Desenvolvimento de Aplicações Utilizando Recursos de Geolocalização de Dispositivos Android

Hellysson Lucas Ramos de Araújo, Marcos Alberto Lopes da Silva

Instituto de Informática – Centro Universitário do Triângulo (UNITRI)
Caixa Postal 309 – 38.411-106 – Uberlândia – MG – Brasil

lucasra1313@hotmail.com, malopes21@gmail.com

Resumo. *Devido a crescente evolução de dispositivos móveis e a necessidade de localizar pessoas, a integração entre estas tecnologias e a Internet tende a ser cada vez mais utilizadas nas mais diversas áreas, inclusive para auxiliar profissionais na área de educação. Este artigo mostra o uso de uma aplicação Web utilizando recursos de geolocalização de dispositivos móveis com Android para localizar aparelhos celulares no globo terrestre.*

1. Introdução

A mobilidade e a tecnologia estão cada vez mais conquistando as pessoas através das facilidades que os mesmos proporcionam e mostra que os dispositivos móveis atuais conseguem fazer praticamente todas as tarefas de um computador doméstico. A evolução dos sistemas nos leva a admitir que tudo gira em torno da Internet, e atualmente com a integração de dispositivos móveis com tecnologias como o *GPS* (*Global Positioning System* ou Sistema de Posicionamento Global), mapas, redes sociais, reconhecimento de voz, dentre inúmeras outras tecnologias, torna isso uma tendência praticamente impossível de conter.

Localizar dispositivos através de satélites é possível nos dias de hoje, podendo citar como exemplo empresas que buscam controlar sua frota de caminhões, para os pais localizarem seus filhos, usado pela polícia no combate ao crime, usado em veículos como guia rodoviário, obtendo a localização atual dos mesmos.

Neste artigo será demonstrada a utilização do GPS dos s*martphones* com sistema operacional *Android* como meio de interação com um sistema Web que recebe os dados geográficos fornecidos por estes dispositivos, obtendo a localização destes aparelhos e mostrando no mapa a posição atual dos mesmos.

2. Surgimento das Tecnologias

Os mapas surgiram antes mesmo da escrita, com objetivo de orientação no espaço geográfico, antigamente os mapas eram feitos através de estudos de campo por geólogos e cartógrafos, com o passar do tempo, foram surgindo novas formas de construir mapas, os aviões também contribuíram através de radares antes mesmo do surgimento dos chamados satélites artificiais [SHVOONG.COM 2008] [EDUCA 2009].

Auxiliando na evolução dos mapas, os satélites artificiais são objetos construídos pelo ser humano e colocado em órbita de algum planeta, sendo utilizados para diversos fins [GFORUM 2011].

A partir da construção de satélites entramos em um novo conceito de mapas, possibilitando o surgimento do GPS criado pelo Departamento de Defesa dos Estados Unidos no início da década de 1960, com função básica identificar a localização de um receptor que capte sinais emitidos de Satélites, tecnologia desenvolvida pra fins bélicos durante a Guerra do Golfo (1990-1991) facilitando a orientação em lançamento de mísseis, porém o uso do GPS foi declarado totalmente operacional somente a partir de 1995. O GPS é um sistema de posicionamento geográfico que disponibiliza as coordenadas de um determinado local na Terra [ZETTEL 2002] [UOL EDUCAÇÃO 2009].

A Internet, na época conhecida como ARPANET criada pelo Departamento de Defesa dos Estados Unidos, surgiu baseada em objetivos militares, em plena Guerra Fria, período compreendido entre o final da Segunda Guerra Mundial (1945) e a extinção da União Soviética (1991), com o propósito de manter as comunicações caso o inimigo destruísse os meios de telecomunicações convencionais e após alguns anos, também se tornou importante para o meio acadêmico.

Conhecida também como Rede Mundial de Computadores, a Internet foi se popularizando a partir do ano de 1990, onde o engenheiro inglês Tim Bernes Lee desenvolveu a WWW (*World Web Wide* – Rede de Alcance Mundial) juntamente com o surgimento de navegadores como Internet Explorer da Microsoft e o Netscape Navigator, possibilitando a interação de internautas com o mundo externo sem mesmo sair de casa [SUAPESQUISA 2012]. A partir destes conceitos básicos, iremos aprofundar um pouco mais nas tecnologias envolvidas para obter melhor compreensão sobre o assunto e abordar o conjunto para satisfazer o estudo do artigo.

2.1. Sistemas Web e conceitos baseados em Requisição e Resposta

A princípio a Internet juntamente com Web sites de modo geral tinha como objetivo principal divulgar informações promovendo o marketing digital. Os sistemas baseados em Web surgiram devido a necessidades de alavancar negócios e possibilitar o uso de sistemas únicos corporativos capazes de controlar e integrar os trabalhos e processos encurtando o caminho entre as empresas e suas filiais, contribuindo na redução de custos com transporte e locomoção, gerando melhor gestão do tempo, possibilitando o crescimento no mercado e garantindo maior destaque entre os concorrentes.

Servidores e clientes Web interagem entre si através de componentes de comunicação chamados protocolos que são baseados em pedidos e mensagens de requisição e resposta utilizando a Internet. Os servidores Web contam com os *Servlets* que são pequenos servidores que trabalham em um container (*Apache Tomcat, Glassfish* dentre outros) e que tratam as requisições e respostas direcionando os dados aos seus devidos locais de acordo com as regras definidas pelo negócio.

2.2. Dispositivos Móveis

A informação na palma da mão é o que todos atualmente possuem com estes pequenos e poderosos dispositivos de comunicação. Através da evolução da telefonia móvel, a integração destes dispositivos com a Internet tornaram-se fundamentais para o aparecimento de *smartphones*, *ipads*, *tablets*, entre outros. Dentre eles vamos destacar o

s*martphone* onde a tecnologia celular aliada a outras tecnologias e aplicações, conseguiu transformar um simples celular nesta moderna ferramenta de trabalho e entretenimento.

Simon, o primeiro Smartphone do mundo criado pela IBM em 23 de novembro de 1992, já possuía alguns recursos como a tela sensível ao toque, câmera, mapas e player de música. Não fez muito sucesso, pois além de ser um aparelho muito pesado, a bateria durava pouco tempo e apresentava muitos defeitos o que não chamou muita atenção bastando apenas seis meses para sair do mercado. Atualmente graças à tecnologia, os *smartphones* estão muito melhores e mais rápidos [CANALTECH 2012].

Um aparelho integrado com câmera, conexão de rede sem fio *(*w*i-fi)*, *bluetooth*, GPS, mapas, televisão e vários outros recursos torna-o cada vez mais completo. Existem vários sistemas operacionais no mercado para s*martphones* como o *Blackberry*, *Symbian*, *Android* da *Google*, *IOS* da *Apple*. Cada dispositivo possui um sistema operacional de acordo com o seu fabricante [SIGNIFICADOS 2012].

2.2.1. Google Android

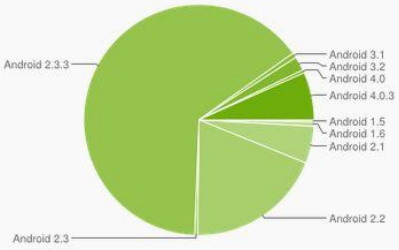
Em Agosto de 2005 a Google anunciou a compra da Android Inc., empresa que desenvolvia sistemas para celular situada em Palo Alto Califórnia (Estados Unidos). Em 2007 foi criada a OHA (*Oppen Handset Alliance*) que consiste na união de várias empresas com o objetivo de prover melhores recursos e menor custo para usuários finais, que é a responsável pelo desenvolvimento do Android, mas a Google é quem lidera e gerencia seus produtos e projetos. O HTC-G1 conforme na Figura 1 foi o primeiro aparelho celular com Android. O Google Android pode ser considerado a primeira plataforma livre e completa para dispositivos móveis.

Figura 1. HTC - G1 Primeiro celular com Google Android [GO ANDROID 2012].

2.3. Sistema Operacional Android

O Android possui um sistema operacional "aberto" baseado no kernel do Linux, eliminando o custo com a compra do sistema operacional, a evolução de suas versões é batizada com nomes de doces. A Figura 2 mostra como exemplo as versões e seus respectivos nomes e também traz dados estatísticos da utilização das mesmas.

Version	Codename	API Level	Distribution
1.5	Cupcake	3	0.3%
1.6	Donut	4	0.6%
2.1	Eclair	7	5.2%
2.2	Froyo	8	19.1%
2.3 - 2.3.2	Gingerbread	9	0.4%
2.3.3 - 2.3.7		10	64.6%
3.1	Honeycomb	12	0.7%
3.2		13	2%
4.0 - 4.0.2	Ice Cream Sandwich	14	0.4%
4.0.3 - 4.0.4		15	6.7%



Data collected during a 14-day period ending on June 1, 2012

Figura 2. Estatística de utilização das versões Android [CELULAR ANDROID 2012].

Além de ser um sistema livre, ele não possui amarração a um hardware específico, suas aplicações na maioria são desenvolvidas em Java, mas executadas através da Máquina Virtual Dalvik (*Dalvik Virtual Machine*) responsável por rodar todos os aplicativos do Android através da conversão do arquivo executável em .dex (*Dalvik Executable)*, formato otimizado que utiliza um mínimo de memória. Para melhorar a experiência com o usuário, a Google utiliza aparelhos modernos já com recursos como câmera, GPS, recursos multitoques, wi-fi e vários outros recursos já consagrados no mercado de dispositivos móveis [ANDROID TECH 2011] [TECHNÈ 2010]. Para os desenvolvedores a plataforma conta com um ambiente de desenvolvimento SDK (*Software Development Kit* ou Pacote de Desenvolvimento de Software) criado no final de 2007 também livre, facilitando o desenvolvimento de aplicações para os dispositivos Android e facilitando a customização tanto do sistema operacional como a de aplicativos.

2.4. Geolocalização e o GPS

A vontade da Google em lançar um aparelho com serviços baseados em localização se deu graças à compra Android Inc. que desenvolvia suas plataformas baseada em Linux. Com isso a Google Android passou a disponibilizar seus aparelhos já com estes serviços [WIKIPEDIA 2012]. A geolocalização é um processo pela qual identifica o local exato onde você está (latitude e longitude) através de seu IP (*Internet Protocol*) [DICIONÁRIO INFORMAL 2012].

Baseado nesta definição, Geolocalização nada mais é que encontrar um dispositivo em um local através de coordenadas geográficas, os smartphones na sua maioria já possuem aplicações que utilizam este recurso como o *Foursquare*, GPS, *Instagram*, aplicativos de rastreamento por localização.

O GPS, comumente usado para obter localização é um elemento importante que utiliza métodos baseados em dados geográficos. Seu funcionamento depende de 3 (três) componentes: o espacial, o de controle e o utilizador.

O espacial é composto de 27 satélites em órbita, o de controle as são estações de controle dos satélites com o objetivo de manter atualizadas as posições dos satélites ao todos são 5 (cinco) espalhados no Globo Terrestre. Por ultimo é o que temos que adquirir para poder usufruir desta tecnologia, o utilizador que é um receptor GPS que obtém sua posição atual. Para que o dispositivo seja localizado, é utilizado um relógio interno que satélites e receptores possuem que marcam a hora de quando o sinal é emitido, e a hora que ele sai do satélite e o tempo que demorou pra chegar, o receptor obtém a distância em relação ao satélite, e como as posições dos satélites mudam constantemente é possível ter a localização exata do dispositivo. Para melhor precisão no cálculo é utilizado um sistema de triangulação de sinais, onde se utiliza 3 (três) Satélites como demonstrado na Figura 3, no caso para encontrar a altitude seria necessário mais 1 (um) satélite, mas utilizando os mesmos princípios de cálculo [TECMUNDO 2009].

Figura 3. Triangulação de Sinais dos Satélites [TECMUNDO 2009].

Com estes conceitos foi possível conhecer algumas das tecnologias que atualmente fazem parte de nosso meio, e que com a crescente evolução dos dispositivos móveis aliados a grandes empresas de âmbito mundial é possível ampliar cada vez mais a conectividade entre países e pessoas utilizando recursos de geolocalização e Internet.

3. Arquiteturas baseadas em Web e Tecnologias

Como foi visto anteriormente os conceitos básicos das tecnologias envolvidas no artigo, agora serão abordados os princípios fundamentais para o desenvolvimento de aplicações no Android, com conectividade via Web e recursos de geolocalização.

3.1. Arquitetura Web

A estrutura básica de funcionamento desta arquitetura pode ser visualizada na Figura 4.

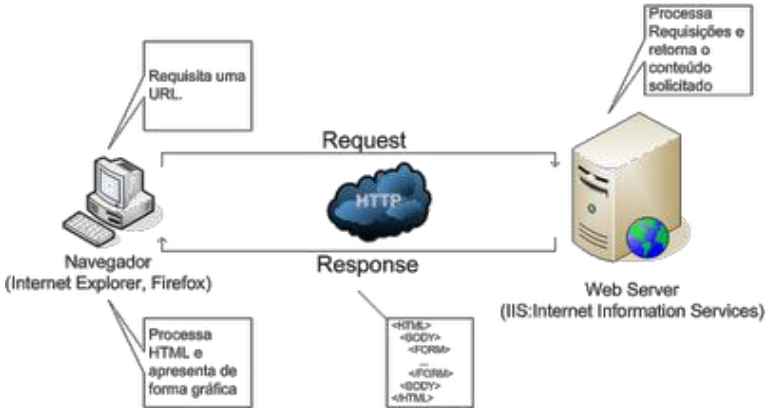


Figura 4. Arquitetura Web [CLIENTE WEB 2008].

Com ambos conectados a provedores de serviços de Internet que utilizam o protocolo TCP/IP (*Transfer Control Protocol / Internet Protocol* ou Protocolo de Controle de Transmissão / Protocolo de Interconexão) responsável pelo envio de dados pela Web, a requisição é feita através do cliente navegador (*Internet Explorer*, *Firefox*, *Google Chrome,* e outros) que envia um pedido via protocolo HTTP (*Hipertext Transfer Protocol* ou Protocolo de Transferência de Hipertexto) ao servidor Web, onde o mesmo processa o pedido e devolve o conteúdo requisitado através do mesmo protocolo uma resposta à solicitação apresentando o conteúdo ao navegador. O Protocolo HTTP é o padrão utilizado para a comunicação entre computadores através da Web, sendo assim indispensável para o funcionamento desta arquitetura [WIKIPEDIA 2013].

3.1.2. Desenvolvimento Web baseado na Linguagem Java

Existem várias linguagens de programação no mercado, mas o que chama atenção na linguagem Java, é que ela independe de plataforma e isso se torna uma vantagem a mais para os desenvolvedores. Possui uma máquina virtual que ajuda nessa independência podendo trabalhar em qualquer sistema operacional. E com essa independência, como hoje o uso da Internet é muito frequente a linguagem Java se torna interessante [CRIARWEB.COM 2004].

3.1.3. Servlets

O principal intuito do Servlet é receber chamadas HTTP, processando e devolvendo uma resposta para o cliente. Na Figura 5 é possível ter uma noção básica do funcionamento desta classe Java:

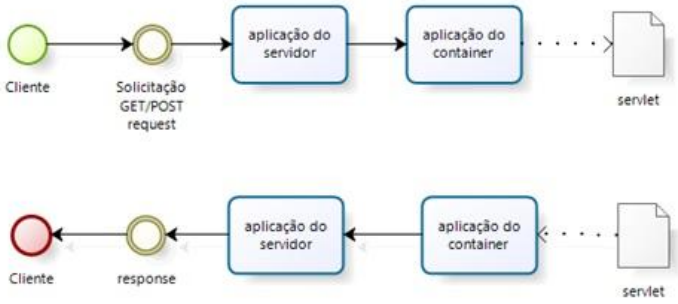


Figura 5. Arquitetura de um Servlet [DEVMEDIA 2013].

É uma classe que traz benefícios como a portabilidade, facilidade de programação, desempenho. Trabalha com a ajuda de um container que auxilia no processamento das requisições, gerencia a memória utilizada, e mais [DEVMEDIA 2013].

3.1.4. Java Server Faces (JSF) e Persistência em Java (JPA)

O Java Server Faces é um framework para desenvolvimento Web baseado em Java e tem como objetivo simplificar o desenvolvimento de interfaces de usuário baseados em Web, possui características que são favoráveis a este conceito como a reutilização de componentes da página, disponibiliza a separação de funções das aplicações envolvidas, permite inserção de comandos em Java Script, aceita a inserção de Folhas de Estilo

(*CSS*) e várias outras funcionalidades. Persistência em Java é uma forma simples e prática de trabalhar com objetos e bancos de dados relacionais, utilizando o mapeamento dos objetos para armazenar os dados em um banco de dados [NETBEANS 2013] [WIKIPEDIA2013].

3.2. Arquitetura Android

A Arquitetura Android é composta pelas seguintes camadas:

- *Applications* (Aplicações) ficam os aplicativos essenciais nativos como navegador, calendário, cliente de email, gerenciador de contatos, fica também os aplicativos desenvolvidos por terceiros como *facebook*, *skype*, aplicativos de bancos. Local que fornece interação direta entre o dispositivo e o usuário.
- *Application Framework* (Framework de Aplicação) ajuda na utilização do reuso de componentes utilizando estas *API's (Application Programming Interface* ou Interface de Programação de Aplicativos) que auxilia os desenvolvedores na criação de aplicativos.
- *Android Runtime* responsável por proporcionar a execução dos aplicativos baseados na plataforma. Conta com o *Core Libraries,* bibiliotecas que trabalha a relação de programação API Java com o Android, e com a *Dalvik Virtual Machine* que libera varias maquinas virtuais separando processos e melhorando o desempenho em temos de memória.
- *Libraries* (Bibliotecas) que fazem parte da plataforma escritas em C/C++ que possuem bibliotecas de renderização 3D, trabalha com arquivos multimídia em diversos formatos, possui o *SQL Lite*, um leve banco de dados relacional, disponibiliza recursos para áudio e vídeo.
- *Linux Kernel* que disponibiliza os serviços essenciais do sistema como gerenciamento de memória, redes, drivers em resumo ele trabalha com serviços de baixo nível.

3.2.1. Componentes de uma Aplicação Android

Figura 6. Componentes de uma aplicação Android [DICAS-L 2011].

Para construir uma aplicação Android é necessário utilizar blocos de construção de componentes nos quais são:

- Activity (Atividades) é a parte onde fica a interface com o usuário, a tela da aplicação, cada activity é independente apesar de trabalharem juntas.
- Service (Serviço) não expõe uma interface para o usuário, roda em segundo plano, sendo possível utilizar uma activity e ao mesmo tempo executar um serviço.
- Content Provider (Provedor de conteúdo) responsável por gerenciar conteúdos de aplicação como, por exemplo, o banco de dados SQL Lite nativo do Android, onde o content provider permite fazer consultas e modificações dos conteúdos destes dados do banco de dados.
- Broadcast Receiver (Receptor de Broadcast) são componentes que ficam inativos e respondem a eventos.
- Android Manifest possui informações importantes e essenciais do aplicativo, como permissões, informações sobre pacotes da aplicação [EMERSON C. LIMA 2012].

3.2.2. Ciclo de vida uma Aplicação Android

Como a classe Activity é considerada a mais importante em uma aplicação Android, responsável por gerenciar a interface do usuário, ela possui um ciclo de vida que será descrito neste momento. Segue o diagrama na Figura 7:

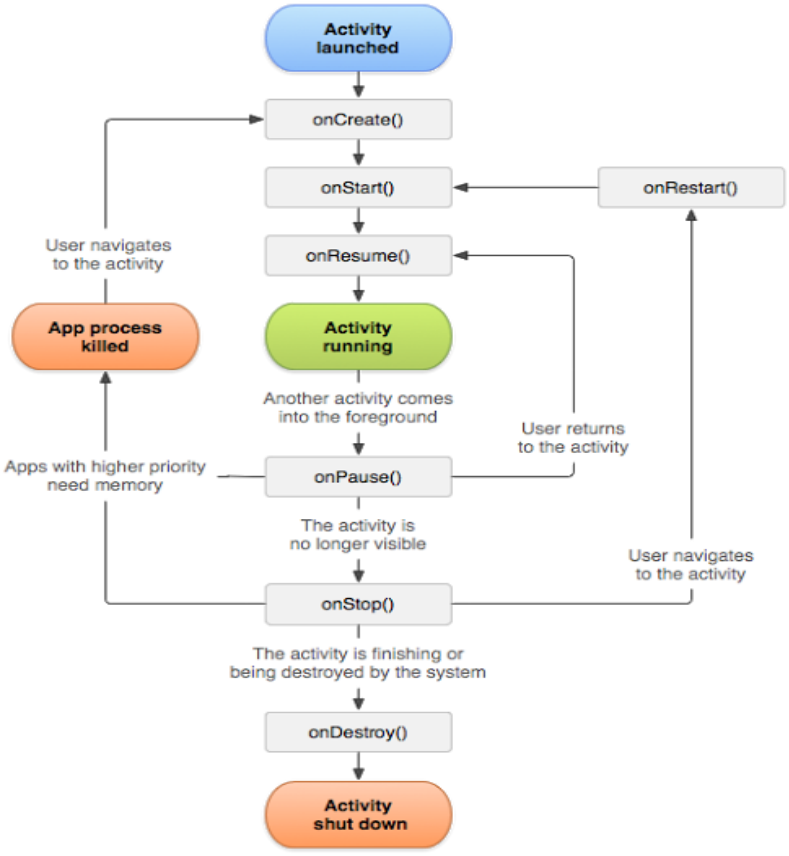


Figura 7. Ciclo de vida de uma Activity [STRANGER DROID 2012].

Métodos da Activity:

- *onCreate()*: responsável por carregar o layout XML da tela e outras operações de inicialização. Executada apenas uma vez no seu ciclo.
- *onStart()*: chamada após o onCreate() e quando não está mais visível na tela e volta a ter o foco.
- *onResume()*: sempre chamada nas retomadas de foco.
- *onPause()*: quando outra Activity é iniciada, ela perde o foco e este método é invocado.
- *onStop()*: chamada quando não é mais visível e encoberta por outra Activity.
- *onDestroy()*: não pode mais ser lançada, considerada terminada.
- *onRestart()*: chamada antes da onStart(), quando o foco retorna a ela depois de estar em segundo plano [FELIPE SILVEIRA 2010].

3.3. API´s para Android

Além das principais *API´s* da camada *Application Framework*, existem várias *API´s* na Internet disponíveis para auxiliar no desenvolvimento de aplicações diversas. Serão citadas somente algumas necessárias para o contexto do artigo que serão utilizadas no desenvolvimento da aplicação que será mostrada no estudo de caso.

3.3.1. API de Localização do Android

A API Location do Android possui classes e interfaces que definem os serviços baseados em localização e serão citadas as mais relevantes.

- *LocationListener* Interface que recebe notificações do *LocationManager* sobre a alteração do local.

  Possui métodos como:

  onLocationChanged, chamado quando existe uma alteração de local.

  onProviderDisabled, chamado quando o provedor está desabilitado pelo usuário.

  onProviderEnabled, chamado quando o provedor está habilitado pelo usuário.

  OnStatusChanged, que é chamado quando há alteração de status do provedor.

  Para que estes métodos do *LocationListener* sejam chamados é necessário o registro com o serviço de localização usando o método *requestLocationUpdates.*
- *Location* Classe que representa uma localização geográfica (latitude e longitude).
- *LocationManager* Classe que fornece o acesso ao serviço de localização do sistema e função de monitorar eventos.
- *LocationProvider* Classe que fornece dados de localização de acordo com critérios pré-estabelecidos [DEVELOPERS ANDROID 2013].

3.4. *Google Maps* usando o *Primefaces*

O Primefaces, que contém uma gama de componentes para *Java Server Faces*, possui alguns componentes construídos pela Google Maps que são de fácil implementação podendo citar o GMap. O Gmap possui recursos como GMap simples, mapa de eventos, polígonos, controles, marcadores e outros. Segue a descrição abaixo de alguns como exemplo:

A GMap Simples é uma classe utilizada para mostrar o mapa com atributos como zoom, centro e tipo.

A classe Marcador (*Markers*) utiliza o *MapModel API* para adicionar marcadores no mapa e possui diversas opções de configuração, opção de ícones personalizados, exibição de nomes [PRIMEFACES 2013].

Estas facilidades garantem aos desenvolvedores uma melhor utilização destes recursos e a possibilidade de uso nas mais diversas áreas que dependem de dados geográficos.

4. Estudo de Caso

Para demonstrar um exemplo de funcionamento do tema proposto juntamente com os conceitos abordados anteriormente, será implementado uma aplicação de forma prática que envia dados de localização de dispositivos móveis com sistema operacional Android (lado cliente) a um servidor Web (lado servidor) que trata os dados recebidos do cliente e disponibiliza as localizações destes dispositivos no mapa via navegador cliente. Estão disponíveis para download todos os arquivos do estudo de caso no seguinte link: *http://code.google.com/p/tcc-hellysson-localizador/*.

A Figura 8 mostra o esquema básico do funcionamento deste exemplo:

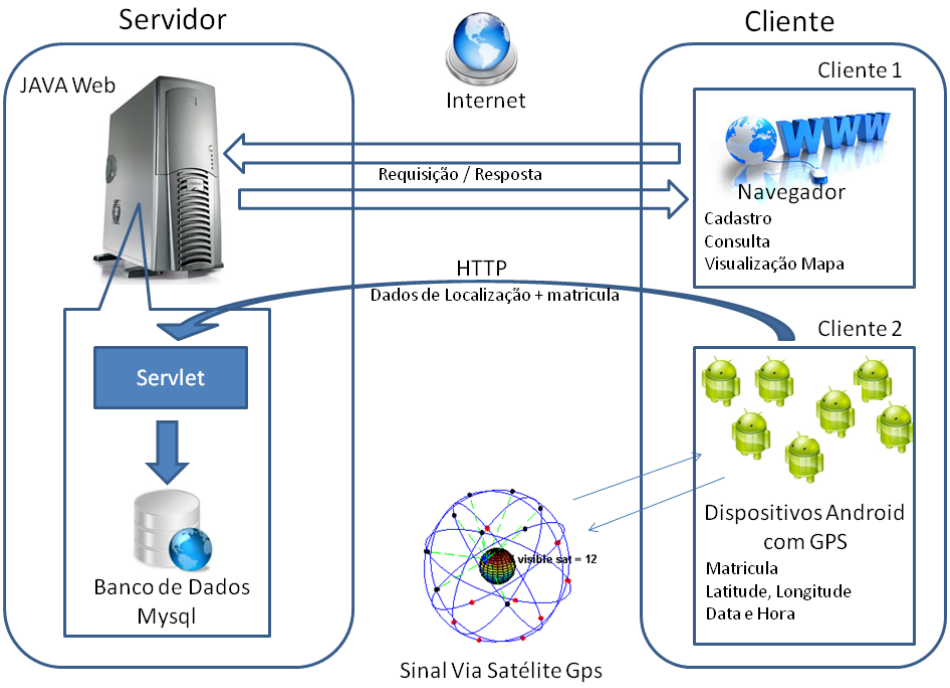


Figura 8. Esquema funcionamento do projeto.

Antes de iniciar o detalhamento do funcionamento deste modelo, é importante salientar que para haver a comunicação entre os dispositivos é necessário ter tanto no cliente como no servidor conexão com provedor de acesso à Internet, pois os protocolos envolvidos dependem desta tecnologia. É necessário também que o dispositivo móvel possua receptor de GPS integrado no aparelho.

Na Figura 8, o esquema mostrado serve como base a uma aplicação de controle de frequência escolar, onde o dispositivo móvel com Android (Cliente2) obtém as coordenadas da localização atual do mesmo, juntamente com a data e hora em que os dados geográficos foram buscados e a matrícula digitada. Os dados são enviados via Internet utilizando o protocolo HTTP para o servlet localizado no servidor Web, que recebe os dados do Android e envia automaticamente os dados para o banco de dados MySql. A partir daí, para consultar os dados do aluno, cadastrar alunos e visualizar a localização dos alunos é utilizado um navegador (Cliente1) que envia pedidos ao servidor que retorna os dados na tela.

Basicamente o funcionamento das aplicações ocorre desta maneira lembrando que os dados recebidos pelo servlet somente são enviados ao banco de dados se a matricula fornecida no dispositivo estiver cadastrada no servidor.

4.1. Ambiente de Desenvolvimento e de Testes

Na Tabela 1, segue a descrição de *Hardwares* e *Softwares* utilizados no desenvolvimento e nos testes.

Tabela 1. Descrição Ambiente de Desenvolvimento e Testes

Desenvolvimento	**Descrição**
Hardware	Processador Intel i5 – M430 CPU 2.27 GHz, 4 GB de RAM (*Random Access Memory* ou Memória de Acesso Randômico), Disco rígido de 500 GB (*Gigabytes*).
Sistema Operacional	Microsoft Windows 7 Professional, 64 bits.
IDE (*Integrated Development Environment* (Ambiente Integrado de Desenvolvimento)	Netbeans IDE 7.3 (Android, Servidor Web e *Servlet*).
SGBD – Sistema Gerenciador de Banco de Dados	MySQL 5.5
Desenvolvimento Android	JDK (*Java Development Kit*) 1.7.0
Adicional	Android SDK Manager Ver. 21.1
Testes	**Descrição**
Hardware	Processador ARM v6 832MHZ(*Mega-hertz*), 289MB de Memória RAM e 180MB(*Megabytes*) de Memória interna.
Sistema Operacional	Android Gingerbread, Firmware 2.3.6.
Dispositivo	Samsung Galaxy Young Duos GT-S6102B

4.2. Desenvolvimento de Aplicações no Android

Para desenvolver aplicações no Android, após instalar o ambiente de desenvolvimento, é necessário definir algumas permissões no arquivo *AndroidManifest.xml* do projeto

para que não ocorram falhas no acesso aos serviços do dispositivo que serão utilizados na aplicação. Parte deste código é mostrada na Figura 9.

```
 <?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
...
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
...
</manifest>
```

Figura 9. Configuração permissões no AndroidManifest.xml.

A classe principal no Android será uma Activity (Atividade) que através do acionamento do Botão pelo usuário inicia os métodos juntamente com a tarefa que será executada em segundo plano. O objetivo da utilização de uma Activity em vez de um serviço totalmente em segundo plano é mostrar o que está acontecendo na aplicação para melhor entendimento do leitor, sendo que poderia a aplicação ser totalmente executada em segundo plano apenas sendo necessária a digitação inicial da matrícula do aluno para que o lado servidor Web receba os dados de acordo com a matrícula. Na Figura 10 é mostrado o trecho principal do código da Activity com o botão.

```
public class MainActivity extends Activity {
...
    Localizacao objLocalizacao = new Localizacao();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        txtMatricula = (EditText) findViewById(R.id.txtMatricula);
        txtMatriculaDigitada = (TextView)
findViewById(R.id.txtMatriculaDigitada);
        txtLatitude = (TextView) findViewById(R.id.txtLatitude);
        txtLongitude = (TextView) findViewById(R.id.txtLongitude);
        txtData = (TextView) findViewById(R.id.txtData);
        txtHora = (TextView) findViewById(R.id.txtHora);
        btnGps = (Button) findViewById(R.id.btnGps);
        btnFinalizar = (Button) findViewById(R.id.btnFinalizar);
        txtResposta = (TextView) findViewById(R.id.txtResposta);

        btnGps.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {

                txtResposta.setText("");
                txtMatriculaDigitada.setText(MostrarMatricula());
                ManterMatricula();
                txtData.setText(RetornaData());
                txtHora.setText(RetornaHora());
                GPS();
                tarefa1();
            }
        });
    }
```

Figura 10. Classe Activity do Android.

Agora será descrito o que faz cada método executado pelo botão da Figura 10:

- MostrarMatricula(): responsável por exibir na tela para o usuário a matricula que ele digitou e automaticamente o Android mantém na memória até que digite uma nova matricula.
- ManterMatricula(): atua com o objetivo de manter a matricula digitada como única até que o serviço seja finalizado.
- RetornaData(): método que retorna a data do sistema em formato texto (String) .
- RetornaHora(): método que retorna a hora do sistema em formato texto(String).
- GPS(): retorna a Latitude e Longitude da localização atual.

Este método utiliza o recurso de geolocalização do Android podendo ser observado o código na Figura 11.

```
public void GPS() {

        LocationManager lManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        LocationListener lListener = new LocationListener() {
            public void onLocationChanged(Location locat) {
                if (locat != null) {
                    Atualiza(locat);
                }

            }

            public void onStatusChanged(String provider, int status, Bundle
extras) {
            }

            public void onProviderEnabled(String provider) {
            }

            public void onProviderDisabled(String provider) {
            }
        };
        lManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
lListener);
    }

    public Double Atualiza(Location location) {

        Double lat = location.getLatitude();
        Double lon = location.getLongitude();
        txtLatitude.setText(lat.toString());
        txtLongitude.setText(lon.toString());

        return null;
    }
```

Figura 11. Código do método GPS().

O método Atualiza() da Figura11 fica por conta de buscar a latitude e longitude quando ele é chamado pela aplicação através do OnLocationChanged(Location Locat), que permite mostrar os dados já atualizados ao usuário na tela.

- Tarefa1() responsável pelo agendamento de tarefas a serem executadas de tempos em tempos, conforme Figura 12.

```
public static final long TEMPO = (200 * 60); // atualiza a cada 12 segundos

    public void tarefa1() {

        Timer timer = null;

        if (timer == null) {

            timer = new Timer();

            TimerTask tarefa = new TimerTask() {

                public void run() {
                    try {

                        executeHttptPostData();

                    } catch (Exception e) {
                        e.printStackTrace();
                        txtResposta.setText(e.getMessage());
                    }
                }

            };

            timer.scheduleAtFixedRate(tarefa, TEMPO, TEMPO);
        }
    }
```

Figura 12. Código do método tarefa1().

Ao definir a variável TEMPO, é possível agendar a tarefa através do timer.scheduleAtFixedRate(nome da tarefa, TEMPO, TEMPO), onde nome da tarefa é o nome dado a função que ativa o Timer, o primeiro TEMPO significa o tempo em milissegundos para iniciar a tarefa e o segundo TEMPO significa o intervalo de tempo para executar o método run(), que no caso da Figura 12 é o mesmo tempo para os dois.

Dentro do método run() encontram-se as tarefas a serem executadas de acordo com a programação do Timer, veremos agora o método responsável pelo envio de dados para o servidor Web que será através do executeHttptPostData(), configurado para ser executado de tempos em tempos ele é o principal componente de comunicação entre o Android e o servidor.

Como o Android envia dados ao servidor Web, o método httppost é utilizado instanciando o endereço (URL) do caminho do servidor. Na sequencia é adicionada uma lista de parâmetros do método POST com os valores em pares com o nome e o conteúdo de cada variável a serem enviadas e por fim é criada uma instância do método httpclient usado para executar o método POST e também a linha de comando que executa este método [FACTORYPATTERN.COM 2007]. Caso ocorra algum erro a exceção é mostrada na tela do dispositivo Android sobre o erro ocorrido.

A Figura 13 mostra o código desta função.

```
public final String URL = "http://192.168.1.100:8080/Localiza/RecebeDados";

 public InputStream executeHttptPostData() {
        HttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(URL);

        objLocalizacao.setMatricula(MostrarMatricula());
        objLocalizacao.setLatitude(txtLatitude.getText().toString());
        objLocalizacao.setLongitude(txtLongitude.getText().toString());
        objLocalizacao.setData(RetornaData());
        objLocalizacao.setHora(RetornaHora());

        String matricula = objLocalizacao.getMatricula();
        String latitude = objLocalizacao.getLatitude();
        String longitude = objLocalizacao.getLongitude();
        String data = objLocalizacao.getData().toString();
        String hora = objLocalizacao.getHora().toString();

        try {
            List nameValuePairs = new ArrayList();
            nameValuePairs.add(new BasicNameValuePair("matricula",
matricula.toString()));
            nameValuePairs.add(new BasicNameValuePair("latitude",
latitude.toString()));
            nameValuePairs.add(new BasicNameValuePair("longitude",
longitude.toString()));
            nameValuePairs.add(new BasicNameValuePair("data",
String.valueOf(data)));
            nameValuePairs.add(new BasicNameValuePair("hora",
String.valueOf(hora)));
            httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
            HttpResponse response = httpclient.execute(httppost);
            return response.getEntity().getContent();

        } catch (ClientProtocolException e) {
            txtResposta.setText(e.getMessage());
        } catch (IOException e) {
            txtResposta.setText(e.getMessage());
        }
        return null;
    }
}
```

Figura 13. Código do método executeHttptPostData().

Na Figura 14, está o resultado do desenvolvimento no Android, sendo que o envio de dados ao servidor não aparece para o usuário, esta tela é somente para o usuário ver os dados que estão sendo utilizados e obtidos na aplicação.

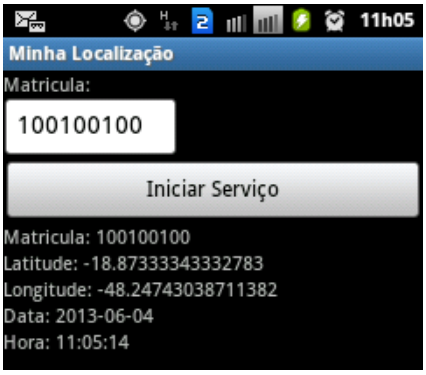


Figura 14. Tela do Dispositivo com Android.

4.3. Desenvolvimento de Aplicações no Servidor Web

Neste passo, serão abordados apenas os códigos com maior foco baseado em geolocalização, sabendo-se que básico para o funcionamento do sistema já foi criado.

Foi implementada no servidor uma aplicação Web que utiliza persistência de dados pra salvar os mesmos no banco de dados MySQL. Seguem na Figura 15 as entidades criadas para atender os requisitos da aplicação.

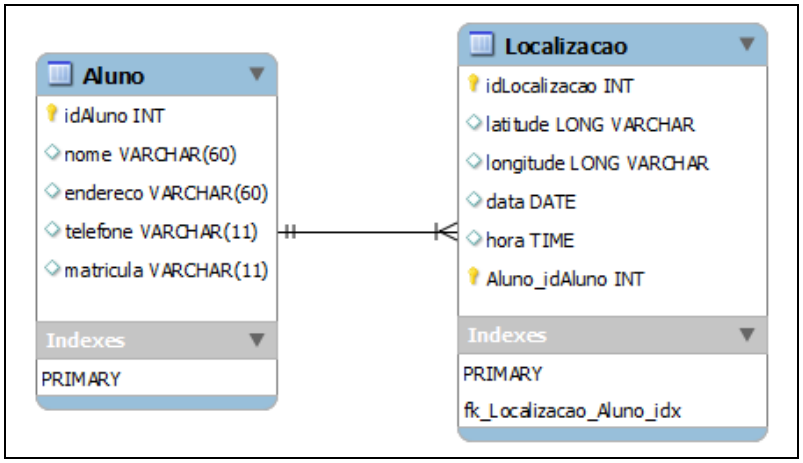


Figura 15. Diagrama entidade relacionamento da Aplicação Servidor.

A partir deste diagrama já foi possível criar o banco de dados, criar dentro do projeto as classes das entidades e os controles (*Beans*) de sessão através do Netbeans que possui componentes que ajudam nesta configuração de forma automática apenas com alguns cliques. No pacote de entidades, concentram-se os atributos das classes Aluno e Localização, e nos controles de sessão encontram-se os métodos que são chamados pela aplicação para executar o CRUD (Create, Read, Update, Delete).

Existem três classes de controle de sessão que são:

- AlunoBean.java, possui os métodos para Ler, Salvar, Excluir e Atualizar o aluno os dados usando Persistência de dados, pois o cadastro de aluno é necessário pra que a matrícula do aluno seja o número de identificação principal dele no programa.
- LocalizacaoBean.java, alguns métodos desta classe serão acessados diretamente pela classe do servlet que será comentado mais adiante.
- MapBean.java, classe do aplicativo que possui os filtros para consulta no banco de dados que mostram os dados das localizações dos alunos na tela.

  Nesta classe o método **mostrarAluno()** traz os registros das ultimas localizações de um aluno específico de acordo com a quantidade de registros e pelo número da matrícula digitadas pelo professor. E existe outro filtro, o método **mostrarUltimoTodos()** que busca na lista de alunos do banco de dados a localização de todos os alunos, mas somente o último registro de cada um.

  A Figura 16 mostra o código deste método.

```
public void mostrarUltimoTodos() {

        simpleModel = new DefaultMapModel();
        EntityManager em = null;
        em = JPAUtil.getEM();

        Query query1 = em.createQuery(" select a from Aluno a ");
        List<Aluno> alunos = query1.getResultList();

        System.out.println("\n\nDados de ultima localização dos Alunos \n");

        for (Aluno dados1 : alunos) {
            for (int i = 0; i < alunos.size(); i++) {

                Aluno alunoBuscado = alunos.get(i);
                int idBuscado = alunoBuscado.getIdAluno();

                Query query = em.createQuery(" select l from Localizacao l
where l.alunoidAluno.idAluno = " + idBuscado + " order by l.data DESC, l.hora
DESC ").setMaxResults(1);
                List<Localizacao> locais = query.getResultList();

                for (Localizacao dados : locais) {
                    Localizacao local = locais.get(0);

                    int idLoc = dados.getAlunoidAluno().getIdAluno();
                    int idAlu = dados1.getIdAluno();

                    if (idLoc == idAlu) {

                        String nome = dados1.getNome();
                        Double lat = dados.getLatitude();
                        Double lng = dados.getLongitude();
                        Date data = dados.getData();
                        Date hora = dados.getHora();

             String novaData = String.format("%1$td/%1$tm/%1$tY", data);
             String novaHora = String.format("%1$tH:%1$tM:%1$tS", hora);

                        LatLng coord1 = new LatLng(lat, lng);
                        simpleModel.addOverlay(new Marker(coord1, " Aluno: " +
nome + " Data: "
                                + novaData.toString() + " Hora: " +
novaHora.toString()));

                        System.out.println("Aluno: " + alunoBuscado.getNome()
+ "\nMatricula: " + alunoBuscado.getMatricula());
                        System.out.println("Latitude: " + local.getLatitude()+
"\nLongitude: " + local.getLongitude());
                        System.out.println("Data: " + novaData + "\nHora: " +
novaHora + "\n");
                    }
                }
            }
        }
        em.close();

    }
}
```

Figura 16. Código de consulta ultimas localizações de todos os Alunos.

A Figura 17 mostra a tela de cadastro de alunos no cliente navegador, onde é feito o cadastro do número de identificação do aluno, a matrícula.

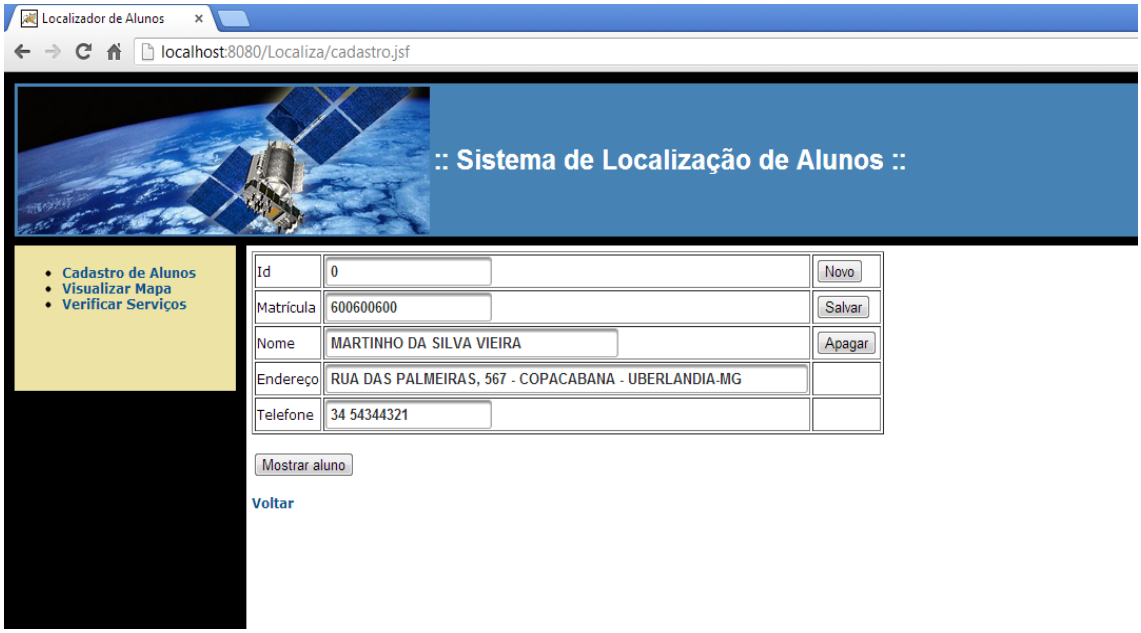


Figura 17. Página JSF de Cadastro de Alunos.

Segue na Figura 18, a imagem da tela de busca do cliente navegador onde as localizações dos alunos são exibidas no mapa de acordo com o tipo de busca efetuada podendo ser tanto individual como coletiva. Mostra a posição do aluno, com o nome data e hora em que ele esteve naquele local.

Figura 18. Página JSF de Busca no Mapa.

4.4. Comunicação entre o Android e o Servidor

A Figura 19 mostra o código do servlet que recebe os dados via POST do cliente Android e os trata verificando através de consulta no banco de dados se a matrícula que chega do dispositivo existe no cadastro de alunos, caso exista os dados são salvos na tabela de localização de acordo com a matrícula encontrada.

```
@WebServlet(name = "RecebeDados", urlPatterns = {"/RecebeDados"})
public class RecebeDados extends HttpServlet {
...
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();

        String matriculaAndroid = req.getParameter("matricula");
        String latitudeAndroid = req.getParameter("latitude");
        String longitudeAndroid = req.getParameter("longitude");
        String dataAndroid = req.getParameter("data");
        String horaAndroid = req.getParameter("hora");

        EntityManager em = null;
        EntityTransaction etx = null;
        try {
            em = JPAUtil.getEM();
            etx = em.getTransaction();
            etx.begin();

            Query query = em.createQuery(" select a from Aluno a");
            List<Aluno> alunos = query.getResultList();

            for (Aluno aluno : alunos) {

                if (aluno.getMatricula().equals(matriculaAndroid.toString()))
{
                    Localizacao loc = new Localizacao();
                    loc.setAlunoidAluno(aluno);

loc.setLatitude(Double.valueOf(latitudeAndroid.toString()));

loc.setLongitude(Double.valueOf(longitudeAndroid.toString()));
                    loc.setData(Date.valueOf(dataAndroid));
                    loc.setHora(Time.valueOf(horaAndroid));
                    em.persist(loc);
                    etx.commit();
                } else {
                    out.print("matricula nao encontrada na base");
                }
            }

        } catch (Exception e) {
            JSFUtil.addMessage("Erro: " + e.getMessage());

        } finally {
            if (em != null) {
                em.close();
            }
        }
        out.close();
    }
}
```

Figura 19. Método RecebeDados() do Servlet Java.

5. Conclusão

Em tempos de evoluções tecnológicas tanto no ambiente acadêmico como no profissional existem diversas maneiras de utilização da geolocalização como meio de auxílio nas atividades diárias.

O Estudo deste artigo demonstrou que os recursos utilizados pelo receptor do GPS de dispositivos móveis juntamente com técnicas e ferramentas de programação já consagradas no mercado de desenvolvimento de softwares, a tecnologia dos mapas e a Internet, contribuem para o enriquecimento de ideias inovadoras, trazendo experiência e incentivando cada vez mais pesquisas no meio tecnológico e cientifico, e mostra também que até mesmo no meio acadêmico esta tecnologia pode ser utilizada e trabalhada de forma a oferecer soluções no controle de frequência escolar.

No decorrer do estudo de caso foram encontrados alguns obstáculos e dificuldades podendo destacar algumas situações:

- Demora do dispositivo móvel para mostrar os dados do GPS (latitude e longitude) devido ao local fechado, sendo mais viável para efeito de teste ter um espaço mais aberto.
- Encontrar um meio de enviar os dados do Android para o servidor através da Internet, sendo necessário conhecer os protocolos de comunicação utilizados para este fim, no caso o HTTP.
- Firewall do Windows e regras do modem dificultando a entrada dos dados para o Servlet, sendo necessário criar regras para a porta utilizada.

Eventos externos como dispositivo móvel desligado, GPS desabilitado, pacote de dados de Internet desativado.

Apesar das dificuldades encontradas, é uma tecnologia que realmente funciona e existem recursos e ideias que ainda podem ser aplicadas e melhoradas neste aplicativo podendo citar como exemplo:

- Criar um controle de presença automatizado, que ao clicar em um botão, o sistema filtra quais alunos está naquele local e horário, e automaticamente consolida a presença dos mesmos e mostra em uma lista ou tabela os alunos presentes.
- Criar relatórios de frequência escolar.
- Efetuar controle de presença separada por turma.
- Utilizar o Banco de Dados nativo do Android para armazenar os dados temporariamente caso ocorra falha no envio no momento da execução da tarefa.

Estas melhorias podem ajudar a deixar o aplicativo mais funcional e mais completo, podendo ser realmente utilizado em ambiente de produção pelos professores para o controle de frequência escolar através dos *smartphones* dos alunos.

6. Referências Bibliográficas

ANDROID TECH. *O que é o Android? Uma visão geral sobre o sistema operacional do Google para smartphones e tablets*. Disponivel em: <http://androidtech.com.br/o-que-e-o-android-2/>. Acessado em: 09 Abril 2013

CANALTECH. *Simon, o primeiro smartphone do mundo, surgia há 20 anos*. Disponível em: <http://canaltech.com.br/noticia/smartphones/Simon-o-primeiro-smartphone-surgia-ha-20-anos/>. Acessado em: 25 Março 2013.

CELULAR ANDROID. *História do sistema operacional Android.* Disponível em: <http://celular-android.com/historia-do-sistema-operacional-android/>. Acessado em: 09 Abril 2013.

CLIENTE WEB. *Arquitetura Web (Navegador, HTML, URL) – Parte 1*. Disponível em: <http://clienteweb.blogspot.com.br/2008/07/arquitetura-web-parte-1.html>. Acessado em: 13 Abril 2013.

CRIARWEB.COM. *O que é Java.* Disponível em: <http://www.criarweb.com/artigos/196.php>. Acessado em: 20 Abril 2013.

DEVELOPERS ANDROID. *Package android.location*. Disponível em: <http://developer.android.com/reference/android/location/package-summary.html>. Acessado em: 01 Maio 2013.

DEVMEDIA. *Conhecendo a tecnologia Servlet.* Disponível em: <http://www.devmedia.com.br/conhecendo-a-tecnologia-servlet/27841>. Acessado em: 01 Maio 2013.

DICIONÁRIO INFORMAL. *Geolocalização*. Disponível em: <http://www.dicionarioinformal.com.br/geolocaliza%C3%A7%C3%A3o/>. Acessado em: 13 Abril 2013.

EDUCA. *Como eram feitos os mapas antigamente?*. Disponível em: <http://www.igeduca.com.br/artigos/desvendamos-misterios/como-eram-feitos-os-mapas-antigamente.html>. Acessado em: 21 Março 2013.

EMERSON C. LIMA. *Componentes de Aplicação.* Disponível em: <http://emersonclima.blogspot.com.br/2012/09/componentes-de-aplicacao.html>. Acessado em: 27 Abril 2013.

FACTORYPATTERN.COM. *How to use Post Method using Apache HttpClient 4*. Disponível em: <http://www.factorypattern.com/how-to-use-post-method-using-apache-httpclient-4/>. Acessado em: 03 Junho 2013.

FELIPE SILVEIRA. *Activity, o que é isso?*. Disponível em: <http://www.felipesilveira.com.br/2010/05/activity-o-que-e-isso/>. Acessado em: 01 Maio 2013.

GFORUM. *O que é um satélite?*. Disponível em: <http://www.geralforum.com/board/543/59356/o-que-e-um-satelite.html>. Acessado em: 21 Março 2013.

GO ANDROID. *"Toma essa, fabricantes"*. Disponível em: <http://letsgodroid.blogspot.com.br/2012/08/toma-essa-fabricantes-desenvolvedores.html>. Acessado em: 26 Março 2013

NETBEANS. *O tutorial do NetBeans E-commerce: Adicionando Classes de Entidade e Beans de Sessão*. Disponível em: <https://netbeans.org/kb/docs/javaee/ecommerce/entity-session_pt_BR.html#jpa>. Acessado em: 13 Maio 2013.

PRIMEFACES. *Google Maps.* Disponível em: <http://www.primefaces.org/showcase/ui/gmapHome. jsf>. Acessado em: 01 Maio 2013.

SHVOONG.COM. *A Evolução dos Mapas*. Disponível em: <http://pt.shvoong.com/exact-sciences/earth-sciences/1854822-evolu%C3%A7%C3%A3o-dos-mapas/>. Acessado em: 20 Março 2013.

SIGNIFICADOS.COM.BR. *Significado de Smartphone.* Disponível em: <http://www.significados.com.br/smartphone/>. Acessado em: 22 Março 2013.

STRANGERDROID. *Entendendo o ciclo de vida de uma Aplicação Android.* Disponível em: <http://blog.strangerdroid.com/2012/09/entendendo-o-ciclo-de-vida-de-uma.html>. Acessado em: 01 Maio 2013.

SUAPESQUISA.COM. *História da Internet.* Disponível em: <http://www.suapesquisa.com/internet/>. Acessado em: 10 Março 2013.

TECHNÈ. *Dalvik: Além, muito além daquela Islândia e daquela Máquina Virtual.* Disponível em: <http://techne.cesar.org.br/dalvik-alem-muito-alem-daquela-islandia-e-daquela-maquina-virtual/>. Acessado em: 31 Maio 2013.

TECMUNDO. *Como funciona o GPS?*. Disponível em: < http://www.tecmundo.com.br/gps/2562-como-funciona-o-gps-.htm>. Acessado em: 13 Abril 2013.

UOL EDUCAÇÃO. *GPS: Sistema de Posicionamento Global tem diferentes utilidades.* Disponível em: <http://educacao.uol.com.br/disciplinas/geografia/gps-sistema-de-posicionamento-global-tem-diferentes-utilidades.htm>. Acessado em: 17 Março 2013.

WIKIPEDIA. *Hypertext Transfer Protocol.* Disponível em: <http://pt.wikipedia.org/wiki/Hypertext_Transfer_Protocol>. Acessado em: 13 Abril 2013.

WIKIPEDIA. *Java Server Faces.* Disponível em: <http://pt.wikipedia.org/wiki/JavaServer_Faces>. Acessado em: 15 Maio 2013.

ZETTEL. *A História do GPS.* Disponível em: <http://www.zettel.com.br/site/index.php?option=com_content&view=article&id=123:a-historia-do-gps&catid=51:nas-trilhas&Itemid=83>. Acessado em: 19 Março 2013.