UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

FERNANDO MEURER

DESENVOLVIMENTO DE UM APLICATIVO PARA A CRIACÃO DE UM DIÁRIO DE LUGARES UTILIZANDO GPS E IMAGENS DA CÂMERA DE DISPOSITIVOS MÓVEIS COM ANDROID

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2012

FERNANDO MEURER

DESENVOLVIMENTO DE UM APLICATIVO PARA A CRIACÃO DE UM DIÁRIO DE LUGARES UTILIZANDO GPS E IMAGENS DA CÂMERA DE DISPOSITIVOS MÓVEIS COM ANDROID

Monografia apresentada como requisito parcial para obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná, Campus Pato Branco.

Orientador: Prof. Robison Cris Brito

PATO BRANCO-PR

2012

ATA Nº: **199**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **FERNANDO MEURER.**

Às 09:00 hrs do dia 11 de outubro de 2012, Bloco S da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Robison Cris Brito (Orientador), Fábio Favarim (Convidado) e Alisson Andrey Puska (Convidado), para avaliar o Trabalho de Diplomação do aluno Fernando Meurer, matrícula 980390, sob o título **Desenvolvimento de um Aplicativo para Criação de um Diário de Lugares Utilizando GPS e Imagens da Câmera do Dispositivo Móvel em Android**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 10:15 hrs foi encerrada a sessão.

Prof. Robison Cris Brito, M.Sc.
Orientador

Prof. Fábio Favarim, Dr.
Convidado

Prof. Alisson Andrey Puska, Esp.
Convidado

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

AGRADECIMENTOS

Ao professor orientador Robison Cris Brito, pela perseverança e a confiança em mim na conclusão deste trabalho, sempre incentivando e motivando, desde o trabalho de estágio até este momento, sempre disponível para ajudar e orientar em tudo o que fosse possível.

A minha família que foi sempre compreensiva e paciente em todos os dias que foram utilizados para desenvolver este trabalho.

A todos os amigos e colegas de trabalho que ajudaram em tudo o que puderam, principalmente nos períodos difíceis.

E a minha namorada, Thais, que não deixou em momento algum que eu pudesse esquecer ou desistir deste trabalho, sacrificando seu tempo e nosso tempo para que pudesse finaliza-lo.

RESUMO

MEURER, Fernando. Desenvolvimento de um aplicativo para criação de um diário de lugares utilizando GPS e imagens da câmera de dispositivos móveis em Android. 2012. 55 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2012.

A interação entre as pessoas acontece desde os primórdios da raça humana. Essa interação nasceu da necessidade de comunicação do homem ao reproduzir sons, a fim de representar seu mundo natural e facilitar a convivência com outros indivíduos. Com o passar do tempo e a evolução dos seres humanos juntamente com o ambiente que os cerca, ferramentas e métodos de comunicação foram sendo criadas e aprimoradas. As tecnologias existentes atualmente proporcionam a capacidade de comunicação das mais variadas formas, tornando qualquer distância praticamente irrelevante, como exemplo a televisão, o rádio, o telefone e a maior de todas as ferramentas de comunicação hoje, a Internet. A fim de elevar o nível de interação e comunicação, o presente trabalho propõe o desenvolvimento de um aplicativo móvel, este baseado no sistema operacional Android, que gerencie um diário de viagens, em que o usuário pode marcar os lugares que já visitou e indexar a este fotos, que posteriormente poderão ser armazenados em um servidor na Internet e compartilhadas com outras pessoas. Para alcançar este objetivo, foram utilizados recursos do Sistema de Posicionamento Global (GPS) e da câmera dos dispositivos móveis, em funcionamento conjunto a uma aplicação *web* que serve apenas como consulta dos dados obtidos, porém com possibilidade de compartilhamento na grande rede. Para compartilhar os dados em um servidor, foi desenvolvida uma aplicação *web* utilizando o *framework* GWT. Como resultado, foi desenvolvido um programa simples e intuitivo, requisitos básicos para aplicativos móveis.

Palavras-Chave: Aplicativo Móvel, Android, Diário de Viagens, GPS, Câmera Digital.

ABSTRACT

MEURER, Fernando. Development of an application to create a diary of places using GPS and images from the camera of mobile devices in Android. 2012. 55 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2012.

The interaction between people happens since the dawn of the human race. This interacion arose from the need of communication from man to play sounds in order to represent their natural world and facilitate interaction with other individuals. With the passage of time and the evolution of humans with the environment that surrounds them, tools and methods of communication were being created and refined. Existing technologies currently provide the ability to communicate in many different ways, making any distance almost irrelevant, for example television, radio, telephone and most of all communication tools today, the Internet. In order to raise the level of interaction and communication, this work proposes the development of a mobile application, this based on the Android operating system, which manages a travel diary, wherein the user can mark the places they have visited and index photos, which can then be stored on a server on the Internet and shared with others. To achieve this goal, the resource was used Global Positioning System (GPS) and the camera of the mobile devices, working together in a *web* application which will serve only as consultation of data, but with the possibility of sharing large network. To share data on a server, a *web* application was developed using the GWT *framework*. As a result, we developed a simple and intuitive program, basic requirements for mobile applications.

Keywords: Internet, Mobile Application, Android, Daily Travel, GPS.

LISTA DE LISTAGENS

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

SDK	*Source Development Kit*
API	*Application Programming Interface*
GWT	Google *Web Toolkit*
XML	*Extensible Markup Language*
WSDL	*Web Services Description Language*
DAO	*Data Access Objects*
SOAP	*Simple Object Access Protocol*
HTTP	*Hypertext Transfer Protocol*
GPS	*Global Position System* (Sistema de Posicionamento Global)
IDE	*Integrated Development Environment*
EPL	*Eclipse Public License*
CPL	*Common Public License*
AJAX	*Asynchonous* Javascript *and* XML
RPC	*Remote Procedure Call*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
HTML	*Hypertext Markup Language*
EXIF	*Exchangeable Image File Format*

SUMÁRIO

1. INTRODUÇÃO

Este capítulo contém as considerações iniciais do trabalho, as quais apresentam uma visão geral do assunto relacionado ao sistema proposto. Os objetivos explicitam as finalidades principais deste trabalho. A justificativa centra-se no tipo de sistema desenvolvido e nas tecnologias utilizadas. Ao final do capítulo, é apresentada a estrutura do trabalho, que apresenta como o texto foi organizado.

1.1 CONSIDERAÇÕES INICIAIS

A interação entre as pessoas acontece desde os primórdios da raça humana. Essa interação nasceu da necessidade de comunicação do homem ao procurar reproduzir sons, a fim de representar seu mundo natural e facilitar a convivência com outros indivíduos. Inicialmente eram usados apelos sonoros e visuais, como o berrante, o gongo e sinais de fumaça, todos obedecendo a um padrão que representava informações relevantes.

Desde então, novas formas de comunicação foram surgindo para facilitar a interação e a comunicação entre as pessoas, mas nada se compara com o advento da informática. As tecnologias existentes atualmente expandiram os horizontes das comunicações e fizeram com que longas distâncias não fossem problemas para promover a comunicação entre as pessoas.

As tecnologias atuais têm sido utilizadas amplamente no nosso cotidiano com o objetivo de reduzir distâncias, como exemplo a televisão, o rádio, o telefone e a maior de todas as ferramentas de comunicação hoje: a Internet.

A Internet provê a possiblidade de comunicação com qualquer pessoa do mundo, entretanto há não muito tempo atrás essa tecnologia era um privilégio de poucos. Atualmente, há um crescimento constante da população que possui acesso a grande rede – Internet - dentro de suas próprias casas e a partir de dispositivos móveis.

No Brasil, houveram crescimentos expressivos entre 2008 e 2012, mais de 24,5 milhões de internautas novos começaram a navegar na rede, um crescimento de quase 45% no período. Além disso, houve crescimento na contratação de Internet Banda Larga em domicílios. Segundo o CETIC.br, em 2010, a utilização de Internet Banda Larga já atingia 21% da população brasileira (SECUNDADOS, 2012).

Entre as principais ferramentas de comunicação disponíveis na Internet hoje, destacam-se as mensagens instantâneas, as redes sociais, a utilização de voz sobre IP, blogs, entre outras.

Entre estas, as redes sociais têm sido muito úteis no propósito de aumentar e dinamizar a interação entre as pessoas e seus contatos, pois proporcionam aos seus usuários, ferramentas que mostrem suas preferências, como perfis públicos, músicas favoritas, filmes preferidos, lugares visitados e frequentados, compartilhamento de fotos pessoais, entre outras.

Com o propósito de aumentar e facilitar a exposição de algumas dessas preferências, como os lugares que uma pessoa frequenta e o compartilhamento de fotos destes, o presente trabalho propõe o desenvolvimento de um sistema móvel, este baseado no sistema operacional Android, em que o usuário pode marcar os lugares que já visitou e indexar a este, fotos, que posteriormente poderão ser postadas em um servidor na Internet e compartilhadas com outras pessoas.

Para utilização deste aplicativo, o usuário pode utilizar seu próprio aparelho celular Android para capturar imagens dos lugares por onde passa e automaticamente registrar o local em que está através do sistema GPS (*Global Position System*) existente nos equipamentos de tecnologia mais novos no mercado, como alguns dos aparelhos celulares com Android.

Utilizando estes recursos, a tarefa de publicação desses dados se torna muito mais fácil e confiável, pois o vínculo da imagem capturada com o lugar ao qual foi obtida é feito automaticamente.

1.2 OBJETIVOS

A seguir são apresentados os objetivos gerais e específicos do trabalho.

1.2.1 Objetivos Gerais

Desenvolver um software que permita a captura de imagens utilizando a câmera do dispositivo móvel e a sua localização global no momento da captura da imagem, através do sistema GPS do aparelho, criando um diário de lugares já visitados.

1.2.2 Objetivos Específicos

Dentre os objetivos específicos do trabalho, destacam-se:

- Estudar Java para o sistema operacional Android, identificando características exclusivas desta plataforma.
- Aplicar o uso de recursos do celular como câmera e GPS para captura de imagens e tratamento de coordenadas geográficas, a fim de visualizar em mapa digital, os lugares onde foram obtidos os dados.
- Integrar informações do dispositivo móvel com o servidor *web* a fim de compartilhar informações na Internet.
- Realizar o desenvolvimento do software que atenda de maneira eficaz, a sua finalidade para posterior comercialização.

1.3 JUSTIFICATIVA

Atualmente existem algumas opções de aplicação que desempenhem a tarefa de registrar os locais por onde um usuário passou e compartilhe suas fotos, de forma simples e rápida. Um sistema similar de diário de lugares é o “Panorâmio” da Google (2012), que proporciona ao usuário a possibilidade de vincular fotos pessoais com lugares. A proposta desse trabalho é de oferecer mais um aplicativo com características próprias para a publicação das fotos.

Um software como o proposto poderia contribuir com a interação entre o utilizador da aplicação e a sociedade da qual participa em vários tipos de ocasiões, sendo que do ponto de vista pessoal, um usuário pode compartilhar suas viagens e locais onde esteve, assim outras pessoas podem usufruir destas informações, verificando os pontos turísticos de locais já visitados. Do ponto de vista profissional, é possível verificar serviços oferecidos em cada local (restaurantes, lojas, etc.), podendo estas informações serem registradas via fotografias, e podendo estes dados serem utilizadas por usuários ou empresas especializadas, como agências de turismo ou pessoas que desejam viajar para o mesmo local.

1.4 ESTRUTURA DO TRABALHO

O trabalho está dividido em 5 capítulos. Destes, o primeiro apresenta a introdução, objetivos e justificativa para a realização deste trabalho.

No Capítulo 2 apresenta uma contextualização teórica sobre os conceitos e características do sistema operacional Android, estes utilizados para o desenvolvimento do aplicativo cliente, bem como do *framework* GWT (Google *Web Toolkit*) que foi utilizado para visualização dos dados no aplicativo servidor, e por fim, apresenta sistemas similares aos propostos pelo presente trabalho.

O Capítulo 3 apresenta os materiais e métodos utilizados para a realização deste trabalho.

O Capítulo 4 apresenta os resultados obtidos, que se resume ao aplicativo cliente, capaz de capturar imagens da câmera do celular e indexá-las a longitude e latitude recuperadas do GPS do aparelho, e o aplicativo servidor, cuja função é apresentar as fotos e o local onde estas foram tiradas.

Finalizando, no capítulo 5 são apresentadas as conclusões, as dificuldades encontradas e as sugestões de trabalhos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho, exibindo as características do sistema Android e também conceitos básicos de seu funcionamento, características do *framework* GWT que é utilizado no sistema *web* desenvolvido para visualização dos dados, e por fim, exemplos de aplicações similares ao software desenvolvido.

2.1 ANDROID

O Android é um sistema operacional para dispositivos móveis baseado em Linux, que revolucionou o mercado mundial de aparelhos celulares por ser a primeira e única plataforma *open-source* existente até hoje no segmento de sistemas operacionais móveis.

A plataforma Android foi criada e é mantida pela Google com a colaboração da Open Handseat Alliance, com a finalidade de acelerar a inovação em dispositivos móveis e oferecer aos consumidores a melhor, mais rica e também mais barata experiência com aparelhos móveis (GARGENTA, 2011).

O Android teve sua primeira versão disponibilizada em Abril de 2009 sob o nome de *Cupcake,* na versão 1.5, seguindo uma lógica alfabética de nomes de sobremesas e bolos nas versões seguintes. Atualmente, a última versão disponível corresponde a 4.1, sob o nome de *Jelly Bean*.

Sua estrutura para funcionamento do sistema operacional consiste basicamente em duas camadas. A primeira camada é constituída por um Kernel Linux, que provê a abstração do hardware do dispositivo móvel, como gerenciamento dos processos do sistema, de memória e manutenção do sistema de arquivos. Na segunda camada há um ambiente de execução de aplicações chamado “Dalvik *Virtual Machine*”, uma máquina virtual que executa as aplicações do sistema e também aplicativos de terceiros, estes desenvolvidos utilizando a linguagem Java (ABLESON, 2012).

A máquina virtual Dalvik foi projetada por Dan Bornstein com contribuições de engenheiros da Google e é otimizada para execuções em dispositivos móveis, pois requer pouca memória e permite que múltiplas instâncias da máquina virtual rodem ao mesmo tempo, permitindo assim o isolamento de processos e fornecendo suporte a *threading*. A estrutura do sistema operacional Android é exemplificada na Figura 1.

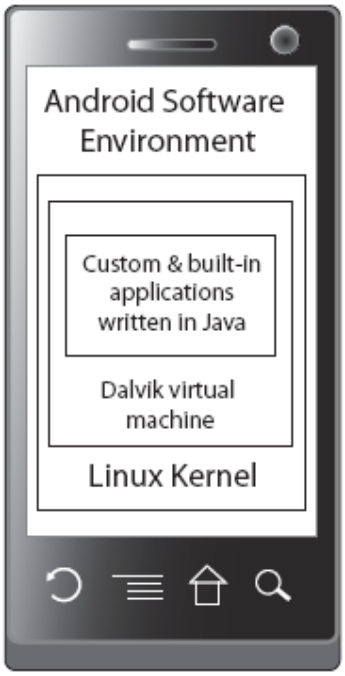

Figura 1 – Estrutura do Sistema Android
Fonte: Ableson (2012, p. 4).

Disponilibiza aos usuários e desenvolvedores uma SDK (*Source Development Kit*) para a criação dos seus próprios aplicativos móveis utilizando a linguagem Java, que é uma linguagem de alto nível, portável para diversas outras plataformas e que é baseada na orientação a objetos.

Uma característica da plataforma Android é que não existem diferenças entre as aplicações e recursos nativos do aparelho em relação às aplicações construídas a partir da SDK. Isso significa que aplicações poderosas podem ser desenvolvidas para serem utilizadas com os recursos disponíveis no aparelho (ABLESON, 2012).

Criar aplicativos para Android garante flexibilidade ao desenvolvedor, dinamismo e a possibilidade de explorar a fundo as capacidades do equipamento, pois permite a utilização sem restrições de qualquer funcionalidade do dispositivo móvel. Todos os recursos de hardware e recursos essenciais do sistema operacional podem ser aproveitados utilizando as API`s (*Application Programming Interface*) nativas do próprio sistema, o que não pode ser feito em outros sistemas operacionais de dispositivos móveis, como o iOS, Windows Mobile, Java ME e Symbian, que bloqueiam a utilização de diversas ferramentas por possuírem código fechado.

2.2 GWT

O GWT é um *framework* de desenvolvimento *web open-source* que consiste em um pacote de ferramentas utilitárias e componentes que tem a finalidade de facilitar a criação de interfaces ricas para desenvolvedores que utilizam a linguagem Java (SMEETS; BONESS; BANKRAS, 2009).

Com o GWT, o desenvolvedor pode utilizar todos os recursos disponíveis a qualquer aplicação Java convencional, como ambientes de desenvolvimento de alta qualidade e depuração do código, contando com um grande benefício até então desconhecido ao desenvolvedor Java: a possibilidade de testar o código em um ambiente emulado que o *framework* dispõe, o qual simula um navegador sem a necessidade de recompilar o código fonte a cada mudança deste, essa facilidade é denominada "*Hosted Mode*" descrita a seguir.

No GWT, existem dois modos de executar uma aplicação, o primeiro é o "*Hosted Mode*", no qual o código Java é executado em um ambiente emulado da mesma forma que seria traduzido para a interpretação de um navegador. Esse modo provê a possibilidade de depurar o código, utilizando recursos como pontos de paradas (*breakpoints)*, podendo eliminar erros do mesmo modo que em um código Java para Desktop, com a exceção de que o resultado será exibido no navegador emulado específico desse modo.

Já em "*Web Mode*", que é o segundo modo de execução disponível, traduz todo o código Java para Javascript e executa efetivamente dentro do navegador, como aconteceria em uma execução normal.

Um dos grandes diferenciais do GWT com relação a outros *frameworks* de desenvolvimento *web* é o modo de funcionamento do seu compilador, que ao compilar o código escrito em linguagem Java, o converte para JavaScript para realizar a execução em um browser. Desse modo, não é necessário ter qualquer conhecimento específico sobre Javascript para trabalhar com GWT, pois é o compilador que se encarrega desta tarefa (HANSON; TACY, 2007).

O compilador GWT trabalha diferente de um compilador Java tradicional em relação ao código a ser compilado, pois ele não compila tudo o que se encontra do descritor de módulo, é incluído somente o que será utilizado. Tal característica permite ao desenvolvedor utilizar grandes bibliotecas com a certeza de que somente as classes e métodos utilizados serão compilados, isso faz que a aplicação tenha a codificação leve,

limpa e evite desperdício de memória e processamento desnecessários (HANSON; TACY, 2007).

2.3 APLICATIVOS MÓVEIS PARA REGISTROS DE LOCAIS EM FOTOS

Tratando-se de aplicações que utilizam recursos de geoprocessamento, armazenamento de fotos e publicação de informações, existem alguns softwares que são populares entre os usuários da Internet, entre esses podem se citar o Panorâmio (PANARAMIO, 2012) e Picasa (PICASA, 2012).

A aplicação Panorâmio, criada pela Google, tem um funcionamento muito parecido com o software proposto por este trabalho. No Panorâmio, qualquer pessoa pode visualizar as fotos enviadas de qualquer lugar do mundo, fotos essas que são enviadas para a aplicação mediante autenticação de usuário, vinculando assim as imagens com os usuários que as postam.

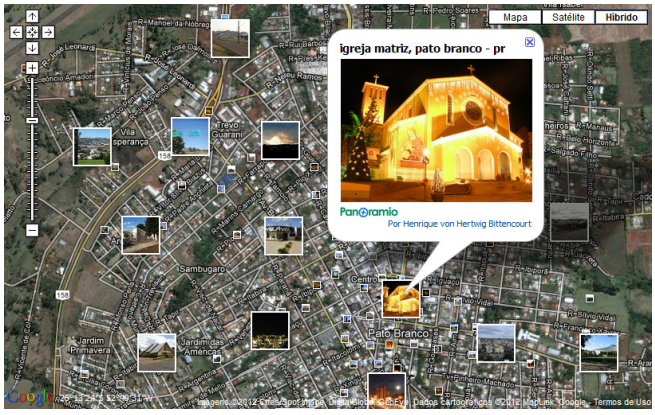

Figura 2 – Exemplo do aplicativo Panoramio.
Fonte: Panoramio (2012).

Outra aplicação que envolve fotos e geoprocessamento é o Picasa, também da Google. Nele, além de visualizar as fotos, há também um jogo que consiste em observar uma sequência de fotos para tentar adivinhar em qual lugar foi capturada, bastando clicar no local desejado no mapa. A resposta é fornecida instantaneamente indicando o local real da foto e também a distância deste local em relação ao ponto marcado pelo usuário.

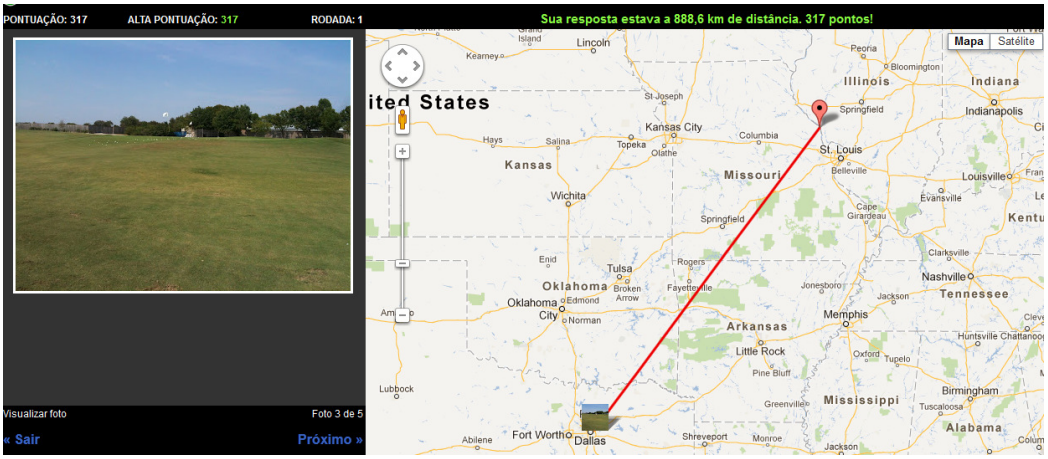

Figura 3 – Exemplo da aplicação Picasa.
Fonte: Picasa (2012).

No sistema proposto, ao capturar as fotos, os dados de geoprocessamento são obtidos automaticamente no mesmo instante da captura e, ao enviar as fotos a partir do celular para o servidor *web* desenvolvido especificamente para a aplicação, os dados de geoprocessamento são enviados junto à imagem e podem ser exibidos no mapa sem necessidade alguma de interação do usuário, estando prontas para serem visualizadas através do mapa.

O envio dos dados da aplicação móvel para a aplicação *web* é uma funcionalidade interessante do aplicativo, pois a praticidade de utilização desse recurso proporciona rapidez na visualização das imagens na aplicação *web*, devido o envio das informações ser realizado com o acionamento simples de um botão.

3. MATERIAIS E MÉTODOS

Este capítulo apresenta as ferramentas utilizadas para a elaboração do aplicativo móvel para diário de viagens, do sistema *web* para exibição das fotos no mapa e do *Web Service* responsável pela comunicação entre o aplicativo móvel e a aplicação *web*. Também é apresentada a metodologia utilizada para atingir o resultado final.

3.1 MATERIAIS

Os materiais e ferramentas utilizadas para desenvolver o aplicativo de diário de viagens foram:

Aplicativo móvel: Android SDK, Banco de dados SQLite e IDE (*Integrated Development Environment*) de Desenvolvimento Eclipse.

Aplicativo Servidor: *Framework* GWT, Banco de Dados MySQL, Ambiente de Gerenciamento Visual de Banco de Dados SQL Front, Servidor de Aplicações *Web* Apache Tomcat.

***Web Service* de Integração:** *Framework* Axis2.

Para o desenvolvimento da aplicação de diário de viagens foi utilizado o sistema operacional Windows 7 (*Seven*) *Professional* e a ferramenta Astah *Community* para análise da aplicação. A utilização das ferramentas citadas acima é apresentada no diagrama da Figura 4:

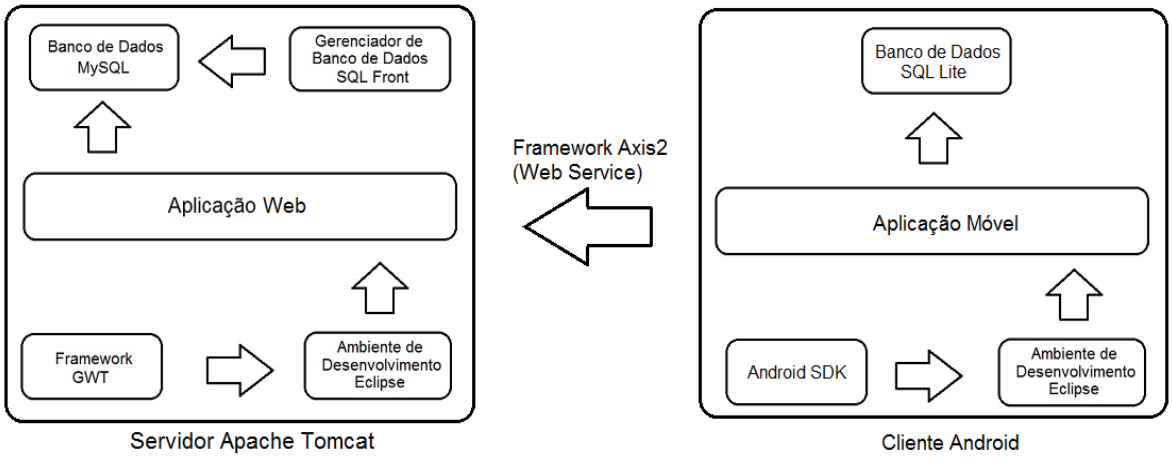

Figura 4 – Esquema dos materiais utilizados

Na aplicação *web,* foi utilizado o *framework* GWT através da IDE Eclipse para realizar o desenvolvimento do *software* web. A aplicação *web* faz o acesso ao banco de dados MySQL para manipulação de dados, sendo esse banco de dados criado utilizando a ferramenta de gerenciamento de banco de dados MySQL Front. O resultado obtido da união das ferramentas mencionadas é executado através do servidor Apache Tomcat.

Na aplicação Android*,* foi utilizado o Android SDK através da IDE Eclipse para realizar o desenvolvimento do *software mobile*. A aplicação móvel faz o acesso ao banco de dados SQLite para manipulação de dados. Por fim o *framework* Axis2 realiza o envio dos dados do aplicativo móvel para a aplicação *web.*

As características e dados relevantes sobre cada ferramenta são apresentadas na sequência:

a) **Android SDK:** O Android SDK (2012), neste trabalho em sua versão 2.2, é uma ferramenta *open-source* que provê bibliotecas e ferramentas de desenvolvimento necessárias para construir, testar e depurar aplicativos para Android. A ferramenta pode ser obtida através do site http://developer.android.com/index.html distribuída sob a licença Apache 2.0.

b) **SQLite:** O SQLite (2012) é um banco de dados transacional nativo do sistema Android que não depende de instalação de servidores e não possui necessidade de configurações. É um banco de dados muito utilizado nas aplicações Android e por grandes corporações como Mozilla e Adobe. Através do site http://www.sqlite.org/ pode ser obtido e utilizado gratuitamente.

c) **IDE Eclipse:** A IDE Eclipse (2012) foi criada pela *International Business Machines* (IBM), com a versão 3.6 utilizada neste trabalho, é um ambiente de desenvolvimento de aplicações feita com a linguagem Java, que permite aos seus utilizadores desenvolverem aplicações em várias linguagens como: C, C++, PHP e Python, disponibilizado sob as licenças CPL (*Common Public License)* e EPL (*Eclipse Public License)*. Este ambiente proporciona ao desenvolvedor, ferramentas úteis de formatação de código, marcação de pontos de parada (*breakpoints*) e outras ferramentas úteis para facilitar o desenvolvimento.

d) ***Framework* GWT:** O Google *Web* Toolkit (2012) é um pacote de ferramentas fornecido gratuitamente através da licença Apache 2.0, com a finalidade de facilitar o trabalho de desenvolvimento de aplicações *web* que possuam interfaces ricas, através de mecanismos próprios que possibilitam a utilização da linguagem Javascript, proporcionando dinamismo a aplicação, utilização de tecnologias que permitam melhor interação de usuário com o sistema, como o AJAX (*Asynchonous* Javascript *and* XML), comunicação rápida e eficiente com o servidor utilizando GWT-RPC, além do já citado Javascript. Para este aplicativo, foi utilizada a versão 2.1.0 do *Framework*.

e) **Banco de Dados MySQL:** O MySQL (2012) é um sistema gerenciador de banco de dados que utiliza a linguagem SQL (*Structured Query Language*) para armazenamento e consulta de dados, sendo a versão 5.1 utilizada nesse trabalho. Tem como principais características o suporte a várias plataformas, ser um software livre que não necessita muitos recursos de *hardware*, além de suporte a várias funcionalidades de grandes sistemas de banco de dados, como *Triggers* e *Stored Procedures*.

f) **Gerenciador de Banco de Dados MySQL Front:** O MySQL Front (2012) é um gerenciador gráfico de banco de dados, de código aberto. Permite a manipulação dos dados com facilidade, sendo possível realizar a criação de tabelas e campos visualmente, bem como demais operações de banco de dados. Caso o MySQL Front não estivesse sendo utilizada, as operações deveriam ser feitas totalmente por linhas de comando.

g) **Servidor *Web* Apache Tomcat:** Apache Tomcat (2012) é um *software open-source,* com suporte à linguagem Java, como as tecnologias JavaServer *Faces* e Java Servlets, e tem a finalidade de interpretar e hospedar aplicações *web* para que sejam publicadas e acessadas através da Internet. O servidor inclui ferramentas para configuração e gerenciamento, o que também pode ser feito editando-se diretamente arquivos em formato XML (*Extensible Markup Language)*. No presente trabalho ele serviu como servidor

de aplicação para a aplicação GWT desenvolvida, bem como servidor para hospedagem do *Web Service* de Integração.

h) ***Framework* Axis2:** O Axis2 (2012) é uma ferramenta de integração ao servidor *web* Apache Tomcat, sendo esta também mantida e desenvolvida pela Apache sob licença Apache 2.0 e utilizada em sua ultima versão, 1.6.2. Através dela, podem ser criados *Web Services* com a publicação de código WSDL (*Web Services Description Language)* gerado automaticamente pelo *framework* ao desenvolver serviços utilizando a linguagem Java.

i) **Astah *Community***: O Astah *Community* (2012) é uma ferramenta que permite a fácil manipulação de todos os modelos de diagramas existentes para modelagem de *software* baseados no padrão UML (*Unified Modeling Language)*. A ferramenta possui recursos que objetivam a praticidade de trabalho, como configuração de atalhos de componentes, bem como o alinhamento automático destes.

3.2 MÉTODO

O desenvolvimento do sistema para registro de fotos e lugares foi dividido em etapas de acordo com o modelo sequencial linear de Pressman (2002). Essas etapas são:

a) **Requisitos:** A definição dos requisitos foi realizada a partir da análise dos aplicativos Panorâmio e Picasa, verificando o modo de utilização dos recursos e como estes são dispostos aos usuários.

b) **Análise**: Os casos de uso foram definidos como forma de representar o problema para o qual seria proposta uma solução. Os casos de uso foram documentados utilizando um editor de textos e a ferramenta Astah *Community* para produzir os diagramas UML.

c) **Projeto**: Os diagramas de classes, definidos com a linguagem UML, foram produzidos utilizando a ferramenta Astah *Community*.

d) **Desenvolvimento**: O desenvolvimento foi realizado utilizando as tecnologias listadas na Seção 3.1 - Materiais.

4. RESULTADOS

Este capítulo apresenta uma visão geral da análise realizada para o sistema bem como sua integração e a utilização de alguns recursos específicos da plataforma Android, como câmera digital e GPS.

Após, é apresentado o sistema desenvolvido, focando nas telas e suas funcionalidades. Por fim, detalhes técnicos de desenvolvimento do aplicativo móvel são detalhados e uma apresentação do aplicativo *web* é apresentada.

4.1 MODELAGEM DO SOFTWARE

Esta seção apresenta a análise do sistema para o diário de viagens, abordando requisitos funcionais e não funcionais que definem as atividades principais do sistema, diagramas de casos de uso e diagramas de classes, a fim de delimitar as operações das aplicações Android e *Web* para melhor entendimento destas e facilitar o desenvolvimento. Foram utilizados apenas estes diagramas, pois a aplicação possui estrutura de banco de dados pequena, em apenas uma tabela de dados para cada aplicação, sendo elas representando os mesmos dados.

4.1.1 Análise da aplicação Android

Os requisitos funcionais e não funcionais são a primeira etapa na análise e modelagem do sistema, através deles torna-se fácil a delimitação de quais funcionalidades o software deve possuir. Estes são apresentados nos Quadros 1 a 5:

F1 Capturar Fotos		**Oculto ( )**		
Descrição: O sistema deve cadastrar fotos capturadas a partir da câmera digital do dispositivo móvel, juntamente com dados de latitude e longitude obtidos através do receptor GPS, e também data e nome de usuário.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF1.1 Identificação do Usuário	O nome do usuário deve ser informado no sistema para realizar captura das fotos	Interface	( )	(X)
NF1.2 Confirmação de Captura	O sistema deve solicitar confirmação de armazenamento das fotos	Interface	( )	(X)
NF1.3 Monitor do GPS	O sistema deve informar o estado e as atualizações de coordenadas do GPS	Interface	( )	(X)
NF1.4 Armazenamento	O sistema deve armazenar os dados em banco de dados	Usabilidade	( )	(X)

NF1.5 Som da câmera	O sistema deve permitir emitir ou não o som do obturador.	Interface	( )	(X)

Quadro 1 – Requisito Funcional Android: Capturar Fotos

F2 Listar Fotos	**Oculto ( )**			
Descrição: O sistema deve listar as fotos capturadas juntamente com os demais dados obtidos.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF2.1 Alteração do usuário	O nome do usuário pode ser alterado pelo usuário	Usabilidade	( )	(X)
NF2.2 Visualização em tela cheia	O sistema deve permitir a exibição das imagens em tela cheia	Interface	( )	(X)
NF2.3 Localização geográfica	Deve ser possível visualizar em mapa o local da captura das fotos.	Interface	( )	(X)

Quadro 2 – Requisito Funcional Android: Listar Fotos

F3 Transmitir Dados	**Oculto ( )**			
Descrição: O sistema deve transmitir as fotos capturadas para um servidor *web* com a finalidade de compartilhar os dados na Internet.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF3.1 Definição do Servidor	O endereço de rede do servidor deve ser definido.	Interface	( )	(X)
NF3.2 Enviar apenas novas fotos	O sistema deve enviar ao servidor *web* apenas dados novos, evitando trafego de rede desnecessário	Eficiência	( )	(X)

Quadro 3 – Requisito Funcional Android: Transmitir Dados

F4 Definição de Configurações	**Oculto ( )**			
Descrição: O sistema deve permitir a configuração e o armazenamento de informações globais para serem utilizadas em todo o contexto da aplicação.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF4.1 Validação das Configurações	A transmissão dos dados e a captura das fotos não podem ser realizadas sem a definição de usuário e endereço de rede.	Usabilidade	( )	(X)
NF4.2 Recurso de som da câmera	O sistema deve permitir retorno de som do obturador conforme opção escolhida pelo usuário.	Interface	( )	(X)

Quadro 4 – Requisito Funcional Android: Definição de Configurações

F5 Visualizar Informações do Sistema	**Oculto ( )**
Descrição: O sistema deve possuir uma tela de informações do sistema, no formato “Sobre”.	
Não há requisitos não funcionais.	

Quadro 5 – Requisito Funcional Android: Visualizar Informações do Sistema

A partir da obtenção dos requisitos funcionais e não funcionais, foi elaborado o diagrama de casos de uso, conforme exibido na Figura 5:

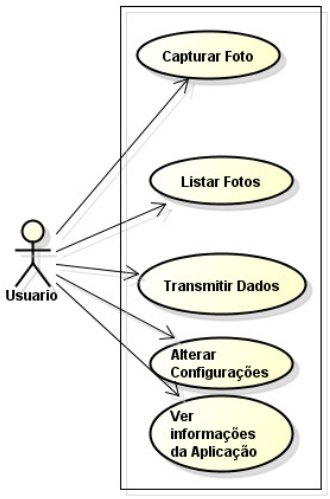

Figura 5 – Diagrama de casos de uso da aplicação Android

Por fim, é apresentado o diagrama de classes referente à aplicação Android na Figura 6, que contém a representação dos atributos e métodos utilizados para atingir o resultado final do sistema.

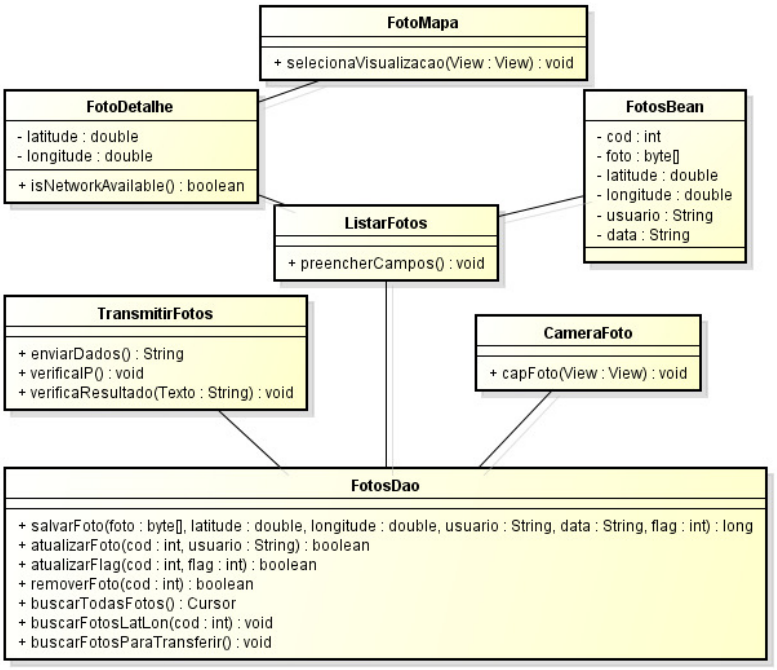

Figura 6 – Diagrama de classes da aplicação Android

O Quadro 6 apresenta a estrutura da única tabela utilizada no programa Android. Esta tabela armazena os registros referente as fotos tiradas pela aplicação.

Nome do Campo	Tipo de Dado
Cod	Int
Foto	Blob
Latitude	Double
Longitude	Double
Data	Date
Usuário	String
Flag	Int

Quadro 6 – Estrutura da tabela do banco de dados da aplicação Android

4.1.2 Análise da Aplicação *Web*

Seguindo a mesma ordem realizada na Seção 4.1.1, são exibidos no Quadro 7 os requisitos funcionais e não funcionais para a aplicação *web*.

Quadro 7 – Requisito Funcional GWT: Listar Fotos

F1 Listar Fotos		**Oculto ( )**		
Descrição: O sistema deve listar as fotos capturadas juntamente com os demais dados obtidos.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF1.1 Filtragem do usuário	A lista de dados pode ser filtrada por nome de usuário.	Usabilidade	( )	(X)
NF1.2 Exibição no Mapa	Ao clicar em um item, a localização geográfica é exibida no mapa	Interface	( )	(X)

A Figura 7 e 8 apresentam os diagramas de casos de uso e de classes, respectivamente nessa ordem:

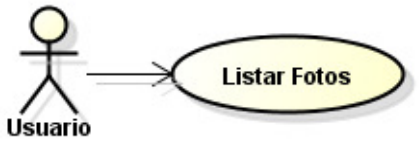

Figura 7 – Diagrama de casos de uso da aplicação *web*

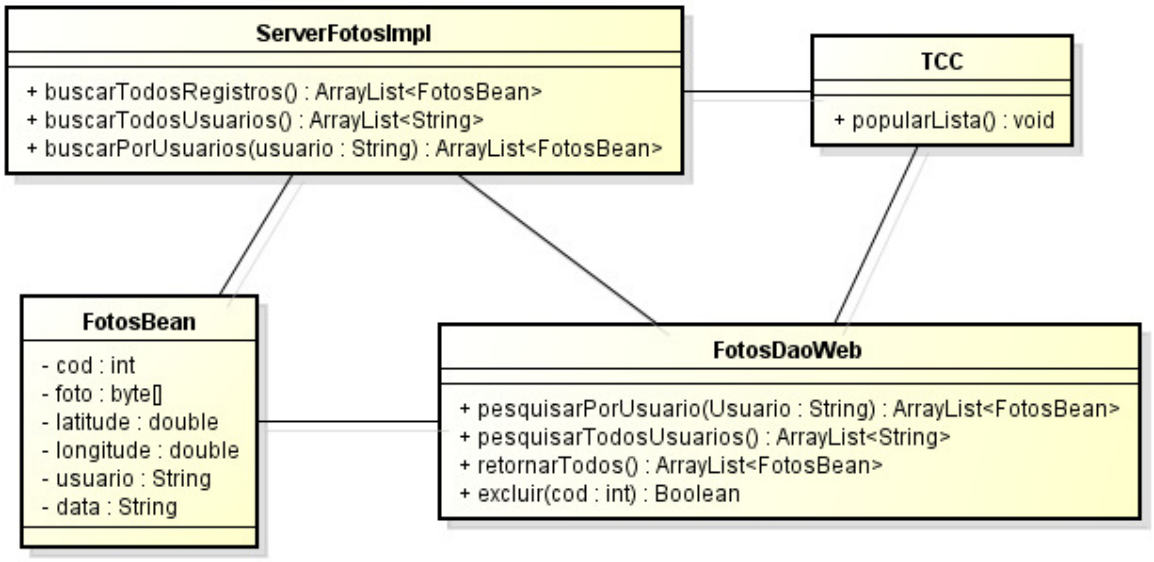

Figura 8 – Diagrama de classes da aplicação *web*

Finalizando esta seção, na Quadro 8 é apresentada a estrutura do banco de dados da aplicação *web*. Nota-se a similaridade com a tabela do banco de dados da aplicação Android, exibida na Quadro 7, pois a finalidade da aplicação *web* é apenas de compartilhar na grande rede, os mesmos dados existentes no aplicativo móvel.

Nome do Campo	Tipo de Dado
Cod	Int
Foto	Blob
Latitude	Double
Longitude	Double
Data	Date
Usuário	String

Quadro 8 – Estrutura da tabela do banco de dados da aplicação *web*

4.2 DESENVOLVIMENTO DO SISTEMA MÓVEL

Para desenvolvimento do sistema móvel, foi utilizada como base a versão 2.2 do Android, sendo esta chamada de “Froyo”. A escolha desta versão foi feita devido à versão do aparelho móvel utilizado para testes do aplicativo possuir esta versão instalada e também porque os *smartphones* comercializados hoje com o sistema Android dificilmente possuem uma versão inferior a esta instalada. O *smartphone* utilizado foi o LG Optimus ME P350, apresentado na Figura 9, que possui todos os recursos abordados nesse trabalho.

Figura 9 – Smartphone LG Optimus ME P350

Na programação para Android, qualquer versão superior do sistema em relação à versão da qual o aplicativo foi desenvolvido é suportada, sendo assim, a aplicação cliente desenvolvida nesse trabalho funcionará em qualquer aparelho com versão igual ou superior à versão 2.2.

Para criação das interfaces visuais de cada item do menu principal e de cada tela do sistema, foi utilizado um assistente visual proporcionado pela própria SDK que é baseada em XML. Através desse assistente, basta arrastar os componentes desejados para a área disponível e organizá-los de acordo com a necessidade enquanto o código XML é gerado automaticamente. A criação das telas é exemplificada na Figura 10.

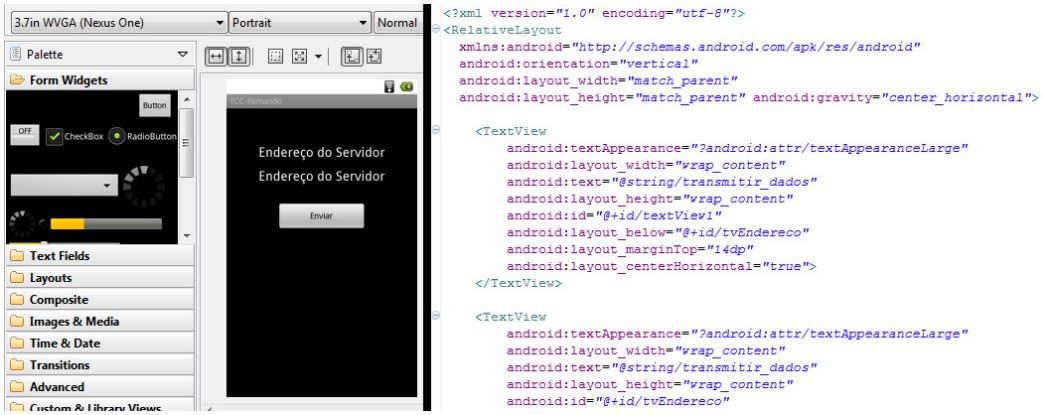

Figura 10 – Assistente de criação de layout das telas

Para a utilização dos recursos de GPS e câmera do dispositivo móvel, é necessário adicionar no arquivo “AndroidManifest”, existente em todos os aplicativos desenvolvidos para Android, uma solicitação de permissão de uso do recurso. Dessa forma, quando um aplicativo for instalado em um aparelho móvel com Android, será informado quais recursos serão usados e a instalação somente será concluída com a confirmação do usuário.

Esse método de segurança evita que aplicações mal intencionadas utilizem recursos do aparelho sem o conhecimento do proprietário, sendo exemplificada na Figura 11.

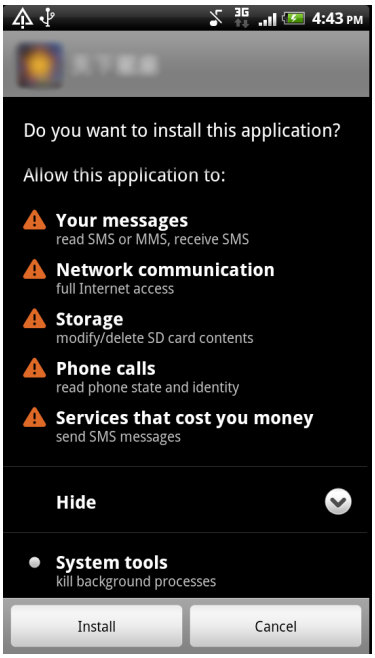

Figura 11 – Tela de confirmação de instalação de aplicativos

4.2.1 Apresentação do cliente Android

O aplicativo móvel possui cinco itens em seu menu principal, sendo o primeiro item para a captura de fotos, que no momento em que a foto é obtida, dados da latitude e da longitude são recebidas pela funcionalidade de GPS do dispositivo. Esse conjunto de dados é armazenado em banco dados existente no próprio sistema do dispositivo móvel. O segundo item do menu exibe a listagem de fotos disponíveis no banco de dados local com seus respectivos dados, que são eles: latitude, longitude, data e usuário, sendo a informação de usuário apenas de texto simples, pois não há validação ou controle de usuários. As imagens podem ser visualizadas em tela inteira e também é possível ver o ponto exato do local da captura da imagem no mapa.

A terceira opção do menu principal é referente a transferência dos dados do aparelho celular para o sistema *web*. Nesta tela, o usuário informa o endereço de rede da aplicação *web* e os dados são transferidos utilizando Wi-Fi ou conexão de dados da operadora de telefonia. Somente são enviadas as fotos que ainda não estão na aplicação *web*, evitando assim tráfego de rede desnecessário. A quarta opção tem a função de ajustar configurações gerais do aplicativo móvel, como nome padrão de usuário e o endereço de rede para transferência dos dados. Por fim, a ultima opção apresenta algumas informações sobre o sistema.

Ao iniciar o aplicativo através do dispositivo móvel, a tela principal é apresentada. Nela, encontram-se as funcionalidades do sistema representadas pelos menus, sendo elas: “Tirar Foto”, “Listar Fotos”, “Transmitir Dados”, “Configurações” e “Sobre”, conforme ilustrado na Figura 12. As funcionalidades desta figura são abordadas com maiores detalhes no transcorrer deste capítulo.

Figura 12 – Tela principal do sistema

Ao acessar através da tela principal o item “Tirar Foto”, o aplicativo ativa a câmera do celular, apresentando para o usuário a imagem em tempo real. O software apresentado foi desenvolvido para que atualize as coordenadas de onde o usuário está localizado a cada 20 segundos, e a cada atualização, uma mensagem é apresentada na tela, informando o utilizador que sua posição está atualizada, conforme exemplifica a Figura 13,

sendo a apresentação desta e também da Figura 14 com a imagem quadriculada ao fundo devido a uma limitação do emulador em exibir imagens reais, pois o emulador não fornece suporte para exibir a imagem da câmera do computador utilizado no desenvolvimento.

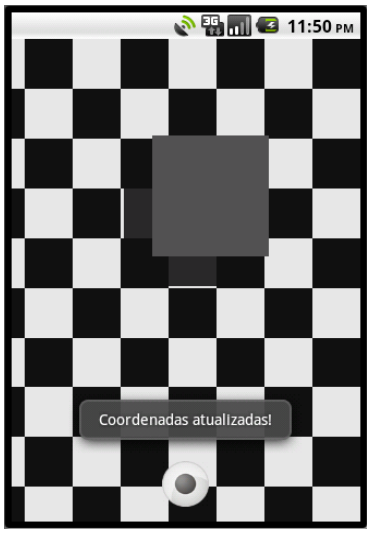

Figura 13 – Atualização das coordenadas GPS na aplicação

Quando o usuário tira a foto, ele pode optar por descartá-la ou salvá-la no dispositivo. No caso de optar por salvá-la, uma mensagem é apresentada ao usuário, informando que sua foto foi salva e qual é a latitude e longitude de onde a mesma foi tirada, conforme ilustra a Figura 15.

Figura 14 – Momento da captura da imagem na tela de fotos

O item “Listar Fotos” da tela principal, quando acessado, apresenta ao usuário as fotos por ele tiradas, mostrando, ao lado de cada foto, informações referente a quando (data e hora), onde (latitude e longitude) e por quem (usuário) estas fotos foram tiradas. A Figura 15 exemplifica a listagem de fotos, embora por possuir a limitação do emulador em não permitir a exibição de imagens a partir da câmera do computador, ao serem capturadas imagens a partir do emulador, estas recebem uma imagem padrão como resultado.

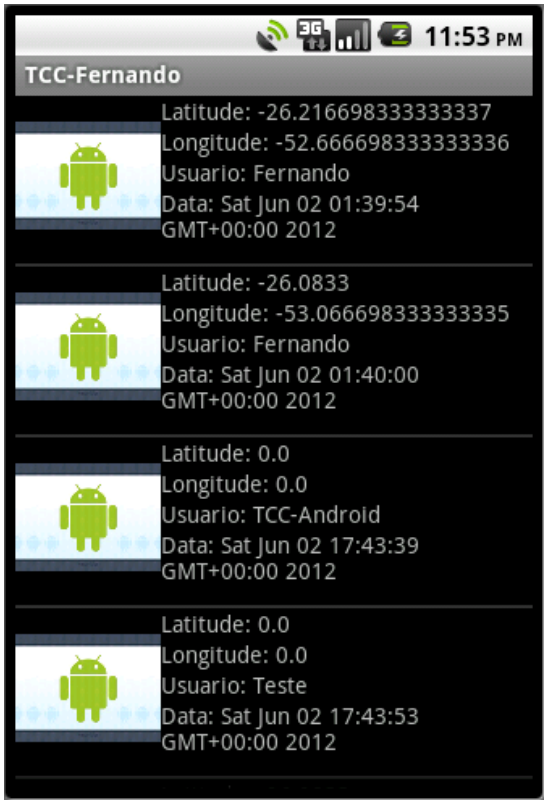

Figura 15 – Tela de listagens das informações

Na tela de listagem de fotos, ao selecionar uma foto e segurar o botão de contexto do dispositivo pressionado, uma tela de ação é apresentada. Esta tela permite ao usuário apagar o registro em questão, ou alterar o usuário vinculado a foto, conforme exemplificado na Figura 16.

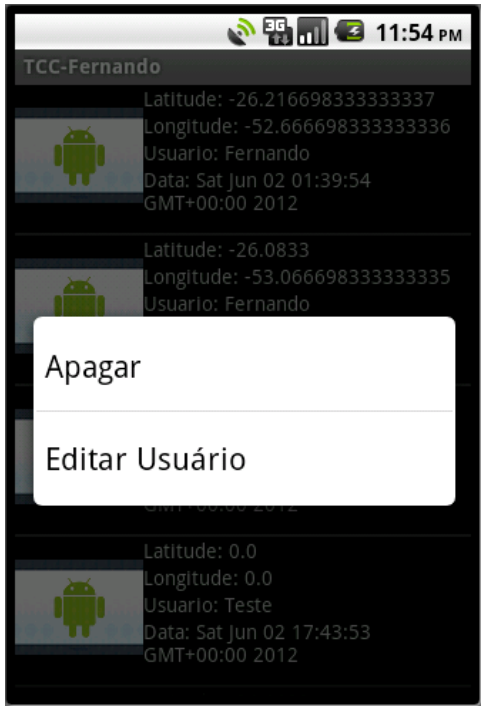

Figura 16 – Menu de Contexto da lista de dados

Ainda na tela de listagem de fotos, ao selecionar uma das fotos, ela é exibida em tela cheia, ocupando toda a tela do dispositivo. Nesta tela, ao pressionar o botão de menu do aparelho, uma tela de ação é apresentada, conforme ilustra a Figura 17.

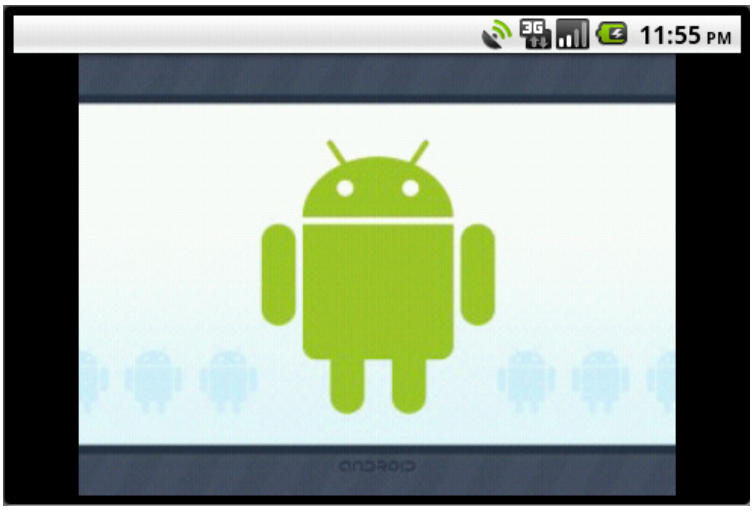

Figura 17 – Exibição da imagem capturada

Figura 18 – Menu de visualização da localização em mapa

Esta tela apresenta o botão “Ver no Mapa”, que ao ser clicado, mostra onde a foto foi tirada no mapa, com as opções de apresentar o mapa na forma de fotos de satélites ou através de mapas, conforme mostra a Figura 18. A exibição da localização no mapa é apresentada na Figura 19.

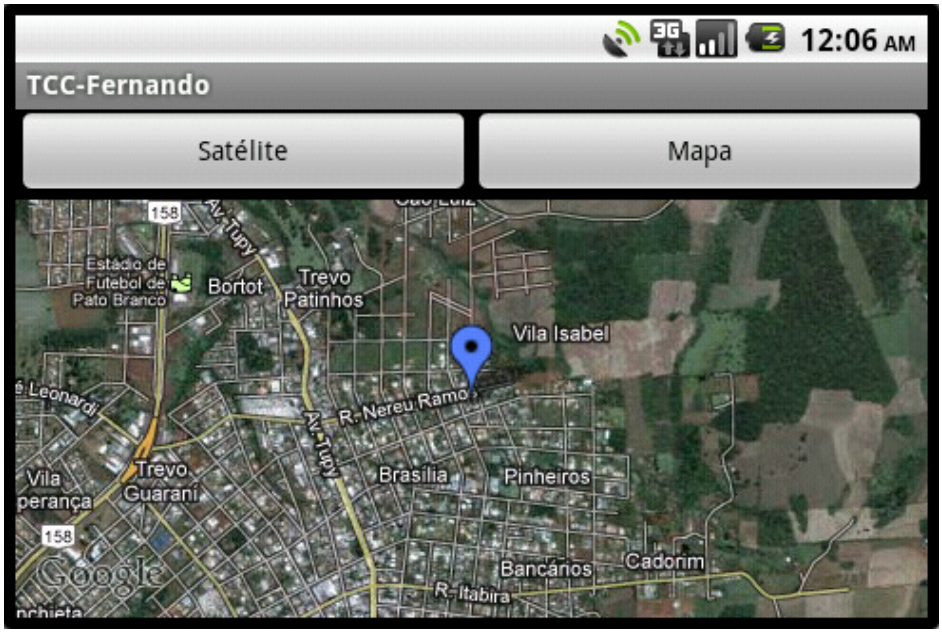

Figura 19 – Localização da foto no mapa

Acessando o terceiro item da tela principal - Transmitir Dados, uma nova tela é apresentada. Nela, é informado para qual servidor as fotos e suas respectivas informações serão enviadas e ainda nesta tela, ao pressionar o botão menu, uma tela de ação apresenta o botão “Alterar IP do Servidor”. Selecionando este botão, e possível digitar um novo endereço IP, este, por sua vez, será o novo servidor. Ao selecionar o botão “Enviar”, as fotos e suas informações serão enviadas para o servidor definido, conforme ilustra a Figura 20.

Figura 20 – Tela de alteração de endereço do servidor e envio dos dados

Ao acessar o menu “Configuração”, uma nova tela é mostrada em que é possível alterar variáveis do sistema, como o nome do usuário, endereço IP padrão para o servidor e habilitar ou desabilitar ou som do obturador da câmera, conforme ilustra a Figura 21.

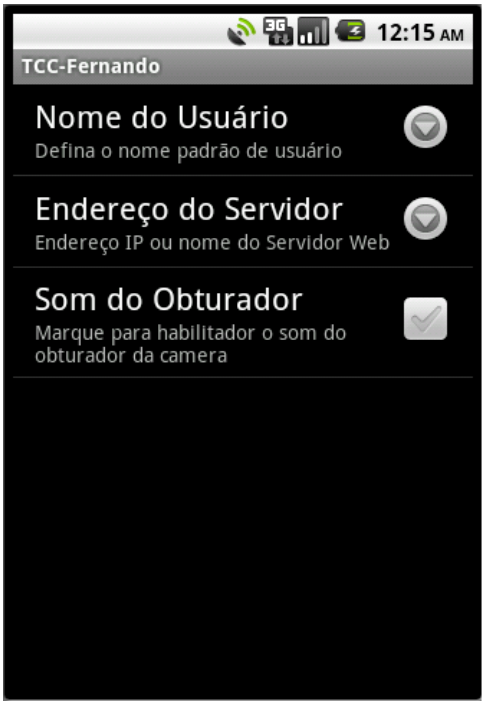

Figura 21 – Tela de configurações

O quinto e último item da tela principal, referenciado como “Sobre”, apresenta informações sobre o desenvolvedor do aplicativo, como ilustra a Figura 22.

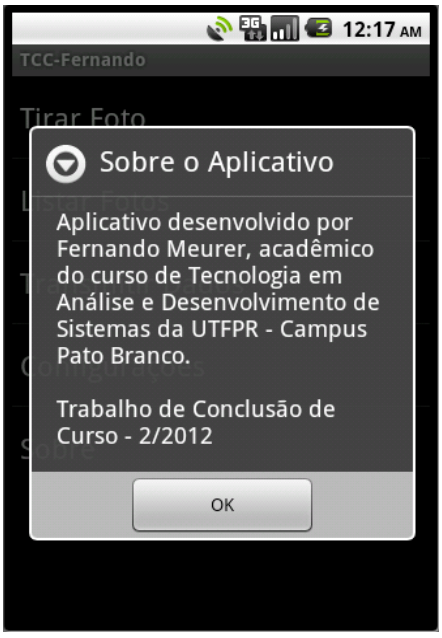

Figura 22 – Tela de informações do sistema

A seguir serão apresentadas as soluções criadas, juntamente com seus respectivos códigos, que tornaram o aplicativo móvel funcional.

4.2.2 Captura da Foto

Ao criar a interface visual responsável pela captura das imagens através do assistente visual, foi inserido na programação um objeto da classe “SurfaceView”, que consiste em um simples *frame* dimensionável em tempo de execução, sendo esta a área onde será mostrada a imagem que estiver sendo obtida através da câmera. Dessa forma, juntamente com a classe “Camera”, que é instancida com o método “Camera.open()”, devem ser criados objetos das classes “SurfaceView” e “SurfaceHolder” no código a ser desenvolvido.

Embora esteja criado o objeto principal do recurso, os objetos do tipo “SurfaceView” e “SurfaceHolder” são necessários para mudar atributos de orientação, sendo o primeiro para receber a instância direta do componente da interface visual e o segundo é o responsável direto pelo monitoramento das ações geradas pelo primeiro. Tal monitoramento é importante, pois a configuração da orientação padrão da imagem é invertida, podendo ser percebida com uma simples rotação do aparelho móvel. Esta falha acontece nos eixos de 0º - 90º e 180º - 270º, que é solucionada com um *listener* para verificar a atual rotação da câmera e corrigi-la, conforme mostrado na Listagem 1.

```
1.      Camera.Parameters parametros = camera.getParameters();
2.      List<Camera.Size> previewSizes = parametros.getSupportedPreviewSizes();
3.      Camera.Size pSize = previewSizes.get(0);
4.
5.      parametros.setPreviewSize(pSize.width, pSize.height);
6.
7.      if (getResources().getConfiguration().orientation == Configuration.ORIENTATION_PORTRAIT) {
8.        parametros.set("orientation", "portrait");
9.      } else if (getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE) {
10.       parametros.set("orientation", "landscape");
11.     }
12.
13.     int rotation = getWindowManager().getDefaultDisplay().getRotation();
14.     int degrees = 0;
15.
16.     switch (rotation) {
17.       case Surface.ROTATION_0: degrees = 90; break;
18.       case Surface.ROTATION_90: degrees = 0; break;
19.       case Surface.ROTATION_180: degrees = 270; break;
20.       case Surface.ROTATION_270: degrees = 180; break;
21.     }
22.
23.     camera.setDisplayOrientation(degrees);
24.     parametros.setRotation(degrees);
```

Listagem 1 – Solução da orientação da imagem da câmera

Com a classe câmera, o desenvolvedor pode executar diversas funções utilizando métodos simples, como ampliação e redução de zoom, iniciar e pausar a visualização da câmera, definir a orientação da imagem baseada em graus e várias outras funções.

Após a correta definição do objeto “SurfaceView” e do ajuste da orientação da imagem, basta invocar o método startPreview() da câmera e a imagem será exibida.

Para realizar a captura da foto é necessário um objeto do tipo “Camera.PictureCallback”, que é uma interface da classe “Camera” responsável por fazer tratamentos de imagem após evento de captura de fotos, para ser atribuído no método takePicture() da câmera, que foi chamado a partir de um evento de botão que foi adicionado na interface visual. O objeto do tipo “Camera.PictureCallback” possui um método abstrato chamado onPictureTaken(), para que a imagem capturada seja manipulada, sendo neste caso, armazenada em banco de dados local (SQLite) junto com demais informações, como as coordenadas de GPS, data e usuário. O processo de captura da foto é exemplificado na Listagem 2.

```
1.      Camera.PictureCallback pictureCallback = new Camera.PictureCallback() {
2.
3.      @Override
4.      public void onPictureTaken(final byte[] foto, Camera camera) {
5.
6.        AlertDialog.Builder dialogConfirmacao = new AlertDialog.Builder(CameraFoto.this);
7.        dialogConfirmacao.setMessage("Deseja salvar essa imagem?").setCancelable(false);
8.
9.        dialogConfirmacao.setPositiveButton("Sim", new DialogInterface.OnClickListener() {
10.
11.       @Override
12.       public void onClick(DialogInterface dialog, int which) {
13.         fotosDao.salvarFoto(foto, ultimaLoc.getLatitude(),
14.         ultimaLoc.getLongitude(), preferences.getString("nomeUsuario", ""), new Date().toString(),
0);
15.         Toast.makeText(getBaseContext(), "Imagem salva: Lat: " + ultimaLoc.getLatitude()
16.           + " Lng: " + ultimaLoc.getLongitude(), Toast.LENGTH_SHORT).show();
```

Listagem 2 – Codigo da captura das fotos

4.2.3 Obtenção das coordenadas do GPS

Para a obtenção das coordenadas de GPS, são utilizados dois objetos nativos da própria SDK chamados “LocationManager” e “LocationListener”, destinados ao tratamento deste recurso. O primeiro é responsável por gerenciar o serviço do sistema GPS, que permite adicionar *listeners* de status do serviço, alertas de proximidade de pontos de referência ou pontos especificados através de coordenadas e o principal recurso deste objeto, que é a requisição de atualização da localização através do método requestLocationUpdates(). Este método permite definir o tempo mínimo das atualizações

de localização e também, se desejado, que as atualizações só ocorram após percorrer uma distancia mínima.

Nesse método, é definido e implementado o segundo objeto que é um *listener* para o tratamento dos dados obtidos pelo GPS e do estado do recurso, sendo possível estar desativado, ativado ou ter sofrido outros tipos de alteração. A Listagem 3 mostra a instanciação dos objetos e a codificação destes.

```
1.      locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
2.      locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 20000, 0,
locationListener);
3.
4.      locationListener = new LocationListener() {
5.
6.        @Override
7.        public void onStatusChanged(String provider, int status, Bundle extras) {
8.          // TODO Auto-generated method stub
9.        }
10.
11.       @Override
12.       public void onProviderEnabled(String provider) {
13.         Toast.makeText(getBaseContext(), "Receptor GPS Ativo!", Toast.LENGTH_SHORT).show();
          }
14.
15.       @Override
16.       public void onProviderDisabled(String provider) {
17.         Toast.makeText(getBaseContext(), "Receptor GPS Desativado!", Toast.LENGTH_SHORT).show();
          }
18.
19.       @Override
20.       public void onLocationChanged(Location location) {
21.         ultimaLoc = location;
22.         Toast.makeText(getBaseContext(), "Coordenadas atualizadas!", Toast.LENGTH_SHORT).show();
23.       }
24.     };
```

Listagem 3 – Codigo da obtenção dos dados do GPS

Ainda codificado na Listagem 3, a cada atualização recebida pelo GPS, é chamado o método onLocationChanged(). Neste método é utilizada uma variável do tipo “Location” que recebe as coordenadas de latitude e longitude atualizadas, sendo esta a variável que será utilizada no momento de salvar os dados no aparelho móvel.

Essa foi a abordagem de obtenção dos dados utilizada no sistema, embora a captura das coordenadas possa ser feita unicamente no momento da captura da foto através do método getLastKnownLocation(), o qual recupera as últimas coordenadas obtidas pelo receptor GPS. Nas formas de tratamento de dados relatadas acima, podem ocorrer divergências da localização real do dispositivo móvel em relação à localização armazenada na variável que contém a ultima localização, pois as atualizações só acontecem em locais onde o receptor GPS consegue se comunicar com o satélite. Assim, pode haver períodos de tempo indeterminados sem receber atualizações, e no momento da captura da foto, a última atualização ser diferente da posição real.

Este problema não pôde ser tratado devido a versão 2.2 da SDK Android não possuir nenhum modo de realizar requisições singulares de coordenadas como pode ser feito na versão 2.3, através do método requestSingleUpdate(), que recupera os dados de latitude e longitude no momento em que é chamado, podendo assim ser utilizado no instante da captura da foto.

4.2.4 Persistência no Banco de Dados

A persistência dos dados foi realizada utilizando o banco dados nativo do sistema Android chamado SQLite. Para a criação da tabela do banco de dados responsável por armazenar os dados da aplicação foi utilizada uma classe da própria aplicação, em que foram definidas as *Strings* de criação das tabelas, como mostrado na Listagem 4.

```
1.      private static final String DATABASE_NAME = "fotos";
2.      private static final int DATABASE_VERSION = 1;
3.      private static final String DATABASE_CREATE = "create table if not exists "+DATABASE_NAME+"
(_id integer primary key autoincrement, "
4.        + "foto blob not null, latitude float, longitude float, usuario text, data date, flag
integer);";
5.
6.      public FotosDatabase(Context context) {
7.        super(context, DATABASE_NAME, null, DATABASE_VERSION); }
8.
9.      @Override
10.     public void onCreate(SQLiteDatabase database) {
11.     database.execSQL(DATABASE_CREATE); }
```

Listagem 4 – Strings de criação do banco de dados

O momento da criação da tabela acontece através de uma classe DAO (*Data Access Objects*) criada no projeto, pois no momento em que o objeto da classe implementada na Listagem 4 é instanciada na classe DAO, o método onCreate() é executado, criando assim o banco de dados.

Após o processo de criação do banco de dados for realizada, este já está funcional e pronto para receber dados. A partir desse momento, a persistência de dados e recuperação destes é muito similar a qualquer aplicação Java tradicional. Nessa aplicação, a classe DAO possui todos os métodos referentes à inclusão, alteração e exclusão de registros da tabela.

Para persistir os dados utilizando o banco SQLite, deve ser criado e valorizado um objeto do tipo “ContentValues” com todas as informações desejadas, e então executar a operação inserindo este objeto como parâmetro. A exemplificação desta abordagem é exibida na Listagem 5.

```
1.      public long salvarFoto(byte[] foto, double latitude, double longitude, String usuario, String
data, int flag) {
2.        ContentValues initialValues = criarValores(foto, latitude, longitude, usuario, data, flag);
3.        return database.insert(TABELA_BD, null, initialValues);
4.      }
5.
6.      public boolean atualizarFoto(int cod, String usuario) {
7.        ContentValues updateValues = new ContentValues();
8.        updateValues.put(FOTOS_USUARIO, usuario);
9.        return database.update(TABELA_BD, updateValues, FOTOS_COD + "=" + cod, null) > 0;
10.     }
```

Listagem 5 – Exemplo de manipulação de dados no banco de dados

4.2.5 Exibição das Imagens

Existe no SDK Android um grupo de componentes chamados *Adapters*, que mais são assistentes de manipulação de dados. Quando é necessário preencher uma lista de dados obtida através de banco de dados, sendo que há mais de um registro nessa tabela, as aplicações Java tradicionais precisam fazer um tratamento especial dessa tarefa passando registro a registro para destinar os dados a seus respectivos campos, comumente utilizando laços de repetições que requerem mais processamento do computador.

Ao desenvolver aplicações para Android, essa tarefa é mais fácil, pois os *Adapters* funcionam como um grupo fechado de dados. Neles, basta definir o conjunto total de registros em forma de *array* sem necessidade de tratamento direto dos dados, pois o tratamento é realizado por métodos internos do *Adapter*. A única tarefa necessária é criar uma classe separada para identificar quais componentes de entrada de texto da interface visual vão receber cada tipo de dados.

No aplicativo foi utilizado o *Adapter* do tipo “ListAdapter”. Para o funcionamento correto deste, deve ser adicionado a um componente do tipo “ListView” na interface visual, em que serão exibidas as informações e atribuir seu nome como “list”. Na classe que está sendo codificada a listagem dos dados, deve ser herdada a classe “ListActivity”, que é a responsável por buscar na interface visual o componente “list”. Nesse momento, a classe já está vinculada na tela visual do sistema bastando, através do método setListAdapter(), definir qual é o vetor de dados e qual a classe que possui as informações de quais campos visuais vão receber os dados.

Em relação à classe de identificação dos campos, como em qualquer outra funcionalidade do sistema desenvolvido que requer a busca de componentes visuais, o Android possui um tratamento diferenciado desses componentes, pois a chamada não é feita diretamente no componente, mas sim em uma classe gerada automaticamente

quando é feita no momento da construção e execução da aplicação, chamada “R”. Essa classe gera um endereçamento hexadecimal para cada componente global do aplicativo, sendo nesta classe que serão buscadas as vinculações com os componentes visuais. A codificação da classe de vinculação de dados e o método de captura dos componentes visuais são exemplificados na Listagem 6.

```
1.      public FotoArray(Activity context, List<FotosBean> dados) {
2.        super(context, R.layout.fotos_linha, dados);
3.        this.context = context;
4.        this.dados = dados;
5.      }
6.
7.      static class ViewFotos {
8.        public ImageView ivFoto;
9.        public TextView txtLatitude;
10.       public TextView txtLongitude;
11.       public TextView txtUsuario;
12.       public TextView txtData;
13.     }
14.
15.     @Override
16.     public View getView(int position, View convertView, ViewGroup parent) {
17.       ViewFotos fotos = null;
18.       View rowView = convertView;
19.       if (rowView == null) {
20.         LayoutInflater inflater = context.getLayoutInflater();
21.         rowView = inflater.inflate(R.layout.fotos_linha, null, true);
22.
23.         fotos = new ViewFotos();
24.         fotos.ivFoto = (ImageView) rowView.findViewById(R.id.imagem);
25.         fotos.txtLatitude = (TextView) rowView.findViewById(R.id.txtLatitude);
26.         fotos.txtLongitude = (TextView) rowView.findViewById(R.id.txtLongitude);
27.         fotos.txtUsuario = (TextView) rowView.findViewById(R.id.txtUsuario);
28.         fotos.txtData = (TextView) rowView.findViewById(R.id.txtData);
29.         rowView.setTag(fotos);
30.       } else {
31.         fotos = (ViewFotos) rowView.getTag();
32.       }
33.
34.       FotosBean bean = (FotosBean) dados.get(position);
35.
36.       byte[] fotoByte = bean.getFoto();
37.       Bitmap bitmapImage = BitmapFactory.decodeByteArray(fotoByte, 0, fotoByte.length);
38.
39.       fotos.ivFoto.setImageBitmap(bitmapImage);
40.       fotos.txtLatitude.setText("Latitude: " + bean.getLatitude());
41.       fotos.txtLongitude.setText("Longitude: " + bean.getLongitude());
42.       fotos.txtUsuario.setText("Usuario: " + bean.getUsuario());
43.       fotos.txtData.setText("Data: " + bean.getData());
44.
45.       return rowView;
```

Listagem 6 – Vinculação de dados de lista e recuperação de componentes visuais

Juntamente com exibição da imagem no aparelho móvel, foi inserida a funcionalidade de visualizar o ponto da captura da foto utilizando o mapa digital da Google chamado Google Maps (2012). Por serem ambas ferramentas desenvolvidas pela Google: o Android e o Google Maps, estes possuem uma integração muito simples de serem utilizadas, bastando o desenvolvedor declarar no arquivo “AndroidManifest” a utilização da biblioteca respectiva ao Google Maps, sendo esta obtida através da própria página da

desenvolvedora, mediante a aceitação de um contrato gratuito de utilização da biblioteca em aplicações terceiras.

Para exibir o marcador da posição geográfica no mapa, foi criada uma classe chamada “Marcador”, responsável por criar o item. Juntamente a este, foi utilizado um objeto do tipo “OverlayItem”, que é o objeto que carrega a imagem do marcador, sendo este atribuído ao marcador através do método addOverlay().

4.2.6 Configurações utilizando *Shared Preferences*

Na grande maioria dos softwares, é necessário armazenar informações e configurações que possam ser utilizadas em todo o escopo da aplicação. Normalmente, o método utilizado para armazenar tais dados é através de persistência em banco dados para então recuperá-los quando necessário. Utilizando o Android, essa tarefa torna-se mais fácil, pois é possível utilizar um recurso chamado *Shared Preferences*, assim toda a atividade de persistência dos dados no banco e posterior recuperação são substituídas por um tratamento que fica sob a responsabilidade do Android.

Com *Shared Preferences*, basta criar a tela de configurações através do assistente visual, utilizar componentes do tipo “Preference” e a cada um deles, atribuir um nome único como identificação do campo sendo que este nome pode ser recuperado em todo o ambiente do software. Esse tratamento das configurações de dados globais realiza o armazenamento das informações em arquivos XML dentro da própria pasta do aplicativo no dispositivo móvel.

Na Listagem 7 pode ser visto que o tipo de componente utilizado não foi um “EditText” tradicional, mas sim um “EditTextPreference”, que é definido o título do componente. Também foi definido um comentário para auxílio na utilização de usuários e por fim, o atributo mais importante definido na tag “android:key”, em que se atribui o nome com o qual a recuperação dos dados poderá ser feita.

Foi utilizado também um componente do tipo “CheckBoxPreference” que possui a mesma metodologia de uso, permitindo recuperar as informações armazenadas automaticamente pela aplicação, apenas referenciando o nome-chave definido no interface visual na tag “android:key”. A Listagem 7 mostra o funcionamento da criação da tela de preferências, da definição da palavra-chave utilizada para obter o valor armazenado e os demais atributos pertinentes.

```
1.      <?xml version="1.0" encoding="utf-8"?>
2.      <PreferenceScreen
3.        xmlns:android="http://schemas.android.com/apk/res/android"
4.        android:orientation="vertical"
5.        android:layout_width="match_parent"
6.        android:layout_height="match_parent">
7.
8.        <EditTextPreference
9.         android:title="Nome do Usuário"
10.        android:summary="Defina o nome padrão de usuário"
11.        android:key="nomeUsuario"
12.        android:defaultValue=""
13.       />
14.
15.       <EditTextPreference
16.        android:title="Endereço do Servidor"
17.        android:summary="Endereço IP ou nome do Servidor Web"
18.        android:key="ipServidor"
19.        android:defaultValue=""
20.       />
21.
22.       <CheckBoxPreference
23.        android:title="Som do Obturador"
24.        android:summary="Marque para habilitador o som do obturador da camera"
25.        android:key="somCamera"
26.       />
27.
28.     </PreferenceScreen>
```

Listagem 7 – Criação da tela de configurações

Após realizada a criação da interface visual, basta acessar a tela criada quando o aplicativo estiver em execução para alterar as preferências sem precisar se preocupar com o modo como esses dados são tratados. A recuperação dos dados requer a utilização de um objeto do tipo *Shared Preferences*, que é valorizado através do método getDefaultSharedPreferences() existente na classe “PreferenceManager”. A partir desse momento, a variável passa a atuar como um vetor de dados, que pode ter seus valores obtidos através do método getString(), definindo nesse método o nome do componente que foi definido no momento da criação da interface visual.

Essa abordagem de utilização de configurações é uma característica importante exclusiva do sistema Android, pois o trabalho do desenvolvedor é reduzido drasticamente, otimizando o tempo e o processamento da aplicação.

4.2.7 Comunicação com o *Web Service*

A transferência das informações armazenadas no dispositivo móvel para o servidor *web* foi feita utilizando *Web Service*. Para realizar a comunicação, foi adicionada às bibliotecas do aplicativo uma nova biblioteca chamada kSOAP2 (2012), responsável por consumir *Web Services* a partir do sistema Android utilizando protocolo SOAP (*Simple Object Access Protocol)*, que é um protocolo simples de acesso a objetos, normalmente

transmitido através do protocolo HTTP (*Hypertext Transfer Protocol*) em formato XML, como é o caso deste aplicativo.

Depois de adicionada a biblioteca no projeto, chamada “ksoap2-j2se-full-2.1.2.jar”, foi criado um objeto do tipo “SoapObject” no qual serão inseridos os dados a serem enviados. Para realizar o envio, é necessário encapsular esse objeto em um “SoapSerializationEnvelope”, que será o objeto transportado, o qual receberá através do método setOutputSoapObject() o objeto do tipo “SoapObject” definido anteriormente. Por fim, foi utilizado um objeto do tipo “HTTPTransportSE” que faz a invocação do *Web Service* através do método call(), passando por parâmetro o “SoapSerializationEnvelope”. Neste momento os dados já estão em tramitação e um novo “SoapObject” é criado para receber a resposta do servidor, sendo este objeto devidamente tratado na continuidade da aplicação. O processo completo de definição dos objetos relatados e a comunicação é exibida na Listagem 8.

```
1.      SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
2.      request.addProperty("cod", cursor.getInt(cursor.getColumnIndex(FotosDao.FOTOS_COD)));
3.
4.      ByteArrayOutputStream bao = new ByteArrayOutputStream();
5.      byte[] foto = cursor.getBlob(cursor.getColumnIndex(FotosDao.FOTOS_FOTO));
6.      Bitmap fotoBitmap = BitmapFactory.decodeByteArray(foto, 0, foto.length);
7.      fotoBitmap.compress(Bitmap.CompressFormat.JPEG, 80, bao);
8.
9.      byte[] byteAux = bao.toByteArray();
10.
11.     String fotoPronta = Base64.encodeToString(byteAux, Base64.DEFAULT);
12.
13.     request.addProperty("foto", fotoPronta);
14.     request.addProperty("latitude",
cursor.getString(cursor.getColumnIndex(FotosDao.FOTOS_LATITUDE)));
15.     request.addProperty("longitude",
cursor.getString(cursor.getColumnIndex(FotosDao.FOTOS_LONGITUDE)));
16.     request.addProperty("usuario",
cursor.getString(cursor.getColumnIndex(FotosDao.FOTOS_USUARIO)));
17.     request.addProperty("data", cursor.getString(cursor.getColumnIndex(FotosDao.FOTOS_DATA)));
18.
19.     SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
20.     envelope.dotNet=false;
21.     envelope.setOutputSoapObject(request);
22.
23.     HttpTransportSE ht = new HttpTransportSE("http://" + preferences.getString("ipServidor", "") +
24.                     ":8080/WebServiceFotos/services/TransmiteFotos");
25.
26.     try {
27.       ht.call(SOAP_ACTION, envelope);
28.       SoapObject resposta = (SoapObject) envelope.bodyIn;
29.
30.       if (resposta.toString().contains("false") || resposta.toString().equals("") || resposta ==
null) {
31.         return "Falha no envio, confira sua conexão!";
32.       } else {
33.         fotosDao.atualizarFlag(cursor.getInt(cursor.getColumnIndex(FotosDao.FOTOS_COD)), 1);
34.         flag = "OK";
35.       }
36.     } catch(Exception e) {
37.       e.printStackTrace();
38.       return e.getMessage();
39.     }
```

Listagem 8 – Comunicação da aplicação cliente com o *Web Service*

O método de recepção do envelope no lado servidor apenas faz conexão com o banco de dados da aplicação servidor e realiza a persistência dos dados. Dessa forma, os dados já estão prontos para serem exibidos na *web*. O objeto retornado para o cliente Android é do tipo *boolean* apenas informando se a inserção no banco de dados foi realizada com sucesso ou não.

4.2.8 Execução e depuração do aplicativo no dispositivo móvel

Para desenvolvimento desse aplicativo foi utilizado um *smartphone* LG Optimus ME P350 com a versão 2.2 do Android instalado, a fim de realizar testes reais da aplicação, principalmente pelo fato do emulador que o SDK Android não possuir suporte a emulação da câmera para exibir imagens e também serem possíveis apenas envios fictícios de localizações do sistema de posicionamento global - GPS.

A execução e depuração do aplicativo podem ser feitas instalando corretamente os drivers do *smartphone* fornecidos pela fabricante e habilitando no aparelho a função de depuração USB, sendo esta uma opção disponível em todas as versões do Android. Com os drivers instalados, basta executar o aplicativo através da IDE, o aparelho móvel será detectado e o ambiente de desenvolvimento pedirá se deseja executar a aplicação usando o aparelho ou um emulador virtual.

Também usando o aparelho móvel, é possível realizar a depuração de código da mesma forma como é feito no emulador, adicionando pontos de parada (*breakpoints*) ao código, com a finalidade de encontrar erros de programação.

Esse modo de execução da aplicação é muito útil para o desenvolvedor, pois permite solucionar vários possíveis problemas que através do emulador não seriam descobertos, como os ajustes de layout ou até mesmo falhas dos recursos da câmera e do GPS, por exemplo.

4.3 SERVIDOR GWT

Ao acessar a página *web* do aplicativo servidor, as fotos enviadas pelos usuários da aplicação móvel são apresentadas, sendo possível filtrá-las pelo nome do usuário. Estas fotos ficam localizadas na parte esquerda da página, enquanto a parte direita reserva-se a ilustrar o mapa, sendo possível vê-lo na forma de mapas ou representado por fotos de satélites, conforme exemplifica a Figura 23.

Ao clicar em uma foto, a localização de onde está foto foi tirada é indicada no mapa. A Figura 24 representa esta ação.

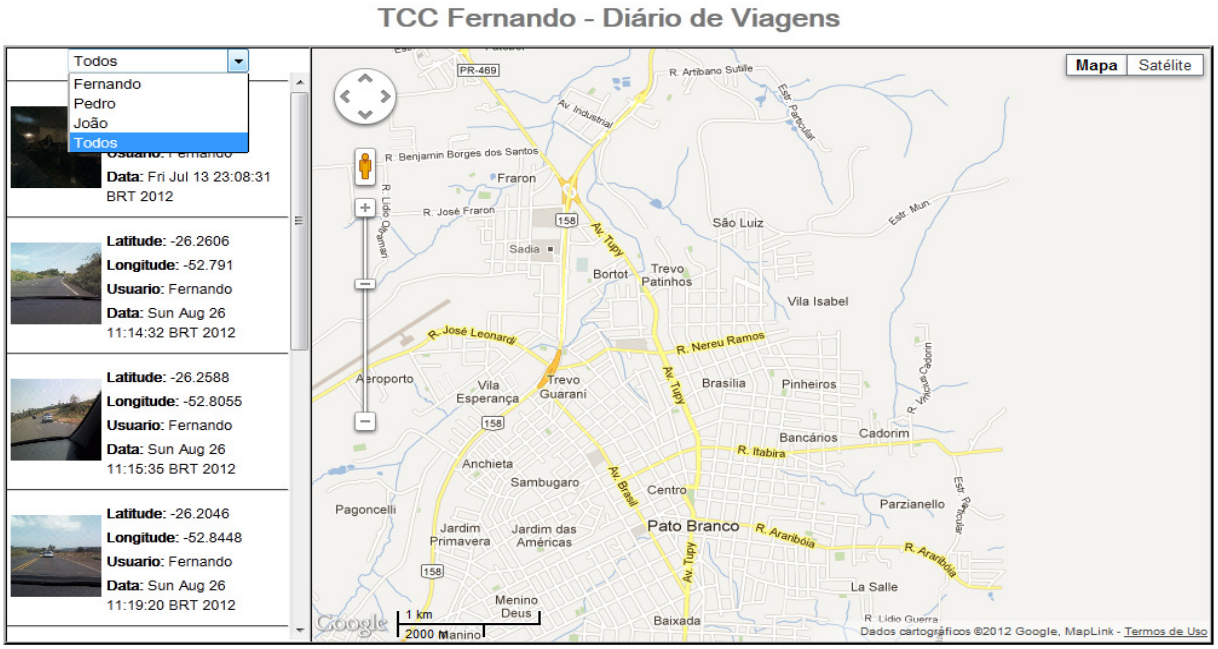

Figura 23 – Listagem de fotos na aplicação *web*

Figura 24 – Exibição da localização no mapa

4.3.1 Desenvolvimento do aplicativo *Web*

A aplicação *web* tem a finalidade de exibir os dados que foram transferidos do dispositivo móvel para o servidor através de *Web Service*. Devido o *Web Service* fazer a persistência no banco de dados, foi necessário apenas abrir uma conexão com o banco de dados e recuperar as informações da única tabela existente.

A aplicação *web* conta com vários recursos avançados, como comunicação RPC (*Remote Procedure Call*), que faz a comunicação entre o lado cliente e servidor através de invocações de procedimentos remotos. Essa característica torna o código desenvolvido muito mais fácil de ser interpretado, pois não há a necessidade de desenvolver a camada de transferência (*servlets*), apenas chamando métodos diretos como se estivesse trabalhando em qualquer aplicação Java Desktop convencional.

O GWT também possui suporte a AJAX nativo, pois em seu funcionamento, o *framework* converte todo o código Java em código Javascript, sendo esta a linguagem padrão do AJAX, já que vem do acrônimo "*Asynchronous* Javascript and XML".

Desenvolver aplicações utilizando GWT torna o trabalho do desenvolvedor mais prático, pois um dos grandes problemas das aplicações *web* é o suporte aos mais diversos navegadores existentes. O *framework* GWT se destaca neste ponto por fornecer suporte aos mais famosos navegadores do mercado gerando vários arquivos no projeto, sendo cada um deles destinado a um navegador, para garantir a melhor integração e compatibilidade possível.

Da mesma forma como se procedeu a utilização da API da Google na utilização dos mapas no cliente Android, para a aplicação servidora também foi necessária a adição da biblioteca responsável no projeto desenvolvido, nesse caso, a biblioteca específica para o *framework* GWT, e também aceitar um termo de utilização através da página da Google, no qual foi gerado um código único de identificação, que foi adicionado na página HTML (*Hypertext Markup Language*) da aplicação, apontando também o endereço onde está localizada a API para manipulação do mapa, conforme mostrado na Listagem 9.

Listagem 9 – Inclusão da API Google Maps ao Servidor *Web*

```
1.      <meta http-equiv="content-type" content="text/html; charset=utf-8">
2.      <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
3.      <script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?v=3&key=AIzaSyAfqwab6jnYtTJUZDlpkjHL-
Mct_vINW1U&sensor=false"></script>
```

A partir desta etapa, a API do Google Maps já está disponível para ser trabalhada. Para exibir o mapa na aplicação foi usado um componente do tipo “MapWidget”, instanciando-o com a utilização de outra classe do tipo “MapOptions”, que é responsável pela manipulação do mapa, como a inserção de controles de zoom, tipos de terreno do mapa, eventos do mouse, entre outros.

A criação da lista de fotos foi feita utilizando uma classe abstrata chamada “CustomCell”, que requer do desenvolvedor a montagem completa da estrutura desejada baseada em código HTML, como mostrado na Listagem 10.

Listagem 10 – Criação da célula do componente “CustomCell”

```
1.      @Override
2.      public void render(com.google.gwt.cell.client.Cell.Context context,
3.        FotosBean value, SafeHtmlBuilder sb) {
4.        if (value == null) {
5.          return;
6.        }
7.
8.        String foto = value.getFoto();
9.        sb.appendHtmlConstant("<hr/>");
10.       sb.appendHtmlConstant("<table>");
11.
12.       // Imagem
13.       sb.appendHtmlConstant("<tr><td rowspan=\"5\">");
14.       sb.appendHtmlConstant("<img width=\"75\" height=\"75\" src=\"" + foto + "\"/>");
15.       sb.appendHtmlConstant("</td></tr>");
16.
17.       // Latitude
18.       sb.appendHtmlConstant("<tr><td style='font-size:95%;'>");
19.       sb.appendHtmlConstant("<b>Latitude:</b> ");
20.       sb.appendEscaped(String.valueOf(value.getLatitude()));
21.       sb.appendHtmlConstant("</td></tr>");
22.
23.       // Longitude
24.       sb.appendHtmlConstant("<tr><td style='font-size:95%;'>");
25.       sb.appendHtmlConstant("<b>Longitude:</b> ");
26.       sb.appendEscaped(String.valueOf(value.getLongitude()));
27.       sb.appendHtmlConstant("</td></tr>");
28.
29.       // Usuario
30.       sb.appendHtmlConstant("<tr><td style='font-size:95%;'>");
31.       sb.appendHtmlConstant("<b>Usuario:</b> ");
32.       sb.appendEscaped(String.valueOf(value.getUsuario()));
33.       sb.appendHtmlConstant("</td></tr>");
34.
35.       // Data
36.       sb.appendHtmlConstant("<tr><td style='font-size:95%;'>");
37.       sb.appendHtmlConstant("<b>Data:</b> ");
38.       sb.appendEscaped(String.valueOf(value.getData()));
39.       sb.appendHtmlConstant("</td></tr></table>");
40.     }
```

Depois de a célula abstrata ser definida e o vetor de informações ser atribuído a esta através do método setRowData() desta classe, todos os itens da lista estão prontos para serem exibidos.

No desenvolvimento da localização das coordenadas geográficas no mapa foram utilizados componentes do tipo “Marker” e “InfoWindow”, sendo o primeiro, o marcador que

aparecerá no local do mapa onde a foto foi capturada e o segundo refere-se à janela informativa que aparece logo acima do marcador. A fim de não sobrecarregar o mapa com marcadores após várias aberturas de imagens, foi mantida apenas uma instancia de cada tipo destes objetos para que somente um marcador seja exibido.

A marcação do mapa é definida simplesmente chamando o método setPosition(), passando por parâmetro um objeto do tipo “Location” que está armazenando as informações das coordenadas geográficas.

5. CONCLUSÃO

O sistema desenvolvido tem a finalidade de publicar fotos capturadas a partir de dispositivos móveis que utilizem o sistema Android para serem visualizadas em uma aplicação *web*, que pode ser consultada a partir de qualquer computador conectado a Internet.

Para a análise e criação das aplicações foram utilizados conceitos obtidos através das disciplinas de Tecnologia de Orientação a Objetos, Programação Orientada a Objetos, Banco de Dados, Computação Móvel, Aplicações Distribuídas e *Web Design*.

Durante o desenvolvimento da aplicação, foi realizado um estudo aplicado ao sistema operacional Android, que mostrou alguns diferenciais da plataforma, como ser baseada em Linux, permitir a manipulação de todos os recursos de hardware sem qualquer restrição do sistema operacional e também pelo código do ambiente Android ser livre e gratuito. Tais características fazem que a plataforma Android esteja em forte ascendência no mercado, juntamente com a popularização dos *smartphones* que proporcionou a abertura de demanda para sistemas operacionais que rodem nestes aparelhos, implicando assim na notável aceitação dos usuários.

Através deste estudo foi possível também utilizar recursos atualmente presentes na grande maioria dos *smartphones*, como a câmera digital e também do GPS, realizando a captura de fotos que podem ser tratadas e manipuladas com facilidade com a utilização da SDK Android e também das coordenadas geográficas obtidas pelo receptor GPS para armazenamento em banco de dados e posterior exibição em mapa.

Por fim, os dados obtidos a partir do dispositivo móvel foram publicados em um servidor *web* utilizando o *Framework* GWT, este que foi tema de estágio realizado sob estudo dirigido, que aborda recursos de alto nível no contexto de aplicações de Internet, como comunicação RPC, interação com o servidor utilizando AJAX e suporte a múltiplos navegadores, atendendo assim a proposta da aplicação de forma satisfatória e eficaz.

A forma de trabalhar com a SDK Android é muito similar a qualquer outra aplicação Java, pois a sintaxe é a mesma, bastando apenas realizar estudo específico das novas classes proporcionadas por essa ferramenta, a fim de garantir a possibilidade de exploração total dos recursos que são disponibilizados.

5.1 PROBLEMAS ENCONTRADOS

Durante o desenvolvimento do trabalho, foram encontradas algumas dificuldades, como a dificuldade de encontrar documentação GWT com foco na utilização da API do Google Maps, pois esta tem sua utilização nativa baseada em Javascript, e apesar do *framework* GWT converter seu código para Javascript, o desenvolvimento é realizado através da linguagem Java.

Outro problema foi da necessidade de testar a funcionalidade da câmera do dispositivo móvel e de não existirem alternativas desses testes serem realizados com o emulador da API, devido o emulador não gerar imagem, mesmo que fictícia, da câmera. O código foi desenvolvido sem a certeza de que a funcionalidade funcionaria e somente após algum tempo buscando uma solução para este problema é que foi descoberto que era possível fazer a execução e depuração da aplicação no próprio dispositivo móvel.

5.2 TRABALHOS FUTUROS

Para trabalhos futuros serão realizadas algumas melhorias na aplicação, bem como a utilização de EXIF (*Exchangeable Image File Format*), que é um formato de metadado existente em imagens, que pode ser utilizado para armazenar informações de data e geolocalização embutidos na imagem, a parametrização do tempo de atualização da captura das coordenadas de GPS, pois o tempo foi definido estaticamente no código do sistema e também a possibilidade de definição manual das coordenadas geográficas, como forma de contornar possíveis problemas de instabilidade ou falta de comunicação do receptor GPS com o satélite.

Também como trabalho futuro, vislumbra-se a possibilidade de comercialização deste software, pois já houveram pessoas e empresas interessadas na essência da funcionalidade da aplicação, sendo necessário realizar algumas implementações e alterações na codificação para adequar o aplicativo à realidade destas pessoas, mas possivelmente abordando funcionalidades relativas ao Sistema de Posicionamento Global.

REFERÊNCIAS

ABLESON, W. F. *et al.* **Android in Action**, 3. Ed, Nova York: Manning, 2012.

Albuns da *web* do Picasa – Disponível em <http://picasaweb.google.com/lh/explore> Acesso em 22 set. 2012.

ANDROID SDK. **Android Developers**. Disponível em <http://developer.android.com/sdk/index.html> Acesso em: 18 set. 2012.

Apache Axis2 - **Apache Axis2/Java - Next Generation *Web* Services** – Disponível em <http://axis.apache.org/axis2/java/core/> Acesso em 24 set. 2012.

Change Vision – **Astah Community, UML, Professional, Share and iPad** – Disponível em < http://astah.net/> Acesso em 01 out. 2012.

GARGENTA, M. **Learning Android**, 1. Ed, Sebastopol: O`Reilly, 2011.

Google Maps – Disponível em <https://maps.google.com.br/maps?hl=pt-BR> Acesso em 24 set. 2012.

HANSON, R; TACY, A. **GWT In Action: Easy Ajax with the Google *Web* Toolkit.** Greenwich: Manning, 2007.

kSOAP2 – Disponível em <http://sourceforge.net/projects/ksoap2/> Acesso em 24 set. 2012.

Panoramio – **Photos of the World** – Disponível em <http://www.panoramio.com/map> Acesso em 22 set. 2012.

PRESSMAN, R. **Engenharia de software**, 5ª ed., Rio de Janeiro: McGraw-Hill, 2002.

Secundados | **Dados secundários, estatísticas e informações sobre a internet no Brasil**. - Disponível em <http://www.secundados.com.br/> Acesso em 25 set. 2012.

SMEETS, B; BONESS, U; BANKRAS, R. **Programando Google *Web* Toolkit: Do Iniciante ao Profissional.** Rio de Janeiro: Alta Books, 2009.

SQL Lite Home Page – Disponível em <http://www.sqlite.org/> Acesso em 24 set. 2012.