UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA
ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

LUIZ MARCELO BALDI

APLICAÇÃO DE LOCALIZAÇÃO PARA PRESTADORES DE SERVIÇO NA AREA DE ALIMENTAÇÃO BASEADOS NA LOCALIZAÇÃO LOCATION-BASED SERVICES (LBS)

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITBA
2011

LUIZ MARCELO BALDI

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA LOCALIZAÇÃO DE FAST FOOD ESPECIALIZADO EM CACHORRO QUENTES

Monografia apresentada ao Curso de Especialização em Tecnologia Java da Universidade Tecnológica Federal do Paraná, como requisito para conclusão de curso.

Orientadores:
Prof. Leandro Batista de Almeida
Prof. Robson Ribeiro Linhares

CURITBA

2011

AGRADECIMENTOS

Agradeço a minha família, e principalmente a minha esposa Renata que me apoiou e me incentivou mesmo nos momentos mais difíceis, e, ao meu irmão Luciano que sempre está pronto para ajudar.

Baldi, Luiz Marcelo. DESENVOLVIMENTO DE UMA APLICAÇÃO PARA LOCALIZAÇÃO DE FAST FOOD **(**2011).Monografia (Especialização Tecnologia Java - Programa de Pós-Graduação em Tecnologia da Informação, Centro Federal de Educação Tecnológica do Paraná.

Curitiba, 2011.

RESUMO

Devido à constante evolução das tecnologias relacionadas aos *smartphones* e *tablets,* a demanda por busca de localização geográfica de serviços tornou-se cada vez maior e consequentemente chamaram a atenção das empresas que se dedicam à pesquisa e desenvolvimento de aplicações voltadas a este mercado emergente.

Neste contexto, um grande destaque é o Sistema Operacional Android, patrocinado pelo Google, cujos conceitos foram utilizados para o desenvolvimento deste trabalho.

Como modelo de aplicação para Android, o Catch Dog é uma ferramenta que permite localizar barracas de cachorros quentes na cidade do usuário, bastando que para isso este possua um aparelho com o sistema instalado e em contrapartida empresários inscritos para serem localizados desta forma. A ideia de buscar esse tipo de serviço, cachorro-quente, serve apenas como molde para a diversidade de serviços existentes nos centros urbanos.

Assim, este trabalho foi implementado utilizando o emulador de celular disponível no kit de desenvolvimento do *Android*, possibilitando que sejam construídas um rol amplo de aplicações baseados nesta tecnologia.

ABSTRACT

Due to the constant evolution of technologies related to smart-phones and tablets, the demand for geo-location search service has become increasingly larger and consequently caught the attention of companies engaged in research and development of applications related to this emerging market.

In this context, a major highlight is the Android Operating System, sored by Google, whose concepts were used to develop this work.

As a model application for Android, Catch Dog is a tool that allows to locate hot dog stands in the city of the user. It's only necessary that the device has the system installed and registered business to be located in. The idea of seeking this type of service, hot dogs, is only an example for the diversity of existing services in urban centers.

This work was implemented using the emulator cellular available in the An-droid development kit, allowing it to be build for a broad list of applications based on LBS (location-based services).

ÍNDICE

Índice de Figuras

INTRODUÇÃO

1.1) MOTIVAÇÃO

O uso intensivo da tecnologia fez da mobilidade uma máxima entre os usuários desta funcionalidade. Assim, *notebooks*, laptops, *handhelds*, smartphones e tablets, cada vez menores possibilitaram que fossem carregados para onde fossem necessários.

Além da miniaturização dos *hardwares*, a evolução do *software* precisou acompanhar a necessidade de consumo de informação de uma nova sociedade totalmente dependente de tecnologia, para trabalho ou lazer.

A grande demanda por esta tecnologia aumentou também a oferta, tornando o acesso facilitado a dispositivos móveis à praticamente toda a população, que antes procurava apenas telefones móveis, agora busca serviços e fontes de informação e entretenimento.

A expansão da necessidade dos usuários levou seus aparelhos móveis celulares à integração de várias funcionalidades como o GPS, mas saber apenas como chegar a determinado local já não é mais o suficiente. Será necessário que a informação seja cada vez mais rápida e objetiva, como localizar o que existe no endereço desde o momento da busca. Assim, o usuário fará a busca pelo serviço que precisa e saberá na sequência como chegar ao endereço, e não o contrário.

1.2) OBJETIVO

A evolução constante das tecnologias móveis, tanto em *hardware* como *software*, vem ampliando a capacidade dos dispositivos, oferecendo uma nova gama de recursos.

O tema deste estudo consiste na evolução da tecnologia de localização, LOCATION-BASED SERVICES (LBS), onde o *software* possibilita a localização através de mapas do serviço pretendido. O crescimento da utilização da Plataforma Android nos *smartphones* e *tablets* em conjunto com os recursos disponibilizados para o desenvolvimento de aplicações móveis fornecido pelo Google, permitiu um grande avanço na padronização e no desenvolvimento, sendo um dos motivos da escolha dessa tecnologia neste estudo.

O objetivo deste trabalho será a elaboração de um protótipo cuja aplicação oferecerá os serviços baseados na localização, utilizando a Plataforma Android disponibilizada pelo Google para seu desenvolvimento.

Assim, o modelo escolhido na aplicação para Android neste trabalho, chamado Catch Dog, permite localizar barracas de cachorros quentes na cidade do usuário, servindo como molde para a diversidade de serviços existentes nos centros urbanos e possibilidades de expansão dessa forma de localização.

1.3) ESTRUTURA TEXTUAL

O texto foi organizado inicialmente com uma abordagem teórica dos conceitos da tecnologia de localização, noções de *hardware*, software e a evolução desta tecnologia. Na sequencia, a parte prática é elaborada gradualmente conforme o desenvolvimento da proposta do *software*, objetivo do presente trabalho.

No capítulo 2, as definições que tratam dos LBS (location-based services) serão trabalhadas de maneira abrangente, mostrando as áreas onde são aplicados e as potencialidades existentes para esta tecnologia ao uso destes. Será analisado como esta tecnologia funciona e como serve de base para estes serviços, onde é apresentado como exemplo o Sistema de Posicionamento Global

(GPS), abordando outras características desses serviços, bem como suas categorias.

No capítulo 3, uma vez que o trabalho foi baseado na Plataforma Android, a descrição desta tecnologia será trabalhada, tratando de sua constituição e as possibilidades das para o desenvolvimento de *software.* Esta plataforma disponibiliza bibliotecas (API) que permitem a utilização de suas classes para consultar seus mapas possibilitando a integração de aplicações com propósito específico.

No capítulo 4 são descritas as principais características da plataforma Android, sua origem, seu propósito e seu futuro, devido a sua versatilidade tanto a nível operacional (software) quanto comercial (aplicativos).

No capítulo 5 é abordado o desenvolvimento do *software*, descrevendo sua modelagem, tratando de aspectos de sua arquitetura e o modo como foi organizado. Também é descrita a implementação do modelo envolvendo todos os quesitos necessários para comprovar a sua viabilidade.

A conclusão deste trabalho é destacada no Capítulo 6, demonstrando a impressão do autor sobre o trabalho desenvolvido utilizando essa tecnologia para o protótipo do *software* Catch Dog.

2) LOCATION-BASED SERVICES

O surgimento de tecnologias capazes de dar a exata localização de um dispositivo móvel (celulares e navegadores GPS) criou possibilidades para que novos serviços surgissem também nesta área e, assim, nasceram os *Location Based Services* (LBS), que também podem ser definidos como georeferenciamento de serviços.

No início deste capítulo serão esclarecidas as principais definições para o georeferenciamento dos serviços, na sequência como são utilizadas essas características, e finalmente como utilizar essas técnicas como opção ao GPS.

2.1) DEFINIÇÃO DE LOCATION-BASED SERVICES (LBS)

Os serviços de localização podem ser definidos como serviços que integram a localização de um dispositivo móvel com outras informações de modo a proporcionar rapidez na consulta baseada em georeferenciamento, transmitindo para o usuário a localização dos serviços que sejam de utilidade para suas necessidades diárias.

Apesar de parecer um tema atual, os serviços baseados na localização remontam a década de 1970. O primeiro sistema desenvolvido foi o GPS (Sistema de Posicionamento Global), inicialmente com fins militares, a pedido do Departamento de Defesa dos EUA. Na década de 1980 o governo dos EUA decidiu liberar o sistema para uso civil e logo a indústria do mundo inteiro, nos mais diversos ramos começou a utilizá-lo como um diferencial em seus produtos e serviços.

Devido ao grande potencial das tecnologias móveis que podem ser utilizadas em um mercado em constante expansão, o Sistema de Posicionamento

Global (GPS) acabou atraindo a atenção para a pesquisa e o desenvolvimento de aplicações voltadas para os Serviços Baseados em Localização.

Dependendo do enfoque encontraremos definições diferentes para o *Location Based Services*, apesar de tratarem do mesmo assunto. O *Group Special Mobile* (GSM WORLD, 2011) define LBS como sendo a prestação de serviços personalizados a pedido do usuário, com base em sua posição atual. Isto pode incluir informações sobre restaurantes, hotéis ou conteúdos específicos do local, tais como mapas, sendo ativados automaticamente pela aproximação de uma determinada localização.

A *Group Special Mobile Association* (GSMA, 2011) é uma associação comercial global que representa os interesses de mais de 800 operadoras de telefonia móvel GSM e mais de 200 fabricantes e fornecedores de *hardwares* e software voltadas para esta área em todo o mundo.

Dentre as entidades existentes podemos citar a *3rd Generation Partnership Project* (3GPP, 2011) criada em 1988, que foi idealizada para padronizar os arquivos multimídia (criação, envio e reprodução vídeos) em telefones celulares e outros aparelhos *wireless* GSM.

Ao acessar a página principal da 3GPP percebe-se que sua intenção original era produzir Especificações Técnicas e Relatórios Técnicos para um Sistema 3G *Mobile* baseado em redes GSM e as tecnologias de acesso de rádio que são utilizadas, ou seja, *Universal Terrestrial Radio Access* (UTRA) ambos os modos *Frequency Division Duplex* (FDD) e *Time Division Duplex* (TDD).

O escopo foi posteriormente alterado para incluir a manutenção e desenvolvimento do Sistema Global para Comunicações Móveis (GSM) e as Especificações Técnicas e Relatórios Técnicos, incluindo tecnologias de rádio e sua evolução de acesso (por exemplo *General Packet Radio Service* (GPRS) e *Enhanced Data Rates for GSM Evolution* (EDGE). O enfoque é a obtenção da

localização das pessoas ou serviços (alvos), tornando disponíveis os resultados dessas informações para atores externos, preocupando-se apenas com os dados referentes a localização.

Nota-se que a localização de serviços pode ser um serviço baseado na localização ou um sub serviço do mesmo, pois nesse caso se não houver um serviço de localização, o *Location Based Services* deverá requisitar ao usuário uma localização. Esse conceito de serviço é chamado *Context-Aware Services* (sensibilidade ao contexto).

Os *Location Based Services* (LBS) se definem na localização da posição ou na localização do dispositivo móvel e as informações que forem necessárias para que o usuário possa usá-las para localizar os serviços existentes em uma região, baseada em aplicações desenvolvidas para essa finalidade, agregando valor a informação solicitada.

2.2) APLICABILIDADE E USO

Os *Location Based Services* são divididos em duas áreas de aplicação: públicas e comerciais. A primeira consiste na aplicação dessa tecnologia para atender a necessidade do cidadão em relação aos serviços prestados pelo poder público, oferecendo qualidade e agilidade com redução de custos.

Para o setor comercial a principal motivação de sua utilização é obter lucro utilizando o potencial de atração dessa nova tecnologia, levando o consumidor a utilizar os serviços prestados por um tempo bem longo, estabelecendo uma relação de custo/benefício para o usuário, refletindo numa lucratividade bem maior para quem oferecer esses serviços baseados na localização. Abrindo várias possibilidades, desde o fornecimento do serviço propriamente dito ou fornecer dados para análise de novos empreendimentos em uma determinada região.

O GPS foi o primeiro sistema desenvolvido para atender primeiramente as necessidades dos militares norte americanos, mas na década de 1980 o governo dos EUA decidiu torná-lo disponível para o mundo, desencadeando uma onda de inovações em torno das tecnologias que utilizavam o posicionamento via satélite, pois, seria mais vantajoso para o governo se o setor privando desenvolvesse novas tecnologias reduzindo custos de desenvolvimento para o mesmo, tornando o modelo americano um padrão desenvolvimento para os setores privado e público.

Ao adotarem o padrão americano, inúmeras indústrias e governos tornaram-se dependentes dele, sendo utilizado numa ampla gama de aplicativos, que vão desde o controle de tráfego aéreo, marítimo e terrestre ajudando na navegação e também auxiliando na gestão logística de recursos (fretes e alocação de material) chegando até aos serviços públicos de emergência.

Hoje é possível localizar pessoas em situação de risco através dos celulares, porque as operadoras de telefonia têm condições de determinar a localização da chamada. Outra forma de aplicação são sistemas desenvolvidos e amplamente utilizados para cobrança de pedágios (CONTROLE DE PEDÁGIOS, 2011).

As necessidades dos governos tanto na esfera de atendimento ao cidadão quanto na esfera administrativa oferece várias oportunidades para o setor privado possibilitando o desenvolvimento de aplicações e sistemas que atendam essas necessidades.

Dessa forma, serve de incentivo ao setor privado investir em pesquisa e desenvolvimento de *software* baseados no *Location Based Services (*LBS), propiciando um aumento de receita, devido ao amplo espectro de serviços que podem ser oferecidos.

A computação móvel vem crescendo nos últimos anos e a constante evolução das transações comerciais na *Internet* (*e-business*) que estão migrando para os dispositivos móveis (*mobile-business)*, os celulares, *smartphones* e agora os *tablets* possuem a capacidade de acessar informações em qualquer lugar e a qualquer momento.

Baseado em todos os exemplos expostos até agora nas mais diversas áreas percebemos que o uso do *Location Based Service*, além de aumentar a receita das empresas oferece aos cidadãos atendidos pelos governos e aos consumidores uma nova gama de serviços.

2.3) CONSIDERAÇÕES FINAIS

As várias situações e aspectos mostrados neste capítulo constituem apenas uma parte das possibilidades do *Location Based Services*, pois devido ao vasto campo de aplicação dessa tecnologia e a diversidade de soluções que podem ser oferecidas fica difícil cobrir toda a capacidade de desenvolvimento e pesquisa sobre a tecnologia LBS, que foi utilizada para o desenvolvimento do aplicativo descrito neste trabalho.

3) ANDROID - INTRODUÇÃO

Neste capítulo será possível ver algumas características do Sistema *Android* desenvolvido pelo *Google*, baseado no Sistema Operacional Linux, que é a plataforma utilizada pela aplicação descrita neste trabalho. As razões de sua escolha são muitas, entre elas podemos citar o fato de utilizar a linguagem de programação Java, ser de código aberto e estar em constante evolução tanto em vendas quanto em desenvolvimento de novas tecnologias baseada nessa plataforma.

3.1) CONHECENDO O *ANDROID*

Android é um *software* que foi desenvolvido para dispositivos móveis sendo constituído por: um sistema operacional, *midleware* e aplicativos. Além disso possui um Kit de Desenvolvimento de Programas (SDK) que fornece ferramentas e as APIs necessárias para o desenvolvimento de aplicações na plataforma Android usando a linguagem Java.

Esse sistema operacional não possuia uma máquina virtual Java (JVM), então o Google desenvolveu uma máquina virtual chamada DALVIK, que foi otimizada para executar nos dispositivos móveis que funcionam junto com um *kernel* Linux, permitindo que os programas desenvolvidos para o *Android* tenham grande portabilidade, isto é, possam ser utilizados em qualquer dispositivo *Android*, independentemente do *hardware* utilizado.

3.2) AS ORIGENS DO *ANDROID*

A história do *Android* começou quando o Google comprou a Android Inc, co-fundada por Andy Rubin, em julho de 2005, no Vale do Silício, na Califórnia. Logo surgiram especulações sobre quais seriam suas intenções. Como a *Apple* já tinha desenvolvido seu *smartphone*, era natural que o Google percebesse o potencial disso e passasse a desenvolver o seu também. Devido a essa aquisição gerou expectativas no mercado para o surgimento do *gPhone* (*Google's Phone*), o qual agregaria além das funções normais uma série de aplicativos como *Google Maps, Gmail,* GPS, etc.

Então no início de novembro de 2007 o Google anunciou a criação de um consórcio de empresas, num total de trinta e quatro empresas, como *Texas Instruments*, *Intel*, *T-Mobile, Sprint* e *Nextel*, para juntos criarem uma interface baseada em *software* de código aberto.

Esse grupo originou o *Open Handset Aliance* (OHA, 2011), que logo após lançou o *Android Developer's Kit* em seu *site*, e anunciou o *Developer Challenge*, com US$ 10 milhões de dólares em prêmios a serem divididos entre os criadores das melhores aplicações para o novo sistema.

Como já existiam milhões de *Iphones* da *Apple* no mercado, a única maneira de rivalizar e obter uma grande fatia dele seria ter vários fabricantes despejando ao mesmo tempo uma grande quantidade de smartphones utilizando o *Android*.

No início de novembro de 2008, o grupo *Open Handset Aliance* (OHA, 2011), anunciou a adesão de mais 14 empresas. Isto demonstrou ao mercado o poderoso apoio ao *Android* como uma plataforma móvel e seu compromisso para com o seu sucesso comercial.

3.3) PORQUE USAR O *ANDROID*

Atualmente existem muitas plataformas móveis no mercado, dentre elas destacam-se, *Symbian, iPhone*, o *Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile* (*Limo*), e muito mais. Então porque o *Google* criou mais uma plataforma? Abaixo temos algumas características que tornam o *Android* especial em relação as outras, embora algumas já existissem ele é o primeiro ambiente que juntou todas combinando-as:

- Baseada na Plataforma Linux que é aberta, livre para o desenvolvimento permitindo aos fabricantes de dispositivos móveis personalizarem a plataforma sem pagamento de royalties e garantindo aos desenvolvedores que será uma evolução constante sem apresentar obstáculos

- Baseado na arquitetura inspirada pela Internet chamada *mash-ups*, isto é, algumas partes de uma aplicação podem ser reutilizadas de outra forma não prevista inicialmente pelo desenvolvedor da aplicação. Possibilitando a substituição dos componentes internos com seus próprios componentes.

- Disponibiliza aos usuários vários serviços internos baseados na localização tais como GPS, *Location Based Services* e Mapas. Através da integração de um visualizador de mapas com aplicações pode-se integrar todas essas capacidades, criando uma vasta gama de funções com baixo custo de desenvolvimento.

- Os programas são combinados na forma de pilha, formando camadas seguras, isolados uns dos outros, aumentando a estabilidade, liberando a memória de acordo com a utilização, ora fechando alguns ou colocando em segundo plano de acordo com a ordem de prioridade. Além de ser otimizado para economizar energia

- Apresenta alto desempenho e ótima qualidade nos gráficos de duas e três dimensões e animações em flash. Também possui integrados os *codecs* mais comuns para áudio e vídeo adotados como padrão pela indústria.

- Possui uma vasta compatibilidade com *hardwares* atuais e em desenvolvimento, porque seus aplicativos são escritos em Java e possuem em comum a máquina virtual desenvolvida pelo Google, a *Dalvik*, suportando várias arquiteturas. Devido a esses fatores possui uma compatibilidade bem grande com os vários tipos de dispositivos básicos de entrada existente: teclado, câmera, voz e etc. Oferecendo ao usuário uma interface de fácil personalização permitindo uma interação muito intuitiva.

3.4) DISPOSITIVOS MÓVEIS NA ATUALIDADE

Acompanhamos diariamente o crescimento constante das vendas de telefones celulares, estudos estimam que mais de 3 bilhões de pessoas possuam um celular.

O segmento que mais cresce na telefonia móvel é o de smartphones e tablets, o primeiro possui um formato mais compacto, os modelos variam de 4 a 5 polegadas, o segundo é bem maior ficando em torno de 10 polegadas, além de possuírem um *design* atraente, ambos possuem várias características dos computadores, tornando-se um diferencial muito atrativo para os consumidores.

Somente no primeiro trimestre deste ano de acordo com o Gartner Group (Technology Research/Gartner Inc, 2011), foram vendidos 427,8 milhões de dispositivos móveis, representando um acréscimo nas vendas de 19%, comparando-se com o mesmo período de 2010. Desse total, os smartphones foram responsáveis por 23,6% das vendas de celulares no primeiro trimestre de 2011, um aumento de 85% em relação ao último ano.

Neste ano foram vendidos nos últimos quatro meses 36,2 milhões de equipamentos que utilizam o *Android* como o Sistema Operacional, o que representou uma de participação de 36% do mercado.

Baseando-se nesses dados percebemos porque o desenvolvimento de novas aplicações para esta plataforma se torna atraente.

3.5) ANDROID COM O LOCATION BASED SERVICES

A plataforma *Android* disponibiliza os recursos necessários para utilização dos *Location Based Services* através das APIs que se encontram nos pacotes *android.location* e *com.google.android.maps*.

O pacote *android.location* contem as informações necessárias para obter a localização do aparelho, o com.*google.android.maps* controla e fornece os meios para apresentar os mapas. Para as aplicações que façam uso das funcionalidades de *Location Based Services* deverão utilizar as classes existentes nesses dois pacotes.

Se uma aplicação for desenvolvida para oferecer serviços baseados na localização é necessário que ela obtenha a posição atual do aparelho e o responsável por isso é o pacote *android.location*. A classe *LocationManager* que faz parte desse pacote exerce o papel central, ela provê uma lista dos provedores de localização disponíveis e registra de como receber essas informações sobre a localização, definindo o local através da latitude e da longitude. O *Google Maps* disponibiliza uma biblioteca de mapas para exibir na tela. Para isto é necessário obter uma chave (CHAVE PRIVADA PARA MAPVIEWS, 2011) e registrá-la **(**REGISTRANDO UMA API KEY PARA MAPVIEW, 2011) e para usá-la numa aplicação.

Aplicações desenvolvidas usando o *Location Based Services* trabalham com coordenadas geográficas para determinar a localização do aparelho, mas nós sabemos que na vida cotidiana das pessoas elas não usam com coordenadas geográficas, mas sim com os endereços das ruas para localizar o local onde querem chegar. Como os serviços por localização trabalham com coordenadas surge a necessidade de converter as coordenadas geográficas com os endereços nas cidades.

Para poder trabalhar com esta situação é fornecido no pacote de localização a classe *Geocoder* que se encarrega de tratar estes tipos de informações. Como vimos também é disponibilizada uma biblioteca do *Google Maps* (*com.google.android.maps*) para que se tenha acesso aos serviços existentes nele, onde é oferecida uma classe que desempenha papel central que é a *MapView*, que gerencia como uma aplicação vai trabalhar com os mapas, lembrando sempre que é necessário obter uma chave para poder desenvolver o *software* que usufruirá desses serviços através do Google em seu site (*http://code.google.com/intl/pt-BR/apis/maps/signup.html).*

3.6) CONCLUSÕES FINAIS

Percebemos que esta plataforma possui uma grande variedade de temas que não são citados aqui, pois, estaríamos fugindo do escopo do trabalho. Foram citados apenas as partes que eram necessárias para a compreensão desta tecnologia e como se relacionam essas características com o tema do presente trabalho. Agora depois de fundamentar a parte teórica que embasa este estudo, passaremos no próximo capítulo para o desenvolvimento da aplicação Catch Dog.

4) A APLICAÇÃO E SEU MODELO

Será trabalhado neste capítulo a demonstração da modelagem proposta para a aplicação *Cach Dog*, para consultar a localização dos *fast food* especializados em cachorro quente presentes em uma determinada região, a partir de um aparelho celular (*smartphone*) baseado no *Android* usando os serviços baseados na localização como citado nos capítulos anteriores.

Inicialmente será tratado o problema principal para que o *software* desenvolvido encontre uma solução de acordo com a proposta do trabalho. Tendo como tema central a resolução do problema como base para o desenvolvimento da modelagem. O modelo será estruturado no início baseado na arquitetura pesquisada demonstrando como ela funciona, detalhando as partes principais da aplicação e como o modelo se divide.

4.1) CONSIDERAÇÕES INICIAIS

Para desenvolver todo o processo de elaboração do *software* ele foi dividido em quatro etapas: identificação do problema, análise do problema e sua solução, análise dos requisitos e suas restrições e finalmente a implementação do projeto.

4.2) FASE 1 - IDENTIFICAÇÃO DO OBJETIVO (PROBLEMA)

Como exposto até agora, a orientação deste estudo é sobre os *Location Based Services* (LBS) no qual será desenvolvida uma aplicação que use esses serviços, seguindo essa proposta o problema escolhido para ser resolvido,

leva o processo de desenvolvimento a ter características que usem o georeferenciamento para produzir o resultado esperado pelo usuário.

A identificação do problema consiste no seguinte: como um usuário usando seu celular (smartphone) poderá obter a localização dos serviços de *fast food* (cachorro quente) mais próximos da região que se encontra.

4.3) FASE 2 – MODELAGEM E ARQUITETURA

4.3.a) ARQUITETURA

Sabemos que existem diversas definições sobre arquitetura de *software*, oriundas de diversas pesquisas que variam de autor para autor, que partem das mesmas ideias fundamentais.

Como vemos se os conceitos fundamentais têm interpretações diferente relativas ao seu uso, na arquitetura não seria diferente, os conceitos servem de alicerce para a definição da arquitetura no desenvolvimento dos sistemas.

Devido a isso foi criado o padrão ISO/IEEE 1471-2000 (ISO – IEEE 1471, 2011), esse padrão foi criado para explicar para que serve e definir o que é arquitetura de *software*, facilitando a compreensão de sua definição e seus conceitos, auxiliando os autores, estudantes e profissionais.

Arquitetura é o modo de organização de um sistema em relação aos seus componentes, seus relacionamentos e os princípios que norteiam seu *design* e construção.

Sabemos que a arquitetura é um conjunto de abstrações que formarão a base do sistema através de seus elementos, de como se relacionam com as entidades externas durante a operação do sistema.

Esses elementos são os componentes e os conectores. Os componentes são as funcionalidades que são encapsuladas e que possuem suas responsabilidades na execução sistema, para que estes componentes possam se comunicar os responsáveis são os conectores que coordenam e colaboram para essa finalidade.

A arquitetura de um sistema é elaborada para se obter um modelo da solução que será utilizada para a resolução do problema, propondo quais componentes serão integrantes da modelagem e como eles se relacionarão através dos conectores, definindo quais utilizar e quais as restrições levantadas durante a fase do levantamento dos requisitos.

A definição das arquiteturas levou a padronização da organização dos sistemas e como utilizá-las para a resolução de problemas que apresentam similaridades durante o desenvolvimento da modelagem.

4.3.b) ARQUITETURA DA APLICAÇÃO

Após a identificação do problema na fase inicial de desenvolvimento, conforme descrito no capítulo 4.2, a ideia principal do *software* de busca para *fast food Catch Dog*, será descrito de acordo com o diagrama de caso de uso abaixo:

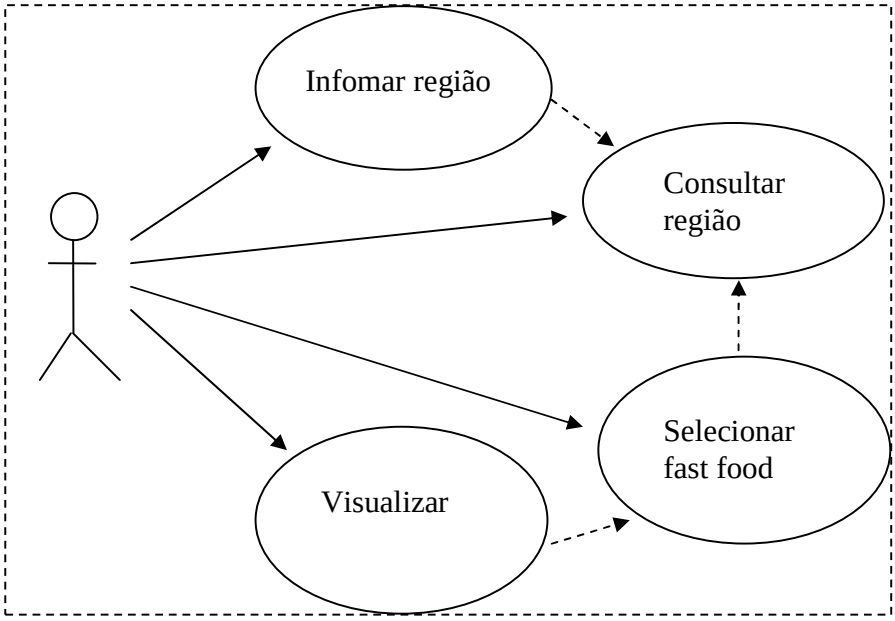

Figura 1 – Diagrama de caso de uso

Baseando-se neste diagrama percebemos que a descrição do problema foi feito de uma maneira ampla, mostrando como funcionará o software para encontrar a solução para o problema proposto. Analisando o diagrama de caso de uso montado, vemos que o foco da aplicação é o de procurar em uma determinada região e informar a localização do fast food (cachorro quente) mais próximo.

Como será uma aplicação de consulta de fast foods (cachorro quente) existentes em uma determinada localização, será necessário o armazenamento dessas coordenadas e dos dados adicionais (horários de funcionamento, etc) que se fizerem necessários para oferecer ao usuário a localização do que ele procura.

Devido a natureza desta aplicação e o tipo de equipamento (smartphone) que será utilizado para fazer a referida consulta torna-se necessário o uso de um banco de dados que ficará disponível on-line para o usuário, devido a limitação da capacidade de armazenamento das informações e poder de processamento. Aqui esbarramos em outro problema, como trata-se de uma aplicação baseada na

localização de um serviço através de dispositivo móvel (smartphone) ela terá que ser desenvolvida levando em consideração o gerenciamento de memória evitando prejudicar a performance do *hardware*.

Aqui nesta fase fica evidente uma divisão entre o usuário (cliente) e a consulta (banco de dados). Para realizar uma consulta torna-se necessário um canal de comunicação entre eles, o que influencia a construção do modelo e consequentemente sua arquitetura.

O modo como será trabalhado a construção do aplicativo se baseará nos detalhes demonstrados na figura abaixo, onde se percebe que a arquitetura que será utilizada adota o padrão cliente-servidor aplicado a um serviço de telefonia móvel.

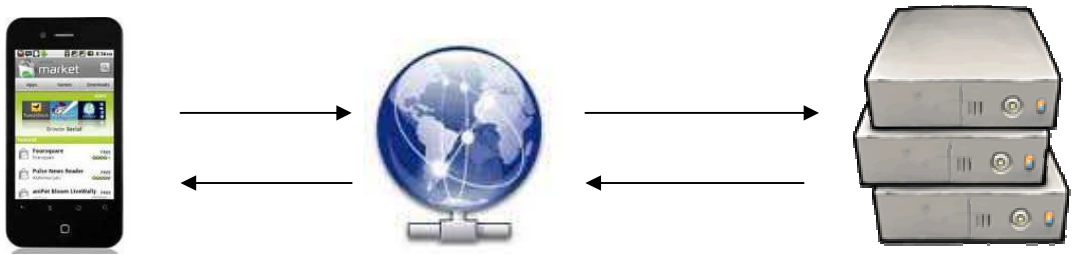

Figura 2 – Exemplo de cliente-servidor

Vemos que o modelo adotado pela aplicação se baseia na arquitetura cliente-servidor e neste caso como se tratar de um *software* de localização de serviços para ser usado nos celulares (smartphones), onde diversos clientes acessão simultaneamente o servidor para o obter o serviço solicitado a arquitetura mencionada se encaixa perfeitamente. Neste tipo de arquitetura o controle (segurança) e manutenção (atualização) dos dados ficam centralizados num servidor facilitando

a vida do usuário, pois, o mesmo não será obrigado a ficar atualizando as informações em seu celular (smartphone).

Como esta aplicação será utilizada apenas para consultas, o serviço não será prejudicado por causa do fluxo de informações, algo comum em aplicações maiores onde o tráfego de informações podem prejudicar a disponibilidade do serviço devido ao grande volume de transações, o que não é caso desta aplicação.

4.3.c) O CLIENTE E SUA ARQUITETURA

Será o cliente desta aplicação o responsável pela obtenção dos dados necessários para a localização geográfica de origem e destino durante a busca que será de grande importância neste contexto.

Como o cliente será utilizado num celular (*smartphone*) utilizando a plataforma Android, estes temas tem papel central no desenvolvimento do modelo proposto pela arquitetura de acordo com seus padrões.

Seguindo os padrões estabelecidos a estrutura do cliente será organizado em uma pilha que conterá três entidades, que serão os componentes da plataforma *Android*, que são denominadas *Actvities*, cada vez que for mencionada esta palavra ela vai indicar as telas da aplicação, *Actvities* são sinônimos para tela (LECHETA,2011). Veremos abaixo a estrutura dessas *Actvities*.

Actvity de Consulta: vai receber o endereço de destino, em seguida obterá as coordenadas de origem e destino e depois fará a consultará chamando a *Activity* Lista.

Acitvity de Listagem: efetuará a consulta remota (http), apresentando a lista de fast foods (cachorro quente), ao ser selecionado um item da lista será chamado a *Actvity* Mapa.

Actvity Mapa: efetuará a consulta remota (http) e fará a apresentação de um mapa contendo a posição do usuário e os *fast foods* (cachorro quentes) mais próximos.

Na descrição acima das *Acvities* fica claro a estrutura e sua organização como cliente, oferecendo ao usuário a localização de serviços (*fast foods*) em uma determinada região, utilizando o georeferenciamento para obter essas informações.

A listagem dos *fast foods* (cachorro quente) será fornecida pelo servidor (banco de dados), enquanto a outra informação que contém a localização do usuário será atribuição do cliente, isto se dá em virtude de sua principal característica que é a mobilidade que influenciará diretamente a obtenção desses dados.

A primeira tela é a *Actvity* de Consulta, onde o usuário irá digitar o local onde se encontra ou para onde pretende ir acionando a consulta, chamando a segunda tela de listagem a *Actvity* de Listagem. Ao ser realizada esta operação já estão pré definidas as posições de origem e destino que servirão de parâmetros para esta consulta. Na figura abaixo temos a tela de consulta a *Actvity* de Consulta:

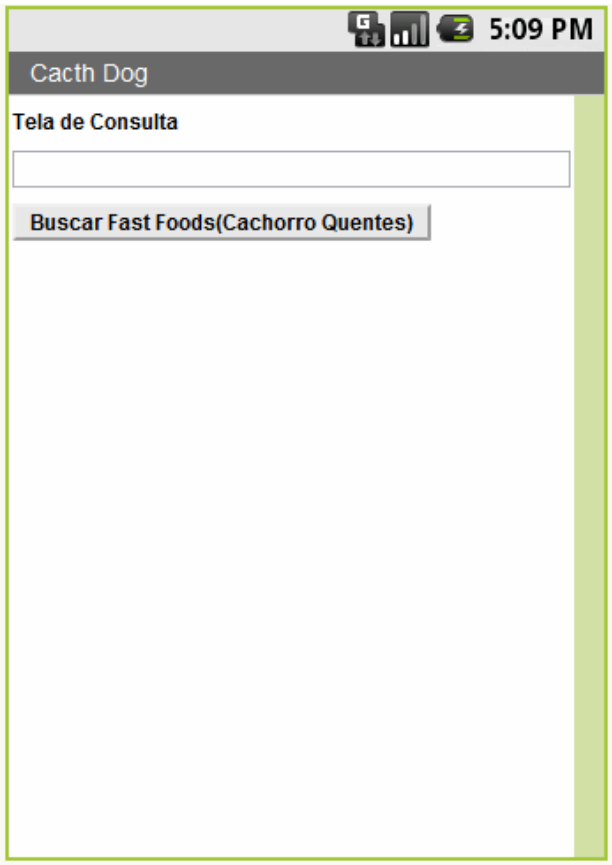

Figura 3 – *Activity* (tela) de Consulta

A activity de Consulta (tela) como vista acima possui uma interface bem simples e fácil de usar, após digitar o endereço e clicar no botão Buscar *Fast Foods da Activity* de Consulta, a partir da qual serão repassados a localização de origem e destino, acionando a chamada para a próxima tela (*Actvity* Listagem) onde aparecerá uma listagem contendo todos os *fast foods* (cachorro quentes) naquela região, como visto na figura abaixo:

Figura 4 – Activity (tela) de Listagem

Percebe-se que os dados de localização da origem e destino são os parâmetros que serão repassados para a terceira tela (*Actvity* Mapa), quando o usuário clicar em um dos itens da lista, neste mapa poderemos visualizar a localização do usuário e os *fast foods* (cachorro quentes) mais próximos, como visualizado na tela abaixo:

Figura 5 – Activity (tela) Mapa

Para desenvolver essa aplicação o modelo de arquitetura adotado foi a estrutura cliente-servidor, como se trata de estabelecer um *link* entre o cliente e o servidor o canal de comunicação torna-se primordial para se obter essas informações. Como se trata de um *software* que será utilizado em um celular (*smartphone*) no qual será feita uma consulta a um banco de dados remoto que será realizado através da internet, também deverá ser levado em conta a utilização de outros meios de comunicação em rede.

Como o objetivo da aplicação é oferecer serviços baseados na localização através da consulta a uma lista em banco de dados remoto, torna-se

necessário implementar uma web *service* que possibilite essa comunicação. Como estamos tratando de serviços, esse *software* se encaixa no modelo de Arquitetura Orientado a Serviços (SOA, 2011).

Na Arquitetura Orientada a Serviços o objetivo são os processos do negócio, como a organização dos componentes que formam o modelo utilizado no desenvolvimento do *software*, não exigindo o uso de uma determinada tecnologia para ser implementar as aplicações SOA, com a intenção de facilitar a interoperabilidade entre os diversos aplicativos, possibilitando o reúso desses serviços de maneira desacoplada e distribuída, tornando-se a principal característica dessa arquitetura.

Se fosse optado por uma determinada tecnologia, não seria possível a utilização e a integração entre os outros processos do negócio, provocando uma situação em que se uma aplicação não se adaptasse a esta tecnologia, resultaria num problema conhecido como restrição de interoperabilidade.

Como se trata do relacionamento entre os diversos tipos de serviços, que é um dos focos da arquitetura SOA, baseando-se nas descrições dos mesmos, onde essa descrição contém: o nome do serviço; seu endereço; os parâmetros que são passados e o tipo de resultado mostrado.

As tecnologias empregadas pela web *services* viabilizaram a construção de aplicativos de acordo com o modelo proposto pela arquitetura SOA.

Baseando-se nessa abordagem dentre as tecnologias que são utilizadas na implementação do modelo de arquitetura SOA será usado o REST (*Representational State Transfer* - Transferência de Estado Representacional), também chamado de *RESTful*, pois o mesmo se apoia fundamentalmente no uso da internet.

O *Web Service* REST (WS RESTFULL, 2011) surgiu de uma proposta na tese de doutorado de Roy Fieding (FIELDING 2011), que é um dos principais autores das especificações utilizadas no http, baseando-se na utilização do protocolo HTTP.

A utilização da arquitetura do *Web Service* REST torna mais ágil a aplicação pois permite a identificação dos recursos utilizados nos serviços, definindo qual será o protocolo apropriado para a interação destes recursos e também tornando-o disponível a partir de um único identificador.

Para tornar o acesso a esse identificador único Fielding propõe a utilização de URIS (Uniform Resource Identifier), como a arquitetura Web Service REST utiliza o protocolo http, cada requisição deve ser composta pelo recurso URI usado. Como cada serviço utilizado pelo sistema é composto por um único URI ele se torna acessível via *hyperlinks*.

A arquitetura *Web Service* REST propõe que uma interface seja alocada somente para as ações relacionadas ao protocolo dessa aplicação, simplificando a comunicação de seus componentes.

Para obter essas informações via HTTP o *Web Service* REST faz o uso dos métodos *GET*, *POST*, *PUT* ou *DELETE* na requisição HTTP, que poderão ser utilizados de acordo com método escolhido. Como o objetivo da aplicação CATCH DOG é buscar um recurso (informação) que será disponibilizado para o usuário utilizaremos o método *GET* que será usado para obter esse recurso, que poderá ser fornecido em vários formatos como HTML ou XML.

4.3.d) MODELAGEM DO BANCO DE DADOS

A partir daqui abordaremos como será trabalhada a base de dados da aplicação onde ficarão armazenadas as informações necessárias para localização,

procurando modelar em função do contexto de sua utilização, separando em partes de acordo com a funcionalidade de cada elemento que será utilizado para fornecer a localização baseado no serviço que será disponibilizado.

Sua modelagem se baseará na possibilidade de localizar os *fast foods* (cachorro quentes) disponíveis em uma determinada região, então, baseados na localização essa base será capaz de armazenar as informações das posições geográficas e através destas obter a localização do *fast food* mais próximo.

Basicamente será composto de uma tabela que conterá os fast foods cadastrados e a partir disso será fornecida a listagem de acordo com a posição do usuário.

É um banco de dados bem simples, composta de três tabelas, na tb_nomefastfood onde serão cadastrados os nomes dos fast foods que farão parte da listagem que será fornecida para consulta, a tb_localizacaofastfood onde serão armazenados os dados geográficos de localização dos *fast foods* e a tb_usuario, onde será inserida a localização do usuário.

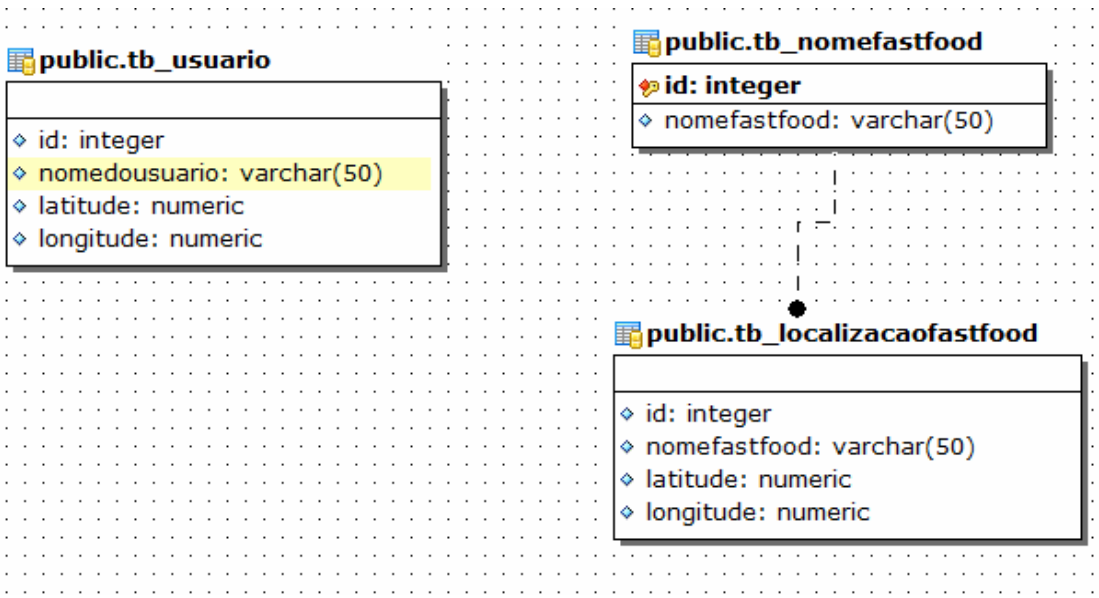

Figura 6 – Modelo do banco de dados

As tabelas tb_localizacaofastfood e tb_usuario serão fundamentais para obter o serviço desejado, pois será através de seus relacionamentos que obteremos os *fast foods* mais próximos do usuário.

Esta aplicação trabalha com duas posições geográficas (coordenadas) que são a origem e o destino, que serão os parâmetros que serão repassados para aplicação, através da consulta que será realizada no banco de dados que contém essas informações.

Como se tratam de coordenadas geográficas que ligam dois pontos sobre esfera terrestre, poderemos trabalhar com a Fórmula de Haversine que é a equação utilizada em navegação, a partir de suas latitudes e longitudes.

Quando aplicada à Terra, ela representa apenas uma aproximação, pois o nosso planeta não é uma esfera perfeita. O raio da Terra é variável, sabemos que nos polos é da ordem de 6357 km; enquanto no equador é de 6378 km. Como o raio é variável teremos que calcular utilizando o valor médio do raio que é de 6371km.

A imprecisão dos cálculos aumenta conforme nos afastamos da linha do equador. Se fosse necessária uma precisão ainda maior teríamos que utilizar a Fórmula de *Vincenty*, que leva em consideração o achatamento da Terra nos polos, por causa de sua característica elíptica. Mas devido a complexidade da segunda fórmula optei em utilizar a fórmula de *Haversine*, para facilitar o entendimento do código. Abaixo segue um exemplo como são realizados esses caculos utilizando a fórmula de *Haversine* a seguir têm uma explicação de como funciona essa fórmula (HAVERSINE, 2011).

Essa fórmula é definida como sendo o seno verso de um ângulo α, onde temos a seguinte relação:

$\text{versin}(\alpha) = 1 - \cos(\alpha)$.

Haversine significa a metade do seno verso (half versine). Assim, temos a seguinte relação:

(1-cos (α)) /2 = sen (α/2) * sen (α/2)

Abaixo temos o exemplo de como se aplica a fórmula de *Haversine*:

- O raio da terra é de 6371 quilômetros, então:

R = 6371;

A diferença das latitudes dos pontos em radianos, então:

dlat = lat2 - lat1

A diferença das longitudes dos pontos em radianos, então:

dlong = long2 - long1

O valor de **a** é o quadrado da metade do arco dos pontos.

a = sen(dlat/2) * sen(dlat/2) + cos(lat1) * cos(lat2) * sen(dlong/2) * sen(dlong/2)

- O valor de **c** é o resultado da distância em radianos.

c = 2 * atan2(sqrt(a), sqrt(1-a));

distancia = R * c;

Para utilizarmos essa fórmula as coordenadas geográficas serão representadas em graus decimais.

O algoritmo de consulta que recebe inicialmente as coordenadas de origem e destino, procura os fast foods (cachorro quentes) localizados próximos as esses pontos fornecidos. O algoritmo de consulta foi elaborada em função do banco de dados, as funções se encontram dentro do banco facilitando o encapsulamento dos dados, ele foi concebido desta maneira levando sempre em conside-

ração a arquitetura orientada a serviço (SOA), que é a base deste trabalho, onde a base de dados funciona de maneira independente do resto da aplicação.

5) PROTÓTIPO – CONSTRUÇÃO

A partir deste item do trabalho será abordado a parte da implementação dos métodos que compõem a aplicação, dividindo a aplicação em três partes de acordo com a modelagem inicial: cliente móvel; consulta via web e banco de dados. A premissa básica sempre vai partir da posição do cliente e como será feita a busca dos fast foods. Sendo exposto como a aplicação foi construída baseada na arquitetura REST e como o serviço de consulta vai mostrar a lista de fast foods (cachorro quentes). Depois as funções do banco de dados que contem as consultam (função).

5.1) LOCALIZAÇÃO – OBTENÇÃO DAS COORDENADAS

A proposta deste trabalho é desenvolver a aplicação utilizando a plataforma Android, pois, para que se possa obter a localização dos serviços mais próximos através dos mapas, foram utilizadas as APIs fornecidas pelo Google, será demonstrado adiante como essas classes auxiliaram na implementação dos métodos que trabalham com as coordenadas geográficas (origem e destino).

A *Activity* de Consulta (tela inicial) contém um campo onde é digitado o endereço de busca para obter a localização dos fast foods (cachorro quentes) mais próximos, sempre levando em consideração a posição inicial do usuário e o destino desejado, que serão os parâmetros repassados para a classe CDConsulta.java, então essa informação de origem é repassada ao método obterPosicaoInicial() que foi implementado as utilizando classes que integram o pacote android.location, o trecho do código que o contem o método citado pode-se visualizar na figura abaixo:

```
49    private CDLocalizacao obterPosicaoInicial(){
50        LocationManager locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
51
52        LocationProvider locationProvider = locationManager.getProvider(LocationManager.GPS_PROVIDER);
53
54        Location  currentLocation = locationManager.getLastKnownLocation(locationProvider.getName());
55
56        CDLocalizacao acharOrigem = new CDLocalizacao();
57
58        if(currentLocation != null){
59
60            acharOrigem.setLatitude(currentLocation.getLatitude());
61
62            acharOrigem.setLongitude(currentLocation.getLongitude());
63        }
64
65        Geocoder geocoder = new Geocoder(this, Locale.getDefault());
66
67        try{
68            List<Address> address = geocoder.getFromLocation(acharOrigem.getLatitude(),acharOrigem.getLongitude(), 1);
69
70            if(address.size()>0){
71
72                acharOrigem.setAddresses(address.get(0).getAddressLine(0));
73
74            }
75        }catch (IOException e) {
76            e.printStackTrace();
77        }
78        return acharOrigem;
79    }
```

Figura 7 – Método obterPosicaoInicial

O método obterPosicaoInicial é acionado quando o usuário clica no botão “CONSULTAR” na tela (actvity de consulta). Neste método quando a classe LocationManager é instanciada são disponibilizados os serviços de localização. Dentro do método temos outro método que é utilizado, o locationManager, que obtém um LocationProvider, esta classe encapsula o serviço de localização.

O método currentLocation obtém a informação (dados) das coordenadas obtidas na última leitura do provedor de localização. Estes dados possibilitam a instanciação da classe CDLocalizacao (CDLocalizacao.java) que faz parte da aplicação Catch Dog, onde são encapsulados os dados necessários (latitude, longitude e localização urbana) para a localização dos fast foods (cachorro questes) em uma determinada área.

A classe Geocorder é a responsável pelo mapeamento das coordenadas geográficas e do endereço urbano.

```
81
82    private CDLocalizacao obterPosicaoFinal(){
83
84        Geocoder geocorder = new Geocoder(this, Locale.getDefault());
85
86        CDLocalizacao acharDestino = new CDLocalizacao();
87
88        String endereco = incluirEndereco.getText().toString();
89
90        acharDestino.setAddresses(endereco);
91
92        try{
93
94            List<Address> addresses = geoCorder.getFromLocationName(endereco, 4);
95
96            acharDestino.setLatitude(addresses.get(0).getLatitude());
97
98            acharDestino.setLongitude(addresses.get(0).getLongitude());
99
100       } catch (IOException e) {
101           // TODO: handle exception
102           e.printStackTrace();
103       }
104
105       return acharDestino;
106   }
107
```

Figura 8 – Método obterPosicaoFinal

Na figura acima, obtemos os dados do destino que foi digitado pelo usuário na tela inicial de consulta (actvity de consulta), dentro do método obter-DestinoFinal na linha 93 a informação (objeto) digitada na tela de consulta será armazenada (posição de destino). Através da classe Geocorder obtém-se os dados de longitude e latitude do endereço digitado na tela de consulta.

Estes dois métodos que obtém as coordenadas das posições de origem e destino são responsáveis pelo resultado da consulta. Estas informações encapsuladas pela classe (CDLocalizacao) proporcionam ao usuário o fornecimento de uma informação que ele possa compreender, transformando as coordenadas em um endereço urbano.

Isso foi possível graças à classe Geocorder que faz parte do pacote android.location, ela possibilita o mapeamento entre os dados que são passados para o métodos que obtêm as posições, este serviço de geocoding não é implementado pela classe, sendo fornecida pela API do Google Maps, que é importado no arquivo AndroidManifest.xml, através da tag <uses-library>, como demonstrado a seguir:

```
<uses-library  android:name=”com.google.android.maps” />
```

5.2) WEB SERVICES RESTFUL – SERVIÇO DE CONSULTA

Na proposta de modelagem para esta aplicação, foi demonstrado que o serviço de consulta de localização de fast foods (cachorro quentes) em uma determinada região, baseava-se em uma arquitetura tipo cliente-servidor, onde a arquitetura REST se encaixa perfeitamente.

Como esta aplicação foi desenvolvida utilizando a linguagem de programação JAVA usaremos o API JAX-RS que usa anotações desta linguagem simplificando o desenvolvimento da web service. A utilização dessas anotações serão feitas nas classes para definir quais recursos e ações elas poderão executar, gerando as classes e artefatos para ele, sendo organizados em um arquivo de extensão .WAR (web application archive), através deste arquivo, estes recursos serão disponibilizados aos usuários (clientes) através de um servidor da aplicação ou de um servidor web.

```
 9 @Path("/fastFoods")
10 public class FastFoodsResourceServer {
11
12     @GET
13     @Produces({Media.APPLICATION_XML})
14     public List<FastFoods> buscarFastFoods
15         (@QueryParam("latitudeOrigem")String latitudeOrigem,
16          @QueryParam("longitudeOrigem")String longitudeOrigem,
17          @QueryParam("latitudeDestino")String latitudeDestino,
18          @QueryParam("longitudeDestino")String longitudeDestino){
19
20         FastFoodsDAO  fastFoodDAO = new FastFoodsDAO();
21
22         List<FastFoods> fastFoods = fastFoodDAO.ConsultarFastFoods
23                                     (latitudeOrigem,longitudeOrigem,
24                                     latitudeDestino,longitudeDestino);
25         return fastFoods;
26
27     }
28 }
```

Figura 9 – Código serviço Rest

Como se vê na figura acima (figura nr 9) temos a representação da classe que fornece a lista de fast foods (cachorro quentes) para um cliente que faz uma consulta (requisição http) para o servidor web onde se encontra a aplicação.

Seu funcionamento consiste simplesmente em uma classe Java que busca outra que traz uma lista de objetos (endereços).

Como é utilizada as anotações (API JAX-RS) nas classes como @Path, @Get, @QueryParam e @Produces veremos como elas se comportam, conforme descrição abaixo:

@Path indica ao servidor onde se encontra a operação e o recurso desta classe, essas requisições são feitas via http para esta URI que usa esse caminho tenham acesso aos serviços fornecido por esta classe.

@Get faz que um requisição (request http get) feita por uma URI que acionará o método buscarFastFoods.

@Produces determina que o retorno da consulta contendo a lista de objetos FastFoods seja no formato XML.

@QueryParam declara para o método que os parâmetros serão passados via URI.

Na figura abaixo vemos a como a listagem é passada via a http (request) procurando o recurso.

```
123    private List<Listagem> buscarFastFoods(){
124        HttpHost host = new HttpHost(localhost, 8080, "http");
125
126        String coordenadas = "?latitudeOrigem="+obterPosicaoInicial().getLatitude()
127                            +"&longitudeOrigem="+obterPosicaoInicial().getLongitude()
128                            +"&latitudeDestino="+obterPosicaoFinal().getLatitude()
129                            +"&longitudeDestino="+obterPosicaoFinal().getLongitude();
130
131        HttpClient cliente = new DefaultHttpClient();
132        String url = "/Catchdog/rest/fastFoods"+coordenadas;
133        System.out.println(url);
134        HttpGet get = new HttpGet(url);
135
136        ArrayList<Listagem> fastFoodsObtidos = new ArrayList<Listagem>();
137
138        try{
139            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
140            DocumentBuilder builderFactory = factory.newDocumentBuilder();
141            HttpEntity entity = cliente.execute(host,get).getEntity();
142
143            fastFoodsObtidos = obterRestXmlParaListarFastFoods(builderFactory, entity);
144
145
146        }
147    }
```

Figura 10 – Código do cliente http do serviço Rest

Ao serem acessados os recursos através da camada de serviço, via http, qualquer usuário que enviar uma requisição GET usando o protocolo obterá os fast foods mais próximos.

Na linha 124 da figura acima, é montado um objeto que contém o host para onde será enviada a requisição. Nas linhas 126 a 132 é montado a URI de acesso para a parte que contém o serviço, sendo passados os parâmetros para a consulta. Na linha 141 o objeto HttpEntity realiza a consulta requisitando ao serviço REST uma listagem contendo os fast foods mais próximos.

5.3) FUNÇÃO SQL DE LOCALIZAÇÃO

O banco de dados da aplicação foi construído de tal maneira que o encapsulamento dos dados e a lógica utilizada para obter a listagem ficassem sob sua responsabilidade, como foi exposto no capítulo anterior o serviço REST obtém a lista através da conexão com banco de dados e da operação executada pela função buscar_fast_food

A consulta gerada pela estrutura do código utiliza a linguagem SQL, onde sua função principal é buscar os fast foods (cachorro quentes) mais próximos de acordo com os parâmetros passados, de origem e destino.

5.5) TESTE REALIZADO NA APLICAÇÃO

O protótipo da aplicação Catch Dog foi desenvolvido usando o SDK do Google projetado para a plataforma Android que contém um emulador que simula um aparelho celular onde a aplicação deste estudo foi executada, possibilitando a realização dos testes.

Os dados utilizados para testes foram cadastrados no banco utilizando referências (endereços) próximos a minha residência, onde a minha casa representava a origem e os destinos eram locais próximos que possuíam fast foods (cachorro quentes) mais próximos, pois a ideia desta aplicação é buscar o serviço que estiver mais próximo do ponto inicial.

5.5) POSIÇÃO INICIAL (ORIGEM)

A posição inicial (origem) onde o aparelho (smartphone) se encontra localiza-se à rua Egídio Carrara, 21, Jardim São Gabriel, Colombo-PR, local da

minha residência, identificada da mesma maneira, possuindo a seguintes coordenadas 25º20’035.58’’ S de longitude e 49º11’27.30’’ O de longitude. Mas para que estas coordenadas possam ser usada pela aplicação Catch Dog elas foram convertidas em graus decimais -25,343217 de latitude e -49.190917 de longitude.

A seguir temos a lista dos fast foods (cachorro-quentes) próximos a minha residência:

Id	endereco	número
1	Rua Heitor Busato	300
2	Rua Alfredo Miguel Baduy	250

5.5.1) TESTE NÚMERO 1

O primeiro teste foi feito passando o destino da rua Heitor Busato,300, onde se localiza o fast food mais perto como pode ser visto na figura abaixo

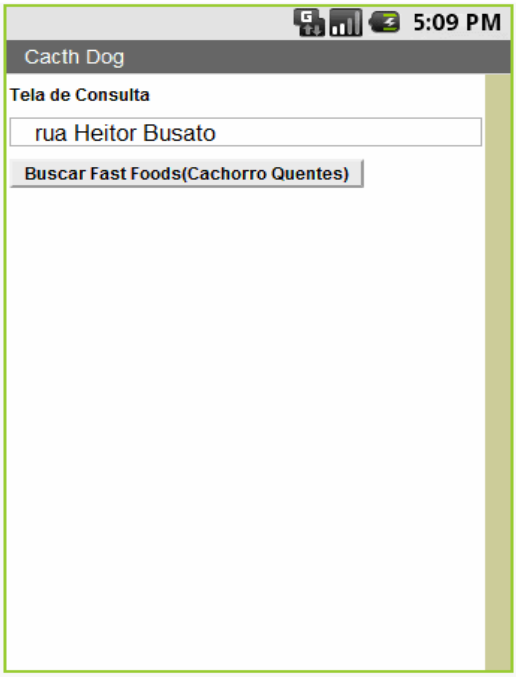

Figura 11 – Activity (tela) de Consulta

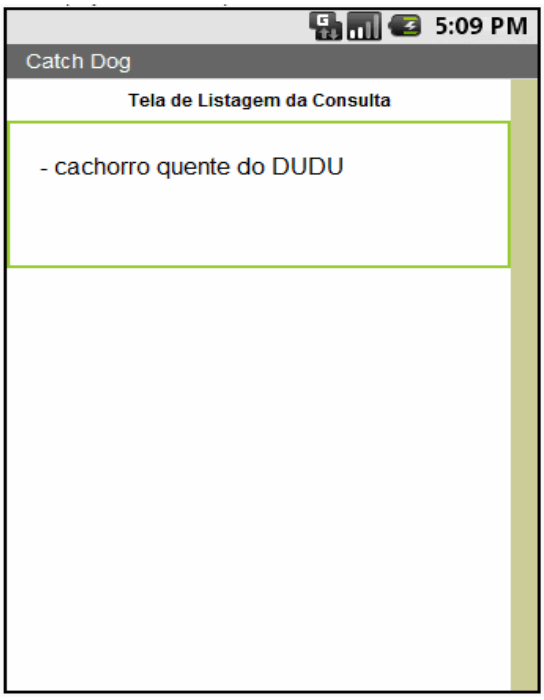

Figura 12 – Activity (tela) de Listagem

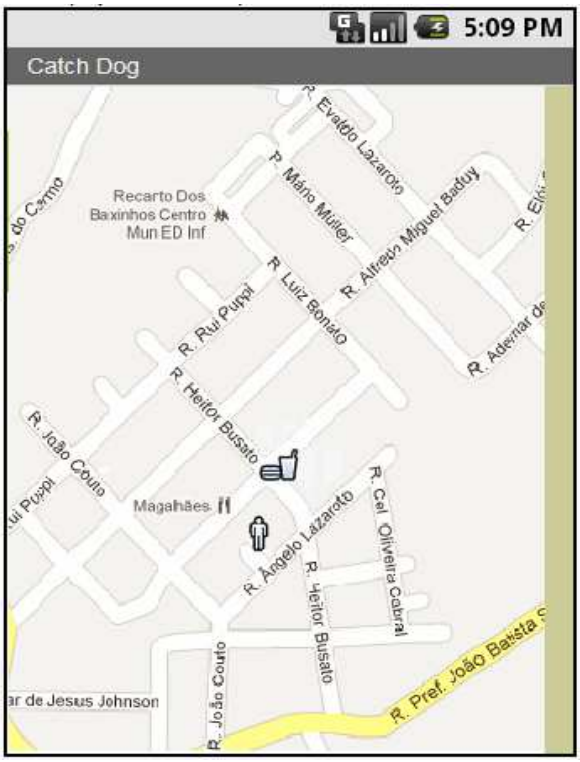

Figura 13 – Activity (tela) do Mapa

Na figura 11 pode-se ver a primeira actvity (tela) de consulta, onde digitamos o endereço, na figura seguinte (Figura 12) temos a actvity (tela) de Listagem com a lista de fast foods (cachorro quentes) mais próximos e na última actvity (tela) do Mapa (Figura 13) temos um mapa onde fica identificada a posição do usuário e dos fast foods mais próximos.

6) CONCLUSÃO

Como visto na introdução deste trabalho a escolha deste assunto foi motivada pela constante evolução da tecnologia ligada à telefonia móvel (*smart-phones* e *tablets*) e a crescente demanda dos usuários por serviços que estejam disponíveis através destes dispositivos, influenciando diretamente os setores ligados a tecnologia da informação, oferecendo ao setor uma grande oportunidade de negócios através de produtos e serviços que podem ser disponibilizados aos usuários.

Os serviços baseados na localização cuja demanda é crescente fizeram despertarem o interesse do uso da tecnologia LBS para atender essa questão, facilitando a vida das pessoas. Percebe-se que a gama de aplicativos que podem ser desenvolvidos, tanto podem atender usuários comuns quanto empresas ou órgãos públicos.

Aparentemente simples, a aplicação Cath Dog foi desenvolvida para a busca por fast foods (cachorros quentes) localizados em uma determinada região possibilitando ao usuário escolher o mais próximo de sua localização.

Todo este trabalho foi desenvolvido e modelado visando o fornecimento desse serviço e o protótipo da aplicação foi implementado baseando-se nessa modelagem. Após implementado, procurou-se verificar a funcionalidade da aplicação através de um simples teste onde se alcançou o resultado esperado no fornecimento deste serviço. A plataforma Android disponibiliza um emulador de celular para auxiliar no desenvolvimento de aplicativos, como não se trata de um aparelho normal alguns fatores presentes neste dispositivo não puderam ser testados.

A proposta do trabalho foi desenvolver uma aplicação que usasse a tecnologia Android e fosse desenvolvida usando a linguagem de programação Java de tal maneira que todo o trabalho se enquadrasse no conteúdo oferecido pelo curso de especialização.

Percebe-se que outro tema também abordado durante o curso, SOA/BEPEL, facilitou a busca e a compreensão das arquiteturas e tecnologias disponíveis que trabalham com aplicações distribuídas e intercambiáveis para a troca de dados, que possuam pouca dependência entre seus módulos.

Sabe-se que o SOA (*Service Oriented Architeture* – Arquitetura Orientada a Serviços), não se foca na linguagem que será utilizada para construir a aplicação e sim nos processos de negócio, e, de como esses serviços serão implantados pela aplicação para atingir esses objetivos. Uma das arquiteturas que se encaixam perfeitamente nesta linha é arquitetura Web Service Restful (REST).

Levando em consideração estes princípios, toda a aplicação que for desenvolvida baseada no fornecimento de serviços de maneira remota, como neste trabalho onde o serviço é fornecido através da internet, os recursos utilizados para o fornecimento de serviços deverá obrigatoriamente ser desacoplado de tal maneira que não existam dependências entre os componentes do sistema.

Devido ao foco deste trabalho ser a utilização da plataforma Android para oferecer serviços não importando onde o cliente (usuário) se encontre, este estudo utilizou os conceitos desta para a elaboração da presente aplicação com a utilização de recursos remotos.

Neste trabalho, a estruturação do algoritmo se dividiu em duas partes: uma no cliente e outra no banco de dados da aplicação. Dessa forma, na camada do cliente que contém os serviços de requisição e consulta ocorrem apenas o trânsito dessas informações, liberando recursos do equipamento e tornando o funcionamento do aplicativo mais leve, deixando a parte do processamento no banco de dados da aplicação. Isto ajuda a manter a performance do aplicativo em um nível aceitável, mesmo que as condições da rede de telecomunicações não sejam estáveis.

Com este trabalho procurou-se demonstrar que apesar da aplicação trabalhar com uma quantidade reduzida de serviços, não impede que aplicações com grande quantidade de serviços sejam desenvolvidas baseadas nos conceitos expostos.

Referências:

LECHETA, Ricardo R. Google ANDROID – Aprenda a criar aplicações para dispositivos móveis com o Android SDK - 2ª Edição – São Paulo: Novatec ,Junho 2010

ISO – IEEE 1471. Disponível em:
<www.iso-architecture.org/ieee-1471/>. Acesso em: abril de 2011

GSM WORLD. Disponível em:
<gsmworld.mobi/roaming/>. Acesso em: abril de 2011.

3GPP. Disponível em:
<http://www.3gpp.org/> . Acesso em: abril de 2011

OHA. Disponível em:
<www.openhandsetalliance.com>. Acesso em: maio de 2011.

Technology Research/Gartner Inc. Disponível em:
<http://www.gartner.com/>. Acesso em: maio de 2011.

CONTROLE PEDÁGIOS. Disponível em:
<http://www2.uol.com.br/canalexecutivo/notas11/150220112.htm>.
Acesso em: maio de 2011.

CHAVE PRIVADA PARA MAPVIEWS (API KEY). Disponível em:
<http://code.google.com/intl/pt-BR/android/add-ons/google-apis/mapkey.html>. Acesso em: maio de 2011

REGISTRANDO UMA API KEY PARA MAPVIEW. Disponível em:

<http://code.google.com/intl/pt-BR/android/maps-api-signup.html>**.** Acesso em: maio de 2011

HAVERSINE. Disponível em : **<**http://www.plugmasters.com.br/plugfeed/post/86430/postgresql-obter-locais-proximos-formula-de-haversine>. Acesso em: junho de 2011.

SOA. Disponível em: <www.oracle.com/**soa**>. Acesso em: julho de 2011.

SOA. Disponível em: <http://soasimples.com/blog/>. Acesso em: julho de 2011.

WS RESTFULL. Disponível em: **<**www.oracle.com/technetwork/articles/javase/index-137171.htm>. Acesso em: julho de 2011

WS RESTFULL. Disponível em: **<**www.ibm.com/developerworks/webservices/ library/ws-restful>. Acesso em: julho de 2011

FIELDING, Roy. Disponível em: **<**http://www.ics.uci.edu/~fielding/> . Acesso em: junho de 2011.