi selecionado a opção localização atual, trazendo assim a imagem de um mapa na tela, com a marcação da localização do *smartphone* por um objeto da cor amarela.

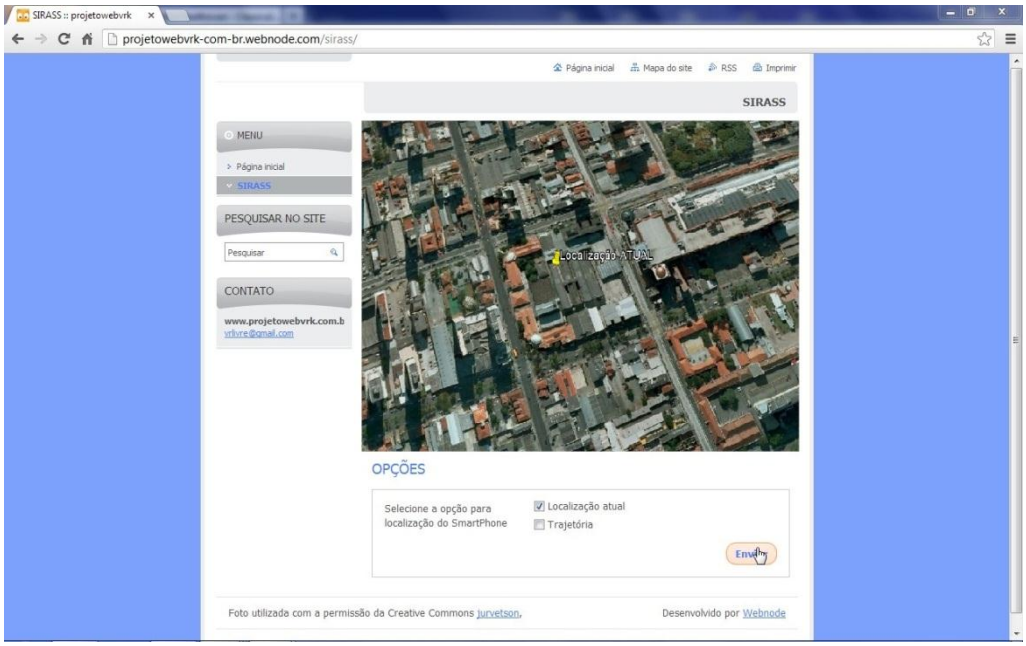

Figura 32 - Tela seleção da localização atual do serviço web SIRASS
Fonte: Autoria própria

Na Figura 33 temos a segunda tela para o serviço do SIRASS web, onde foi selecionada a opção trajetória, trazendo assim a imagem de um mapa na tela com as últimas localizações do *smartphone*, com a marcação das localizações feita por um objeto da cor amarela.

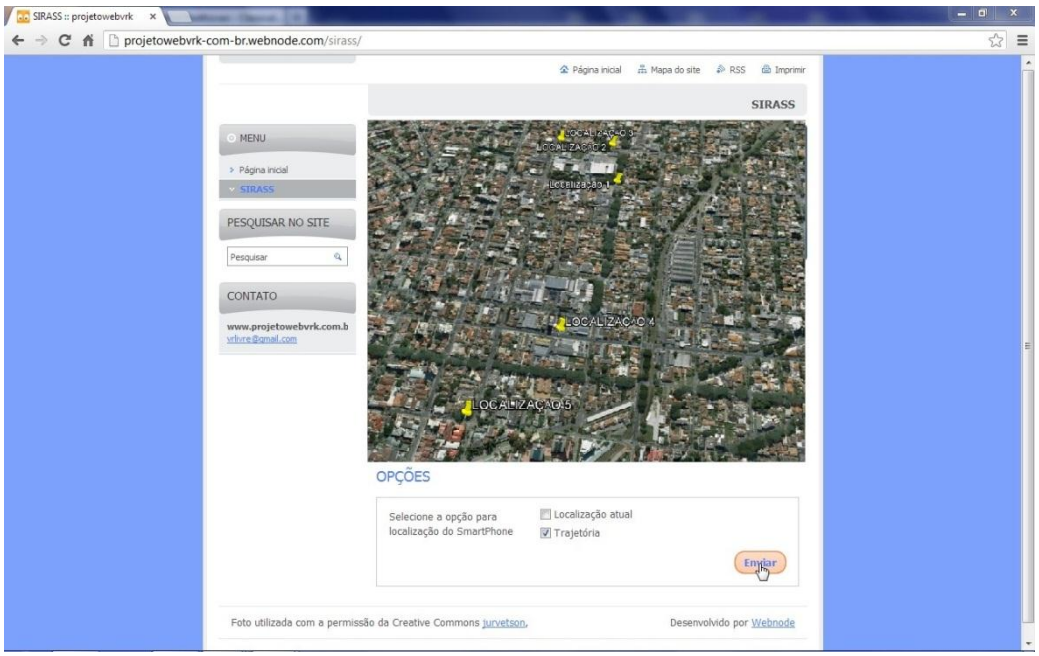

Figura 33 - Tela seleção da Trajetória do serviço web SIRASS
Fonte: Autoria própria

5 CONSIDERAÇÕES FINAIS

Com o desenvolvimento do aplicativo permitiu ao autor ter uma visão das ferramentas disponíveis no mercado, que puderam esclarecer novas ideias para as principais funcionalidades propostas. Notou-se a integração entre sistema de plataformas e linguagens diferentes, devido a uma aplicação construída em uma linguagem consumir o serviço de outra diferente, sem problemas, gerando assim a interoperabilidade.

As tecnologias utilizadas atenderam as necessidades do projeto e foram motivadas pelo fato de ser um projeto de código aberto. A plataforma Android possui uma excelente qualidade nas ferramentas disponíveis e seus frameworks, mas na documentação deixa a desejar. Para o WebService as ferramentas são acessíveis, mas a documentação para seu desenvolvimento é pouco relatada nos meios de comunicação.

Após o desenvolvimento do software, pode-se concluir que o sistema SIRASS consiste, em termos da ferramenta em questão, em uma melhoria na questão de praticidade e de fácil operalização para os usuários, permitindo localizar smartphones perdidos. O sistema proposto busca atender aos objetivos traçados, tendo como base o conhecimento teórico adquirido e o documentado.

A análise do sistema foi feita a rigor, porém houve dificuldades durante o desenvolvimento do sistema e gastou-se mais tempo que o previsto em pesquisa de referencial teórico para um melhor entendimento e funcionamento das tecnologias, causando atraso no cronograma e consequentemente teve impacto em todo o projeto.

As funcionalidades que não estão no escopo deste trabalho, estão listadas na próxima seção que aborda os trabalhos futuros.

5.2 TRABALHOS FUTUROS

Por fim, ao longo desse trabalho foram observadas melhorias que devem ser feitas no aplicativo, assim entendidos como interessantes fontes de trabalhos futuros, há oportunidades no sentido de adicionar recursos para torna-lo completo, tais como:

- Disponibilizar o envio de mensagens e sons para o smartphone perdido ou roubado, a fim de ajudar ainda mais para encontrar o aparelho móvel;
- Desenvolver a opção para salvar todo o conteúdo do celular, tanto da memória interna e externa, através de um comando externo;
- Opção de apagar todo o conteúdo do smartphone presente nas memórias externa e interna, e também os dados presentes no chip (agenda);
- Concluir a conexão do site com o WS e também com o google maps;
- Disponibilizar no site a opção para a autenticação de usuário, assim como a consulta da localização e trajetória do smartphone, disponíveis no banco de dados;

Além disso, evoluir o estudo de caso de forma a agregar aspectos de segurança, confiabilidade, tolerância a falhas e disponibilidade é outro ponto interessante, visto como trabalho futuro.

REFERÊNCIAS BIBLIOGRAFICAS

A. DEITEL, Paul J.; DEITEL, H. M. **Java: how to program.** EUA: Prentice Hall. 2012. 9 ed.

BORRIELO, Daniela. GOMES, Rafael V. A.. JUNIOR, Antonio P. Castro. SANTOS, Roberto F. T. **Análise comparativa entre servidores de aplicação livres que seguem a plataforma J2EE**. Anais da 58ª Reunião Anual da SBPC, Florianópolis, SC, 2006. Disponível em <http://www.sbpcnet.org.br/livro/58ra/SENIOR/RESUMOS/resumo_394.html>. Acessado em 10/05/2013.

CAELUM. **FJ-21: Java para Desenvolvimento Web**. Apostilas gratuitas. Disponível em http://www.caelum.com.br/download/caelum-java-web-fj21.pdf Acessado em 11/05/2013.

DATE, C. J. **Introdução a sistemas de bancos de dados.** 7. ed. Rio de Janeiro: Campus,2000.

DEITAL, H. M.; DEITEL, P.J. **Java como programar.** 3. ed. Porto Alegre: Bookman, 2001.

DEITEL, H.M. e DEITEL, P.J. **JAVA Como Programar**, 4ª edição. trad. Carlos Arthur Lang Lisbôa. Porto Alegre: Bookman, 2003.

ELMASRI, Ramez. **Sistemas de banco de dados**. São Paulo: Pearson, 2005.

EL-RABBANY, Ahmed. **Introduction to GPS**: **The Global Positioning System.** Boston: Artech House, 2000.

FISCHER, Alice E.; GRODZINSKY, Frances. ***The Anatomy of Programming Languages.*** Englewood Cliffs, New Jersey: Prentice Hall, 1993.

FOLHA DE S.PAULO. **Uso de dados móveis em smartphones quase dobra em um ano**. Disponível em:<http://www1.folha.uol.com.br/tec/1239448-uso-de-dados-moveis-em-smartphone-quase-dobra-em-um-ano.shtml>. Acessado em 20/04/ 2013.

FRENCH, Gregory T. **GPS An Introduction to the Global Positioning System**: **What It Is and How It Works**. Bethesda: GeoResearch, 1996.

GOMES, Handerson Ferreira. **A plataforma J2EE**. Webinsider, 2000. Disponível em <http://webinsider.uol.com.br/2000/12/05/a-plataforma-j2ee/>. Acessado em 10/05/2013.

GOOGLE Android. **Estrutura de desenvolvimento poderoso**. Disponível em <http://developer.android.com/about/index.html>. Acessado em 20/05/2013.

HARTMAN, Bret et al. **Mastering Web Services Security.** 1. ed. Indianapolis: Wiley, 2003. 436p.

KALIN, Martin. **Java Web Services: Implementando.** Rio de Janeiro, RJ. Ed. Altabooks,2010.

KSOAP, **Biblioteca para comunicação com Web Services em Android.** Disponível em <http://code.google.com/p/ksoap2-android/>. Acesso em 10/05/2013.

LECHETA, Ricardo R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** São Paulo, SP. Novatec, 2010.

JENDROCK Eric; EVANS, Ian; GOLLAPUDI, Devika; HAASE, Kim; CHINMAYEE, Srivathsa. ***The Java EE Tutorial****:* Basic Concepts. 4 ed. *Boston: Addison-Wesley, 2010.*

MELO, Ana Cristina Vieira de; SILVA, Flávio Soares Corrêa da. **Princípios de Linguagens de Programação.** São Paulo: Edgard Blücher Ltda, 2003.

MOREIRA, Nilton Stringasci. **Segurança Mínima: uma visão corporativa da segurança de informações.** 1. ed. Rio de Janeiro: Axcel, 2001.

ORACLE. **Enterprise JavaBeans Technology.** Disponível em <http://www.oracle.com/technetwork/java/ejb-141389.html>. Acessado em 15/05/2013.

POSTGRESQL. **Introdução.** [S.l.], [2003]. Disponível em: <http://wiki.postgresql.org/wiki/Introdução_e_Histórico>. Acessado em 20/04/2013.

POSTGRESQL. **Triggers.** [S.l.], [2005]. Disponível em: <http://www.postgresql.org/docs/8.0/interactive/triggers.html>. Acesso em: 22/05/2013.

POTTS, Stephen. KOPACK, Mike. **Aprenda Web Services em 24 horas**. Tradução de Marcos Vieira. Ed. Campus, 2003.

SCHILDT, Herbert. ***Java The Complete Reference****. 7ed. Nova York: McGraw-Hill, 2007.*

SILVA, Luciano Alves da. **Programando passo a passo**. Apostila de Android. Disponível <http://www.portalandroid.org/comunidade/viewtopic.php?f=7&t=2528 Acessado em 20/05/2013.

SRIGANESH, Rima P.; BROSE, Gerald; Silverman, Micah. ***Mastering Enterprise JavaBeans 3.0****. 1 ed. Indianópolis: Wiley Publishing, 2006.*

TECMUNDO. **25% dos brasileiros já tiveram seu celular roubado ou perdido**. Disponível em: <http://www.tecmundo.com.br/celular/31075-25-dos-brasileiros-ja-tiveram-seu-celular-roubado-ou-perdido.htm>. Acessado em 20/04/2013.

VASCOLCELOS b, Marcos Antonio. **Android – Estrutura e organização da aplicação**. Mark Vasconcelos Creative Solutions. Publicado em 24/01/2011.

VOHRA, Deepak. **Acessando um Serviço da Web em JAX-WS a partir do Android**. Publicado em 01/07/2011. Disponível em <http://www.ibm.com/developerworks/br/library/ws-android/index.html?ca=drs>. Acessado em 17/05/2013.

Web Services Architecture Usage Scenarios, W3C Working Group Note, H. He, H. Haas, D. Orchard, 11 February 2004 <http://www.w3.org/TR/2004/NOTE-ws-arch-scenarios-20040211/>. Acessado em 20/05/2013.