UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALVARO SAMUEL KAMPA

EVANDRO PINTO PORTUGAL

MAPEAMENTO DAS COORDENADAS GEOGRÁFICAS DE ESCOLAS ESTADUAIS UTILIZANDO WEB CRAWLER E GOOGLE MAPS API EM DISPOSITIVOS MÓVEIS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2013

ALVARO SAMUEL KAMPA

EVANDRO PINTO PORTUGAL

MAPEAMENTO DAS COORDENADAS GEOGRÁFICAS DE ESCOLAS ESTADUAIS UTILIZANDO WEB CRAWLER E GOOGLE MAPS API EM DISPOSITIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Thalita Scharr Rodrigues

Co-orientador: Prof. Wellton Costa de Oliveira

PONTA GROSSA

2013

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Graduação e Educação Profissional

TERMO DE APROVAÇÃO

MAPEAMENTO DAS COORDENADAS GEOGRÁFICAS DE ESCOLAS ESTADUAIS UTILIZANDO WEB CRAWLER E GOOGLE MAPS API EM DISPOSITIVOS MÓVEIS

por

ALVARO SAMUEL KAMPA
EVANDRO PINTO PORTUGAL

Este Trabalho de Conclusão de Curso (TCC) foi apresentadoem26demarçode2013 como requisito parcial para a obtenção do título de Tecnólogoem Análise e Desenvolvimentode Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Thalita Scharr Rodrigues
Profa. Orientadora

Wellton Costa de Oliveira
Membro titular

Danillo Leal Belmonte
Membro titular

HelyaneBronoski Borges
Responsável pelos Trabalhos
de Conclusão de Curso

Simone de Almeida
Coordenador do Curso
UTFPR - Campus Ponta Grossa

Dedico este trabalho à minha querida mãe, Eliza Beatriz Gomes da Costa, que sempre me incentivou a seguir em frente mesmo quando tudo parecia difícil demais de ser alcançado. Mãe, a sua insistência em fazer-me perceverar me ajudou a vencer! Obrigado!

Alvaro Samuel Kampa

Dedico este trabalho aos meus mestres. A palavra mestre, nunca fará justiça aos professores dedicados, aos quais, sem nominar terão meu eterno agradecimento! Dedico também à minha família, que nos momentos de minha ausência, sempre fizeram entender que o futuro, é feito a partir da constante dedicação no presente. Aos meus amigos e amigas, minha segunda família. Jamais lhes esquecerei. Dedico por fim à minha namorada, a qual compartilhou de momentos difíceis, onde por vezes eu estava desacreditado, ajudando a me levantar novamente, para que outro desafio fosse superado.

Evandro Pinto Portugal

AGRADECIMENTOS

Primeiramente, agradecemos à Deus, por sempre estar presente em nossas vidas e nos presentear com dádivas. Sem Suas bençãos nada seríamos.

Agradecemos aos nossos professores, que tanto nos auxiliaram durante o curso. Tenham a certeza de que o conhecimento adquirido ao longo dessa jornada será de grande proveito em nossas vidas profissionais.

Ao Professor Mestre Wellton Costa, que prestou grande ajuda e sempre nos auxiliou no entendimento de novas tecnologias, das quais fizemos uso no trabalho.

À Professora Mestre Thalita Scharr Rodrigues, por sua dedicação em orientar-nos sabiamente e nos dar forças para vencer esta etapa da vida.

Aos nossos amigos e colegas de classe, pela amizade que construímos durante o curso e pela troca de conhecimento dentro das salas de aula.

Agradecemos também às nossas famílias, que sempre nos deram apoio, almejando o nosso sucesso futuro, como profissionais e bons cidadãos.

À todos que, de alguma forma participaram dessa jornada, o nosso muito obrigado.

"Só sabemos com exatidão quando sabemos pouco; à medida que vamos adquirindo conhecimentos, instala-se a dúvida."

Johann Wolfgang von Goethe

RESUMO

KAMPA, Alvaro Samuel; PORTUGAL, Evandro Pinto. **Mapeamento das Coordenadas Geográficas de Escolas Estaduais Utilizando Web Crawler e Google Maps API em Dispositivos Móveis**. 2013. **51 folhas**. Trabalho de Conclusão de Curso Tecnologia em Análise e Desenvolvimento de Sistemas - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2013.

No universo *Web,* inovação é o que não tem faltado e, todos os dias novas ferramentas são criadas para promover a satisfação do usuário, visando conquistar cada vez mais espaço na vida cotidiana das pessoas. A vasta quantia de modelos de dispositivos móveis no mercado e a alta procura por esse tipo de meio de comunicação e entretenimento nos faz imaginar quantas possibilidades existem para facilitar o trabalho diário de profissionais, como o de motoristas e empresários. Uma delas é o Geoprocessamento, o qual está presente na lista de inovações que esses aparelhos nos trazem e será sempre muito bem utilizado. Também nos deparamos com tecnologias da *internet* que poupam os usuários de extensas e cansativas pesquisas, como é o caso dos *Web Crawlers*, que têm o poder de navegar incansavelmente pela *web* atras de conteúdos que seriam difíceis de se conseguir por meio do simples acesso à *sites*. Neste trabalho será apresentada uma proposta para unir essas tecnologias com o intuito de facilitar a localização de escolas e colégios paranaenses, dando também uma visão geral do funcionamento de um *Crawler,* desenvolvido em linguagem Java, de forma prática e simples.

Palavras-chave: Web Crawler. GPS. Google Maps. Coordenadas Geográficas. Android. Java. Escolas. Colégios.

ABSTRACT

Kampa, Alvaro Samuel; PORTUGAL, Evandro Pinto. **Geographical Coordinates Mapping of Public Schools Using Web Crawler and Google Maps API in Mobile Devices**. 2013. **51 pages**. Trabalho de Conclusão de Curso Tecnologia em Análise e Desenvolvimento de Sistemas - Federal Technology University - Parana. Ponta Grossa, 2013.

In the Web universe, innovation is not what has been lacking, and every day new tools are designed to promote user satisfaction, aiming to conquer more and more space in people's everyday lives. The vast amount of models of mobile devices in the market and the high demand for this type of communication devices and entertainment makes us imagine how many possibilities exist to facilitate the daily work of professionals, like drivers and businessmen. One of these is the Geoprocessing, which is present in the list of innovations that these devices bring to us and will always be very well used. We also face with Internet technologies that save users of extensive and exhausting research, as is the case of Web Crawlers, which have the power to navigate tirelessly searching for web content that would be difficult to achieve through simple access to sites This work presents a proposal to unite these technologies in order to facilitate the location of schools and colleges of Paraná, also giving an overview of the functioning of a Crawler, built in Java language, in a practical and simple way.

Keywords: Web Crawler. GPS. Google Maps. Geographical Coordinates. Android. Java. Schools. Coleges.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

Inc.	Incorporated

LISTA DE SIGLAS

API	Application Programming Inteface
FTP	File Transfer Protocol
GPS	Global Positioning System
HTML	HyperText Markup Language
KML	Keyhole Markup Language
OGC	Open Geospatial Consortium
URL	Uniform Resource Locator
XML	eXtensible Markup Language

LISTA DE ACRÔNIMOS

NAVSTAR	NAVigation System with Time And Ranging

SUMÁRIO

1 INTRODUÇÃO

Com os avanços da era digital, diversas tecnologias foram desenvolvidas para facilitar o dia-a-dia das pessoas e empresas. O mapeamento geográfico por meio de satélites é um exemplo de como os países evoluiram a partir da popularização do sensoriamento remoto com o passar dos anos (FITZ, 2008), principalmente facilitando o trabalho de motoristas. Uma das provas desse acontecimento é que mapas impressos e guias rodoviários estão cada vez mais raros. Um fator que contribuiu para a ampla utilização destes mapas digitais foi a chegada de dispositivos móveis, tais como *smartphones* e *tablets*, os quais possibilitam acessar tais mapas em qualquer lugar, desde que se obtenha acesso à *internet*.

Nos últimos anos, entre as inovações das tecnologias web encontram-se os *Crawlers*. Segundo Arasu et al.(2000), *Crawlers* são pequenos sistemas que percorrem a *Web* seguindo *links*, afim de alcançar diferentes páginas. São amplamente utilizados em *engines* de pesquisas para manter uma base atualizada de páginas à serem pesquisadas.

Analisando essas afirmações nota-se que, no Brasil, existem inúmeras escolas e colégios estaduais espalhados por todas as regiões do país, porém não se tem um mapeamento exato de todas as instituições de ensino, ainda mais se tratando de mapas digitais. Partindo deste pressuposto, o presente trabalho de conclusão de curso tem como objetivo o mapeamento geográfico das escolas estaduais do estado do Paraná.

1.1 MOTIVAÇÃO

A iniciativa de realizar o mapeamento geográfico das escolas e colégios estaduais do Paraná, surgiu visando facilitar a pesquisa da localização de uma dessas instituições. Além disso, espera-se que o aplicativo seja de grande valia para ógãos do governo dos estados brasileiros, como suas respectivas Secretarias da Educação, sendo que a aplicação localizará automaticamente a instituição de ensino selecionada.

1.2 OBJETIVO

Esta seção apresenta os objetivos do Trabalho de Diplomação, passos que devem ser seguidos para que o projeto possa ser realizado de maneira eficaz.

1.2.1 Objetivo Geral

Desenvolver um sistema para auxiliar na localização das escolas e colégios estaduais do estado do Paraná.

1.2.2 Objetivos Específicos

- Utilizar um *Web Crawler* para coletar as informações das escolas e colégios estaduais, a partir de uma página na internet que contenha os dados necessários para a localização das mesmas, em seguida criando um arquivo na estrutura XML(*eXtensible Markup Language*) para o armazenamento desses dados.
- Acessar a plataforma *Maps*, a partir do mesmo *Crawler* para obter as coordenadas do endereço das escolas, adicionando-as também ao arquivo XML.
- Disponibilizar o arquivo XML em um servidor *web*.
- Criar um aplicativo para o sistema Android, responsável por utilizar os dados da escola ou colégio selecionado pelo usuário, gerar um arquivo KML(*Keyhole Markup Language*) e acessar o Google Maps, apontando o local exato conforme as coordenadas.

1.3 JUSTIFICATIVA

A escolha do tema teve por base o desenvolvimento de um robô *web*, o qual teria o trabalho de acessar uma página na internet e baixar um conteúdo específico. Então, encontrou-se a necessidade do desenvolvimento de um aplicativo que fosse útil na rápida localização de escolas e colégios estaduais, afim de facilitar a pesquisa da localidade de uma instituição de ensino pública.

Visando tal objetivo, fora escolhida a plataforma *Maps* da Google para a localização destas unidades, cuja utilização está em alta no desenvolvimento para dispositivos móveis.

1.4 ESTRUTURA DO TRABALHO

Este trabalho se encontra dividido em 5 capítulos, sendo o primeiro, Introdução, o segundo, Fundamentação Teórica, o terceiro, Desenvolvimento, o quarto, Resultados e o quinto, Conclusões.

No Capítulo 1 é apresentada uma breve introdução, apresentando assuntos relacionados ao tema e que serão amplamente utilizados durante o desenvolvimento do projeto, deixando claras as áreas em que este trabalho estará participando.

No segundo capítulo, serão tratadas as principais tecnologias utilizadas no desenvolvimento do trabalho.

O terceiro capítulo apresenta as etapas do desenvolvimento do projeto, bem como trechos de código implementados dentro do sistema.

Então, têm-se os resultados obtidos apresentados no quarto capítulo. Nesse capítulo mostra-se o que foi possível de ser desenvolvido dentro do escopo planejado.

As conclusões do projeto, bem como trabalhos que possam ser desenvolvidos futuramente, estão no quinto capítulo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os conceitos das principais tecnologias utilizadas para o desenvolvimento do projeto, tais como Geoprocessamento, Desenvolvimento Android, *Web Crawlers*, entre outros.

2.1 MAPEAMENTO E COORDENADAS

Mendonça et al.(2008) consideram o Google *Maps* como um aplicativo a partir do Google *Earth*. Os autores exemplificam a popularização dos serviços de mapeamento junto aos usuários da rede mundial de computadores sendo que "Este serviço constitui-se no conjunto de especificações que permitem a criação, desenho em tela e distribuição de dados geoespaciais a partir de uma ou mais fontes de dados remotas".

2.1.1 Sistema de Coordenadas Geográficas

Segundo Rosa (2004), "Trata-se do sistema mais antigo de coordenadas. Nele, cada ponto da superfície terrestre é localizado na interseção de um meridiano com um paralelo. Suas coordenadas são a latitude e a longitude", números que consistem em graus(º), minutos(') e segundos(').

Meridianos são as linhas imaginárias que cortam o globo terrestre de sul a norte. Paralelos são as linhas, também imaginárias, paralelas com a linha do Equador (este também é um paralelo).

2.1.1.1 Latitude

Define-se como latitude o ângulo formado entre o ponto do marco zero (linha do Equador) até a posição desejada na superfície terrestre, tendo por base o centro da terra(o maior ângulo, que é o Polo Norte ou Polo Sul, é equivalente à 90º).

Paulo Rosa(2004) define que:

> Latitude geográfica é o ângulo (medido ao longo do meridiano que passa pelo lugar) formado entre o equador terrestre e o ponto considerado. Todos os pontos do equador terrestre têm latitude geográfica igual a 0º. Pontos situados ao norte do equador têm latitudes maiores que 0º variando até 90º que é a latitude do pólo geográfico norte. Da mesma forma variam as latitudes ao sul do equador terrestre, desde 0º a 90º, latitude do pólo geográfico sul. Para se diferenciar os valores, atribui-se sinal positivo para as latitudes norte e negativo para as latitudes sul. Simboliza-se a latitude pela letra grega ϕ. (ROSA, 2004, p. 35).

2.1.1.2 Longitude

Os graus da longitude são medidos pelo ângulo que se forma desde o meridiano de Greenwich até o ponto desejado, por meio da linha imaginária paralela à linha do equador.

Segue abaixo a explicação dada por Paulo Rosa (2004):

> Longitude geográfica é o ângulo (medido ao longo do equador) formado entre o meridiano que passa pelo lugar e o meridiano que passa pela cidade de Greenwich, Inglaterra. A longitude é medida de 0o a 180o, para leste ou para oeste de Greenwich. Por convenção, atribui-se também sinais para as longitudes: negativo para oeste e positivo para leste. A longitude é simbolizada pela letra grega λ. (ROSA, 2004, p. 35).

2.1.2 Sistema GPS

O Sistema de Posicionamento Global, mais conhecido pela sigla GPS (abreviação para *Global Positioning System*), é atualmente o sistema de localização geográfica mais utilizado. Ele usa como base o posicionamento de satélites que estão em órbita ao redor do planeta.

> GPS é a abreviatura de NAVSTAR GPS (NAVSTAR GPS - *NAVigation System with Time And Ranging Global Positioning System*). É um sistema de radio-navegação baseado em satélites desenvolvido e controlado pelo departamento de defesa dos Estados Unidos da América (U.S. DoD) que permite a qualquer usuário saber a sua localização, velocidade e tempo, 24 horas por dia, sob quaisquer condições atmosféricas e em qualquer ponto do globo terrestre. (ROSA, 2004, p. 56).

2.1.2.1 Funcionamento

O funcionamento do sistema GPS é baseado na localização dos satélites, sendo necessários ao menos três satélites para que o calculo da posição seja possível. No mínimo 4 satélites seriam ideais, resultando em melhor posicionamento com pouca margem de erro.

Os satélites emitem ondas de rádio que são quase randômicas. Porém, apresentam um padrão que apenas os aparelhos GPS podem codificar, dificultando assim que ruídos interfiram no sinal e no cálculo da distância entre o aparelho receptor e os satélites. Tal cálculo é efetuado através das variáveis tempo e velocidade (velocidade da luz) (ROSA, 2004).

2.1.3 Arquivos KML

A abreviação KML vem do termo inglês *Keyhole Markup Language*, que significa Linguagem de Marcação Buraco-de-Fechadura. Nesses arquivos existem dados de coordenadas geográficas, nome do local, entre outros atributos necessários para a identificação e localização de um lugar específico.

> KML é um formato de arquivo usado para exibir dados geográficos em um navegador da Terra, como o Google *Earth*, Google *Maps* e Google *Maps* para celular. É possível criar arquivos KML para indicar locais, adicionar sobreposições de imagem e expor dados avançados de novas maneiras. KML é um padrão internacional mantido pelo *Open Geospatial Consortium*, Inc. (OGC). (GOOGLE, 2013).

A estrutura de arquivos KML é baseada na linguagem XML e utiliza, tal como, tags criadas para a definição dos atributos a serem resgatados posteriormente, através de uma nomenclatura adequada(sintaxe), a qual fora definida pela OGC (GOOGLE, 2013).

A seguir, apresenta-se um exemplo de arquivo KML, forncecido pela Google. Nesse trecho de código, notam-se as *tags* utilizadas na estrutura deste

formato:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <kml xmlns="http://www.opengis.net/kml/2.2">
3    <Placemark>
4      <name>Simple placemark</name>
5      <description>Attached to the ground. Intelligently places itself at the height of the underlying terrain.</description>
6      <Point>
7        <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
8      </Point>
9    </Placemark>
10 </kml>
```

Código Fonte

Na linha 1 nota-se a presença de um cabeçalho XML. Esta linha sempre estará presente nos arquivos KML. Na linha 2, existe a declaração de *namespace* do KML.

A *tag Placemark* inicia os dados de localização. No exemplo de código acima, pode-se conferir as *tags name* (nome), *description* (descrição) e *point* (ponto). Esta última *tag* determina a posição do marcador no mapa, utilizando-se as coordenadas geográficas (latitude e longitude) e, opcionalmente, altitude.

2.1.4 Google *Maps* API

Uma API (*Application Programming Interface*) tem a função de "mascarar" o funcionamento complexo de um sistema, fornecendo apenas um conjunto de rotinas e padrões para que suas principais funcionalidades possam ser acessadas mediante de programação.

O sistema *Maps*, da Google Inc., foi desenvolvido inicialmente para a plataforma *deskto.* Entretanto, com a chegada dos *smartphone* e *tablets*, e após o desenvolvimento do sistema operacional Android, esse sistema de navegação em mapas tornou-se uma ferramenta de grande utilidade na linha de dispositivos móveis.

Com a popularização do serviço de GPS, juntamente com a ampla utilização do sistema *Maps*, foi possível o desenvolvimento de aplicativos que calculam a localização aproximada do aparelho e calculam a rota entre os ponto o mapa.

Para isso são utilizadas outras APIs que a Google disponibiliza, tais como a *Places* API(retorna informações sobre o local desejado), *Geocoding* API(converte endereços em coordenadas, ou vice-versa). Outra API denominada *Directions* API calcula a rota que será exibida ao usuário, além das rotas alternativas, tanto para

veículos como para andar a pé, utilizando-se das coordenadas geográficas ou endereço dos dois pontos.

A API da Matriz de Distância apresenta informações de tempo e distância necessários para percorrer o trajeto, e a API de Elevação consulta a elevação nos terrenos com a possibilidade de calcular a elevação em um trajeto a ser percorrido (GOOGLE, 2013).

2.2 ROBÔS DA INTERNET E *WEB CRAWLERS*

Robots, ou robôs, são também conhecidos como "Indexadores", "*Crawlers*", "*Web Crawlers*" ou "*Bots*".

> Um *Web Crawler* é um programa que realiza downloads de páginas da *Web*, usualmente para um motor de busca *Web* e/ou um *cache* local. (...) Cada página recuperada é entregue a um *client* que salva a página, cria um índice das mesmas, ou analiza o conteúdo das páginas. (CHO, 2001, tradução propria)

Pode-se dizer que um robô é um programa automatizado de busca e coleta de dados de páginas da *web*. Eles armazenam as mesmas em seu formato original e indexando-as, podendo posteriormente extrair informações específicas dentre todo o conteúdo armazenado.

2.2.1 *Crawler* Incremental x *Crawler* Periódico

Toda a coleção de páginas criada por um C*rawler* precisa ser mantida e atualizada. Assim, as informações e estatísticas obtidas a partir destas coleções se mantêm igualmente atualizados na medida do possível.

Existe básicamente duas maneiras de um *Crawler* se manter atualizado, e dependendo do método utilizado, o mesmo recebe nomenclaturas diferentes:

a) *Crawler* incremental - ao atingir o limite de páginas a serem buscadas, através de métricas de seleção, ele executa atualizações constantes da coleção de

páginas, substituindo as páginas consideradas de menor importancia através das métricas. Sua implementação é mais complexa.

b) *Crawler* periódico - com o número máximo de páginas atingido, o programa para. A cada n dias o *crawler* é executado novamente para renovar a coleção de páginas, apagando toda a coleção atual e utilizando da mesma busca anterior para criar uma coleção completamente nova. A implementação deste tipo de robô é simples, devido à ausência de regras para a atualização.

Na Figura 1 pode-se observar uma relação da atualização da coleção *versus* o tempo em que a mesma permanece atualizada. A parte cinza representa o período em que o robô está ativo.

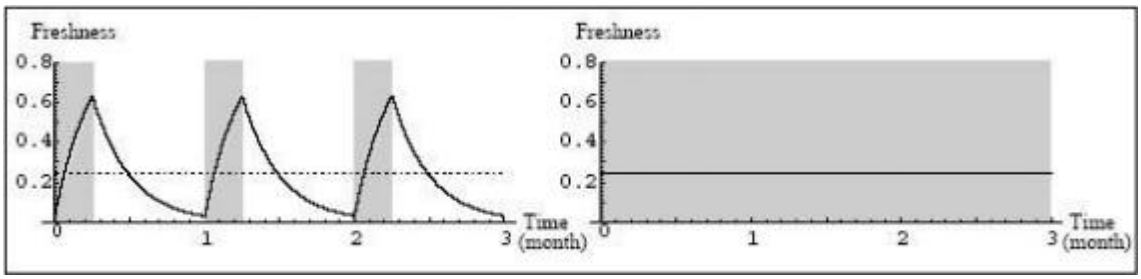

Figura 1 - Atualização da coleção de páginas - Crawler Periódico x Incremental.
Fonte: CHO(1999, p.11)

O primeiro gráfico representa um *crawler* periódico, enquanto o segundo representa um *crawler* incremental. É possível notar que no primeiro caso, as páginas se mantêm atualizadas somente enquanto o *crawler* está em atividade, e após este período toda a coleção fica desatualizada até a próxima execução do robô. Já no segundo caso, com a constante execução do robô é mantido um padrão de atualização da coleção.

2.3 DISPOSITIVOS MÓVEIS

São equipamentos portáteis, capazes de realizar processamento de informações. Como exemplos temos os celulares, *smartphones*, *tablets*, *laptops*, entre outros.

> A utilização de dispositivos móveis é cada vez mais comum, aliando as facilidades de pesquisa de informação à mobilidade. Estes dispositivos permitem a localização automática do utilizador, proporcionando a indicação de pontos de interesse local sem requerer a sua intervenção. No entanto, apresentam limitações que têm de ser tomadas em conta quando se desenvolvem aplicações de visualização, nomeadamente, no que diz respeito à dimensão reduzida do ecrã, à menor capacidade de desempenho do equipamento, aos mecanismos de entrada pouco adequados para tarefas complexas e aos problemas de conectividade. Consequentemente, a interface tem de ser especialmente concebida para este tipo de dispositivos, exigindo que os mecanismos de filtragem e as técnicas de visualização sejam adaptados às características do equipamento. (MATOS, P.; CARMO, M.; AFONSO, A.; 2007, p. 1).

2.3.1 Android

É o sistema operacional *mobile* de código aberto mais utilizado no mercado de *smartphones* e *tablets* e é desenvolvido pela Google. Segundo a empresa, devido à alta procura por aparelhos que utilizem este sistema, todo dia mais de 1 milhão de novos dispositivos móveis portando Android são ativados.

Sua arquitetura, representada pela Figura 2, é baseada no Kernel Linux 2.6, "que é responsável pelos serviços de segurança, gerenciamento de memória, processos, rede e drivers, este último componente é muito importante, pois garante que o desenvolvedor não precisará se preocupar em como acessar ou gerenciar dispositivos específicos do celular, produzindo assim uma abstração entre o hardware e o software", de acordo com Silva(2009).

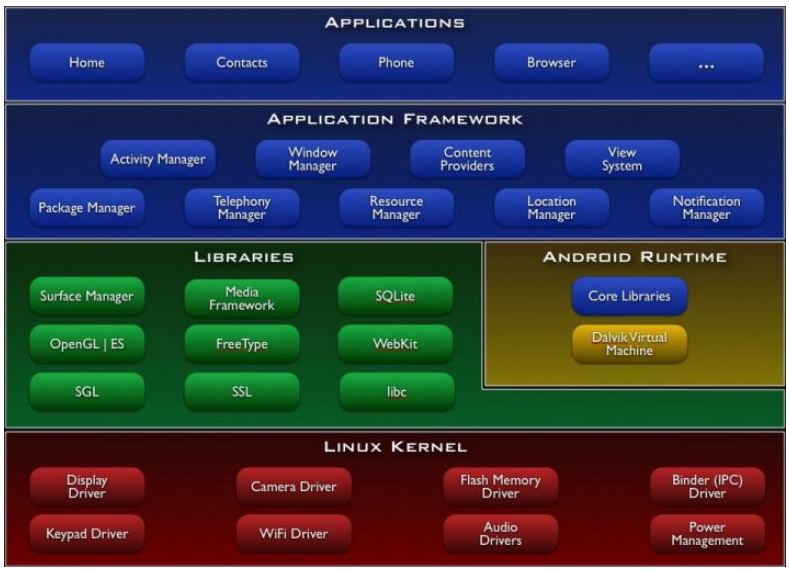

Figura 2 - Arquitetura Android.
Fonte: BORDIN(2012, p. 2)

3 METODOLOGIA

Neste capítulo estão sendo descritos os materiais e métodos empregados para o desenvolvimento do *Crawler* e do aplicativo móvel apresentados neste trabalho, que ilustram a integração das tecnologias e dos conceitos mostrados anteriormente.

No *Crawler* está definida a URL alvo, a qual o mesmo irá acessar, efetuando o download da página e gravando em memória para em seguida verificar todo o código em busca dos dados referentes ás escolas estaduais do Paraná, como nome e cidade de cada uma delas.

Por fim, os dados extraídos serão utilizados para tentar recuperar a localização geográfica de cada escola por meio da API do Google *Maps* e arquivos KML. Os dados buscados são armazenados em KML's para a cada escola localizada, para por fim ser gerado um único KML com todas as localizações encontradas das escolas estaduais do Paraná no Google *Maps*.

3.1 VISÃO GERAL

O desenvolvimento do projeto pode ser dividido em cinco partes, como representado na Figura 3. Mediante esse diagrama, foi possível definir o passo à passo para a implementação do projeto.

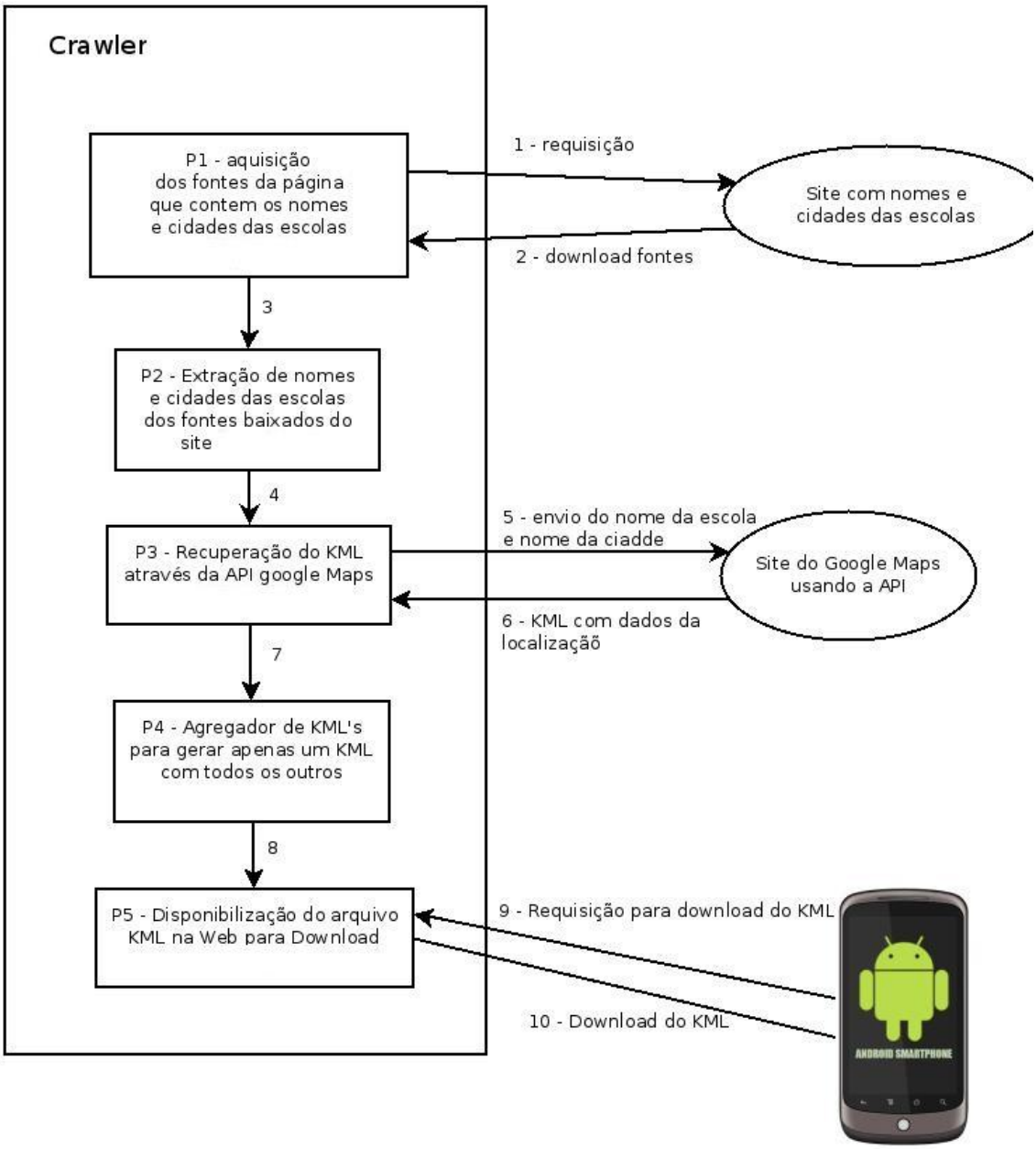

Figura 3 - Ilustação do funcionamento do Crawler
Fonte: Autoria Própria

Como ilustrado na Figura 3, as cinco partes podem ser descritas como segue:

1) O ponto de início do desenvolvimento foi estabelecido como a obtenção da URL do estado do Paraná que contém todas as escolas estaduais em funcionamento no ano de 2013.

2) A segunda parte do desenvolvimento consiste em extrair da URL obtida, todas as informações referentes ao nome e a cidade das escolas. São utilizados

padrões encontrados em meio ao código HTML, e armazenar na memória para uso posterior.

3) Com a API do Google Maps, se inicia a terceira parte do desenvolvimento. Nessa etapa, os dados das escolas, que estão armazenados em memória, serão utilizados para a recuperação dos arquivos KML's referente à localização geográfica de cada escola, transferindo o arquivo e armazenando em disco para cada uma delas.

4) Com as KML's salvas, inicia-se o processo em que é extraído apenas as informações relevantes, neste caso todo o conteúdo das tags *Placemark.* Gera-se assim um novo e único arquivo KML, que contém todas as informações obtidas pelo *Crawler*.

5) Por fim, o arquivo KML final é disponibilizado em um servidor FTP para posterior uso do aplicativo móvel desenvolvido. O aplicativo faz o *download* do respectivo KML, possibilitando por fim, a visualização dos dados obtidos em uma interface móvel, utilizando novamente a API Google Maps.

3.2 DESENVOLVIMENTO

Dado o início do desenvolvimento, foram utilizados passos padrões da metodologia empregada, de acordo com a Figura 4.

Figura 4 - Metodologia empregada no desenvolvimento
Fonte: SILVA, Marcelo(2012)

3.2.1 Fase Inicial - Planejamento

Inicialmente foi elaborada uma pesquisa na internet, procurando por *sites* que contivessem informações de endereços de escolas e colégios estaduais. Infelizmente, esses dados só foram encontrados para as instituições de ensino público do estado do Paraná, disponíveis no site da Secretaria da Educação do Estado do Paraná.

Então, iniciou-se um estudo relacionado às ferramentas que pudessem realizar a extração de informações de uma página na *web.* Foram encontrados sistemas como o jSpider e o WGet, que são dois exemplos de *Crawlers*. Contudo, esses sistemas não poderiam executar as tarefas necessárias para o perfeito funcionamento do projeto, então foi decidido que seria implementado um *Crawler* próprio pelos autores do presente trabalho, adicionando somente as funcionalidades necessárias para tal.

Outro ponto importante foi a disponibilização do arquivo KML, contendo as informações das escolas, tais como endereço, coordenadas, nome, entre outros dados necessários para a localização. Esse arquivo está disponível em um servidor na *web* contendo em seu nome e a versão em que este se encontra. Isto é necessário para que, quando o sistema desenvolvido para Android acessá-lo, possa analisar se é preciso realizar a atualização.

A plataforma Android fora escolhida para a criação do sistema, que será responsável por pesquisar em sua base de dados a escola selecionada e acessar o Google Maps, focando o mapa por meio das coordenadas. Seu uso está em alta no momento e milhares de novos aparelhos são ativados todos os dias. Outro fator que ajudou na escolha é pelo fato de ser um sistema *open source*.

3.2.2 Requisitos

- O *Crawler* deverá prover um arquivo KML com todas as escolas públicas do Paraná.
- O C*rawler* deverá disponibilizar a base de dados da escolas encontradas em KML para *download* no servidor FTP, sempre que houver alteração ou adição nos dados das escolas.

- O aplicativo em Android deverá atualizar a base KML sempre que houver atualização disponível.
- O aplicativo em Android deverá abrir a interface do Google *Maps* quando uma escola for selecionada.
- Deverá ser permitida a saída imediata do sistema Android quando solicitada pelo usuário.

3.2.3 Análise e Projeto

Foi elaborado um diagrama de Casos de Uso e um diagrama de Atividades para análise e entendimento do projeto. A Figura 5 demonstra as funcionalidades do robô C*rawler*.

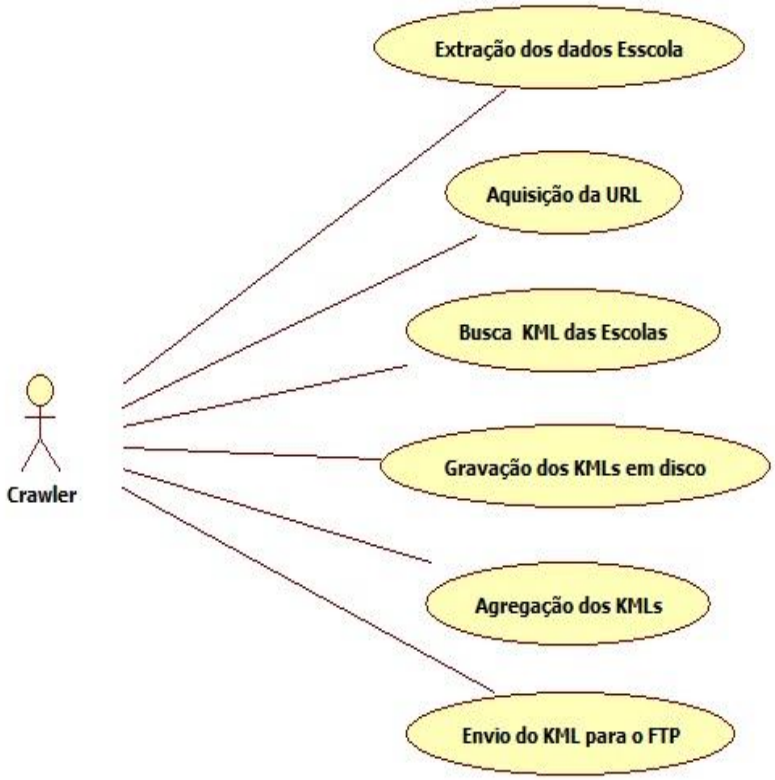

Figura 5 - Funcionalidades do *Crawler*
Fonte: Autoria Própria

Como pode ser visualizado na Figura 5, existe um caso de uso denominado Extração dos Dados da Escola, tarefa inicial e de grande importância para todo o processo. Para isto, estão englobadas as funções de buscar a URL e o arquivo KML

das escolas da página requerida. Feito isso, ainda existe a necessidade de gravar o arquivo, realizar a agregação dos KMLs e envio dos mesmos para o servidor FTP.

A Figura 6 apresenta o Diagrama de Atividades do aplicativo Android.

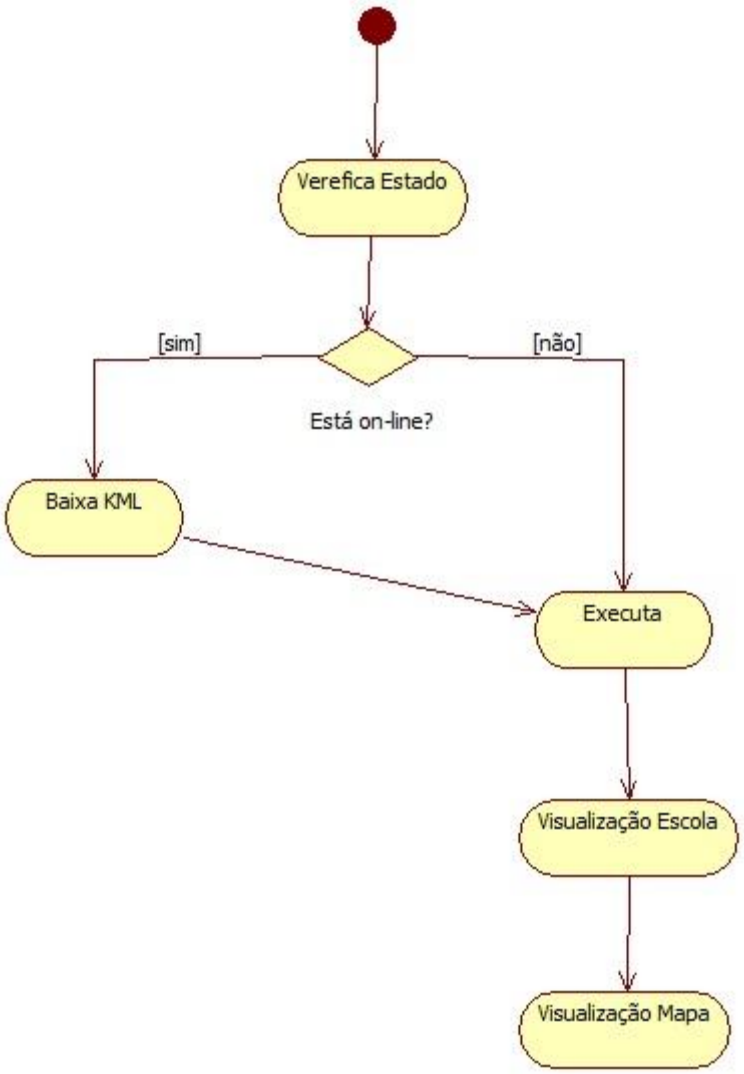

Figura 6 - Diagrama de atividades do aplicativo móvel
Fonte: Autoria Própria

O Diagrama de Atividades apresenta a ordem em que as ações são executadas, depois de o aplicativo ser iniciado e ser feita a busca por uma escola estadual específica.

3.2.4 Implementação

Esta seção apresenta a implementação do C*rawler* utilizando Java, assim como do aplicativo Android com a API Google *Maps*.

3.2.4.1 Programação das classes do *Crawler*

Como o C*rawler* foi dividido em cinco partes, foram necessárias diversas classes e funções relativas a cada etapa do robô, as quais são responsáveis pela obtenção dos dados desejados para o projeto.

A Figura 7 mostra a função obterEscolas, a qual inicia o processo da obtenção da página a partir da função extrairDados. Posteriormente, a função gravaArquivoEscolas é executado, extraindo os dados das escolas encontradas na página, salvando em memória as informações obtidas.

Na Figura 11 é apresentada a função obterKML, sendo que a partir dos endereços armazenados e da URL fornecida pela função extrairLocalizacao, é efetuada uma pesquisa pelo arquivo KML deste local , por meio do nome da escola e da cidade da mesma. Quando não for possível localizar utilizando essas informações, utiliza-se a classe *Maps* e a função pegarEndereco. Por meio dessa função, efetua-se uma nova tentativa à partir do endereço da escola em questão, posteriormente sendo armazenando o mesmo em disco. Em seguida, por meio da função gerarKML, todos os KML's são carregados e busca-se pelas *tags* referentes a *Placemark*. Essa tag contém as informações de localização como latitude e longitudade, além do endereço completo. Então, todas essas tags são reunidas em um único arquivo KML para *upload* em um servidor de FTP, o qual o aplicativo móvel irá utilizar para efetuar o *download* e ter acesso às informações obtidas pelo *crawler*.

3.2.4.2 Aquisição da página

A Figura 7 mostra a função obterEscolas, que é responsável pela chamada da função que adquire o endereço da página alvo, ou seja, URL que contém as informações das escolas estaduais do Paraná.

```
83      public String[] obterEscolas() {
84          String endereco[] = extrairDados("PR");
85          String endereco20[] = new String[endereco.length];
86
87          for (int x = 0; x < endereco.length; x++) {
88              endereco20[x] = endereco[x].replace(" ", "%20");
89              System.out.println("\n---------------------------------\nescolas " + x + ": " + endereco[x]);
90              if (!extrairLocalizacao(endereco20[x], x)) {
91                  Maps m = new Maps();
92                  endereco20[x] = m.pegarEndereco(endereco20[x]);
93                  extrairLocalizacao(endereco20[x], x);
94              }
95          }
96
97          return endereco;
98      }
```

Figura 7 - obterEscolas
Fonte: Autoria Própria

Na linha 84 é inicializada a variável que armazena as informações utilizadas para a obtenção dos arquivos KML's na terceira etapa do robô. Nesta respectiva linha também se inicia o processo de acesso e *download* da página alvo.

Na Figura 8, a função extrairDados é responsável por armazenar e localizar as informações das escolas estaduais. Na linha 16 apresenta-se a variável endereco[], um vetor de *strings* onde fica armazenado dinamicamente essas informações. Nas linhas 21 e 23 tem-se a criação e a instanciação de uma variável do tipo URL**,** a qual armazena do endereço da página.

```
14      public String[] extrairDados(String estado) {
15          List retorneEscolas = new ArrayList();
16          String endereco[] = new String[1000000];
17
18          if (estado.equals("PR")) {
19              try {
20
21                  URL url = null;
22                  File arquivoEscolas = new File("escolasParana.html");
23                  url = new URL("http://www4.pr.gov.br/escolas/listaescolas.jsp?NomeNre=Todos&NomeMun=Todos&DescrRedeEnsino=Estadual&CodNre=0&CodMun=0&CodRedeEnsino=2&Ano=2013");
24                  System.out.println("Baixando fonte de http://www4.pr.gov.br/escolas/listaescolas.jsp ...");
25                  endereco = gravaArquivoEscolas(url, arquivoEscolas);
26
27              } catch (MalformedURLException e) {
28                  e.printStackTrace();
29              } catch (Exception e) {
30                  e.printStackTrace();
31              }
32          }
33          return endereco;
34      }
```

Figura 8 - extrairDados
Fonte: Autoria Própria

Na linha 25 é efetuada a chamada da função gravarArquivoEscolas, a qual é responsável pela segunda etapa do projeto.

3.2.4.3 Extração de nomes, cidades e endereço das escolas estaduais

A Figura 9 mostra a função gravaArquivoEscolas, que mediante um *BufferedReader* (na linha 38), carrega o fonte da URL armazenada anteriormente. Para a extração dos dados dos municípios e das escolas, são utilizadas variáveis do tipo *string* e *list*, as quais são utilizadas dentro do laço *while* na linha 47. Dentro deste laço, são utilizados padrões de *strings* encontrados no HTML da URL, para extrair as informações referentes aos municípios e das escolas. Essas informações são guardadas na variável listaEscolas, as quais são retornadas para a função extrairDados, que por sua vez repassa para a função obterEscolas**,** função que iniciou o processo para a aquisição dos dados.

```
36      public String[] gravaArquivoEscolas(URL url, File arquivoEscolas) throws IOException {
37
38          BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
39          BufferedWriter out = new BufferedWriter(new FileWriter("arquivoEscolas.html"));
40          String inputLine;
41          int contadorCidade = 1;
42          int contadorEscolaDaCidade = 1;
43          List listaEscolas = new ArrayList();
44          int codigoEscola = 0;
45          String escolaAtual = "";
46
47          while ((inputLine = in.readLine()) != null) {
48              String municipio[] = inputLine.split("Munic□pio: ");
49              if (municipio.length == 2) {
50                  contadorEscolaDaCidade = 1;
51                  String municipio2[] = municipio[1].split("</b></td>");
52                  escolaAtual = municipio2[0];
53
54                  System.out.println("\n\n================================================================================");
55                  System.out.print("Cidade: " + municipio2[0] + "(" + contadorCidade + ")");
56                  contadorCidade++;
57              }
58
59              String escolas[] = inputLine.split("<font class=\"verdeclaro11\">");
60
61              if (escolas.length == 2) {
62                  String escolas2[] = escolas[1].split("</font>");
63                  String escolas3[] = escolas2[0].split(",");
64                  System.out.println("\nescola[" + contadorCidade + "][" + contadorEscolaDaCidade + "]=(" + codigoEscola + ")" + escolas3[0]);
65                  listaEscolas.add("colegio " + escolas3[0] + "," + escolaAtual);
66                  contadorEscolaDaCidade++;
67                  codigoEscola++;
68              }
69
70              out.write(inputLine);
71              out.newLine();
72          }
73
74          System.out.println("\n DOWNLOAD COMPLETO!");
75          in.close();
76          out.flush();
77          out.close();
78          String[] escolasConvertidas = (String[]) listaEscolas.toArray(new String[0]);
79
80          return escolasConvertidas;
81      }
```

Figura 9 - gravaArquivoEscolas
Fonte: Autoria Própria

Com os dados retornados a função inicializadora e salvos em memória, o *Crawler* inicia a terceira etapa do projeto.

3.2.4.4 Recuperação do arquivo KML das escolas estaduais

Retornando a Figura 7, onde exibi-se a função obterEscolas, tem-se na linha 85 a criação de uma variável *string* chamada endereco**.** Essa variável recebe as informações das escolas que estavam salvas dinamicamente, de modo que essa variável é utilizada dentro de um laço for na linha 87. A cada ciclo do laço, utiliza-se a função extrairLocalizacao**,** que a partir do nome da cidade e da escola, efetua uma tentativa de obter o respectivo KML.

```
100     public boolean extrairLocalizacao(String endereco, int contador) {
101
102         URL url = null;
103         File arquivo = new File("kml" + endereco + ".xml");
104         int naoEncontradas = 0;
105         float percentual;
106         boolean encontrouOuNao = false;
107
108         try {
109             arquivo = new File("kml" + endereco + ".xml");
110             url = new URL("http://maps.google.com/maps/geo?q=" + endereco + "&output=xml&key=ABQIAAAAAaVFxs6kNq7gWY59qf5XMxSec6s_uUscdbTyPSy8(
111
112             //percentual = (naoEncontradas*100)/(endereco.length - 1);
113
114             System.out.println("-------numero escola: " + contador + " (nao encontradas:" + naoEncontradas + ")----------------------");
115
116             if (!obterKML(url, arquivo, contador)) {
117                 naoEncontradas++;
118                 System.out.println(naoEncontradas);
119                 encontrouOuNao = false;
120             } else {
121                 encontrouOuNao = true;
122             }
123
124         } catch (MalformedURLException e) {
125             e.printStackTrace();
126         } catch (Exception e) {
127             e.printStackTrace();
128         }
129         return encontrouOuNao;
130
131     }
```

Figura 10 - extrairLocalizacao
Fonte: Autoria Própria

Na Figura 10 apresenta-se a função extrairLocalizacao, a qual usa uma variável do tipo URL que contém os dados de *placemark* da escola que está sendo procurada. Esta URL é formada por um endereço padrão, que tem o nome e a cidade da escola, como mostrado na linha 110.

Na linha 116 é chamada a função que a partir da URL acessada irá efetuar o *download* do arquivo KML referente ao local pesquisado.

A função obterKML, apresentada na Figura 11, de posse da URL acessada, utiliza uma variável do tipo *BufferedReader* para varrer o KML encontrado e salvar o mesmo em disco para um uso futuro. Podemos observar que está função retorna um valor *boolean*, o qual indica se foi possível resgatar o arquivo KML a partir da pesquisa pelo nome e cidade da escola que foi utilizado, como é mostrado na linha

147, caso na URL o status for de código 602, o *Crawler* assume que não foi possível recuperar os dados, e com isto voltando uma resposta negativa para a função obterEscolas, a qual por sua vez, como observado na Figura 7 na linha 91, instancia uma classe *Maps*.

```
133     public boolean obterKML(URL url, File arquivo, int codigoEscola) throws IOException {
134         BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
135         BufferedWriter out = new BufferedWriter(new FileWriter("arquivosKML/kml" + codigoEscola + ".xml"));
136         String inputLine;
137         boolean encontrado = true;
138 
139         while ((inputLine = in.readLine()) != null) {
140             // Imprime página no console
141             System.out.println(inputLine);
142             String linhaDividida[] = inputLine.split("<code>");
143 
144             if (linhaDividida.length == 2) {
145                 String linhaDividida2[] = linhaDividida[1].split("</code>");
146 
147                 if (linhaDividida2[0].equals("602")) {
148                     System.out.println("béé... Dados não encontrados");
149                     encontrado = false;
150                     break;
151                 }
152             }
153 
154             out.write(inputLine);
155             out.newLine();
156         }
157         in.close();
158         out.flush();
159         out.close();
160 
161         return encontrado;
162     }
```

Figura 11 - obterKML
Fonte: Autoria Própria

3.2.4.5 Classe Maps

Na classe *Maps*, representada na Figura 12, de posse das informações do endereço da escola, através da função pegarEndereco, é efetuada uma nova consulta a partir de uma nova URL, representada na linha 16, onde a partir desta página é extraído novas informações sobre o endereço da escola pesquisada, retornando as mesmas para a função obterEscolas, a qual na linha 93 da Figura 7, é efetuado uma nova tentativa de *download* do KML da escola em questão.

Caso nessa tentativa não seja possível obter novamente o KML, essa escola é descartada. Dessa maneira estabelece-se que, não foi possível, por meio de padrões de pesquisa, encontrar as informações do KML automáticamente.

```
public class Maps {
	public String pegarEndereco(String escolaNaoEncontrada){
			String enderecoRetornado="";
			try{
				/*String escolaNaoEncontrada[]={
					"escola CEEBJA WENCESLAU BRAZ,WENCESLAU BRAZ".replace(" ","+"),
					"escola MILTON BENNER,WENCESLAU BRAZ".replace(" ","+"),
					"escola  PATRIMONIO SAO MIGUEL,WENCESLAU BRAZ".replace(" ","+")
				};*/
				escolaNaoEncontrada = escolaNaoEncontrada.replace(" ","+");

				URL url = null;
				File arquivoEscolas = new File("escolasParana.html");

				url = new URL("https://maps.google.com.br/maps?q="+escolaNaoEncontrada+"&ie=UTF-8&hl=pt-PT");
				System.out.println(url);

				BufferedReader in=new BufferedReader(new InputStreamReader(url.openStream()));
				BufferedWriter out=new BufferedWriter(new FileWriter("arquivoEscolas.html"));
				String inputLine;
				int contadorCidade=1;
				int contadorEscolaDaCidade=1;
				List listaEscolas = new ArrayList();
				int codigoEscola=0;
				String escolaAtual="";

				while ((inputLine = in.readLine()) != null) {
					String enderecoEscola[] = inputLine.split("<span class=\"pp-place-title\"><span>");
					 if(enderecoEscola.length==2){
						String novoEndereco = enderecoEscola[1];
						//novoEndereco = novoEndereco.replace("<b>", "");
						//novoEndereco = novoEndereco.replace("</b>", "");
						//novoEndereco = novoEndereco.replace("<div>","");
						//novoEndereco = novoEndereco.replace("</div>", "");
						//novoEndereco = novoEndereco.replace("<span>", "");
						//novoEndereco = novoEndereco.replace("</span>", "");
						//novoEndereco = novoEndereco.replace("<span dir=\"ltr\" class=\"pp-headline-item pp-headline-address\">", "");

						//String enderecoEscola3[] = novoEndereco.split("<");
						String enderecoEscola2[] = novoEndereco.split("<span dir=\"ltr\" class\"pp-headline-item pp-headline-address\">");

						System.out.println(enderecoEscola2[0]);
						enderecoRetornado = enderecoEscola2[0];
					 }
					// Grava pagina no arquivo
					out.write(inputLine);
					out.newLine();
				}
			} catch (MalformedURLException e) {
				 e.printStackTrace();
			} catch (Exception e) {
				e.printStackTrace();
			}

		return enderecoRetornado.replace(" ", "%20");
	}
```

Figura 12 - Classe Maps e Função pegarEndereco
Fonte: Autoria Própria

3.2.4.6 Agregar KML's

A quarta etapa do C*rawler* consiste em agregar todos os *placemarks* encontrados entre todas as KML's armazenadas em apenas um único arquivo KML. Para isto é utilizado a função gerarKML, representada na Figura 13.

```
167     public void gerarKML() throws IOException {
168         File[] fileList = new File("arquivosKML").listFiles();
169         File newFile = new File("placemarks"+new SimpleDateFormat("yyyyMMdd").format(new Date())+".xml");
170         BufferedWriter out = new BufferedWriter(new FileWriter(newFile));
171         String inputLine = "";
172         int pos1 = 0;
173         int pos2 = 0;
174         // Percorre a lista de arquivos
175         for (int c = 0; c < fileList.length; c++) {
176             String outputLine = "";
177             String fileName = fileList[c].getName();
178             BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream(fileList[c])));
179             if (fileName.contains("kml")) {
180                 while ((inputLine = in.readLine()) != null) {
181                     outputLine += inputLine;
182                     inputLine = "";
183                 }
184                 pos1 = outputLine.indexOf("<address>");
185                 pos2 = outputLine.indexOf("</Placemark>");
186                 if (pos1 != -1) {
187                     outputLine = outputLine.substring(pos1, pos2);
188                     out.write("<Placemark id=\"p" + (c + 1) + "\">" + outputLine + "</Placemark>");
189                     out.flush();
190                 }
191             }
192         }
193         out.close();
194         enviaFTP(newFile);
195     }
```

Figura 13 - gerarKML
Fonte: Autoria Própria

Nesta função, na linha 168 é criado a variável *fileList* do tipo *File array*, onde ao instanciá-la é carregado todos os arquivos encontrados na pasta arquivosKML, pasta esta utilizada pela função obterKML para armazenar todos os KML's que obtidos .

Na linha 169, é criado o arquivo que será salvo após a agregação de todo o conteúdos dos KML's das escolas estaduais, onde vale notar a nomenclatura utilizada para o nome do arquivo, formado pela *string placemark* concatenado a data em que o arquivo foi gerado, composto por ano, mês e dia respectivamente, com o objetivo de ser um controle de versão do arquivo KML, para servir de auxílio para o aplicativo móvel posteriormente.

Na linha 175, se inicia um laço for, no qual se repete até que todos os arquivos contidos na variável *fileList* sejam percorridos, e a cada arquivo percorrido é feito a leitura do mesmo, verificado se o arquivo é um dos KML's gerados pela função obterKML do *Crawler*, e em seguida extraído toda informação contida entre as *tags* <Placemark> contídas no arquivo.

Na linha 194, é chamada a função enviaFTP, a qual é a responsável pela última etapa do *Crawler*.

3.2.4.7 Disponibilização do arquivo KML na web para *download*

A quinta e última etapa consiste em enviar para um servidor de FTP o arquivo KML gerado anteriormente pela função gerarKML.

```
196     public void enviaFTP(File file) {
197         String nomeArquivo = null;
198         FTPClient ftp = new FTPClient();
199         try {
200             ftp.connect("ftp.etechsistemas.com.br");
201 
202             //verifica se conectou com sucesso!
203             if (FTPReply.isPositiveCompletion(ftp.getReplyCode())) {
204                 ftp.login("etechsistemas", "1qasw2");
205             } else {
206                 //erro ao se conectar
207                 ftp.disconnect();
208                 System.out.println("Conexão recusada");
209                 System.exit(1);
210             }
211 
212             //abre um stream com o arquivo a ser enviado
213             InputStream is = new FileInputStream(file);
214             nomeArquivo = file.getName();
215 
216             //ajusta o tipo do arquivo a ser enviado
217             if (file.getName().contains(".txt")) {
218                 ftp.setFileType(FTPClient.ASCII_FILE_TYPE);
219             } else if (file.getName().contains(".jpg")) {
220                 ftp.setFileType(FTPClient.BINARY_FILE_TYPE);
221             } else {
222                 ftp.setFileType(FTPClient.ASCII_FILE_TYPE);
223             }
224             System.out.println("Enviando arquivo " + nomeArquivo + "...");
225 
226             ftp.changeWorkingDirectory("/www/download/Evandro");
227             //faz o envio do arquivo
228             ftp.storeFile(nomeArquivo, is);
229             System.out.println("Arquivo " + nomeArquivo + " enviado com sucesso!");
230 
231             ftp.disconnect();
232             System.out.println("Fim. Tchau!");
233         } catch (Exception e) {
234             System.out.println("Ocorreu um erro: " + e);
235             System.exit(1);
236         }
237 
238     }
```

Figura 14 - enviaFTP
Fonte: Autoria Própria

Podemos observar na Figura 14 o procedimento enviaFTP responsável pelo envio deste arquivo final, tornando assim disponível para o aplicativo mobile efetuar o *download* do mesmo.

Na linha 198 podemos observar que é instanciado uma variável da classe FTPClient, esta classe será a responsável pela conexão com o servidor FTP e por todo o procedimento de envio e navegação por este servidor. Esta classe, que não é nativa do Java, foi incorporada ao projeto através da biblioteca commons-net-3.2.jar (Apache Commons Net, 2012), para tornar possível o uso da classe e suas funções pelo robô.

```
12 import org.apache.commons.net.ftp.FTPClient;
13 import org.apache.commons.net.ftp.FTPReply;
```

Figura 15 - *Imports* da biblioteca apache.commons.net
Fonte: Autoria Própria

Na Figura 15 temos os imports necessários para os usos das classes FTPClient e FTPReply.

Na linha 200 é estabelecido a conexão com o FTP através da função connect, seguido pela verificação do sucesso ou não da conexãon na linha 203 através da função isPositiveCompletion da classe FTPReply, passando como parâmetro a variável ReplyCode da classe FTPClient**.**

Antes de ser efetuado o *upload* do arquivo KML, é efetuado uma verificação do tipo do arquivo na linha 217, e na linha 226 é navegado pelos diretórios do FTP até o local onde será salvo o arquivo.

Por último o procedimento envia o arquivo ao FTP através da função storeFile, como mostrado na linha 228, se desconetando do servidor com a função disconnect na linha 231.

3.2.5 Integração com a API Google *Maps*

Com o aplicativo móvel desenvolvido em Android, utilizamos da API Google *Maps* para criar a integração entre os resultado obtidos através do *Crawler* com um dispositivo móvel.

Iniciamos o aplicativo implementando em sua atividade princial um procedimento de busca pelo arquivo KML gerado pelo *Crawler*, efetuando em seguida um *download* do mesmo como mostrado na Figura 16**.**

```
if (isNetworkAvailable()){
        try {
                URL url = null;
                url = new URL("http://www.etechsistemas.com.br/download/Evandro/placemarks20130314.xml");

                BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
        BufferedWriter out = new BufferedWriter(new FileWriter("placemarks20130314.xml"));
        String inputLine;

        while ((inputLine = in.readLine()) != null) {
            // Grava KML
            out.write(inputLine);
            out.newLine();
        }

        in.close();
        out.flush();
        out.close();
        } catch (MalformedURLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
```

Figura 16 - Busca e *download* do arquivo KML
Fonte: Autoria Própria

Podemos observar que o método utilizado para efetuar o *download* do arquivo segue o padrão utilizado durante o desenvolvimento do *Web Crawler*, porém antes de efetuarmos o *download*, o aplicativo verifica se o aparelho móvel está com conexão disponível através da função isNetworkAvaible**()**.

```
private boolean isNetworkAvailable() {
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
    return activeNetworkInfo != null && activeNetworkInfo.isConnected();
}
```

Figura 17 - isNetworkAvaible
Fonte: Autoria Própria

Na Figura 17 vemos como é efetuada esta verificação a partir de uma instanciação da classe *ConnectivityManager* e da classe NetworkInfo, a quais em conjunto recuperam o status de conexão do dispositivo móvel através de qualquer meio, seja por conexão *wireless*, conexão 3G, entre outras.

Passando para a segunda atividade implementada no aplicativo, temos então a atividade responsável pela leitura do arquivo, a qual faz o armazenamento das informações de cada escola estadual contida no arquivo KML e em seguida listando as mesmas tornando disponível para consulta na API do *Google Maps.*

```
private void buscarTodas() {
        new Thread() {
                @Override
                public void run() {
                        try {
                        String inputLine;
                        AssetManager am = getAssets();
                        InputStream is = am.open("placemarks20130314.xml");
                                BufferedReader in = new BufferedReader(new InputStreamReader(is));
                        while ((inputLine = in.readLine()) != null) {
                                String dadosEscola[] = inputLine.split("<Placemark id=");
                                String nome = inputLine.substring(inputLine.indexOf("<AddressLine>")+13
                                                                , inputLine.indexOf("</AddressLine>"));
                                String coordenadas = inputLine.substring(inputLine.indexOf("<coordinates>")+13
                                                                        , inputLine.indexOf("</coordinates>"));
                                coordenadas = coordenadas.substring(0, coordenadas.length()-3);
                                float latitude = Float.parseFloat(coordenadas.substring(0, coordenadas.indexOf(',')-1));
                                float longitude = Float.parseFloat(coordenadas.substring(coordenadas.indexOf(',')+1, coordenadas.length()));
                                Escola escola = new Escola(nome, latitude, longitude);
                                        listaEscolas.add(escola);
                        }
                        //fi.close();
                        is.close();
                        } catch (FileNotFoundException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                        atualizaListaEscolas();
                }
        }.start();
}
```

Figura 18 - buscaTodas
Fonte: Autoria Própria

Na Figura 18 vemos o procedimento responsável por efetuar a leitura do arquivo KML e em seguida armazenar as informações das escolas na varíavel listaEscolas do tipo *List*. As informações são armazenadas como objetos da classe Escola.

```
package com.example.evandro_android;

public class Escola {

        String nome;
        float latitude;
        float longitude;

        public Escola(String nome, float latitude, float longitude){
                this.nome = nome;
                this.latitude = latitude;
                this.longitude = longitude;
        }

        public void setNome(String nome){[...]
        }

        public void setLatitude(float latitude){[...]
        }

        public void setLongitude(float longitude){[...]
        }

        public String getNome(){[...]
        }

        public float getLatitude(){[...]
        }

        public float getLongitude(){[...]
        }

}
```

Figura 19 - Classe Escola
Fonte: Autoria Própria

A classe Escola, representada na Figura 19, possui os atributos nome do tipo *String*, e latitude e longitude do tipo *float*.

Por último, temos a classe Gmaps, que é a atividade responsável por criar uma instância de um mapa da API *Google Maps* e efetuar as configurações para tornar possível o uso do mapa junto a aplicação desenvolvida.

```
public class Gmaps extends Activity implements LocationListener  {

        Location mostRecentLocation;

        private static final String MAP_URL = "http://gmaps-samples.googlecode.com/svn/trunk/articles-android-webmap/simple-android-map.html";
        private void getLocation() {
            LocationManager locationManager =
              (LocationManager)getSystemService(Context.LOCATION_SERVICE);
            Criteria criteria = new Criteria();
            criteria.setAccuracy(Criteria.ACCURACY_FINE);
            String provider = locationManager.getBestProvider(criteria,true);
            //In order to make sure the device is getting the location, request updates.
            locationManager.requestLocationUpdates(provider, 1, 0, this);
            mostRecentLocation = locationManager.getLastKnownLocation(provider);
          }

        public void setLocation(Escola escola){
                mostRecentLocation.setLatitude(escola.latitude);
                mostRecentLocation.setLongitude(escola.longitude);
        }

          private WebView webView;
          @Override
          /** Called when the activity is first created. */
          public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_gmaps);
            setupWebView();
            this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
          }
          /** Sets up the WebView object and loads the URL of the page **/
          private void setupWebView(){
            final String centerURL = "javascript:centerAt(" +
                        mostRecentLocation.getLatitude() + "," +
                        mostRecentLocation.getLongitude()+ ")";
            final String GotoLatLng = "javascript:GotoLatLng(" +
                                mostRecentLocation.getLatitude() + "," +
                                mostRecentLocation.getLongitude()+ ")";
            webView = (WebView) findViewById(R.id.webview);

            webView.setWebViewClient(new WebViewClient());
            webView.loadUrl(MAP_URL);
          }
```

Figura 20 - WebView e setLocation
Fonte: Autoria Própria

Na Figura 20 podemos observar como é implementado as classes necessárias para o uso da API *Google Maps*, destacando o *WebView***,** que é o responsável por gerenciar a visualização do mapa. Vemos que dentro do método de criação da classe Gmaps é feito a chamada do procedimento setupWebView**()**.

```
1           private void setupWebView(){
2             final String centerURL = "javascript:centerAt(" +
3                           mostRecentLocation.getLatitude() + "," +
4                           mostRecentLocation.getLongitude()+ ")";
5             final String GotoLatLng = "javascript:GotoLatLng(" +
6                                   mostRecentLocation.getLatitude() + "," +
7                                   mostRecentLocation.getLongitude()+ ")";
8             webView = (WebView) findViewById(R.id.webview);
9             //webView.getSettings().setJavaScriptEnabled(true);
10            //Wait for the page to load then send the location information
11            webView.setWebViewClient(new WebViewClient());
12            webView.loadUrl(MAP_URL);
13          }
14
15          private class JavaScriptInterface {
16                  public double getLatitude(){
17                    return mostRecentLocation.getLatitude();
18                  }
19                  public double getLongitude(){
20                    return mostRecentLocation.getLongitude();
21                  }
22          }
```

Figura 21 - setupWebView e JavaScriptInterface
Fonte: Autoria Própria

O procedimento setupWebView é o responsável por carregar a URL do mapa a ser utilizado como base, e através dele é possível definir outras informações como zoom inicial, localização inicial, o qual usamos para mostrar a localização das escolas estaduais, entre outras opções. Quanto a localização a ser exibida no mapa, é utilizado de um JavaScript para efetuar esta requisição, sendo assim necessário a criação de uma interface JavaScriptInterface, a qual é responsável por fazer a comunicação entre os diversos dispositivos existentes com a API do *Google Maps*. Sem a utilização de um JavaScript para setar a posição inicial no mapa, por padrão o mesmo seria criado com as coordenadas(0,0).

3.2.6 Testes

Durante o desenvolvimento do sistema, foram executados vários testes unitários. Esses testes tem como objetivo verificar pequenas alterações durante o desenvolvimento e previnir de serem encontrados erros nos testes futuros.

Os testes de caixa-branca vieram em seguida. Ao término da programação e ficando os detalhes a serem corrigidos, foram escolhidas rotinas críticas para o sistema e executada a verificação destes códigos através do modo *debug*.

Por fim, o teste de caixa-preta identificou as últimas inconsistências no funcionamento do sistema. Sua aplicação teve como objetivo testar apenas a interface do sistema, sem levar em consideração o código fonte.

Vale ressaltar que, mesmo que um sistema seja muito bem testado, ele nunca estará livre de falhas.

4 RESULTADOS

Neste capítulo observamos os resultados obtidos com o projeto, passando pelo C*rawler* em funcionamento, a geração do KML final, disponibilização na *web* através de um servidor FTP e o resultado final no aplicativo móvel Android, o qual exibe as localizações de todas as escolas estaduais que foram possíveis de ser encontradas através dos algoritmos usados.

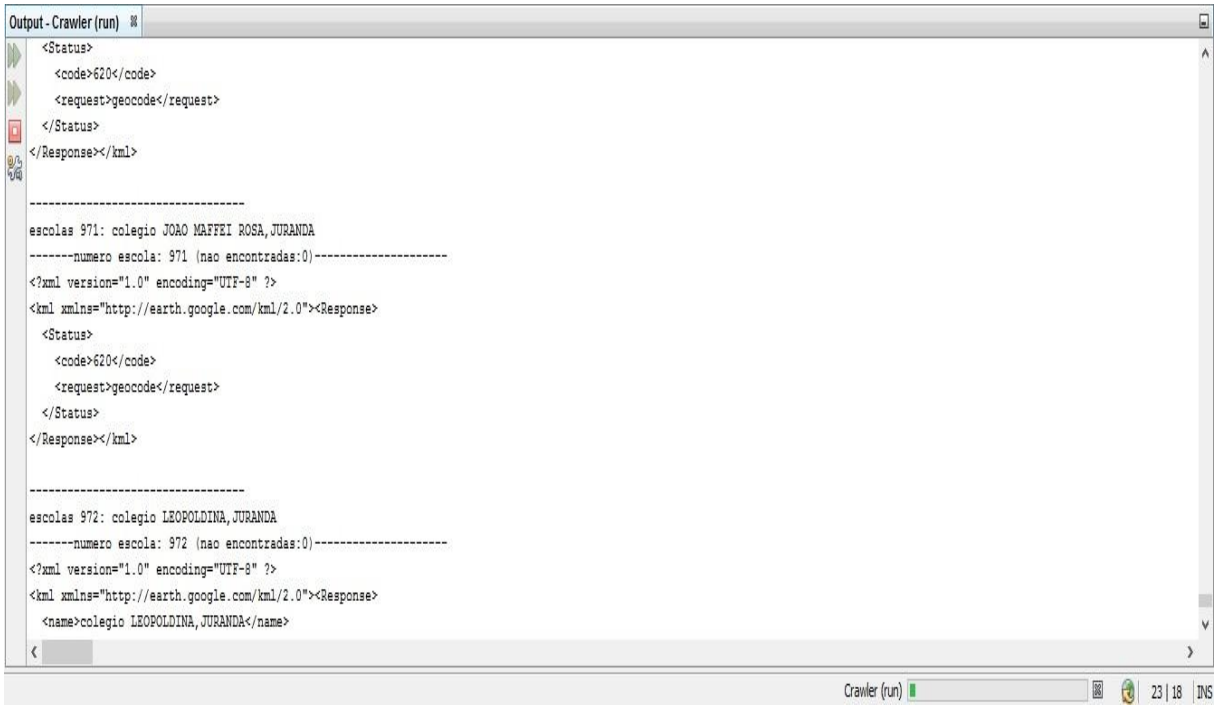

Figura 22 - Robô em execução
Fonte: Autoria Própria

Na Figura 22 é apresentado o C*rawler* em execução, após a obtenção das localizações das escolas. Nela aparece o robô tentando recuperar os arquivos KML's referentes a cada escola incluída na busca, retornando mensagens de erro sempre que não for possível localizar algum dos KML's. A cada KML recuperado o mesmo é mostrado na tela em seu formato original.

Podemos observar na Figura 23 todos os KML's que foram obtidos e salvos em uma respectiva pasta dentro do projeto, o qual serão acessados para a agregação dos mesmos em seguida.

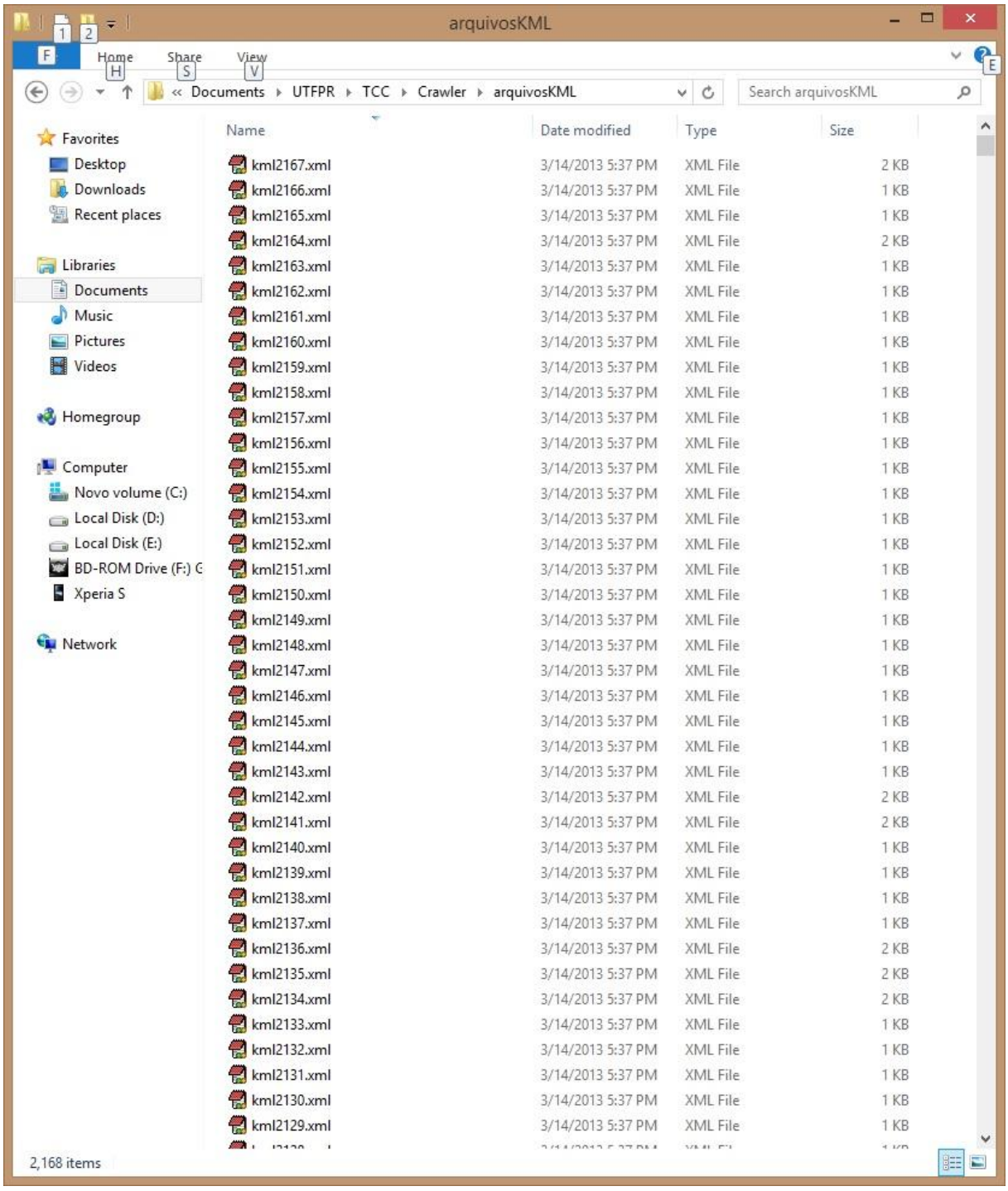

Figura 23 - Arquivos KML
Fonte: Autoria Própria

É possível perceber pela imagem que apesar da grande quantidade de arquivos recuperados, o espaço de armazenamento em disco exigido é pequeno, já que cada arquivo KML ocupa um espaço quase insignificante.

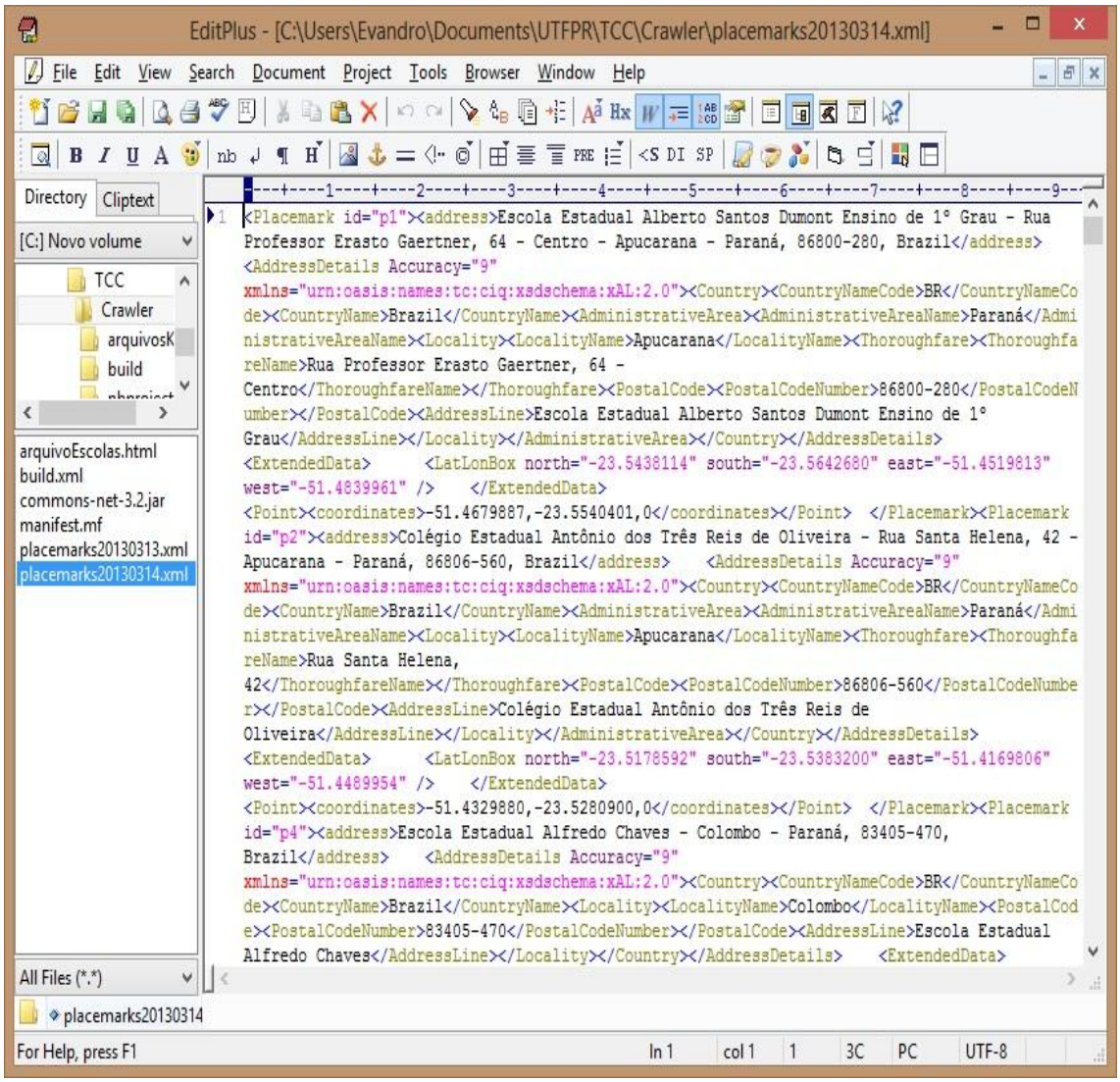

Figura 24 - KML's agregados
Fonte: Autoria Própria

Na Figura 24 pode-se ver o resultado da agregação de todos os KML's das escolas estaduais recuperados até então. Nele estão contidos várias tags <Placemark> com suas respectivas id's, cada uma referente a uma escola estadual diferente, onde todo o conteúdo entre o início e o fim da tag <Placemark> são respectivamente as informações da escola, como o nome, o endereço e outras informações.

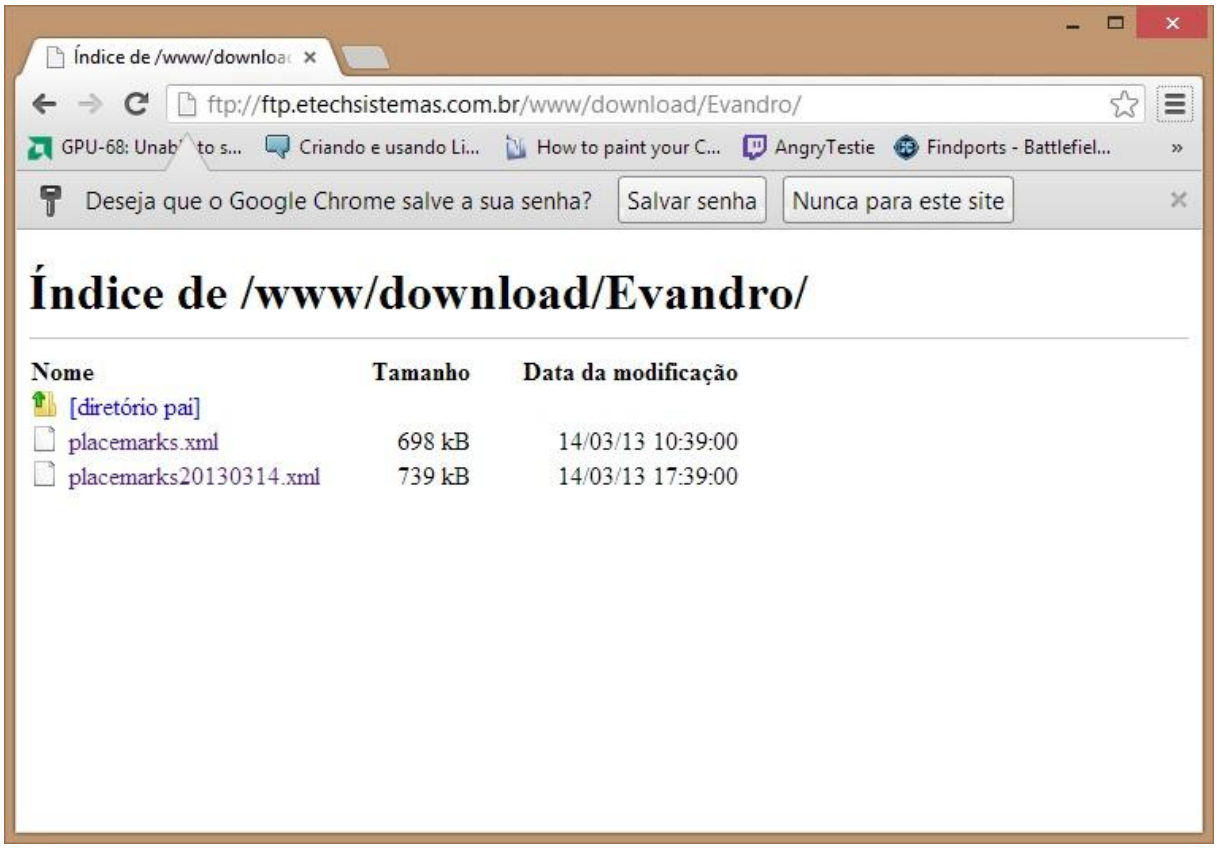

Figura 25 - FTP
Fonte: Autoria Própria

Observamos na Figura 25 o FTP utilizado para a armazenagem do arquivo KML que será efetuado o *download* pelo aplicativo móvel. Pode-se notar que os arquivos não são excluídos a cada nova versão.

Figura 26 - Lista das Escolas Estaduais do Paraná
Fonte: Autoria Própria

Na tela da Figura 26 encontra-se uma lista com todas as escolas estaduais contidas no arquivo KML recuperado do servidor FTP. Ao selecionar uma escola o aplicativo redireciona para a ativiidade seguinte da API do *Google Maps*.

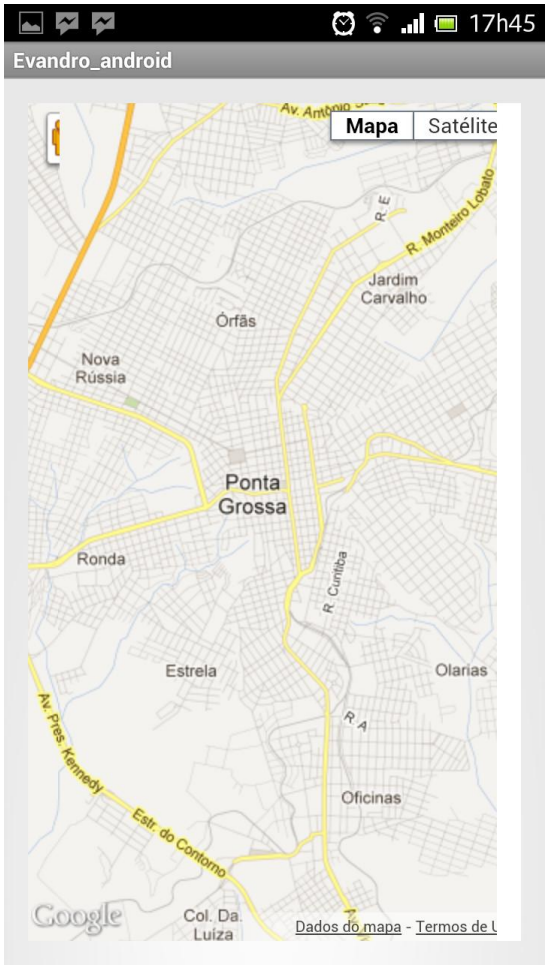

Figura 27 - Mapa do Google Maps API
Fonte: Autoria Própria

Na Figura 19 temos a tela na qual mostra um mapa da API *Google Maps*, sendo este posicionado de acordo com a escola estudual escolhida na atividade anterior.

Durante o desenvolvimento foi possível notar que alguns pontos poderiam ser melhorados, como por exemplo o algoritmo de busca pelo KML das escolas, os quais diversos foram recuperados sem as informações das escolas, por conta de não ter sido possível localiza-lás com o algoritmo atual.

No total, o estado do Paraná conta com duas mil cento e sessenta e sete escolas estaduais em funcionamento no ano de 2013, das quais aproximadamente trinta por cento não foi possível recuperar as informações no arquivo KML.

5 CONCLUSÃO

O objetivo inicial de implementar um sistema com conceitos de *Web Crawler* para extração de dados específicos tendo os mesmos disponíveis para dispositívos mobiles foi alcançado. O *Crawler* e os outros elementos necessários para o projeto foram implementados. Existe a possibilidade de ser feita busca pelas escolas estaduais do Paraná, assim como pelo KML das mesmas.

O *crawler* apresentou um bom desempenho na sua execução, obtendo o resultado final em um curto espaço de tempo, e com um grande número de requisições. Cerca de aproximadamente trinta por cento (30%) das informações não foram obtidas. Desse modo, podemos considerar como um trabalho futuro um estudo para a otimização do algoritmo de busca das informações, visando aumentar o desempenho da mesma.

A criação do C*rawler* foi de grande importância para entendimento dos conceitos de indexação de páginas *web*, compreendendo assim como funcionam os grandes motores de busca *web* e até mesmo de extração de dados e estatísticas. Já o desenvolvimento Android com a API do Google *Maps* foi uma iniciativa significativa para o aprendizado em desenvolvimento *mobile*.

A contribuição desse trabalho é servir de exemplo para desenvolvimento utilizando uma API popular e útil, que agrega diversos serviços, além dos mostrados nesse trabalho. E também, o presente trabalho mostrou uma forma efetiva de como é feita a construção de um *Web Crawler* puro, usando conceitos simples e classes nativas Java.

REFERÊNCIAS

FITZ, Paulo Roberto. **Geoprocessamento sem complicação**. São Paulo: Oficinas de Textos, 160p. 2008.

MATOS, P. P.; CARMO, M. B.; AFONSO, A. P. **Visualização de Informação Georeferenciada em Dispositivos Móveis**. EPCG, 2007. Disponível em: <http://homepages.di.fc.ul.pt/~bc/artigos/epcg07-pombinho-bc-apa.pdf>. Acesso em: 06 mar. 2013.

MENDONÇA, A. L. A; SCHMIDT, M. A. R; DELAZARI, L. S. **Publicação de mapas na web: abordagem Cartográfica com uso de tecnologias código-Aberto**. Boletim de Ciências Geodésicas. Curitiba, v. 15, n. 1, p.103-119, jan. 2009. Disponível em:<http://ojs.c3sl.ufpr.br/ojs2/index.php/bcg/article/view/13903/9358> Acesso em: 13 fev. 2013.

ARASU, Arvind; et al. **Searching the Web**. Relatório Técnico, 2000, Stanford. Disponível em: <http://ilpubs.stanford.edu:8090/457/1/2000-37.pdf>. Acesso em: 06 mar. 2013.

ROSA, Paulo. **Cartografia Básica**. Universidade Federal de Uberlândia, fev. 2004. <http://www.uff.br/cartografiabasica/cartografia%20texto%20bom.pdf>. Acesso em: 09 mar. 2013.

CORRÊA, Iran Carlos Stalliviere. **Coordenadas Geográficas**. Departamento de Geodesia - UFRGS. Porto Alegre, maio de 2009. <http://www.ufrgs.br/museudetopografia/Artigos/Coordenadas_geogr%C3%A1ficas.pdf>. Acesso em: 12 mar. 2013.

GOOGLE Inc. **Keyhole Markup Language**, 2013 <https://developers.google.com/kml/?hl=pt-br>. Acesso em: 11 mar. 2013.

GOOGLE Inc. **Keyhole Markup Language:** Tutorial do KML, 2013 <https://developers.google.com/kml/?hl=pt-br>. Acesso em: 11 mar. 2013.

GOOGLE Inc. **Serviços da Web da API do Google Maps**, 2013. <https://developers.google.com/maps/documentation/webservices/?hl=pt-br>. Acesso em: 13 mar. 2013.

CARVALHO, Leandro A. **Integração da API Google Maps com HTML5 e PHP**, UTFPR, Medianeira, 2011. <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/601/1/MD_COADS_2011_2_06.pdf>. Acesso em: 12 mar. 2013.

SILVA, L. E. P. **Utilização da Plataforma Android no Desenvolvimento de um Aplicativo para o Cálculo do Balanço Hídrico Climatológico**, UEMGS, nov. 2009. <https://bhcmovel.googlecode.com/files/TCC%20-%20Final.pdf>. Acesso em: 12 mar. 2013.

FISCHER, Tiago Roberto. **Implementação de um Crawler Incremental Distribuído**: Um Sistema de Busca na Web, 2006. <http://www.facebook.com/l.php?u=http%3A%2F%2Fwww.bc.furb.br%2Fdocs%2FMO%2F2006%2F306746_1_1.pdf&h=VAQGZiFIo>. Acesso em: 13 mar. 2013.

CHO, Junghoo. **Crawling the Web**: Discovery and Maintenance of Large-scale Web Data, 2001. <http://www.facebook.com/l.php?u=http%3A%2F%2Foak.cs.ucla.edu%2F~cho%2Fpapers%2Fcho-thesis.pdf&h=VAQGZiFIo>. Acesso em: 13 mar. 2013.

SCHEMBERGER, Elder Elisandro; FREITAS, Ivonei; VANI, Ramiro. **Plataforma Android**. UNIOESTE, 2009. <http://www.jornaltech.com.br/wp-content/uploads/2009/09/Artigo_Android.pdf>. Acesso em: 14 mar. 2013.

BATISTA, Dorival José. **Localização Geográfica em Mapa Digital de Núcleos do Programa Segundo Tempo Através de Referência Espacial Indireta e API's de Webservice Google Maps**, 2011. <http://www.facebook.com/l.php?u=http%3A%2F%2Fwww.espweb.uem.br%2Fwp%2Fwp-content%2Fuploads%2F2012%2F05%2FDorival-Jos%25C3%25A9-Batista.pdf&h=VAQGZiFIo>. Acesso em: 14 mar. 2013.

APACHE Commons Net. **Biblioteca commons-net-3.2.jar**. 2012. <http://commons.apache.org/proper/commons-net//download_net.cgi>. Acesso em: 14 mar. 2013.

CAMARGO, Francini D. M.; ZIEGLER, Valdair. **Uma Experiência em Aplicações Georeferenciadas para Empresas de Monitoramento de Alarmes**. Trabalho de Conclusão de Curso - Universidade Tecnológica Federal do Paraná. Ponta Grossa(PR), maio 2008.

SILVA, Marcelo. **UML - Conceitos e Casos de Uso**, mar. 2012. <http://marcelosilvaeti.blogspot.com.br/2012/03/uml-conceitos-e-casos-de-uso.html>. Acesso em: 15 mar. 2013.