colloquium exactarum

DESENVOLVIMENTO DE UM FRAMEWORK PARA O MONITORAMENTO DE DISPOSITIVOS MÓVEIS NA PLATAFORMA ANDROID

A FRAMEWORK IMPLEMENTATION FOR MOBILE DEVICES MONITORING IN ANDROID PLATFORM

Bruna Molina de Oliveira e Silvio Antonio Carro

Faculdade de Informática – Universidade do Oeste Paulista - UNOESTE
Correspondência para: Silvio Antonio Carro - silvio@unoeste.br

RESUMO - MonitoremSempre é um Framework que foi desenvolvido utilizando a plataforma Android com o objetivo de oferecer recursos para amenizar os efeitos de eventuais roubos e violações ocorridos em dispositivos móveis. O Framework pode ser utilizado para criar aplicações de monitoramento por coordenadas geográficas. Oferece ainda uma interface web para realizar o gerenciamento dos dispositivos monitorados.
Palavras-chave: Framework; monitoramento; dispositivos móveis; aplicativos android; segurança de dados.

ABSTRACT - MonitoremSempre is a framework for Android Applications that offers a set of resources to monitoring mobile devices. The MonitoremSempre framework provides a solution monitoring and managing mobile devices. Also offers a web interface to perform a management of monitored devices.
Keywords: Framework; monitoring; mobile devices; android applications; data security.

Recebido em: 21/08/2013
Revisado em: 27/09/2013
Aprovado em: 29/10/2013

1 INTRODUÇÃO

A difusão dos smartphones, tablets e outros dispositivos móveis atingem atualmente boa parte da população e seus proprietários dependem cada vez mais das funções e informações contidas nesses dispositivos. Sendo assim, torna-se necessário manter os dados seguros ou inacessíveis evitando que pessoas sem autorização visualize informações pessoais.

O presente trabalho por intermédio de um framework propõe subsidiar os desenvolvedores no desenvolvimento de aplicativos que visam proteger dispositivos móveis de eventuais roubos e violações. O framework, denominado MonitoremSempre oferece um conjunto de classes que abstraem os processos de monitoramento, segurança e localização de dispositivos na plataforma Android (ANDROID, 2013) e ainda fornece um sistema Web com rotinas prontas para o monitoramento remoto.

Assim como a proposta deste trabalho também existem várias aplicações voltadas para o monitoramento do aparelho com a plataforma Android, com o intuito de preservar a segurança de seu dispositivo e seus dados, como o Avast Mobile Security (AVAST, 2013) e o Zoe Mob (ZOEMOB, 2013) porém algumas dessas aplicações equivalentes são pagas e limitadas na forma de aplicativos prontos, portanto não fornecem subsídios para a elaboração de soluções específicas.

As próximas seções deste trabalho estão organizadas da seguinte forma: a seção 2 apresenta quais foram os materiais e métodos utilizados. A seção 3 discute os resultados obtidos e a seção 4 apresenta a conclusão deste trabalho juntamente com os trabalhos futuros para seguimento de novas versões tanto do framework quanto do protótipo.

2 MATERIAL E MÉTODOS

Esta seção apresenta as tecnologias utilizadas e o desenvolvimento do sistema.

2.1 Tecnologias Utilizadas

No presente trabalho foram utilizadas diversas tecnologias em seu desenvolvimento, dentre elas a linguagem JAVA (ORACLE, 2013), o Software Development Kit – SDK Android que conta com as bibliotecas para o desenvolvimento da plataforma Android e emulador, a linguagem C# (MICROSOFT, 2013b) com o Framework .Net 3.5 (MICROSOFT, 2013d) juntamente com JavaScript (W3SCHOOLS, 2013b) e JQuery (JQUERY, 2013) e por fim SQL Server 2008 (MICROSOFT, 2013c).

2.2 Projeto MonitoremSempre

Para experimentar a proposta foi desenvolvido um framework denominado MonitoremSempre, escrito em Java/Android com recursos para o monitoramento de um dispositivo móvel.

O MonitoremSempre foi projetado para funcionar em três módulos, o módulo Web, o módulo de Web Service e o módulo móvel, como pode ser visto na figura 1.

Dentre as funcionalidades oferecidas no módulo móvel, podemos destacar: a obtenção de coordenadas de um aparelho através do GPS, o envio dessas coordenadas para um Web Service, envio de mensagens de texto, alerta sonoro, disparos de fotos e limpeza de cartão de memória.

O módulo Web oferece uma interface gráfica com diversas funções para conexão com o módulo Web Service. O Web Service recebe e envia mensagens ao módulo Web através de um protocolo convencionado neste trabalho.

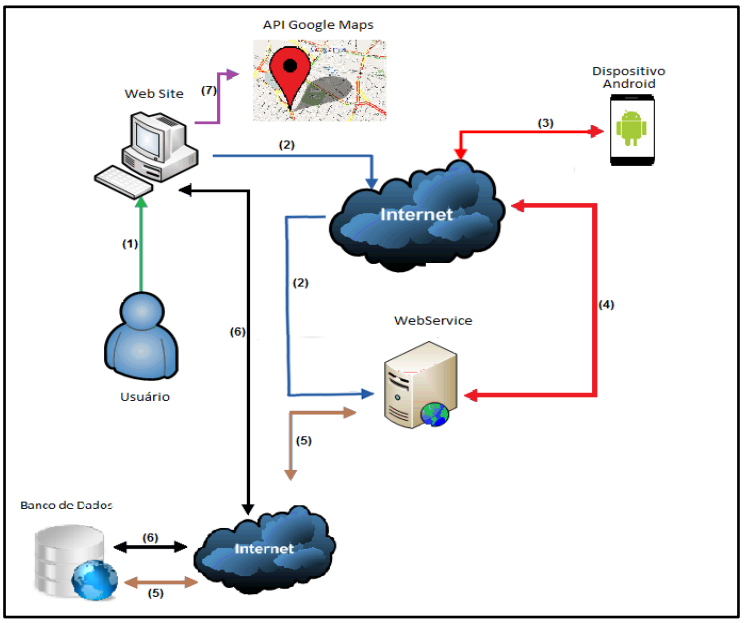

Figura 1. Arquitetura de utilização do Framework.

As próximas seções ilustram o desenvolvimento dos módulos do projeto MonitoremSempre e estão organizados da seguinte forma: seção 2.2.1 descreve o desenvolvimento do módulo Web com sua arquitetura e modo de implementação; a seção 2.2.2 diz respeito ao módulo Web Service; a seção 2.2.3 descreve o banco de dados utilizado e na seção 2.2.4 o desenvolvimento do módulo móvel, detalhando as principais classes.

2.2.1 Módulo Web

O módulo Web foi desenvolvido em linguagem C# com o Framework .Net 3.5 juntamente com JavaScript e JQuery para melhor estilização da pagina. Esse módulo conta com Web Services pertencentes ao módulo Web que realizam operações de Create, Read, Update e Delete – CRUD para registro de usuários e de dispositivos em uma base SQLServer. Para realização das operações CRUD, foi utilizado Language Intergrated Query – LINQ (MICROSOFT, 2013a) que é um componente pertencente ao Framework .Net que converte consultas escritas em C# ou Visual Basic em SQL dinâmico, provendo uma interface que permite mapear os objetos do banco de dados gerando as classes para realizar as operações usando a sintaxe LINQ, também permite realizar alterações nos objetos e atualizar o banco de dados.

Nas realizações das operações CRUD, funções JavaScript são criadas e acionadas ao clique de um botão. Essas funções contam com código Asynchronous JavaScript and XML – Ajax (W3SCHOOLS, 2013a) que realizam as chamadas dos serviços Webs para realização de inserts, updates, deletes e selects, retornando para pagina um JavaScript Object Notation – JSON (W3SCHOOLS, 2013c) que é consumido e apresentado o resultado da operação na pagina para visualização do usuário.

Quando se refere à utilização da pagina para interação com celular, um Web Service desenvolvido em Java entra em operação onde o mesmo será apresentado na próxima seção.

A integração do módulo Web com o Web Service Java é realizado através de uma referencia Web criada na aplicação Web em C#, a partir desse ponto é instanciado um objeto com o tipo da referencia criada e todos os métodos presentes no Web Service Java estarão acessíveis na aplicação Web C# e portanto toda a comunicação entre as plataformas serão realizadas de forma transparente ao desenvolvedor.

2.2.1.1 Arquitetura do Módulo Web

No diagrama de classe da figura 2 e na tabela 1, são mostradas e descritas todas as classes pertencentes ao módulo Web. A arquitetura é responsável pela realização das operações CRUD onde cada classe deve ser instanciada, seus respectivos métodos chamados e os parâmetros corretos devem ser passados.

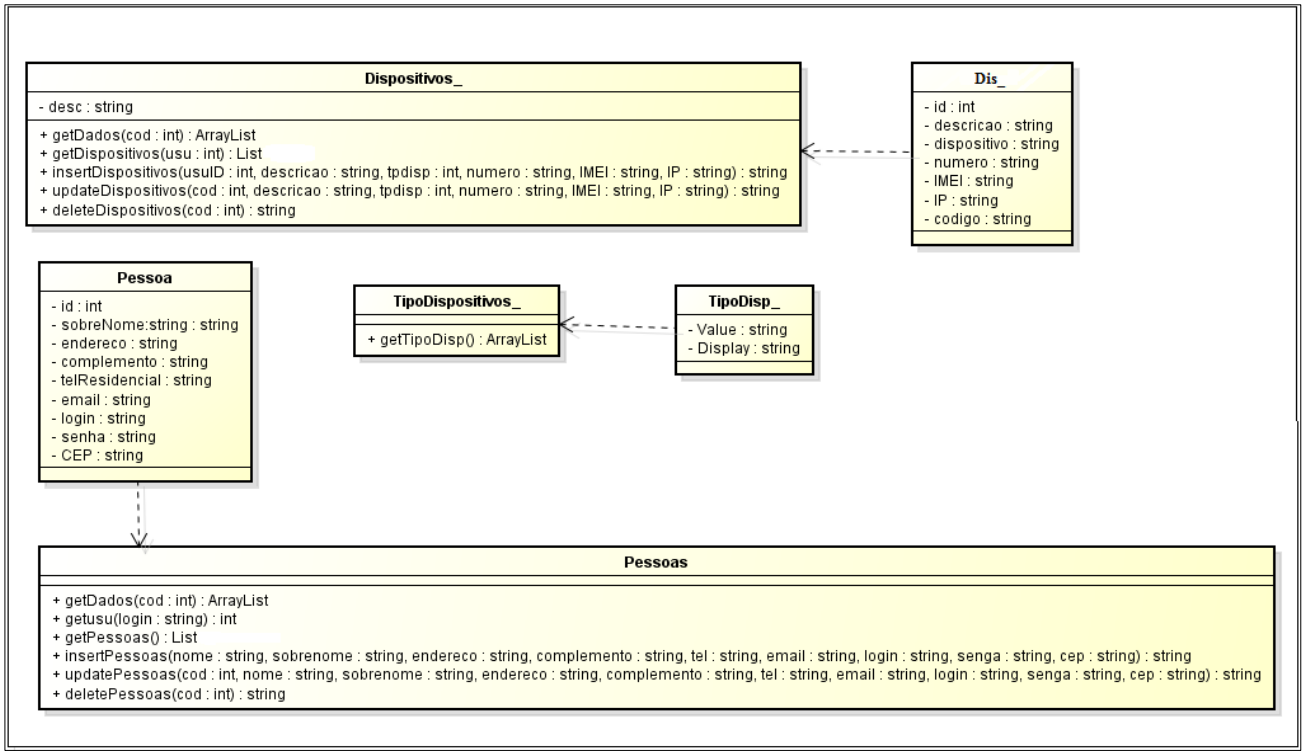

Figura 2. Diagrama de classe do Framework do módulo Web.

Tabela 1. Descrição das classes utilizadas no Framework.

Classes	Descrição
Dispositivos_()	Realiza a inserção, alteração, deleção e listagem dos dispositivos, onde os métodos correspondentes são: getDados(), getDispositivos(), insertDispositivos(), updateDispositivos(), deleteDispositivos().
Pessoas()	Realiza a inserção, alteração, deleção e listagem dos usuários cadastrados, onde os métodos correspondentes são: getDados(), getUsu(), getPessoas(), insertPessoas(), updatePessoas().
Dis_(), tipoDis() e Pessoa()	São classes que representam as entidades do modelo.

A seguir na figura 3, pode ser visto um exemplo de utilização da classe Dispositivos(), onde é instanciado um objeto denominado *d* do tipo Dispositivos_, para posterior utilização do método insertDispositivos() que pertence a classe.

```
1  String retorno;
2  Dispositivos_ d = new Dispositivos_();
3  retorno = d.insertDispositivos(usuID, descricao, tpdisp, numero, IMEI, IP);
4
```

Figura 3. Exemplo de utilização da classe Dispositivos_().

2.2.2 Módulo Web Service

O Web Service é utilizado na integração de sistemas e na comunicação entre aplicações diferentes, com ele é possível que novas aplicações possam interagir com aquelas que já existem. Os Web Services são componentes que permitem às aplicações enviar e receber dados em formato Extensible Markup Language – XML que é considerado uma linguagem universal (W3SCHOOLS, 2013f).

Utilizando a tecnologia Web Service, uma aplicação pode invocar outra para efetuar tarefas simples ou complexas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Por outras palavras, os Web Services fazem com que os seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo Web Service (W3SCHOOLS, 2013d).

Os Web Services usam os protocolos padrões, como Hypertext Transfer Protocol – HTTP, XML, e Simple Object Access Protocol – SOAP responsável em transferir os dados no formato XML e encapsula-los (MEDEIROS, 2012).

"A arquitetura dos Web Services é baseada na interação de três personagens: Provedor de Serviços, Consumidor de Serviços e Registro dos Serviços. A interação destes personagens envolve as operações de publicação, pesquisa e ligação." (RECKZIEGEL, 2006 apud KREGER, 2001).

O Web Service utilizado neste trabalho foi desenvolvido no IDE Eclipse (ECLIPSE, 2012) com a linguagem Java, e possui três classes:

- Classe Connect() que faz a conexão entre o Web Service e o banco de dados, utilizando o drive Java Database Connectivity – JDBC para SQLServer.
- Classe Data Access Object – DAO() é onde são realizadas as manipulações do banco de dados como as funções CRUD.
- Classe WebServiceSecurityPhone() é a responsável pela comunicação entre Web Service e dispositivo Android, possui os seguintes métodos descrito na tabela 2.

A figura 4, demonstra o diagrama de classe do Web Service.

Tabela 2. Descrição dos métodos utilizados na classe WebServiceSecurityPhone().

Método	Descrição
HelloWord()	Criado como exemplo para realizar o primeiro teste e verificar o funcionamento do Web Service.
setLatitudeLongitude()	Chama a classe DAO() para realizar as inserções das coordenadas.
testeConexao()	Instancia um objeto da classe Connect() para verificar a conexão do Web Service;
getTypeOperation()	Resgata a operação que deve ser executada pelo dispositivo.
setTypeOperation()	É utilizado tanto pelo dispositivo móvel, quanto pela pagina Web, setando qual operação deve ser executada.

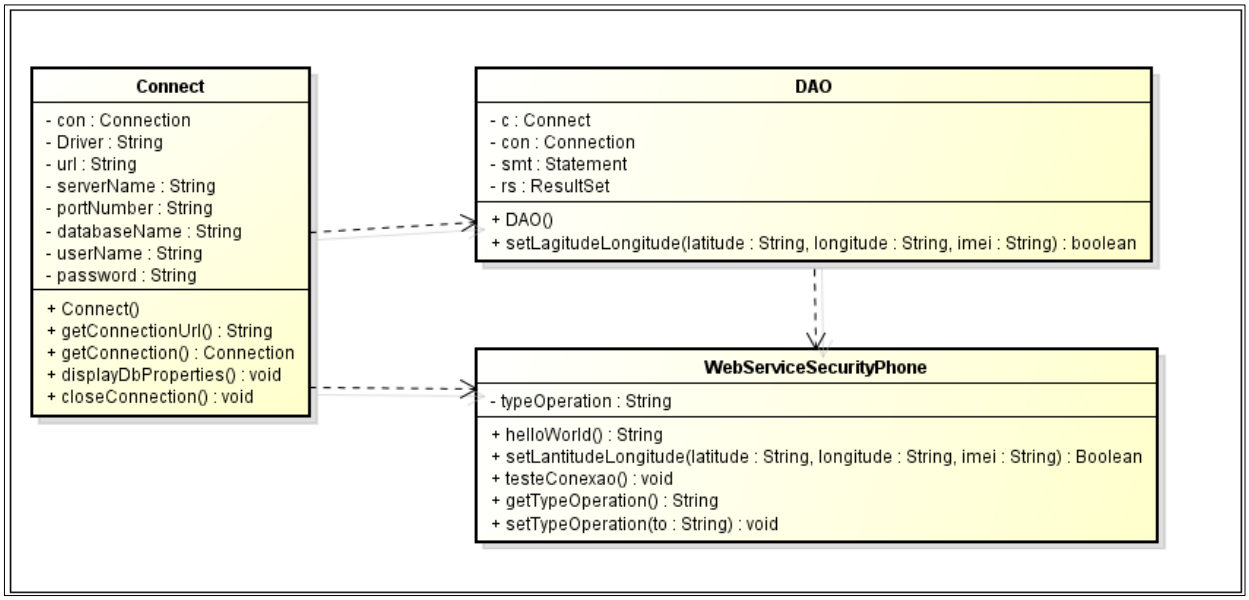

Figura 4. Diagrama de classes do Web Service.

2.2.2.1 Protocolo de Comunicação

Para construção do presente Web Service foi utilizado o servidor Tomcat (APACHE TOMCAT, 2013) que trabalha com um Framework de Web Service chamado Axis. Ambos são subprojetos livres do Apache (APACHE SOFTWARE FOUNDATION, 2013).

De acordo com Pamplona (2010) o Tomcat é um container para JSP e Servlets muito conhecido e utilizado. O Axis é um conjunto de ferramentas para o desenvolvimento de Web Services, dentre suas principais funcionalidades estão:

- Implementação do protocolo SOAP;
- Implementação de classes para agilizar a comunicação e a publicação de Web Services;

- Utiliza containers JSP para disponibilizar os Web Services na rede.

Para que um Web Service seja publicado e que outras pessoas possam utilizá-lo é necessário definir o seu acesso e os valores que serão retornados. Estas definições são descritas em um arquivo XML de acordo com a padronização Web Service Description Language – WSDL, este arquivo deve ser construído para que os usuários do serviço possam entender seu funcionamento (W3SCHOOLS, 2013e).

WSDL é um documento proposto pela World Wide Web Consortion – W3C, escrito em XML que tende padronizar as descrições das funcionalidades oferecidas por Web Services de forma independente de plataforma ou linguagem, onde possui duas finalidades: Expor os métodos que um determinado serviço disponibilizará e possibilitar a localização de um determinado serviço (SANCHEZ, 2011).

Na figura 5, temos um trecho do código do WSDL utilizado neste trabalho.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://modelo" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://modelo" xmlns:intf="http://modelo"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://modelo" xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="helloWorld">
    <complexType/>
   </element>
   <element name="helloWorldResponse">
    <complexType>
     <sequence>
      <element name="helloWorldReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="getTypeOperation">
    <complexType/>
   </element>
   <element name="getTypeOperationResponse">
    <complexType>
     <sequence>
      <element name="getTypeOperationReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="setTypeOperation">
    <complexType>
     <sequence>
      <element name="to" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="setTypeOperationResponse">
    <complexType/>
   </element>
   <element name="setLantidudeLongitude">
    <complexType>
     <sequence>
      <element name="latitude" type="xsd:string"/>
      <element name="longitude" type="xsd:string"/>
      <element name="imei" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="setLantidudeLongitudeResponse">
    <complexType>
     <sequence>
      <element name="setLantidudeLongitudeReturn" type="xsd:boolean"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>
```

Figura 5. Trecho do código do WSDL utilizado no trabalho.

Para que aja a comunicação entre Web Service, o protótipo Android e sistema Web, primeiramente deve-se ativar o servidor Tomcat.

2.2.3. A Base de Dados

O presente trabalho utiliza banco de dados SQLServer pois o mesmo possui integração com o componente LINQ do Framework .Net 3.5.

É utilizado pelas classes Dispositivos() e Pessoas() do módulo Web que fazem o acesso ao banco utilizando o componente LINQ e pelo Web Service Java utilizando o drive JDBC pela classe DAO(). Na figura 6, temos a modelagem do banco de dados que foi utilizada neste trabalho.

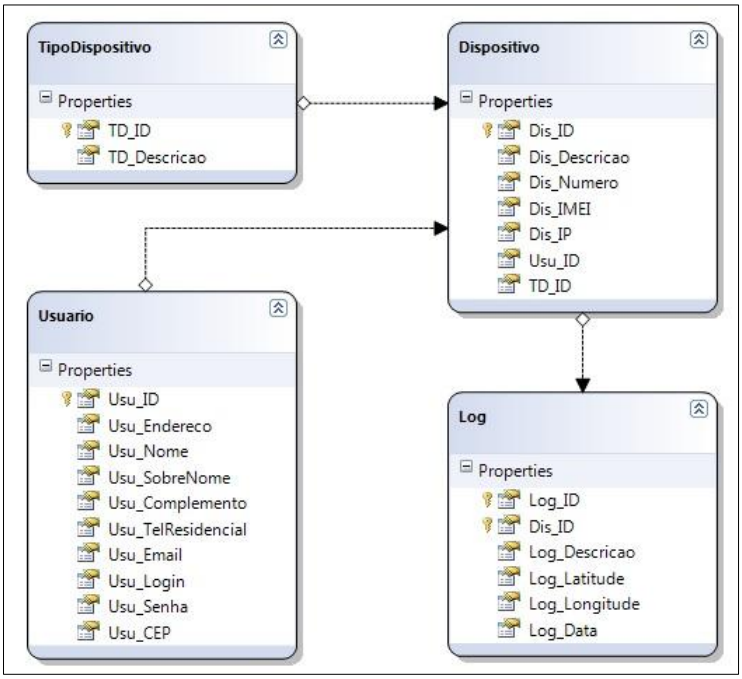

Figura 6. Base de dados utilizado no trabalho.

2.2.4. Módulo Móvel

No desenvolvimento do módulo móvel foi utilizado a linguagem Java no IDE Eclipse com o SDK do Android. O módulo, denominado de Framework é formado por várias classes que estão descritas na tabela 3 e na figura 7 temos o diagrama de classe desse módulo.

Tabela 3. Descrições das classes do Framework.

Classe	Descrição
GPS()	Possui o método que busca as coordenadas no dispositivo e os métodos que enviam a latitude e longitude do dispositivo para o Web Service.
SDCard()	Responsável por realizar as operação de limpeza do cartão de memória onde possui o método que verifica a disponibilidade do cartão e diversos métodos que fazem a exclusão dos dados.
Alerta()	Possui métodos necessários para realizar a operação de alerta sonoro e o método setMSG() que é responsável em exibir a mensagem de texto no dispositivo.
Config()	Possui alguns métodos que capturam informações do dispositivo como o IMEI, o número do chip e verifica se existe alguma conexão com a internet podendo ser WIFI ou 3G.
ValidaAcessoHttpTransport()	O próprio nome já diz ela valida o acesso e é responsável em fazer o transporte das informações ao Web Service.

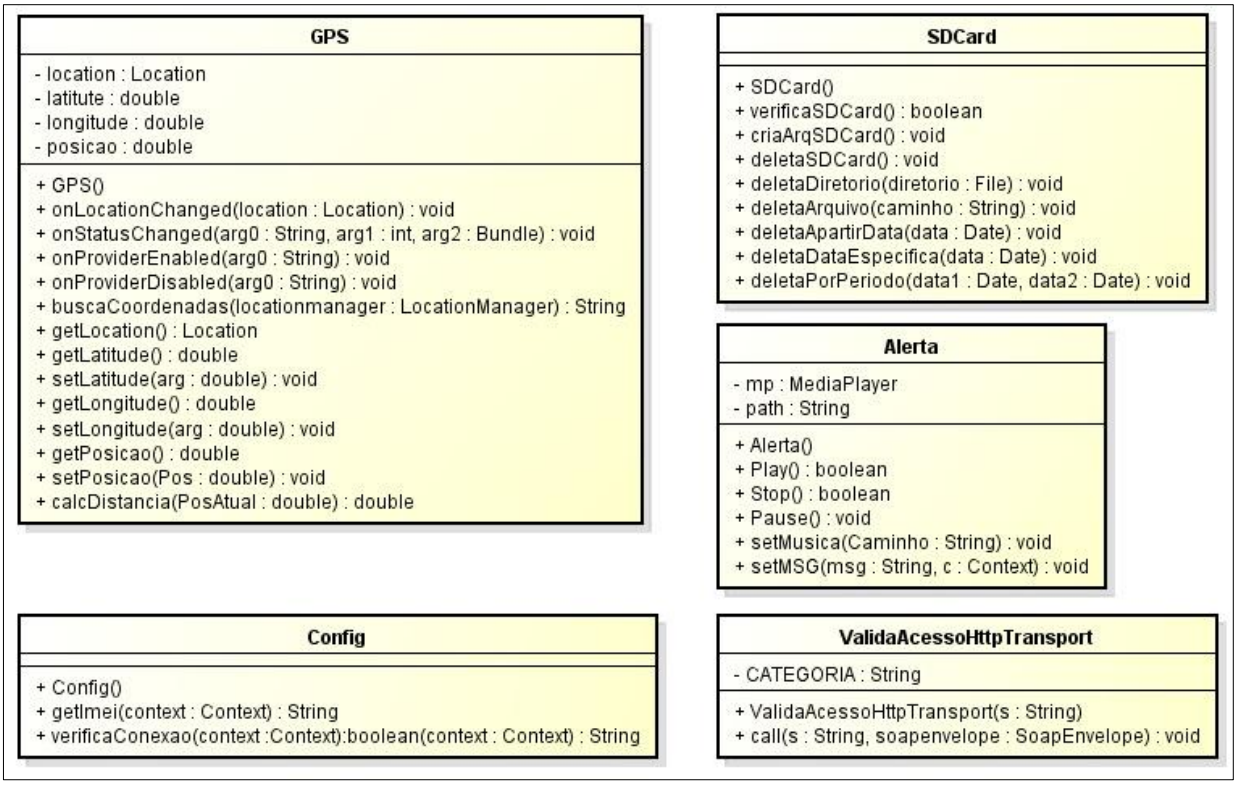

Figura 7. Diagrama de classe do módulo móvel.

2.2.4.1 A Classe Principal

A classe principal do protótipo é a MainActivity() que corresponde a tela activity_main. Essa classe é responsável pelo monitoramento do dispositivo móvel, com os métodos descritos na tabela 4.

Tabela 4. Descrições dos métodos utilizado.

Método	Descrição
onCreate()	É onde as atividades são inicializadas e vinculadas com os componentes da tela.
startOperation()	Chamado pelo método onCreate() após ter verificado se existe conexão WIFI ou 3G, nesse método é realizado uma thread que fica chamando o método callWebservice() a cada 5000ms.
callWebservice()	Realiza a conexão com o Web Service, sua comunicação é através do envelopamento utilizando a biblioteca KSOAP2, após o retorno da resposta do Web Service que é realizado automático pela biblioteca KSOAP2 é verificado a operação desejada e é executado o processo correspondente.
onCreateOptionsMenu()	Responsável em gerenciar a criação do menu.

"O KSOAP2 é uma biblioteca cliente do serviço Web SOAP para ambientes Java" (KSOAP2, 2012). "Permite o consumo de Web Services sem a necessidade de geração de código ou uso de proxies dinâmicos" (USHISIMA, 2011). "Essa biblioteca é muito útil, pois encapsula a geração e leitura dos XMLs (oferecendo os dados em forma de SoapObjects), bem como a comunicação com o serviço, tratando, inclusive, erros levantados pelo servidor e lançando exceções quando apropriado" (SILVA, 2011). A figura 8, demostra um trecho código utilizado na implementação do KSOAP2.

```
//Cria Request Indentificando o Metodo e o Namespace
//SoapObject request =  new SoapObject("http://modelo/", "getTypeOperation");
SoapObject request =  new SoapObject(NAMESPACE, METHOD_NAME);

//Seta Coordenadas Para o WebService Gravar na Base de Dados
if (METHOD_NAME.equals("setLantidudeLongitude"))
{
    Config conf = new Config();
    imei = conf.getImei(this);

    //Alimenta Parametros do WebService Para o Metodo setLatitudeLongitude
    request.addProperty("latitude", String.valueOf(latitude));
    request.addProperty("longitude", String.valueOf(longitude));
    request.addProperty("imei", imei);
}
else
{
    //Zera TypeOperation Para Que Não Seja Requisitado Nada Nesse Momento
    if (METHOD_NAME.equals("setTypeOperation"))
    {
        //Alimenta Parametros do WebService Para o Metodo setTypeOperation
        request.addProperty("to","Aguardando Operação");
    }
}

//Cria Envelope SOAP
SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);

//Envelopa a Requisicao
envelope.setOutputSoapObject(request);

//Cria HTTP Transporte
HttpTransportSE httpTransport = new HttpTransportSE(URL);

//Faz Uma Chamada Para o WebService no Metodo getTypeOperation
//httpTransport.call(http://modelo/getTypeOperation, envelope);
httpTransport.call(SOAP_ACTION + METHOD_NAME, envelope);

//Obtem a Resposta do WebService
Object resultadoValidaAcesso = envelope.getResponse();
String series = resultadoValidaAcesso.toString();
```

Figura 8. Código utilizado na implementação do KSOAP.

O request é feito usando um SoapObject e cada parâmetro da chamada é uma propriedade do SoapObject. A chamada ao servidor de aplicações é feita usando um HttpTransportSE passando um envelope SOAP com a request gerada. Após a execução do serviço, o envelope irá conter o retorno do serviço. No caso, o resultado é uma String.

2.2.4.2 Classe IMEI

A classe IMEIActivity() é a classe da tela activity_imei que mostra o numero do IMEI do dispositivo e inicializa o monitoramento das requisições do usuário. Seu ciclo de vida é o mesmo presente em uma Activity como explicado anteriormente e por fim o método carregaTelaMonitoramento() onde são realizadas as chamadas de inicialização da tela de monitoramento. A figura 9, demostra o código utilizado na implementação.

```
1  public class IMEIActivity extends Activity
2  {
3      private String imei;
4
5      @Override
6      public void onCreate(Bundle savedInstanceState)
7      {
8          super.onCreate(savedInstanceState);
9          setContentView(R.layout.activity_imei);
10
11         Button monitorar = (Button)findViewById(R.id.button2);
12         Button bimei = (Button)findViewById(R.id.button3);
13         final TextView txt = (TextView)findViewById(R.id.textMEI)
14
15         Config conf = new Config();
16         imei = conf.getImei(this);
17
18         monitorar.setOnClickListener(new View.OnClickListener() {
19             public void onClick(View v) {
20                 CarregaTelaMonitoramento();
21             }
22         });
23
24         bimei.setOnClickListener(new View.OnClickListener() {
25             public void onClick(View v) {
26                 txt.setText("IMEI do Celular:\n"+ imei);
27             }
28         });
29     }
30
31     public void CarregaTelaMonitoramento()
32     {
33         setContentView(R.layout.activity_main);
34         Intent i = new Intent(this, MainActivity.class);
35         startActivity(i);
36     }
```

Figura 9. Código da classe IMEIActivity().

2.2.4.3 Classe Câmera

A realização da operação de tirar foto não permite que o usuário visualize e perceba que a foto foi tirada. Para o seu desenvolvimento foi necessário à criação de uma classe que possuísse a extensão de uma Activity, pois é necessário a utilização de um componente de interface o SurfaceView. E que a classe também utilizasse a implementação do SurfaceHolder.Callback necessário para poder adicionar o callback à SurfaceView. Na figura 10, temos a demonstração do código fonte.

```
public class CameraActivity extends Activity implements SurfaceHolder.Callback
{
    Camera camera;  // Objeto que Representa a Câmera
    SurfaceView surfaceView; // View Para Exibição do Preview da Imagem
    SurfaceHolder surfaceHolder;
    File imageFile; // Local de armazenamento da foto tirada

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

            getWindow().setFormat(PixelFormat.UNKNOWN);
            surfaceView = (SurfaceView) findViewById(R.id.preview);

            // Adiciona o callback à SurfaceView, para que a aplicação seja notificada quando
            // a superfície for criada, alterada ou destruída
            surfaceHolder = surfaceView.getHolder();
            surfaceHolder.addCallback(this);
            surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

            // Defino o Local Onde a Imagem será Salva
            this.imageFile = new File(String.format("/sdcard/.projeto/%d.jpg",System.currentTimeMillis()));
            //
            tiraESalvaImagem();
            // Finalizo a Chamada da Activity e Retorno para Activity que a Chamou
            finish();
    }

    public void tiraESalvaImagem()
    {
        try
        {
            if(camera == null)
            {
                // Identifico o Número de Câmera que o Dispositivo Possivel
                int qtdCam = Camera.getNumberOfCameras();

                // Obtém uma Instância da Câmera
                // Se qtdCam retorna 1, instância a Câmera Traseira, se não a Dianteira
                camera = Camera.open(qtdCam-1);
            }
        }
        catch (Exception e)
        {  }

        if(camera != null)
        {
            try
            {
                // Instância um Objeto que Contem as Definições da Câmara
                Camera.Parameters parametros = camera.getParameters();
                // Modifico o Parâmetro
                parametros.setPictureFormat(PixelFormat.JPEG);
                // Seto a Modificação na Câmera
                camera.setParameters(parametros);
                // Seto o Preview
                camera.setPreviewDisplay(surfaceHolder);
                // Inicia o Preview
                camera.startPreview();

                // Chamado quando o obturador é aberto
                ShutterCallback shutterCallback = new ShutterCallback()
                {
                    public void onShutter()
                    {
                      // Não Tenho a Necessidade de Implementar Nada
                    }
                };

                // Manipula dados para retrato cru
                PictureCallback rawCallback = new PictureCallback()
                {
                    public void onPictureTaken(byte[] _data, Camera _camera)
                    {
                      // Não Tenho a Necessidade de Implementar Nada
                    }
                };

                // Manipula os dados para imagens JPEG
                PictureCallback jpegCallback = new PictureCallback()
                {
                    public void onPictureTaken(byte[] _data, Camera _camera)
                    {
                      FileOutputStream fos = null;
                      try
                      {
                          try
                          {
                              // Grava os bytes da imagem no arquivo onde a foto deve ser armazenada
                              fos = new FileOutputStream(imageFile);
                              fos.write(_data);
                          }
                          finally
                          {
                              if (fos != null)
                              {
                                  fos.close();
                              }
                          }
                      }
                      catch (IOException e)
                      {
                          e.printStackTrace();
                      }
                    }
                };
                // Tira uma foto. O callback Fornecido é Chamado Assim que a Imagem JPEG Estiver Disponivel
                camera.takePicture(shutterCallback, rawCallback, jpegCallback);
            }
            catch (IOException e)
            { }
        }
    }
```

Figura 10. Código da classe CameraActivity()

2.3. Protótipo MonitoremSempre Móvel

O Protótipo MonitoremSempre Móvel é baseado no módulo móvel descrito anteriormente. O protótipo se comunica com o Web Service que por sua fez faz a comunicação com o sistema Web. Na figura 11 e 12, podem ser visualizadas as interfaces oferecidas para a manipulação do protótipo. A figura 11 demostra as telas de apresentação do aplicativo, de manipulação e de monitoramento.

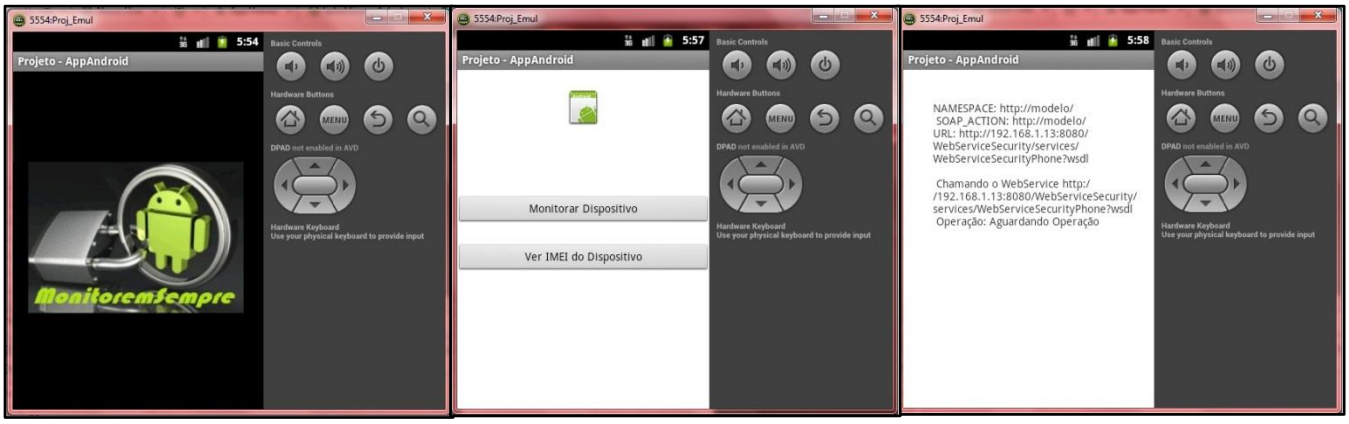

Figura 11. Telas do protótipo Android

Cada tela do protótipo é uma classe devidamente programada para exibição dos componentes, seguindo as especificações do Android e explicada anteriormente.

A figura 12, demostra as principais telas do sistema Web, a tela de cadastro do dispositivo móvel, a tela onde são mostrados os dispositivos cadastrados e por ultimo a tela onde são realizadas as operações. Para que haja a manipulação das telas é necessário realizar um cadastro prévio do usuário e assim terá acesso a uma área restrita para cadastro de seus dispositivos, onde a principal informação é o International Mobile Equipment Identity – IMEI do dispositivo para vinculo entre dispositivos específicos. A partir desse ponto o usuário poderá escolher qual dispositivo cadastrado deseja monitorar e será redirecionado para pagina onde poderá escolher qual procedimento deseja realizar.

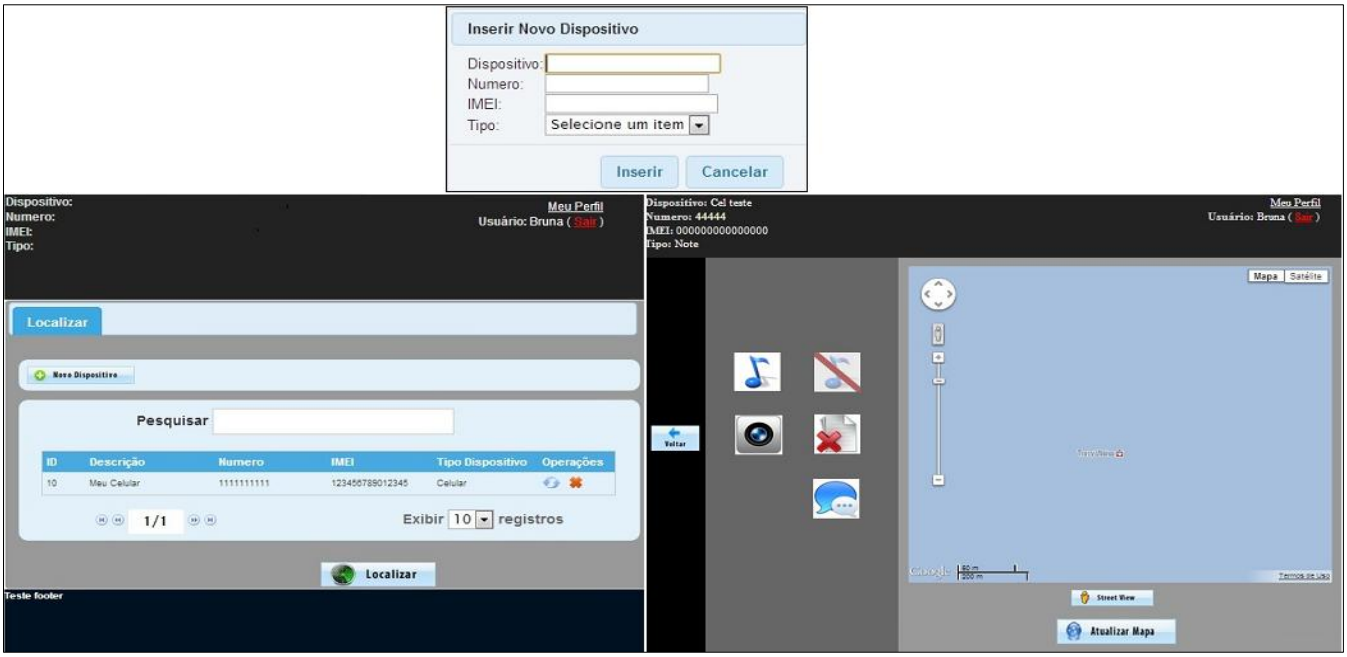

Figura 12. Layout das principais telas da aplicação Web

3 RESULTADOS E DISCUSSÃO

Os resultados obtidos neste trabalho demonstraram que, os métodos desenvolvidos como o monitoramento do dispositivo móvel, a exclusão dos dados do cartão de memória, o envio de mensagens de texto, alerta sonoro e disparos de fotos, foram eficazes no sentido que o monitoramento funcionou de forma adequada, pois as coordenadas foram obtidas de maneira satisfatória permitindo a localização do dispositivo.

Os demais métodos também tiveram seu funcionamento corretos, pois através do seu número IMEI foi possível identificar o dispositivo e realizar tais operações, mostrando a viabilidade do Framework em implementar tais funções, permitindo a ssim o auxílio para o desenvolvimento de aplicações neste seguimento.

A comunicação entre os módulos: Web, Web Service e móvel funcionaram adequadamente, possibilitando assim a utilização do Framework desenvolvido e a execução do trabalho como um todo.

Os testes para a comprovação da eficiência do protótipo MonitoremSempre Móvel foram realizados utilizando aparelhos celulares da marca Samsung com a plataforma Android nas versões 2.3.3 e 4.0. Alguns testes foram realizados utilizado o emulador Android na versão 2.3.3, para que se pudesse procurar possíveis erros utilizando o debug e para verificação da interação e funcionamento do W eb Service entre a página Web e o dispositivo móvel,

não sendo eficaz na visualização de resultados dos métodos desenvolvidos.

O protótipo MonitoremSempre Móvel poderá ocasionar falhas caso não haja a inicialização ou a desinstalação do aplicativo no dispositivo móvel, por falta de conexão com o GPS e internet, ou ainda por não conter os requisitos básicos necessários no dispositivo como a falta do auto falante, câmera, GPS ou Internet.

Tanto o sistema Web quanto o Web Service não foram hospedados em um servidor, sendo assim os testes com ambos e com os dispositivos móveis foram executados localmente. Sendo assim, para que seus módulos se comuniquem, estes devem estar conectados na mesma rede local e o Web Service ter sido inicializado.

4 CONCLUSÕES

Os estudos e o desenvolvimento deste trabalho comprovaram a viabilidade do Framework, que através dos testes as operações foram realizadas com sucesso.

Além da necessidade de um Framework como o proposto, surgem a cada dia novas necessidades, gerando grande oportunidade de mercado para novos e experientes desenvolvedores Android.

Tendo em vista o enorme crescimento dos números de dispositivos Android e a grande importância da proteção destes, é indispensável à existência de Frameworks para facilitar o desenvolvimento de novas e já existentes aplicações que realizem a localização de dispositivos móveis, trazendo tranquilidade e segurança aos usuários. Ficando como sugestões de trabalhos futuros a reescrita do Framework para outras plataformas como o Windows Phone e IOS. Algumas extensões também podem ser desenvolvidas, como envio de fotos tiradas para pagina Web, envios de mensagens sonoras e bloqueio do dispositivo por senha.

REFERÊNCIAS

ANDROID. **Site Oficial do Android**. Disponível em: <http://www.android.com/>. Acesso em: 21 ago. 2012.

APACHE TOMCAT. **Site Oficial do Apcahe Tomcat**. Disponível em: <http://tomcat.apache.org/>. Acesso em: 05 de abr. 2013

APACHE SOFTWARE FOUNDATION. **Site Oficial do Apache**. Disponível em: <http://www.apache.org/> Acesso em: 05 abr. 2013.

AVAST MOBILE SECURITY. **Site Oficial do Avast**. Disponível em: <http://www.avast.com/pt-br/index>. Acesso em: 01 nov. 2013.

DIAS, A.D.L. **Android conceitos e arquitetura**. Disponível em: <http://www.slideshare.net/AnaDoloresLimaDias/android-9149956>. Acesso em: 21 ago. 2012.

ECLIPSE. **Site Oficial do Eclipse**. Disponível em: <http://www.eclipse.org>. Acesso em: 16 out. 2012.

FERNANDES, M. **Configurando o ambiente de desenvolvimento Android com o Eclipse**. Disponível em: <http://pontov.com.br/site/android/276-configurando-ambiente-de-desenvolvimento-android-com-eclipse>. Acesso em: 10 set. 2012.

GOOGLE APIS. **Site do Google Code –APIS**. Disponível em: <https://code.google.com/apis/console/?pli=1#project:231103588716:access>. Acesso em: 21 ago. 2012.

GOOGLE DEVELOPERS. **Obtaining a Google Maps Android v1 API Key**. Disponível em: <https://developers.google.com/maps/documentation/android/mapkey>. Acesso em: 10 set. 2012.

JQUERY. **O que é JQuery?**. Disponível em: <http://jquery.com/>. Acesso em: 10 abr. 2013.

KSOAP2. **Site Oficial do KSOAP2**. Disponível em: <http://ksoap2.sourceforge.net/>. Acesso em: 05 abr. 2013.

LECHETA, R.R. **Google Android**: aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2. ed. São Paulo: Novatec Editora, 2010.

MATIOLLA, M. **Plataforma muda conceitos de inovação em celulares.** 2009. Disponível em: <http://softwarelivre.org/portal/geral/especial-android-plataforma-muda-conceito-de-inovacao-em-celulares>. Acesso em: 21 set. 2012.

MEDEIROS, H. **Conhecendo Web Services**. Disponível em: <http://www.devmedia.com.br/conhecendo-web-services/5070#ixzz2QwiXrvwu>. Acesso em: 08 mar. 2013.

MICROSOFT (2013a). **Introdução ao LINQ**. Disponível em: <http://msdn.microsoft.com/en-us/library/bb397897.aspx>. Acesso em: 10 abr. 2013.

MICROSOFT (2013b). **Introdução à linguagem C# e o Framework .Net**. Disponível em: <http://msdn.microsoft.com/en-us/library/z1zx9t92.aspx>. Acesso em: 10 abr. 2013.

MICROSOFT (2013c). **SQL Server**. Disponível em: <http://www.microsoft.com/sqlserver/pt/br/default.aspx>. Acesso em: 10 abr. 2013.

MICROSOFT (2013d). **.Net**. Disponível em: <http://www.microsoft.com/net>. Acesso em: 10 abr. 2013.

ORACLE. **Java**. Disponível em: <http://www.oracle.com/br/technologies/java/overview/index.html?origref=http://www.oracle.com/br/index.html>. Acesso em: 21 ago. 2012.

OPEN HANDSET ALLIANCE. **Android**. Disponível em: <http://www.openhandsetalliance.com/android_overview.html>. Acesso em: 21 ago. 2012.

PAMPLONA, V.F. **Web Services. Construindo, disponibilizando e acessando Web Services via J2SE e J2ME**, 2010. Disponível em: <http://javafree.uol.com.br/artigo/871485/Web-Services-Construindo-disponibilizando-e-acessando-Web-Services-via-J2SE-e-J2ME.html>. Acesso em: 08 mar. 2013.

PEREIRA, L.C.O.; SILVA, M.L. **Android para desenvolvedores**. Rio de Janeiro: Brasport, 2009.

RABELLO, R.R. **Utilizando Web Services no Google Android**. 2013. Disponível em: <http://www.cesar.org.br/site/files/file/WM23_Android_WebServices.pdf>. Acesso em: 07 mar. 2013.

ROGERS, R. et al. **Desenvolvimento de aplicações Android**. 1. ed. São Paulo: Novatec, 2009.

SANCHEZ, F. **WSDL – O que é? Para que serve? Onde utilizo?;** 2013. Disponível em: <http://fabriciosanchez.com.br/2/wsdl-o-que-e-pra-que-serve-onde-utilizo/>. Acesso em: 08 mar. 2013.

SILVA, B.V.D. **Android+Ksoap - Pt.1:** Transcrevendo SoapObjects através de Annotations e Reflection. 2011. Disponível em: <http://zbra.com.br/2011/07/06/androidksoap-pt-1-transcrevendo-soapobjects-atraves-de-annotations-reflection/>. Acesso em: 05 abr. 2013.

SILVA, F.M.C. **Entendendo um ciclo de vida de uma aplicação Android**. Disponível em: <http://www.devmedia.com.br/entendendo-o-ciclo-de-vida-de-uma-aplicacao-android/22922>. Acesso em: 21 ago. 2012.

SILVA, L.A. **Apostila de Android** - Programando passo a passo. 4. ed. Disponível em: < http://apostilaandroid.ueuo.com/adquirir-apostilas-antigas.php >. Acesso em: 20 ago. 2012.

TOLEDO, R. **Montando um ambiente para o desenvolvimento em Android**. 2012. Disponível em: <http://www.rafaeltoledo.net/montando-um-ambiente-de-desenvolvimento-para-android/>. Acesso em: 10 set. 2012.

USHISIMA, R. **Consumindo Web Service em aplicações Android**. 2011. Disponível em: <http://zbra.com.br/2011/03/30/consumindo-web-service-em-aplicacoes-android/>. Acesso em: 08 mar. 2013.

W3SCHOOLS (2013a). **AJAX Tutorial**. Disponível em: <http://www.w3schools.com/ajax>. Acesso em: 10 abr. 2013.

W3SCHOOLS (2013b). **JavaScript Tutorial**. Disponível em: <http://www.w3schools.com/js>. Acesso em: 10 abr. 2013.

W3SCHOOLS (2013c). **JSON Tutorial**. Disponível em: <http://www.w3schools.com/json>. Acesso em: 08 mar. 2013.

W3SCHOOLS (2013d). **Web Services Tutorial**. Disponível em: <http://www.w3schools.com/webservices>. Acesso em: 08 mar. 2013.

W3SCHOOLS (2013e). **WSDL Tutorial**. Disponível em: <http://www.w3schools.com/wsdl>. Acesso em: 08 mar. 2013.

W3SCHOOLS (2013f). **XML Tutorial**. Disponível em: <http://www.w3schools.com/xml>. Acesso em: 08 mar. 2013.

ZOEMOB. **Site Oficial Zoemob**. Disponível em: <https://www.zoemob.com/pt/>. Acesso em: 01 nov. 2013.