UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Presidente Prudente

CAIQUE CASTANHO BOLOGNESI MELCHIOR

CONCEPÇÃO E IMPLEMENTAÇÃO DE UM SERVIÇO BASEADO EM LOCALIZAÇÃO NA PLATAFORMA ANDROID/GOOGLE

Presidente Prudente 2011

CAIQUE CASTANHO BOLOGNESI MELCHIOR

CONCEPÇÃO E IMPLEMENTAÇÃO DE UM SERVIÇO BASEADO EM LOCALIZAÇÃO NA PLATAFORMA ANDROID/GOOGLE

Trabalho de conclusão de curso apresentado ao Departamento de Matemática, Estatística e Computação (DMEC), da Universidade Estadual Paulista "Júlio de Mesquita Filho" sob o título "Concepção e Implementação de um Serviço Baseado em Localização na Plataforma Android/Google".

Orientador: Prof. Dr. Milton Hirokazu Shimabukuro

Presidente Prudente

2011

TERMO DE APROVAÇÃO

CAIQUE CASTANHO BOLOGNESI MELCHIOR

CONCEPÇÃO E IMPLEMENTAÇÃO DE UM SERVIÇO BASEADO EM LOCALIZAÇÃO NA PLATAFORMA ANDROID/GOOGLE

Monografia aprovada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, da Universidade Estadual Paulista "Júlio de Mesquita Filho", pela seguinte banca examinadora:

Orientador: Prof. Dr. Milton Hirokazu Shimabukuro
Departamento de Matemática, Estatística e Computação (DMEC)

Prof. Dr. Marco Antônio Piteri
Departamento de Matemática, Estatística e Computação (DMEC)

Prof. Dr. Almir Olivette Artero
Departamento de Matemática, Estatística e Computação (DMEC)

Presidente Prudente

2011

AGRADECIMENTOS

Agradeço primeiramente a Deus que me deu a perseverança necessária para a conclusão deste trabalho, além de me dar forças para transpor dificuldades e as minhas próprias limitações.

Aos meus pais que me deram educação e força de vontade para perseguir os meus sonhos, a todos os meus amigos que estiveram ao meu lado nos períodos difíceis.

E ao orientador Prof. Dr. Milton Hirokazu Shimabukuro que contribuiu para o desenvolvimento e a conclusão deste trabalho.

RESUMO

O aplicativo móvel proposto neste trabalho, classificado como um Serviço Baseado em Localização e denominado Traveller, foi desenvolvido para auxiliar usuários de celulares equipados com GPS em locais desconhecidos através do fornecimento de informações climáticas, de localização do usuário e informações de estabelecimentos em áreas urbanas acessando serviços na internet.

Os dispositivos móveis aos quais se destina este trabalho são os equipados com o Android e a linguagem de programação selecionada foi a Java utilizando o ambiente de desenvolvimento Eclipse juntamente com o kit de ferramentas de desenvolvimento Android (ADT - *Android Development Tools*).

Palavras-chave: Serviço Baseado em Localização, *Web Services*, Plataforma Android.

ABSTRACT

The mobile application proposed in this undergraduate project, classified as a Location-Based Service and named Traveller, was developed to support users of mobile phones equipped with GPS in unknown locations by providing information about weather, location of users and stores in urban areas.

The mobile devices whose this project is intended are those equipped with Android. The programming language Java was selected and the Eclipse development environment was used along with the toolkit for developing Android (ADT - Android Development Tools).

Keywords: Location-Based Service, Web Services, Android Platform.

LISTA DE FIGURAS

Figura 2.1 – Arquitetura do Android 5
Figura 2.2 – Latitude e Longitude 10
Figura 2.3 – Triangulação e Trilateração 12
Figura 2.4 - Abordagem do SIG por camadas 19
Figura 2.5 – Modelos de representação de objetos espaciais 19
Figura 2.6 – Modelo de comunicação LBS 24
Figura 2.7 – Arquitetura orientada a serviço 26
Figura 2.8 – Web Services SOAP 28
Figura 3.1 – Ciclo de vida de uma Activity 33
Figura 3.2 – Acitivity com mapa 37
Figura 4.1 – Interações entre o Traveller e os *web services.* 47
Figura 4.2 – Tela de Abertura 49
Figura 4.3 – Localização Atual 50
Figura 4.4 – Previsão de Tempo 51
Figura 4.5 – Tela de Busca de Lugares 52
Figura 4.6 – Classes responsáveis pela busca de lugares. 53
Figura 4.7 – Busca de Cidades 54
Figura 4.8 – A classe geodesia. 55
Figura 4.9 – Catálogo de Pontos 56
Figura 4.10 – Classes de Persistência 57
Figura 4.11 – Lugares Visitados 58
Figura 4.12 – Tabelas da Base de Dados 58

LISTA DE SIGLAS

AAC = *Advanced Audio Coding*
A-GPS = Assisted GPS
AMR = *Adaptive Multi-Rate*
AOA = *Angle of Arrival*
API = *Application Programming Interface*
CGI = *Cell of Global Identity*
COO = *Cell of Origin*
EDGE = *Enhanced Dara rates for GSM Evolution*
E-OTD = *Enhanced Observed Time Difference*
GPS = *Global Positioning System*
GSM = *Global System for Mobile*
HLR = *Home Location Register*
HTTP = *HyperText Transfer Protocol*
IPC = *Inter-Process Communication*
JVM = *Java Virtual Machine*
LBS = *Location Based Services*
LKM = *Loadable Kernel Modules*
MPC = *Mobile Positioning Center*
MPS = *Mobile Positioning System*
PDA = *Personal Digital Assistants*
PNG = *Portable Network Graphics*
PPS = *Precise Positioning Service*
REST = *Representational State Transfer*
SDK = *Software Development Kit*
SIG = Sistemas de Informações Geográficas
SOA = *Service-oriented architecture*
SOAP = *Simple Object Access Protocol*
SQL = *Structured Query Language*
TA = *Timing Advance*
TDOA = *Time Difference of Arrival*
TOA = *Time of Arrival*
TTFF = *Time To First Fix*
UDDI = *Universal Description, Discovery and Integration*
UL-TOA = *Uplink Time of Arrival*
URI = *Uniform Resource Identifier*
UTF-8 = 8-*bit Unicode Transformation Format*
VLR = *Visitor Location Register*
W3C = *Word Wide Web Consortium*
WSDL = *Web Services Description Language*
XML = *Extensible Markup Language*

SUMÁRIO

1 – INTRODUÇÃO 1
1.1 – Objetivos do Trabalho 3
1.2 - Organização do Trabalho 3
2 – REVISÃO BIBLIOGRÁFICA 4
2.1. O Linux e a Arquitetura Básica do Android 4
2.1.1. Linux Kernel 6
2.1.1.1. *Driver* IPC 6
2.1.2. Libraries (bibliotecas) 6
2.1.3. Android Runtime 7
2.1.4. Application Framework 7
2.2. Localização e Posicionamento 8
2.2.1. Sistema de Coordenadas Elipsoidal 9
2.2.2. Sistemas de Posicionamento 11
2.2.2.1. Triangulação e Trilateração 11
2.2.2.2. Técnicas básicas de localização 12
2.2.2.3. Sistemas de posicionamento por satélites 13
2.2.2.4. *Assisted GPS* 15
2.2.2.5. Sistemas de posicionamento baseados em redes 15
2.3. Sistemas de Informações Geográficas (SIGS) 17
2.3.1. Definição de Sistemas de Informações Geográficas (SIGs) 18
2.4. *Location Based Services (LBS)* 20
2.4.1.Taxonomia dos Serviços Baseados em Localização 21
2.4.2. LBS e Privacidade 22
2.4.3. O Mercado LBS 23
2.4.4. O Modelo de Comunicação LBS 24
2.5. Web Services 25
2.5.1. Arquitetura Orientada a Serviço – SOA 26
2.5.2. Web Services SOAP 27
2.6.3. Web Services REST 28
3 – RECURSOS COMPUTACIONAIS PARA A IMPLEMENTAÇÃO 31
3.1 - Ambiente de Desenvolvimento Android 31

3.1.1 - Activity 32
3.1.2 - A Classe R 34
3.1.3 - Definição da Interface Visual 34
3.1.4 - O arquivo AndroidManifest.xml 35
3.1.5 - A classe android.content.Intent 35
3.1.6 - Google Maps 36
3.1.7 - SQLite 37
3.2 - Apontador 38
3.3 – O Serviço Google Weather 43
3.4 – O Portal GeoNames 45
3.4.1 - O método findNearbyPlaceName 46
4 - O SERVIÇO BASEADO EM LOCALIZAÇÃO TRAVELLER 47
4.1 – O serviço de localização e posicionamento 48
4.2 – O Serviço de Informações Climáticas 50
4.3 – O Serviço de Pesquisa de Estabelecimentos 51
4.4 – O Serviço Catálogo de Lugares 54
4.4.1 - Busca e Armazenamento de Cidades 54
4.4.1.1 – Cálculo de Distâncias 55
4.4.2 – Catálogo de Pontos Visitados em uma Cidade 56
4.4.3 - Galeria de Pontos Visitados 57
5 – TESTES E CONCLUSÃO 59
REFERÊNCIAS BIBLIOGRÁFICAS 62

1 – INTRODUÇÃO

Os Serviços Baseados em Localização trouxeram uma nova perspectiva aos aplicativos móveis agregando valor e melhorando a experiência dos usuários através de informações de localização. O GPS é usado há algum tempo em diversas áreas, com o simples propósito de se localizar no espaço e como ferramenta de navegação. A partir do momento em que os fabricantes de dispositivos móveis uniram em um único aparelho um receptor GPS e conexão com a internet, emergiu uma nova classe de aplicativos que fornecem diversas informações em tempo real através da internet em função da localização do usuário (KUPPER, 2005).

Um usuário perdido em uma cidade desconhecida pode se localizar através do GPS presente no seu celular e a partir daí descobrir um meio de chegar ao seu destino. Por outro lado é muito mais interessante se esse mesmo usuário puder se conectar a internet e em função de suas coordenadas, descobrir qual a estação de ônibus mais próxima, por exemplo. Funções como essa só são possíveis devido a crescente evolução dos *web services*.

Um dispositivo capaz de se conectar na internet é suficiente para processar aplicativos que recorrem a conteúdos recuperados a partir de fontes externas para a criação de um serviço novo. Tal classe de aplicativos é conhecida como *Mashup* (IBM, 2006).

Um aplicativo *Mashup* se caracteriza por usar conteúdo de mais de uma fonte para a criação de um novo serviço completo. O princípio básico presente nesse tipo de aplicativo é o reuso de código e novos serviços sofisticados podem ser projetados com relativa simplicidade. Por exemplo, um desenvolvedor de aplicativo que fornece informações climáticas provenientes de *web services* especializados não precisa ter conhecimentos de meteorologia ou de como os dados da condição do tempo são coletados.

Aplicativos *Mashups* podem ser desenvolvidos para serem usados em computadores *desktop* através de navegadores de internet, porém o desenvolvimento de *Mashups* se torna mais abrangente e dinâmico quando a plataforma alvo são dispositivos móveis equipados com GPS e conexão com a internet. Dessa forma, é possível combinar informações de diversas fontes presentes na internet com a localização do usuário, tudo em tempo real criando

desde aplicações direcionadas ao entretenimento, a aplicações militares e de serviços de emergência.

Com a expansão dos celulares e dispositivos móveis de alto desempenho houve a necessidade de criação de sistemas e arquiteturas direcionadas a esses dispositivos. Sistemas que fossem projetados levando em consideração as características desses aparelhos como a relativa baixa capacidade de memória, processamento e autonomia de bateria. Nesse contexto, um consórcio de empresas denominado *Open Handset Alliance* desenvolveu um novo sistema operacional baseado no núcleo do Linux e chamado Android, especialmente para dispositivos móveis.

O Android é uma pilha de software para dispositivos móveis que usa o *kernel* 2.6 do Linux como base, incluindo algumas adaptações para dispositivos móveis. Os serviços críticos do sistema como segurança, gerenciamento de memória, gerenciamento de processos, protocolos de rede e drivers ficam a cargo do *kernel* do Linux, que também atua como uma camada de abstração entre o *hardware* e o restante do *software* (ANDROID DEVELOPERS, 2011).

Apesar de o Android ser baseado no Linux, ele não é uma distribuição Linux, pois não suporta todas as funcionalidades padrões de uma distribuição. O Android é, na verdade, um *kernel* do Linux levemente modificado, descrito como a primeira plataforma para dispositivos móveis verdadeiramente de código aberto (*open-source*) (LECHETA, 2009).

Considerando a pertinência das informações de localização como forma de agregar valor aos aplicativos e o crescente interesse dos usuários em dispositivos móveis, o presente projeto visa desenvolver um aplicativo pertencente a classe dos Serviços Baseados em Localização, tendo como alvo a plataforma Android, para demonstrar como informações de diferentes fontes podem ser combinadas de modo a criar uma experiência de uso sofisticada, colocando em prática as soluções do Android para dispositivos móveis e comprovar a sua eficácia nesse tipo de aplicação.

1.1 – Objetivos do Trabalho

O aplicativo idealizado nesse trabalho, denominado Traveller e classificado como um Serviço Baseado em Localização tem por objetivo auxiliar usuários de celulares equipados com GPS em locais desconhecidos através do fornecimento de informações climáticas, de localização do usuário e informações de estabelecimentos em áreas urbanas, além de apontar no mapa a localização atual do dispositivo e de cidades de interesse sendo complementado com funções relativas à localização, como o cálculo de distâncias entre coordenadas geográficas.

1.2 - Organização do Trabalho

O próximo capítulo traz a fundamentação teórica estudada para o desenvolvimento do trabalho, a Seção 2.1 aborda conceitos básicos de um sistema Linux e a arquitetura básica do Android, seguida pela Seção 2.2 que aborda o tema localização e posicionamento, assunto fundamental contido nesse trabalho. A Seção 2.3 introduz o conceito de Sistemas de Informações Geográficas

A Seção 2.4 explana sobre os Serviços Baseados em Localização, suas características, taxonomia e o modelo de comunicação LBS. A última seção do segundo Capítulo discorre sobre os *web services* e sua arquitetura.

O Capítulo 3 apresenta os recursos utilizados para o desenvolvimento do trabalho. A Seção 3.1 traz as características do ambiente de desenvolvimento Android utilizado para o desenvolvimento da aplicação. A Seção 3.2 mostra o funcionamento do *web service* Apontador, seguida pela Seção 3.3 e 3.4 que apresentam o funcionamento dos serviços *Google Weather e GeoNames* respectivamente.

No Capítulo 4 é exibido o funcionamento do Serviço Baseado em Localização Traveller, as seções de 4.1 a 4.4 demonstram as funções de localização e posicionamento, informações climáticas, pesquisa de estabelecimentos e catálogo de lugares respectivamente. Finalmente o Capítulo 5 traz as conclusões do trabalho.

2 – REVISÃO BIBLIOGRÁFICA

Neste Capítulo é apresentada a fundamentação teórica necessária para o desenvolvimento do trabalho. Na Seção 2.1 são apresentados conceitos básicos de um sistema Linux e a arquitetura básica do Android. A Seção 2.2 discorre sobre fundamentos de localização e posicionamento, a Seção 2.3 introduz o conceito de Sistemas de Informações Geográficas, já a Seção 2.4 apresenta os Serviços Baseados em Localização e finalmente a Seção 2.5 traz os principais tipos de *web services* e suas características principais.

2.1. O Linux e a Arquitetura Básica do Android

O Linux possui um núcleo (*Kernel*) monolítico. As funções principais como escalonamento de processos, gerenciamento de memória, acesso ao sistema de arquivos e comunicação com o hardware são executadas dentro do núcleo do sistema operacional. O motivo principal é o desempenho: como todas as estruturas de dados e códigos do *Kernel* são mantidas em um só espaço de endereçamento não há a necessidade de troca de contexto quando um processo chama uma função do sistema operacional. Além disso, o *Kernel* possui uma característica modular, pois algumas funções como *drivers* de hardware, por exemplo, podem ser compiladas e executadas como módulos denominados LKM (*Loadable Kernel Modules*). Tais bibliotecas são capazes de ser carregadas e descarregadas com o *Kernel* em execução. Os LKM (módulos recarregáveis do *Kernel*) são executados em modo *Kernel* privilegiado e, por conseguinte, tem acesso total à máquina em que se encontram em execução. Em tese não há restrições ao que um módulo do *Kernel* pode fazer.

Os módulos do *Kernel* permitem que um sistema Linux possa ser configurado com um *Kernel* mínimo, sem que qualquer *driver* de dispositivo extra seja incorporado, todos os *drivers* de dispositivos que o usuário venha a precisar poderão ser recarregados explicitamente pelo sistema de inicialização ou carregados automaticamente pelo sistema sob demanda e descarregados quando não

estiverem em uso.

Possuir um sólido sistema de gerenciamento de memória e processos além de drivers confiáveis, são fatores que levaram à escolha do *kernel* para servir de base para o Android e prover confiabilidade e segurança às tarefas críticas do sistema.

"O Android é a nova plataforma de desenvolvimento para aplicativos móveis como *smarthphones* e contém um sistema operacional baseado em Linux, uma interface visual rica, suporte a GPS, diversas aplicações já instaladas e ainda um ambiente de desenvolvimento bastante poderoso, inovador e flexível" (LECHETA, 2009, p. 21).

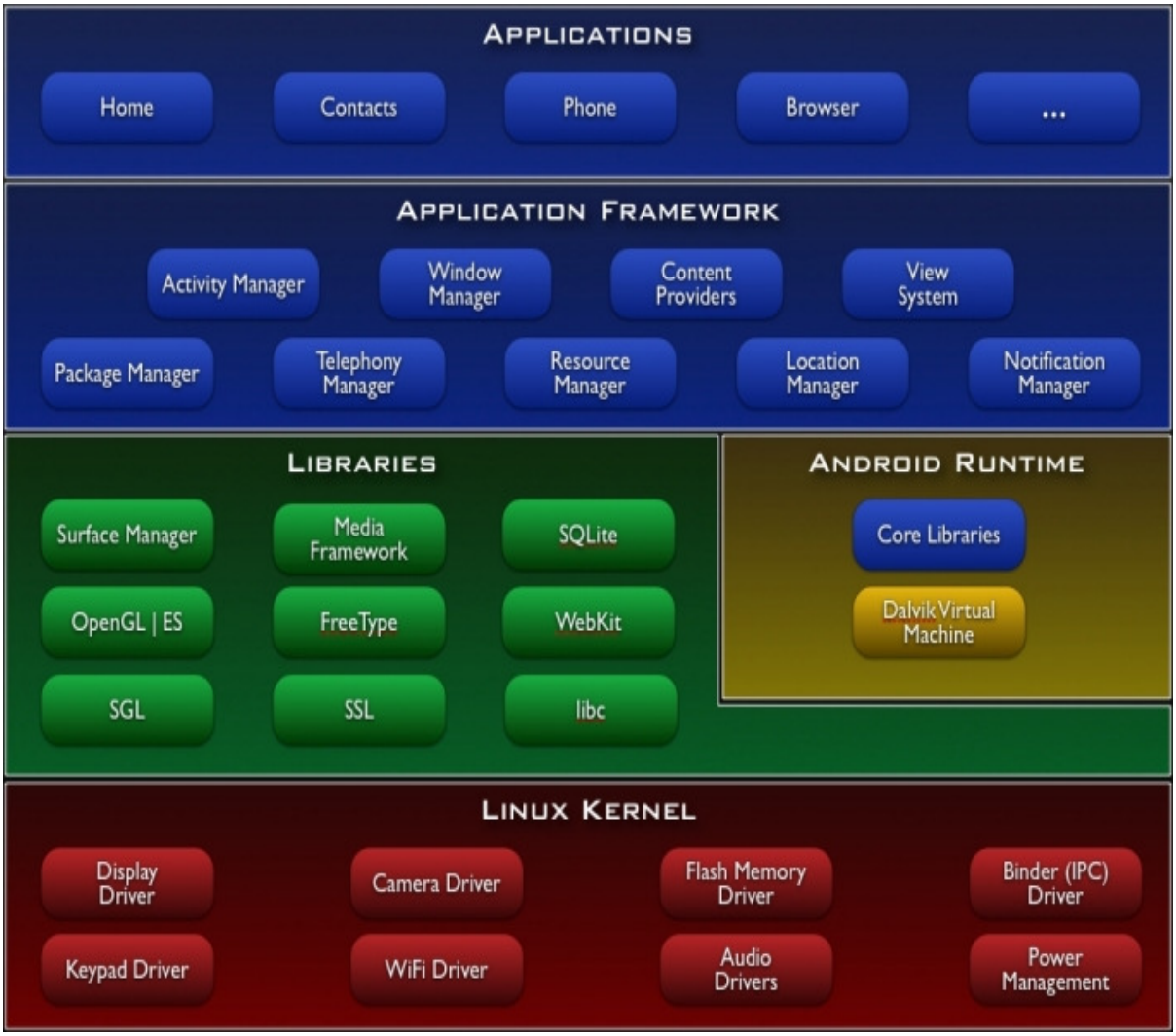

Figura 2.1 – Arquitetura do Android
Fonte: < http://developer.android.com/guide/basics/what-is-android.html>

2.1.1. Linux Kernel

Na base do diagrama da Figura 2.1 está o *Kernel* do Linux, no qual se encontram principalmente os *drivers* de dispositivos além do módulo de gerenciamento de energia modificado e o IPC (*Inter-Process Communication*) que constituem as principais modificações feitas para atender as necessidades dos dispositivos móveis.

2.1.1.1. *Driver* IPC

O *driver* IPC (*Inter-Process Communication*) provê aos aplicativos a capacidade de se comunicar de forma sincronizada. O Android possui um eficiente mecanismo para chamadas de procedimentos remotos, onde um método é chamado localmente, mas é executado remotamente (em outro processo) e o resultado é retornado ao processo que o chamou.

2.1.2. Libraries (bibliotecas)

O Android inclui uma série de bibliotecas C/C++ usadas por vários de seus componentes. Os recursos providos pelas bibliotecas são acessíveis aos programadores através do aplicativo framework do Android, segue abaixo uma lista das bibliotecas presentes no sistema:

- *System C library* – implementação derivada da biblioteca padrão C aprimorada para dispositivos baseados em Linux embarcado;
- *Media Libraries* – baseado no *PacketVideo's OpenCORE*; a biblioteca suporta reprodução e gravação de vários formatos populares de áudio e vídeo incluindo MPEG4, H.264, MP3, AAC, AMR, JPG, e PNG;
- *Surface Manager* – gerencia o acesso ao subsistema de exibição e compõe camada gráfica 2D e 3D de múltiplas aplicações;
- LibWebCore – biblioteca de suporte a navegação na web;

- SGL – biblioteca subjacente de gráficos 2D;
- 3D *libraries* – uma implementação baseada no OpenGL, a biblioteca usa, quando disponível, aceleração 3D por *hardware* e *software* de rasterização altamente otimizado;
- FreeType – biblioteca de renderização de imagens;
- SQLite – um poderoso e leve gerenciador de banco de dados relacional disponível para todas as aplicações.

2.1.3. Android Runtime

A linguagem Java é utilizada para criar aplicativos para o Android, o fato é que o sistema não possui uma maquina virtual Java (JVM)[1], em seu lugar o que se tem é uma máquina virtual chamada Dalvik otimizada para execução em dispositivos móveis.

As aplicações Android possuem seu próprio processo, com sua própria instancia de máquina virtual Dalvik. A máquina virtual Dalvik foi projetada de modo que os dispositivos possam executar múltiplas instâncias de máquina virtual eficientemente.

No processo de desenvolvimento dos aplicativos, o bytecode[2] é compilado e convertido para o formato ".dex" (Dalvik *Executable)*, que representa um aplicativo Android compilado, posteriormente os arquivos ".dex" e outros recursos como imagens são compactados e empacotados em um único arquivo com extensão ".apk" (*Android Package File*), que representa a aplicação final, pronta para ser distribuída e instalada.

2.1.4. Application Framework

O desenvolvedor independente tem acesso total ao mesmo *framework* utilizado pelas aplicações nativas. A arquitetura da aplicação foi desenhada para

[1] Máquina Virtual Java (Java Virtual Machine – JVM) é uma máquina virtual que interpreta bytecodes.
[2] Bytecode é uma forma intermediária derivada de programas fonte Java produzido por compiladores Java.

simplificar o reuso de componentes, qualquer aplicação pode publicar seus recursos para que outros aplicativos façam uso dos mesmos (ANDROID DEVELOPERS, 2011).

Uma coleção de serviços nativos está disponível para todos os aplicativos, tais como:

- Uma série de “Views” que podem ser usados para construir a interface dos aplicativos incluindo listas, caixas de texto botões e até mesmo um navegador web embarcado;
- Provedores de conteúdo (*Content Providers*) que possibilitam as aplicações acessarem dados de outros aplicativos, como a lista de contatos do telefone ou até mesmo compartilhar seus dados com outros aplicativos;
- Um gerenciador de recursos (*Resource Manager*), para prover acesso a recursos como gráficos e arquivos de layout;
- Um gerenciador de notificações (*Notification Manager*) que possibilita todas as aplicações a mostrarem mensagens de alerta personalizadas;
- Um gerenciador de atividade, ou gerenciador de telas (*Activity Manager*) que gerencia o ciclo de vida de uma atividade (tela) e prove a navegação entre as telas dos aplicativos;

2.2. Localização e Posicionamento

Embora a maioria das pessoas pense que estão familiarizadas com o conceito de localização é útil distinguir as diferentes categorias de informações de localização. Segundo Kupper (2005), basicamente o termo localização se refere a um lugar físico no mundo real. A classe de lugares físicos pode ainda ser subdivido em três categorias de localização:

1. Localização Descritiva - uma localização descritiva é uma localização referenciada por descrições, ou seja, nomes, números ou identificadores. É um conceito fundamental no nosso cotidiano usado pelas pessoas para referenciar lugares. Localizações descritivas referem-se geralmente a objetos geográficos naturais como montanhas e lagos ou objetos construídos pelo homem como estradas e prédios;

2. Localização Espacial - estritamente falando, a localização espacial representa um único ponto no espaço euclidiano. É geralmente expressa por meio de coordenadas bidimensionais ou tridimensionais que são dadas como um vetor de números, cada um dos quais fixa a posição em uma dimensão.
3. Locais de Rede - Locais de rede referem-se a topologia de uma rede de comunicação, como a Internet ou sistemas celulares. Estas redes são compostas de várias redes locais conectadas entre si por uma topologia hierárquica de circuitos. A localização de dispositivos nessas redes é conhecida através de endereços de rede baseados em sua topologia. Por exemplo, na internet um determinado computador possui um endereço chamado de IP que serve para localizá-lo no interior da rede;

Os serviços podem ser baseados nas três diferentes categorias de localização, assim uma importante função do LBS é o mapeamento entre as diferentes categorias de locais. Se o posicionamento oferece uma localização espacial ou de rede, muitas vezes será necessário a conversão em um local descritivo, a fim de ser interpretável pelo usuário. Por outro lado, pode ser necessário que um local descritivo tenha que ser mapeado para uma localização espacial para calcular distâncias, por exemplo.

2.2.1. Sistema de Coordenadas Elipsoidal

Sistema de coordenadas no qual a posição é definida pela latitude elipsoidal, longitude elipsoidal e (no caso tridimensional) pela altitude elipsoidal (Kupper, 2005).

- Latitude elipsoidal: Ângulo entre o plano equatorial e a normal ao elipsóide que passa através de um dado ponto. A latitude elipsoidal é positiva se o ponto se situar a norte do equador e negativa caso contrário. Como ilustrado pela Figura 2.2 a latitude é medida na direção vertical.
- Longitude elipsoidal: Ângulo entre o meridiano principal e o meridiano do lugar de um dado ponto. As longitudes consideram-se positivas para leste do meridiano principal. De acordo com a Figura 2.2 a longitude é medida na direção horizontal.

- Altitude elipsoidal: Distância de um ponto ao elipsóide medida ao longo da perpendicular que passa por aquele ponto. A altitude é positiva se o ponto se situar no exterior do elipsóide. A altitude é a altura expressa geralmente em relação ao nível do mar.

Enquanto a altitude pode ser expressa em qualquer unidade de comprimento, tais como metros ou pés, a unidade de latitude e longitude é dada em graus (◦). A latitude tem uma faixa de valores de -90◦ a 90◦, onde -90◦ e 90◦ correspondem aos Polo Sul e Polo Norte da terra respectivamente. A faixa de longitude vai de -180◦ a +180 ◦.

Cada grau é composto de 60 min, e um minuto é então subdividido em 60 segundos. Como as linhas de latitude são organizadas em paralelo, elas têm sempre uma distância constante na superfície da Terra. Por exemplo, um segundo de latitude corresponde a aproximadamente 30 m em todos os lugares da Terra.

Por outro lado, as linhas de longitude convergem nos pólos e, consequentemente, a suas distâncias na superfície dependem da latitude onde a distância é medida. Um segundo de longitude no plano equatorial também é cerca de 30 m, a 45 ◦ de latitude é aproximadamente 22 m, e nos pólos a distância é exatamente 0.

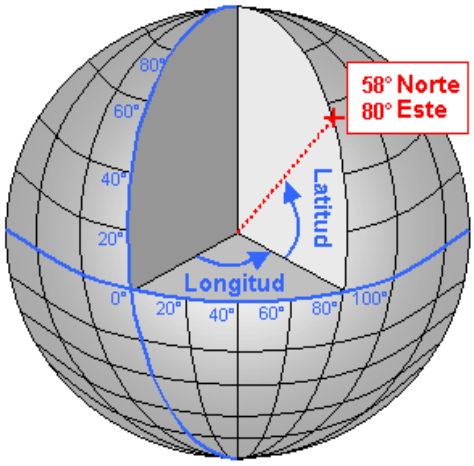

Figura 2.2 – Latitude e Longitude
Fonte: < http://artedanavagacaoaerea.blogspot.com/2011/05/apresentacao-de-coordenadas-geograficas.html>

2.2.2. Sistemas de Posicionamento

Localizar-se na superfície da Terra há muito tempo é uma necessidade do homem, pois, muitas atividades como a navegação precisam de técnicas de posicionamento para se desenvolver. Durante muito tempo as pessoas fizeram uso de técnicas rudimentares para se localizarem, porém, com o advento da informática e da eletrônica, os sistemas se tornaram muito mais confiáveis e precisos. A próxima seção explana sobre os métodos de posicionamento.

2.2.2.1. Triangulação e Trilateração

Métodos de posicionamento precisos, como os que serão abordados na Seção 2.2.2.2, têm suas raízes em agrimensura, na qual abordagens geométricas são usadas para determinar os locais com a ajuda de ângulos e distâncias. Qualquer sistema de posicionamento geográfico que fornece coordenadas ainda faz uso desses princípios.

- Triangulação (Figura 2.3 (a) - *Triangulation*) - são necessárias duas posições fixas, (*p*1 e *p*2) , de cada uma dessas posições mede-se o angulo até a posição *u,* geometricamente falando a localização *u* é obtida com a intersecção das linhas que partem das posições *p*1 e *p*2, com a ajuda de princípios matemáticos é possivel calcular a posição *u*;
- Trilateração (Figura 2.3 (b) - *Trilateration*) - também são necessárias duas posições fixas, porém ao invés de ângulos usa-se duas distâncias até o local desconhecido, a localização *u* é obtida com a intersecção de dois círculos, geralmente existem dois pontos de intersecção, porém é preciso que um deles seja eliminado através de informações adicionais. Em contraste com a triangulação, a trilateração lida com sistemas de equações não-lineares e é necessário empregar métodos numéricos para resolver as equações.

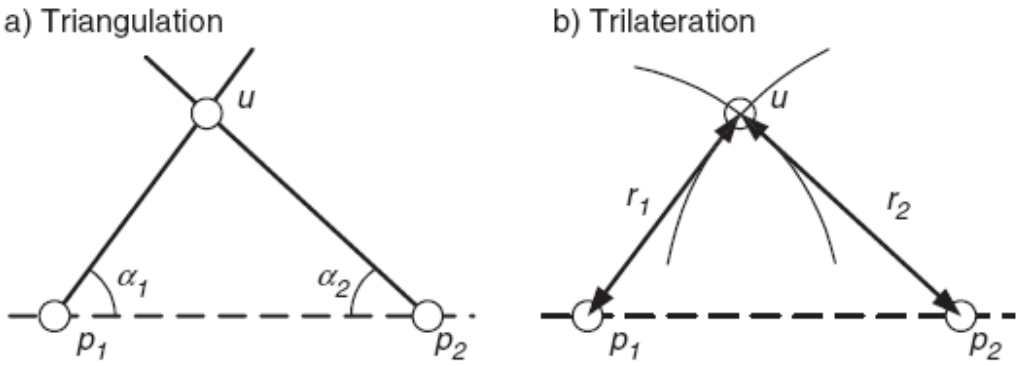

Figura 2.3 – Triangulação e Trilateração
Adaptado de: SCHILLER Jochen; VOISARD Agnès. **Location-Based Services**. São Francisco: Elsevier, 2004, p.180.

2.2.2.2. Técnicas básicas de localização

Segundo Schiller e Voisard (2004), sistemas de localização podem ser divididos em duas categorias básicas: rastreamento e posicionamento. Em sistemas de rastreamento uma rede de sensores é responsável por calcular a posição do usuário e enviá-la ao dispositivo. O dispositivo deve possuir uma identificação especifica para que o sistema de sensores possa identificá-lo e posicioná-lo no espaço.

Em sistemas de posicionamento o dispositivo é responsável por descobrir a sua localização, há uma rede de transmissores que possibilita que o dispositivo se localize.

Há cinco técnicas básicas para a determinação de localização que podem ser usadas em conjunto para aumentar a confiabilidade do sistema:

1. *Cell of Origin* (COO) - esta técnica é usada se o sistema de posicionamento tem uma estrutura celular. Tecnologias de transmissão sem fio tem uma faixa restrita (ou seja, um sinal irradiado está disponível apenas em uma determinada área, da célula). Se a célula tem uma certa identificação ela pode ser usada para determinar uma posição;
2. *Time of Arrival* (TOA), *Time Difference of Arrival* (TDOA) - sinais eletromagnéticos se movem na velocidade da luz. Como esta velocidade é muito alta (cerca de 300.000 km / s), os tempos de execução correspondente são muito curtos. Se assumirmos a velocidade da luz constante, podemos

usar a diferença de tempo entre o envio e a recepção de um sinal para calcular a distância entre o transmissor e o receptor. O mesmo princípio pode ser aplicado em sinais de ultra-som, se a medição for feita em função da diferença entre dois sinais, usa-se o termo TDOA, em redes GSM usa-se o termo *Enhanced Observed Time Difference* (E-OTD) ao invés de TDOA;

3. *Angle of Arrival* (AOA) - antenas direcionais podem determinar a direção de um sinal, caso haja duas ou mais antenas detectando o sinal enviado por um dispositivo é possível calcular a posição do dispositivo;
4. Medição da Intensidade do Sinal - a intensidade de sinais eletromagnéticos mesmo no vácuo decresce na proporção da distância ao quadrado até a fonte, dado um sinal especifico é possível calcular a distância até o seu emissor, esta técnica não é muito precisa, pois os obstáculos reduzem a intensidade do sinal;
5. Processamento de vídeo – padrões de imagens em um vídeo podem ser usados para calcular a posição de um dispositivo móvel, através de técnicas de processamento de imagens é possível identificar os padrões no vídeo, uma vez identificado o dispositivo são empregados as técnicas de AOA para calcular sua posição.

2.2.2.3. Sistemas de posicionamento por satélites

A técnica de posicionamento através de satélites proporciona uma precisão maior quando comparada com as outras técnicas além de funcionarem em qualquer ponto a céu aberto da superfície da Terra.

O princípio de funcionamento é relativamente simples. Em tese o usuário precisa conhecer a distância de ao menos três satélites para calcular sua posição através de simples princípios geométricos. Na prática, porém, um quarto satélite é necessário já que é inviável a sincronização dos relógios dos dispositivos com os relógios dos satélites, o quarto satélite contorna o problema com a sincronia dos relógios.

O Sistema de Posicionamento Global, popularmente conhecido como GPS (*Global Positioning System*) é um sistema baseado em satélites, trata-se de uma

constelação de satélites em órbita ao redor da Terra. O exército americano desenvolveu e implementou essa rede de satélites como um sistema de navegação militar. Posteriormente fora liberado para uso civil.

Cada um dos satélites circunda o globo terrestre a aproximadamente 20.200 quilômetros de altura, completando duas rotações completas a cada dia (GPS.GOV, 2011). As órbitas são dispostas de modo que a qualquer hora do dia, em qualquer lugar na Terra, haja pelo menos quatro satélites "visíveis" no céu.

A função de um receptor GPS é localizar quatro ou mais desses satélites, determinar a distância para cada um e utilizar esta informação para deduzir sua própria posição. Essa operação é baseada em um princípio matemático chamado trilateração, descrito na Seção 2.2.2.1.

O GPS consiste de três segmentos principais: espacial, controle e de usuários (GPS.GOV, 2011).

1. Segmento Espacial - consiste de 24 satélites operacionais, distribuídos em 6 planos orbitais igualmente espaçados. Os planos orbitais são inclinados 55 graus em relação ao equador, dessa forma, a posição dos satélites se repete a cada dia e garante que no mínimo quatro satélites GPS sejam visíveis em qualquer local da superfície terrestre, a qualquer hora;
2. Segmento de Controle - o sistema de controle é composto por 5 estações monitoras, três delas com antenas para transmitir os dados para os satélites além da estação de controle central. As principais tarefas do segmento de controle são monitorar e controlar continuamente o sistema de satélites, predizer as efemérides dos satélites, calcular as correções dos relógios dos satélites e atualizar periodicamente as mensagens de navegação de cada satélite;
3. Segmento de Usuários - é composto pelos receptores GPS, os quais devem ser apropriados para os propósitos a que se destinam. A categoria de usuários pode ser dividida em civil e militar.

O segmento militar faz uso do serviço PPS*(Precise Positioning Service*) que proporciona localização mais precisa, já o segmento civil faz uso do SPS(*Standand Positioning Service*) que proporciona uma precisão menor se comparada ao PPS.

2.2.2.4. *Assisted GPS*

Os receptores GPS convencionais, por diversos fatores, nem sempre conseguem se comunicar com os satélites em condições ideais, fato que resulta em graves atrasos no processo de localização. O *Assisted GPS* ou A-GPS surgiu para contornar esse problema. O A-GPS se comunica com um servidor assistente via conexão de dados para troca de informações de suporte para calcular com mais rapidez e confiabilidade as coordenadas do usuário.

Cada servidor assistente é responsável por fornecer a lista de coordenadas de todos os satélites presentes no espaço. Quando isto acontece, a operadora de telefonia rastreia a torre na qual o celular com A-GPS está conectado. Como a operadora conhece as coordenadas de cada torre, ela processa a lista dos satélites juntamente com outros dados e seleciona as coordenadas dos satélites que provavelmente estão visíveis para o celular requisitante. Após a seleção, essas informações são enviadas ao aparelho por meio da rede de telefonia. Com isso, diminui-se o Tempo de Localização Inicial (TTFF – *Time To First Fix*) de todo o sistema, fato que reduz, conseqüentemente, o tempo que um aparelho leva para receber a sua localização geográfica.

2.2.2.5. Sistemas de posicionamento baseados em redes

A instalação de sistemas de posicionamento demanda um investimento significativo. Para reduzir os custos, as redes sem fio podem ser usadas para serviços de posicionamento. Um exemplo de uso desses sistemas é o das redes GSM já que são usadas em mais de 190 países (SCHILLER;VOISARD, 2004).

Em redes GSM, a localização aproximada de um usuário pode ser obitida sem adaptações adicionais ao sistema já que a localização das antenas é conhecida, e cada um dos usuários da rede econtra-se dentro da área de alcance de uma das antenas da rede.

Quando um usuário adentra uma área especifica da rede ele é registrado em um banco de dados descentralizado chamado *Visitor Location Register* (VLR), esta

informação é repassada ao banco de dados central, o *Home Location Register* (HLR). Cada operadora de telefonia possui o seu HLR, dessa forma é possível localizar usuários com a precisão do tamanho da área de cobertura de uma antena.

Com o propósito de melhorar a precisão da localização de usuários foi desenvolvido um sistema chamado *Mobile Positioning System* (MPS), o MPS não traz grandes alterações à infra-estrutura da rede e não há necessidade de adaptações aos aparelhos celulares já que o cálculo de posicionamento é efetuado pela rede. Segundo Schiller e Voisard (2004), para o cálculo de posicionamento o MPS utiliza os seguintes mecanismos:

- *Cell of Global Identity* (CGI) - técnica baseada no mecanismo COO, possui baixa precisão e somente é usada quando não há disponibilidade de métodos mais precisos;
- *Segment antennas* - as estações base, muitas vezes têm antenas, que dividem os 360 graus em dois, três, ou quatro segmentos. Assim, uma estação base pode limitar a localização de um usuário móvel a um segmento angular de 180, 120, ou 90 graus. Técnica baseada no mecanismo AOA;
- *Timing Advance* (TA) - os celulares e as estações base usam intervalos de tempo determinados para a comunicação, como o tempo deve ser exato, o sistema leva em conta o tempo de execução do sinal entre uma estação base e um celular. Estas informações podem ser usadas para determinar a posição dentro de uma célula com mais precisão, tal método é baseado no mecanismo TOA;
- *Uplink Time of Arrival* (UL-TOA) - quando um celular é visível no raio de quatro estações base a posição do usuário pode ser determinada com precisão de 50 a 150 metros levando em consideração o tempo de execução dos sinais entre o celular e as estações base, tal método é baseado no mecanismo TOA;

Os dados de localização determinada por diferentes procedimentos são transmitidas ao *Mobile Positioning Center* (MPC), dessa maneira as informações de localização não ficam diretamente disponíveis aos dispositivos móveis que precisam solicitá-los ao MPC quando necessário.

2.3. Sistemas de Informações Geográficas (SIGS)

Para o correto entendimento do modelo de comunicação LBS e de sua arquitetura em três camadas que será introduzida na Seção 2.4.4 é necessário primeiramente conceituar Sistemas de Informações Geográficas (SIG).

A localização espacial representa uma forma apropriada de referenciar um local exato na superfície da Terra, a maioria dos métodos de posicionamento usados na área de LBS, como o GPS, obtém as informações de posicionamento através de uma série de medidas e cálculos, no entanto a localização espacial não é uma abordagem intuitiva compreendida em todas as situações. Por exemplo, fornecer a posição N 48 ◦ 21 '17 "L 11 ◦ 47' 15 " a um usuário de LBS é menos siginificativo do que simplesmente informar ao usuário que o determinado local situa-se no aeroporto de Munique na Alemanha (KUPPER, 2005).

Problema semelhante acontece quando da fixação de duas posições para cálculo de distância entre tais posições a fim de mostrar a distância e o tempo gasto entre as duas posições, uma primeira abordagem seria calcular a distância em linha reta entre os dois pontos, no entanto seria bem mais conveniente para o usuário obter a distância em uma rede de estradas ou em uma rede de transportes públicos de preferência exibindo um mapa mostrando inclusive a rota mais curta. Como estes exemplos demonstram, é inevitavel o mapeamento estre localizações espaciais e descritivas a fim de tornar as informações mais convenientes aos usuários.

Bases de dados espaciais e Sistemas de Informações Geográficas (SIGs) são a chave para desempenhar essas tarefas, enquanto eles cobrem uma extensa gama de aplicações, como por exemplo na área de agrimensura, cartografia e transporte, no contexto dos LBSs eles são importantes para indicar as posições de um ou vários locais em relação ao conteúdo geográfico como fronteiras de cidades e países, edifícios, redes de estradas, etc. Eles são usados para o mapeamento entre informações de localização espacial e informações de localização descritiva, o que é chamado de goeocodificação reversa e geocodificação respectivamente, bem como para criação de mapas digitais e informações de roteamento.

2.3.1. Definição de Sistemas de Informações Geográficas (SIGs)

A gama de definições de SIG existente é muito extensa e muitas vezes controversa, uma definição mais abrangente deriva de (RIGAUX[3] et al. 2002 apud KUPPER, 2005) que caracteriza as funções de um SIG como: "sistema que armazena dados geográficos, recupera e combina esses dados para criar novas representações do espaço geográfico, fornece ferramentas para análise espacial e faz simulações para ajudar usuários experientes a organizar seus trabalhos em muitas áreas incluindo administração pública, rede de transportes, aplicações militares e sistemas de informação ambiental".

Outra definição é dada por (MCDONNELL; KEMP[4], 1996 apud KUPPER, 2005), que descreve um SIG como "um sistema de computador para capturar, gerenciar, integrar, manipular, analisar e exibir dados que são espacialmente referenciados na Terra. "

Para entender a idéia por trás do SIG, é necessário distinguir entre dois níveis de abstração, a camada superior em um SIG, o chamado modelo de dados geográficos (*Geographic Data Model*), ilustrado pela Figura 2.4 fornece uma visão conceitual dos conteúdos geográficos em termos de unidades chamadas *features*. Uma *feature* representa uma unidade do mundo real, por exemplo, um edifício, uma estrada, um rio, um país ou uma cidade. Ela consiste de um componente espacial, que corrige a sua localização, forma e relação topológica com outras entidades, bem como uma descrição, que fornece informações não-espaciais sobre a entidade, por exemplo, o nome de uma cidade ou estrada, ou a população de um país.

Cada *feature* tem um conjunto bem definido de operações, que é adaptado ao tipo de entidade do mundo real que ela representa. Por exemplo, uma *feature* que representa uma estrada pode conter uma operação que fornece seu comprimento, enquanto que a de uma cidade pode conter uma operação para a obtenção de sua área. As operações são usadas pelas aplicações para manipular ou solicitar informações espaciais ou descritivas de uma *feature* e relacioná-la com outras.

---

[3] RIGAUX, P.; SCHOLL, M.O.; VOISARD, A. Spatial Databases with Application to GIS. Morgan Kaufmann Publishers, 2002.

[4] MCDONNELL, R., KEMP, K. International GIS Dictionary. John Wiley & Sons. mGain. (2003). Web site of the EU's Information Society Technologies (IST) project mGain.http://www.mgain.org/

Assim, o modelo de dados geográfico representa a interface entre o SIG e a aplicação.

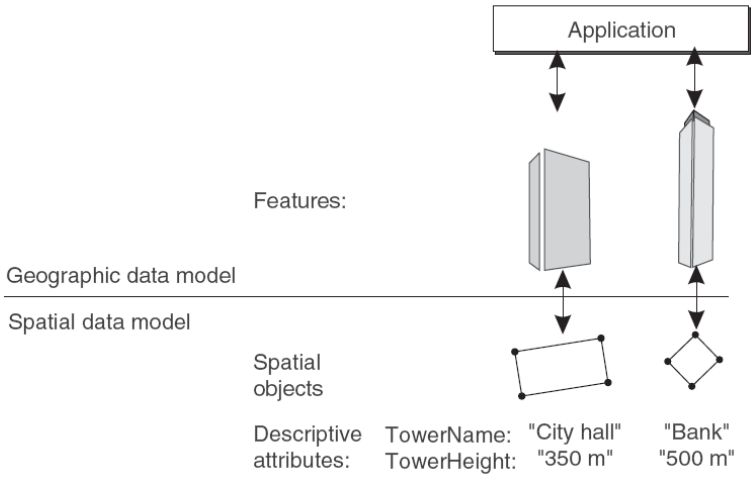

Figura 2.4 - Abordagem do SIG por camadas
Fonte: KUPPER, Axel. Location-Based Services - Fundamentals And Operation. Wiley, 2005,p.37.

A camada inferior, indicada na Figura 2.4 por *Spatial data model* (Modelo de dados espacial), lida com os aspectos do gerenciamento de dados físicos incluindo armazenamento, processamento de consultas e otimização, bem como de concorrência e recuperação. Uma particular função do SIG é a representação dos componentes espaciais das *features*, que são chamados objetos espaciais, bem como sua combinação com atributos descritivos para a manutenção de suas descrições. Objetos espaciais podem ser representados de várias maneiras. Geralmente, ela é classificada em duas categorias de representação: modo de varredura (Raster mode) e modo vetorial (Vector mode) ilustradas pelo Figura 2.5.

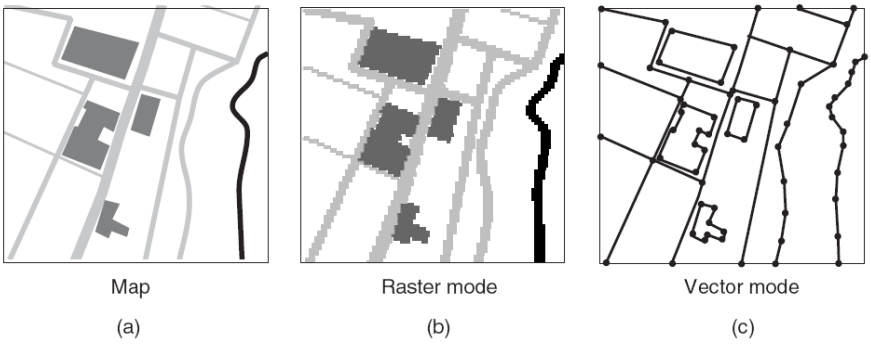

Figura 2.5 – Modelos de representação de objetos espaciais
Fonte: KUPPER, A. **Location-Based Services - Fundamentals And Operation.**Wiley, 2005, p.38.

2.4. *Location Based Services (LBS)*

Segundo Schiller e Voisard (2004), LBS, ou Serviços Baseados em Localização podem ser definidos como aplicações que integram localização geográfica com a noção geral de serviços. Exemplos dessas aplicações são serviços de emergência, sistemas de navegação para carros, aplicativos de planejamento turístico ou os chamados “mapas amarelos” (combinação de páginas amarelas e mapas).

Serviços de localização têm longa tradição, desde os anos 70 o Departamento de Defesa dos Estados Unidos tem operado o Sistema De Posicionamento Global (GPS), uma infra-estrutura de satélites que fornece a localização de pessoas e objetos. Inicialmente, o GPS foi concebido com propósitos militares, porém nos anos 80 o governo dos Estados Unidos decidiu tornar livre o uso do GPS para outros propósitos. Desde então a indústria mundial tem aproveitado o GPS para melhorar seus produtos e serviços, por exemplo, a indústria automotiva que tem integrado sistemas de navegação nos carros há um bom tempo (KUPPER, 2005).

Em sistemas de posicionamento tradicionais, as informações de localização geralmente derivam de um dispositivo com ajuda dos satélites do GPS para um determinado propósito, mas foi a partir do final dos anos 90 que o interesse generalizado em sistemas baseados em localização foi realmente impulsionado por um novo mercado, o da telefonia móvel.

Por volta de 1997, as redes de telefonia móvel já haviam sido amplamente implantadas em todo o mundo e, apesar dos rendimentos com os serviços de comunicação terem provado serem uma enorme fonte de renda, o crescimento das redes começa a se mostrar limitado. Conseqüentemente, as operadoras começaram a procurar por novas fontes de renda e uma das principais formas de colher benefícios financeiros adicionais das redes móveis, além da vóz, é oferecendo serviços de dados.

Para continuar alimentando o crescimento dos negócios, as operadoras precisaram investir mais em novas tecnologias, especialmente em serviços de mensagens e internet móvel, e nesse contexto a localização do usuário se tornou um quesito importante para agregar valor aos serviços e otimizar a experiência de

uso dos consumidores criando assim um novo conceito de serviços, o de serviços baseados em localização.

2.4.1.Taxonomia dos Serviços Baseados em Localização

Analistas e pesquisadores tomam varias abordagens para classificar os aplicativos LBS, uma grande distinção entre os serviços é se eles são orientados a pessoas ou orientados aos dispositivos (SCHILLER; VOISARD, 2004) :

- LBS orientado a pessoa compreende todos os aplicativos baseados no usuário, assim o foco do aplicativo é o posicionamento de uma pessoa ou usar a posição de uma pessoa para melhorar o serviço, geralmente a pessoa localizada controla o serviço;
- LBS orientado ao dispositivo é externo ao usuário, dessa forma esses aplicativos também poderão dar enfase a localização de uma pessoa, mas não necessariamente. Em vez de apenas uma pessoa, um objeto ou um grupo de pessoas podem ser localizados, nesse tipo de aplicativo o objeto ou pessoa localizada não está no controle da aplicação, por exemplo um sistema de monitoramento contra roubo de carros;

Além dessa abordagem, Schiller e Voisard (2004) classificam os serviços em *push* e *pull* :

- *Push Services* compreende os aplicativos em que o usuário recebe sua localização em função do seu paradeiro sem ter que ativamente solicitar por ela, a informação pode ser enviada com ou sem consentimento prévio;
- *Pull Services* por outro lado compreende os aplicativos em que o usuário requisita ativamente informações de localização, como por exemplo, onde encontrar o cimema mais próximo;

Outra maneira de se classificar os LBS é levando em consideração suas funcionalidades:

- Serviços de informação e lazer - serviços que fornecem como chegar a um determinado lugar, onde encontrar determinado serviço ou encontrar outro usuário, ou mostrar a localização no mapa de um usuário perdido são exemplos desses aplicativos. As informações podem ser mais refinadas

correlacionando-as com informações climáticas e as condições de rodovias por exemplo. Aplicativos desse tipo geralmente são de natureza *pull* já que o usuário requisita ativamente as informações de localização;

- Serviços de rastreamento - compreende os aplicativos em que é monitorado o paradeiro geográfico de entidades equipadas com terminais móveis (por exemplo caminhões pertencentes a frota de uma empresa) a fim localizar essas entidades, tomar decisões de logísticas ou evitar que as entidade desviem de uma rota estabelecida previamente. Aplicativos desse tipo podem ser de natureza *pull* ou *push*;
- Serviços de disseminação seletiva de informações - se refere a disseminação de informações e serviços aos usuários de acordo com sua localização, perfil e contexto. Um exemplo é a divulgação de propagandas, pela qual anunciantes poderiam enviar mensagens a usuários próximos aos seus estabelecimentos, essa categoria de aplicativos é de natureza *push* já que os usuários receberiam informações sem requisitar por elas;
- Serviços de tarifação sensíveis a localização - aplicativos dessa categoria poderiam diferenciar a aplicação de tarifas de serviços em função da localização do usuário, um exemplo seria um pedágio que cobraria suas taxas proporcionamente ao espaço percorrido por um automóvel em uma via oferecendo um serviço de tarifação mais justa;
- Serviços de Emergência - em momentos críticos o fator localização aliado ao fator tempo pode ser vital para a solução de problemas, uma infinidade de aplicações podem ser implementadas levando em consideração esses fatores, serviços de socorro, de emergência, acidentes, ocorrências policiais são alguns exemplos.

2.4.2. LBS e Privacidade

Muitos estudos têm mostrado que os consumidores se preocupam com sua privacidade, como resultado as operadoras e agências de propaganda devem ser cuidadosos quando se trata da localização das pessoas. Além da impressão por parte dos usuários de estarem sendo observados, o *spam* também se tornou uma

grande preocupação no contexto dos serviços baseados em localização já que a relevancia das mensagens é bem maior quando se tem a localização do usuário ou do dispositivo (SCHILLER; VOISARD, 2004).

Teme-se que um problema semelhante ao que acontece na internet com relação ao spam possa emergir no mundo móvel, mensagens não solicitadas enviadas para telefones celulares podem ser bem mais prejudiciais do que e-mails indesejados já que o celular é um aparelhor pequeno, confiável e geralmente é mantido junto ao corpo do usuário, consequentemente a atenção do usuário seria tomada por toda e qualquer mensagem de spam recebida pelo aparelho.

Do ponto de vista das operadoras, mensagens indesejadas sendo recebidas por seus usuários poderiam trazer custos enormes em assistência, já que a relação entre os usuários e as operadoras é bem mais íntima do que a relação entre provedores de internet e seus usuários, quando há um problema o usuário de celular é muito mais propenso a contatar a operadora do que o usuário de internet contatar um provedor de internet.

2.4.3. O Mercado LBS

O mercado dos serviços baseados em localicação se desenvolve em torno de negócios e consumidores e podem ser agrupados em duas esferas de mercado (SCHILLER; VOISARD, 2004):

- O mercado vertical é caracterizado por usuários provenientes da indústria em que a gestão da informação de localização é e sempre foi parte do negócio, por exemplo, a localização de um taxi pertencente a uma frota de uma determinada empresa é fundamental para suas operações, da mesma forma que aeroportos precisam de controle de tráfego aéreo e consequentemente necessitam saber a localização das aeronaves;
- O mercado horizontal, por outro, lado se caracteriza por usuários provenientes da indústria onde o uso de celulares e de infomações sobre localização vêm para agregar valor aos serviços ja existentes, por exemplo, um novo sistema de rastreamento de bens de auto valor para manter baixas as taxas de seguro e manter a competitividade dentro do segmento. Hoje o

mercado horizontal tem oferecido um enorme pontencial para desenvolvedores de software de terceiros (SCHILLER; VOISARD, 2004). Como o recurso de localização pode ser usado para melhorar os serviços tradicionais novos canais de comercialização podem ser explorados para atingir um mercado em massa e ao mesmo tempo produtos e serviços já existentes podem ser reinventados usando localização para agregar valor aos serviços.

2.4.4. O Modelo de Comunicação LBS

A fim de tornar possível o desenvolvimento e a expansão do LBS, a indústria teve que superar vários desafios de natureza tecnológica e econômica nos últimos anos. Tecnologicamente é possível definir a arquitetura LBS em um modelo em três camadas, incluindo uma camada de posicionamento, uma camada de *middleware* e uma camada de aplicação, conforme a Figura 2.6.

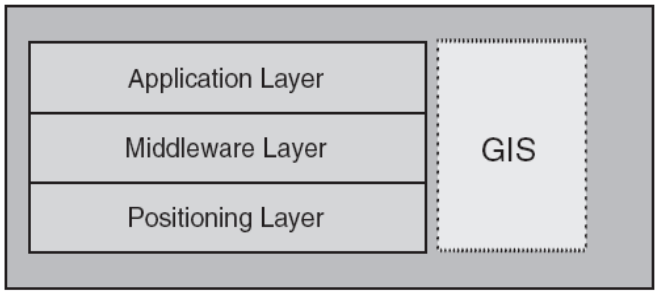

Figura 2.6 – Modelo de comunicação LBS
Fonte: SCHILLER J.; VOISARD A. **Location-Based Services**.São Francisco: Elsevier, 2004, p.23.

A camada de posicionamento (*Positioning Layer*) é responsável por calcular a posição de um dispositivo móvel fazendo uso de equipamentos próprios para a determinação do posicionamento e de dados mantidos em um Sistema de Informação Geográfica (SIG). Os aplicativos e serviços que demandam as informações de localização compõem a camada de aplicação (*Application Layer)*.

A camada de *middleware* é responsável por gerenciar e mediar a comunicação entre as camadas de aplicação e localização, tal camada oculta os

detalhes de implementação fornecendo uma interface padronizada para os desenvolvedores.

Inicialmente a camada de posicionamento desempenharia o papel de gerenciar os pedidos e prover a localização a todos os aplicativos da camada de aplicação que solicitassem a informação, porém com o advento da indústria houve a necessidade de inserir uma camada de *middleware* a fim de facilitar a integração entre os serviços e reduzir a sua complexidade. Promover a fácil integração entre os serviços a fim de oferecer um modelo de acesso "por atacado" as informações de localização é vital para as operadoras de telefonia móvel, já que uma infinidade de aplicativos que por sua vez possuem uma infinidade de usuários pode estar constantemente requisitando informações de localização à rede.

2.5. Web Services

De acordo com o Word Wide Web Consortium (W3C), um *web service* é um sistema de software projetado para suportar interoperabilidade entre máquinas através de redes de computadores. Possuem uma interface descrita em um formato processável por máquina. Outros sistemas interagem com os *web services* de maneira padronizada pela sua descrição usando mensagens SOAP[5], geralmente transmitidas através do protocolo HTTP.[6]

Basicamente, um *web service* é uma espécie de método que recebe valores na forma de parâmetros, executa um determinado processamento com eles e devolve um resultado. Todas a mensagens são baseadas em XML[7].

A disponibilização das operações e a descrição do serviço também ocorrem através do padrão XML sendo que o arquivo descritor do serviço possui todas as informações necessárias para que outros sistemas possam interagir com o serviço, provendo abstração ao modelo e permitindo que o mesmo seja acessado independentemente da plataforma de *software* que foi implementado, bastando que

[5] *Simple Object Access Protocol* (SOAP) é um protocolo destinado a troca de informações estruturadas em um ambiente distribuído e descentralizado.
[6] *HyperText Transfer Protocol* (HTTP) é o protocolo de transferência usado em toda a internet. Ele especifica as mensagens que os clientes podem enviar aos servidores e que respostas eles receberão.
[7] *Extensible Markup Language* (XML) é uma linguagem de marcação auto descritiva para transportar e armazenar dados.

as mensagens sejam baseadas em XML.

Web services, geralmente, são projetados para promover a interoperabilidade de sistemas, já que tornam os serviços ou aplicações fracamente acoplados ao restante do sistema de *software*.

2.5.1. Arquitetura Orientada a Serviço – SOA

A arquitetura SOA (*Service-Oriented Architecture*), é uma arquitetura para construção de aplicações que se comportam como uma coleção de componentes em forma de "caixas-pretas" fracamente acopladas, projetadas para fornecer serviços bem definidos interligando os processos de negócio (HURWITZ et al.,2009).

A arquitetura SOA promove o fraco acoplamento e ligação dinâmica entre os serviços. Alguns fatores importantes do coração da arquitetura SOA são necessários para que ela funcione efetivamente: primeiramente é necessário prover uma definição abstrata do serviço em questão incluindo os detalhes que permitem que qualquer um que queira usar o serviço possa se comunicar com ele de forma apropriada. Em segundo o provedor do serviço (*Service provider*) precisa publicar os detalhes do serviço para que os usuários possam entender mais precisamente o que serviço faz e obter as informações necessárias para se conectar e fazer uso do serviço. Por ultimo é necessário que os usuários (*Service requester*) dos serviços possuam uma forma de descobrir quais serviços estão disponíveis para alimentar suas necessidades, esse conjunto de fatores (*bind/ publish/ find*) ilustrados pela Figura 2.7, compõem a arquitetura SOA.

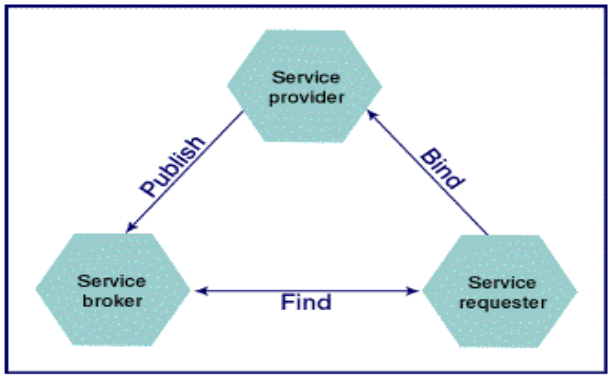

Figura 2.7 – Arquitetura orientada a serviço
Fonte: < http://www.ibm.com/developerworks/ibm/library/i-portletintro/>

2.5.2. Web Services SOAP

Os *Web Services* seguem o estilo de arquitetura SOA, *Web Services* podem ser entendidos como serviços na Internet descritos via WSDL, registrados via UDDI, acessados utilizando SOAP e a transmissão de dados feita através de arquivos XML (W3C TECHNICAL ARCHITECTURE GROUP, 2004). SOAP (*Simple Object Access Protocol*) é o protocolo-padrão para transmissão de dados dentro da arquitetura de *web services* proposta pelo W3C. O SOAP é um protocolo baseado no XML e segue o modelo "REQUEST-RESPONSE" do HTTP.

WSDL (*Web Services Description Language*) é um arquivo do tipo XML, cuja finalidade é descrever detalhadamente um Web Service. Essa descrição especifica as operações que compõem o *web service* e define de forma clara como deve ser o formato de entrada e saída de cada operação.

UDDI (*Universal Description, Discovery and Integration*) é um mecanismo que visa atender tanto ao cliente do *web service* quanto ao provedor. Ele tem de fornecer ao provedor de *web services* meios para que os mesmos sejam registrados e publicados. Isso permitirá que os *web services* sejam pesquisados e localizados pelos clientes, outra finalidade do UDDI é o armazenamento de arquivos WSDL.

A arquitetura de *web services* se baseia na interação de três entidades ilustradas pela Figura 2.8: provedor do serviço, cliente do serviço e servidor de registro. De uma forma geral, as interações são para publicação, busca e uso (*bind*) de operações.

O provedor do serviço representa a plataforma que hospeda o *web service* permitindo que os clientes acessem o serviço. O cliente do serviço é a aplicação que está procurando, invocando ou iniciando uma interação com o *web service*. O cliente do serviço pode ser uma pessoa acessando através de um browser ou uma aplicação realizando uma invocação aos métodos descritos na interface do *web service*.

O servidor de registro é responsável pelo registro e busca de *web services* baseados em arquivos de descrição de serviços (WSDL) que foram publicados pelos provedores de serviços.

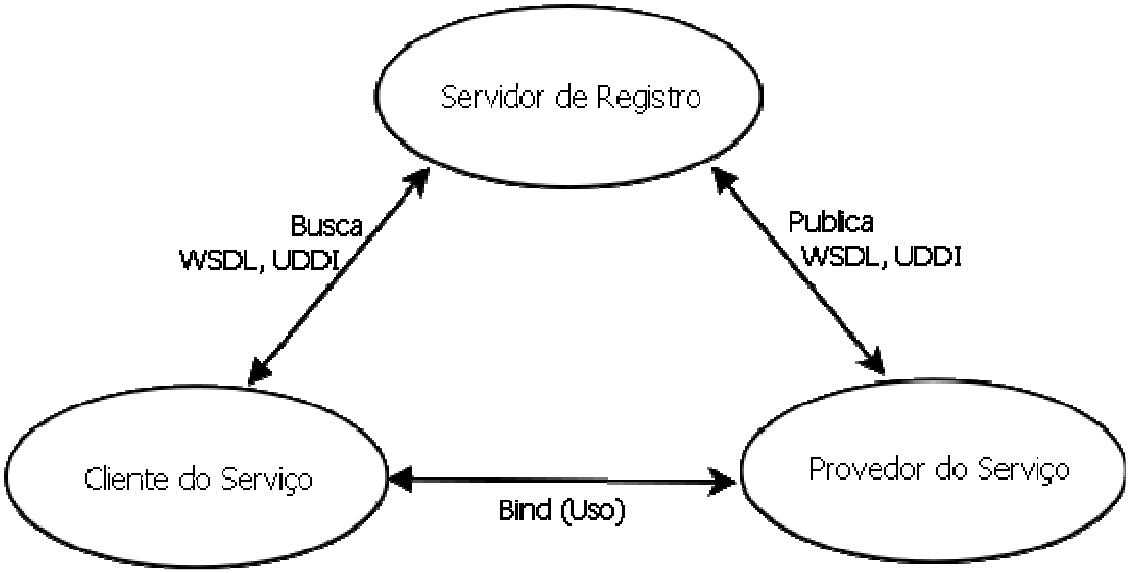

Figura 2.8 – Web Services SOAP

2.6.3. Web Services REST

REST (*Representational State Transfer*), ou Transferência de Estado Representacional, descreve um estilo de arquitetura para sistemas hipermídia derivado de vários estilos baseados em redes como a Web proposto por Roy Thomas Fielding. Nessa técnica as mensagens trocadas são encapsuladas diretamente no protocolo HTTP. O principal diferencial dessa técnica em relação a técnica SOAP é o foco nos recursos e não nas chamadas aos procedimentos/serviços.

> A Transferência de Estado Representacional (REST) é uma abstração dos elementos arquiteturais dentro de um sistema hipermídia distribuído. REST ignora os detalhes da implementação dos componentes e sintaxe do protocolo, a fim de se concentrar sobre os papéis dos componentes, as restrições sobre sua interação com outros componentes e sua interpretação de elementos significativos de dados. Ele abrange as restrições fundamentais sobre os componentes, conectores e dados que definem a base da arquitetura presente na internet, e assim, a essência do seu comportamento como aplicação baseada em rede. (FIELDING, 2000, tradução nossa).

No REST são feitas requisições para recursos e não serviços. Essa abordagem pode ser interessante para aplicações em que é mais importante a interoperabilidade do que um contrato formal entre as partes.

A arquitetura REST é baseada em quatro princípios:

1. Identificação de recursos através de URI[8]. A arquitetura REST foca principalmente em recursos para interação com os clientes. Tais recursos são identificados através de URIs. Com isso, cada recurso tem um endereço global que pode ser usado para descoberta de serviços;
2. Interface Uniforme. Os recursos são manipulados utilizando um conjunto de quatro operações: inserção, atualização, exclusão e listagem/leitura. Cada operação utiliza um método de requisição HTTP diferente. O método GET retorna o estado corrente do recurso em alguma representação, o método PUT insere um novo recurso, que pode ser excluído com o método DELETE, já o método POST transfere um novo estado para um recurso;
3. Mensagens auto-descritivas. Os recursos estão desassociados de suas representações. Dessa maneira, estes podem ser acessados em diferentes formatos. Metadados em relação aos recursos estão disponíveis e são usados, por exemplo, para negociar o formato apropriado da representação;
4. Interações de estado através de *links*. Todas as interações com os recursos são sem estado. Entretanto, os estados podem ser explicitamente transferidos. Os estados podem ser embutidos nas mensagens de resposta para garantir a futura interação com estes.

A arquitetura de *Web Services* em REST é simples, visto que são utilizados padrões bem conhecidos (HTTP, XML, URI, por exemplo).

A implementação, tanto do servidor quanto do cliente, é possível na grande maioria das linguagens de programação e sistemas operacionais. Tendo isso em vista, a barreira para adoção dessa arquitetura é muito pequena. O desenvolvedor pode testar os serviços no próprio navegador Web. A implantação é bem similar à de um *website* dinâmico.

A escolha da arquitetura REST, em relação à baseada em SOAP e WSDL, remove a necessidade de muitas decisões futuras. *Web services* SOAP possuem várias camadas e maior complexidade, em alguns casos, supérflua. Caso seja necessário, a implementação de *web services* SOAP em aplicações que implementam REST é bem mais trabalhosa que o contrário, devido ao maior numero de camadas e protocolos.(XAVIER, 2010).

[8] *Uniform Resource Identifier* (URI) é uma cadeia de caracteres compacta que identifica recursos na internet (W3C TECHNICAL ARCHITECTURE GROUP, 2006).

Web services SOAP consomem maior banda que a arquitetura REST. Com isso, recomenda-se o uso de REST em aplicações que possam ser acessadas por dispositivos móveis ou de maneira ubíqua. Entretanto, *web services* REST não possuem um contrato formal bem definido. Sendo assim, é necessário que o produtor do serviço e seu consumidor (cliente) conheçam-no e estejam contextualizados.

3 – RECURSOS COMPUTACIONAIS PARA A IMPLEMENTAÇÃO

O aplicativo idealizado nesse trabalho, denominado Traveller e classificado como um Serviço Baseado em Localização tem por objetivo auxiliar usuários de celulares equipados com GPS em locais desconhecidos através do fornecimento de informações climáticas, de localização do usuário e informações de estabelecimentos em áreas urbanas, além de apontar no mapa a localização atual do dispositivo e de cidades de interesse sendo complementado com funções relativas à localização, como o cálculo de distâncias entre coordenadas geográficas.

Para o desenvolvimento do Serviço Baseado em Localização Traveller foi usada a ferramenta de desenvolvimento integrada ao Eclipse e a biblioteca de mapas e localização para Android do Google Maps.

Com o objetivo de integrar posicionamento e informações de localização o aplicativo Traveller faz uso de três diferentes *web services*: o Apontador, o Google Weather e o GeoNames.

3.1 - Ambiente de Desenvolvimento Android

O Android SDK (*Software Development Kit*), ou kit de desenvolvimento de software, é o conjunto de aplicativos utilizados para desenvolver aplicações no Android, possui um emulador para simular o celular, ferramentas utilitárias e uma API[9] completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações. Embora o SDK tenha um emulador que pode ser executado como um aplicativo comum, existe um *plugin* para o Eclipse, que visa justamente, integrar o ambiente de desenvolvimento Java com o emulador. Com o *plugin* é possível iniciar o emulador diretamente dentro do Eclipse, instalando a aplicação no emulador automaticamente e, com o *debug* integrado do Eclipse, é possivel depurar o código-fonte como qualquer outra aplicação Java.

[9] API (*Application Programming Interface*) ou interface de programação de aplicativos é um conjunto de padrões e rotinas estabelecidos por um software para sua utlização por aplicativos, a API é composta por uma série de funções acessíveis por programação.

As próximas sub-seções abordarão os principais conceitos relacionados ao desenvolvimento de aplicações para o sistema operacional Android.

3.1.1 - Activity

A classe android.app.activity representa basicamente uma tela de uma determinada aplicação. Uma tela é composta de vários elementos visuais, os quais no Android são representados pela classe android.view.View.

Essas duas classes se relacionam constantemente. A classe Activity define que existe uma tela, controla seu estado e a passagem de parâmetros de uma tela para outra, define os métodos que serão chamados quando o usuário pressionar algum botão, etc. Mas a Activity precisa exibir elementos visuais na tela, e este é o papel da classe View, que tem a finalidade de desenhar algo na tela.

Uma View pode ser um simples componente gráfico ou pode ainda ser um elemento complexo, que atua como um gerenciador de *layout*, a qual pode conter várias Views-filhas e tem a função de organizar as mesmas na tela.

Para cada tela da aplicação existirá uma Activity para controlar seu estado e eventos, mas para definir a interface gráfica da tela é utilizado uma View.

Uma Activity tem um ciclo de vida bem definido, cada Ativity que é iniciada é inserida no topo de uma pilha, chamada de "*activity stack*", assim sempre que uma nova Activity é inserida no topo da pilha, a *Activity* anterior que estava em execução fica logo abaixo da nova.

A Activity que está no topo da pilha é a Activity que está em execução no momento, e as demais podem estar executando em segundo plano, estar no estado pausado ou totalmente paradas.

O sistema operacional é responsável por controlar o ciclo de vida de uma Activity, a Figura 3.1 representa o ciclo de vida completo de uma Activity, exibindo os estados possíveis e a chamada de cada método.

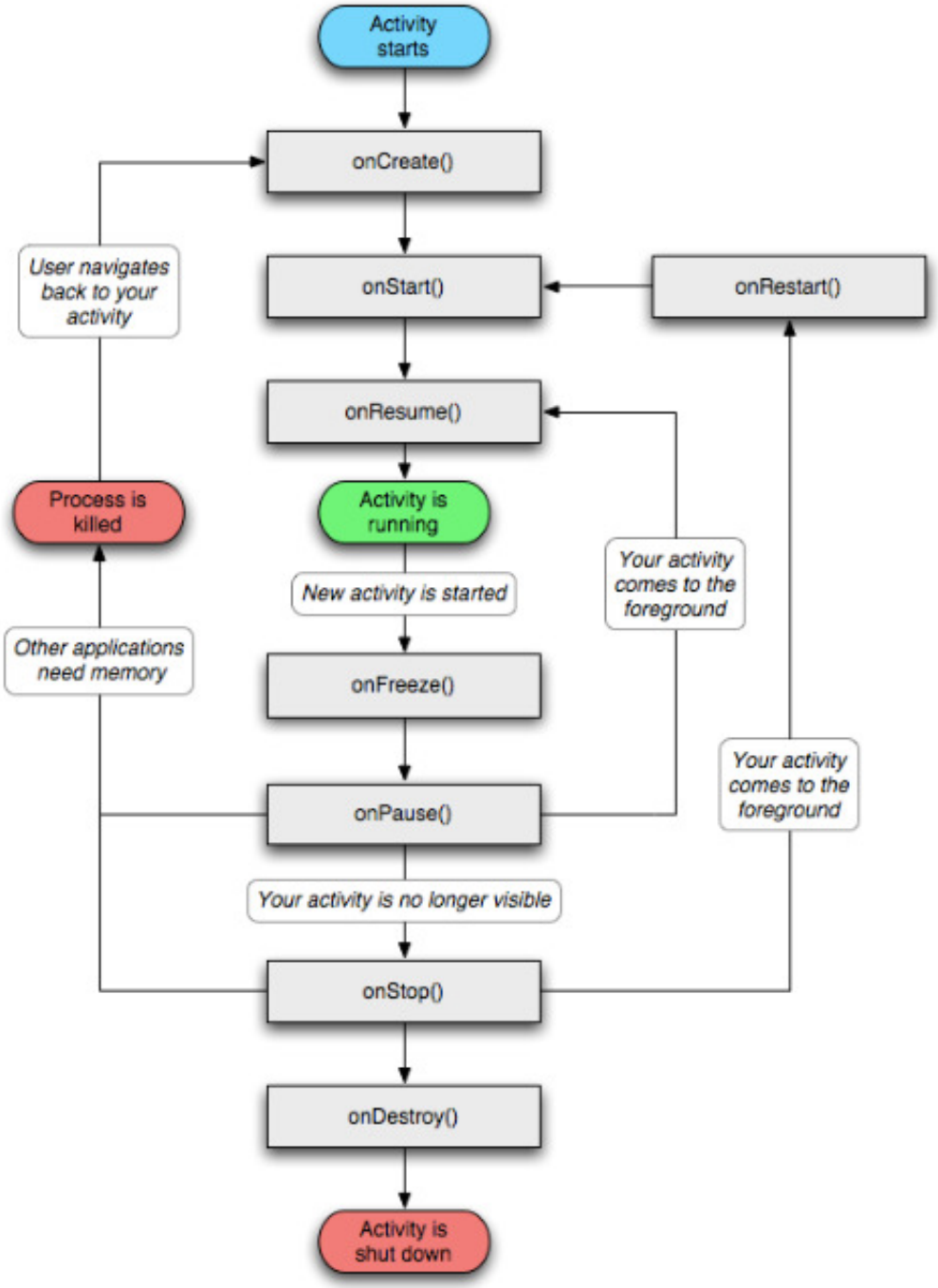

Figura 3.1 – Ciclo de vida de uma Activity
Fonte: < http://developer.android.com/guide/topics/fundamentals/activities.html->

Segundo a documentação do Android, há três subníveis do ciclo de vida principal, que por sua vez ficam se repetindo durante a execução da aplicação. Esses três ciclos são chamados de *entire lifetime, visible lifetime* e *foreground lifetime*.

- *Entire lifetime*: Esse ciclo ocorre apenas uma vez e define o tempo de vida completo de uma Activity. Ele acontece entre as chamadas do método *onCreate()* e *onDestroy(),* os quais são chamados apenas uma única vez, quando a Activity é criada e destruída, respectivamente;

- *Visible lifetime*: A Activity já foi iniciada durante esse ciclo, mas pode estar no topo da pilha interagindo com o usuário ou temporariamente parada em segundo plano, esse ciclo ocorre entre os métodos *onStart()* e *onStop();*

- *Foregroud Lifetime*: Durante esse ciclo a Activity está no topo da pilha e interagindo com o usuário, esse ciclo ocorre entre os métodos *onResume()* e *onPause().*

3.1.2 - A Classe R

A classe R é gerada automaticamente pelo Eclipse e contém constantes para acessar os diversos recursos do projeto, que pode ser simplesmente uma imagem ou um arquivo XML que define alguma tela da aplicação.

3.1.3 - Definição da Interface Visual

O Android é bastante flexível em relação à criação da interface gráfica dos aplicativos já que possibilita que as telas sejam definidas em um arquivo de recurso no formato XML, dessa forma a lógica de negócios e a parte visual do aplicativo ficam separadas deixando o código mais legível.

Ao inserir um novo arquivo de recurso XML na pasta do projeto do aplicativo a classe R, descrita na Seção 3.1.2, é processada automaticamente pelo Eclipse para conter a nova referência ao recurso, nesse caso, a interface da tela.

Após a criação da tela, representada pela Activity, e da definição da estrutura da interface gráfica, representada pelo arquivo de recurso em formato XML, é

necessário que seja feita a ligação entre esses dois componentes. Isso é feito através da chamada ao método *setContentView()* da classe Activity que relaciona uma Activity ao seu respectivo arquivo em XML de definição de interface como demonstrado a seguir:

```
public class TelaDeAbertura extends Activity implements LocationListener
{
	public void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setContentView(R.layout.tela_de_abertura);
	}
}
```

3.1.4 - O arquivo AndroidManifest.xml

O arquivo AndroidManifest.xml é a base de uma aplicação Android. Ele é obrigatório e deve ficar na pasta raiz do projeto, contendo todas as configurações necessárias para executar a aplicação, como o nome do pacote utilizado, o nome das classes de cada Activity e várias outras configurações.

Podemos comparar o arquivo AndroidManifest.xml com o arquivo web.xml utilizado nas aplicações web em Java (LECHETA, 2009).

É obrigatório que cada Activity do projeto esteja declarada no arquivo AndroidManifest.xml, caso contrário não será possível utilizá-la.

3.1.5 - A classe android.content.Intent

A classe Intent é o coração do Android, uma Intent está presente em todos os aplicativos e representa uma mensagem da aplicação para o sistema operacional, solicitando que algo seja realizado, e representa um importante papel na arquitetura do Android para integrar diferentes aplicações.

A classe Intent representa uma ação que a aplicação deseja executar, a intenção sobre determinada ação é enviada ao sistema operacional como uma mensagem, chamada broadcast. Ao receber a mensagem, o sistema operacional

tomará as decisões necessárias, dependendo do conteúdo da mensagem, isto é, de sua intenção.

Uma Intent pode ser utilizada para:

- Enviar uma mensagem para o sistema operacional;
- Abrir uma nova tela da aplicação;
- Solicitar ao sistema operacional que ligue para determinado número de celular;
- Abrir o browser em um determinado endereço na internet;
- Exibir algum endereço no Google Maps;
- Executar algum processamento pesado em segundo plano;
- Enviar uma mensagem para outra aplicação.

3.1.6 - Google Maps

Provavelmente uma das funcionalidades do Android mais interessantes é a integração com o Google Maps e a possibilidade de desenvolver aplicações de localização com o GPS, com poucas linhas de código.

Para inserir um mapa na tela é usada a classe *com.google.android.maps. MapView,* Figura 3.2. O pacote de mapas não é padrão na plataforma do Android, e desta forma é necessário importa-lo no arquivo *AndroidManifest.xml*. O nome do pacote que precisa ser importado é *com.google.android.maps*.

Como internamente essa classe se comunica com os serviços do Google Maps, é necessário declarar a permissão “INTERNET” no arquivo *AndroidManifest.xml,* isso é necessário para que a permissão de acesso a internet seja solicitada ao usuário no momento da instalação do aplicativo. Outra permissão necessária para usar o GPS é a “ACCESS_FINE_LOCATION”.

Ao desenvolver uma aplicação para o Android com os recursos do Google Maps é necessário a obtenção de uma chave de autenticação fornecida pelo Google, para solicitar é preciso antes fornecer o código do certificado digital que é utilizado para assinar a aplicação. Por padrão, o Android cria um certificado digital de testes chamado “debug.keystore” quando um projeto do Android é compilado pelo Eclipse.

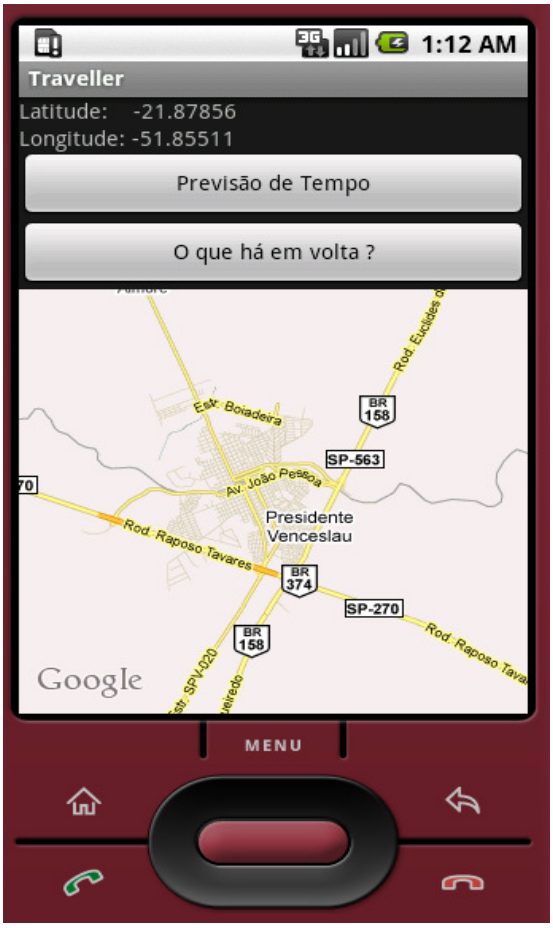

Figura 3.2 – Acitivity com mapa

3.1.7 - SQLite

SQLite é uma biblioteca que implementa um banco de dados relacional que não necessita de servidor. O código do SQLite é de domínio público e, portanto, livre para o uso com qualquer finalidade, comercial ou privada.

O SQLite é um programa de banco de dados SQL embutido, diferentemente da maioria dos outros bancos de dados SQL, o SQLite não tem um processo servidor separado, o programa lê e escreve diretamente em arquivos no disco. Um banco de dados SQL completo, com várias tabelas e índices, está contido em um único arquivo no disco.

O formato do arquivo de banco de dados é multiplataforma, é possível copiar livremente entre um banco de dados 32-bit e 64-bit.

O programa SQLite é uma biblioteca compacta, com todos os recursos habilitados, o tamanho da biblioteca pode ser inferior a 300KB, dependendo das configurações de otimização do compilador (SQLITE, 2011). Se os recursos opcionais são omitidos, o tamanho da biblioteca SQLite pode ser reduzido ainda mais. O SQLite também pode funcionar consumindo pouco espaço na memória, fazendo com que o SQLite seja uma escolha popular para banco de dados em dispositivos de memória limitada, como celulares, PDAs e *MP3 players*. Há um equilíbrio entre o uso de memória e velocidade.

Além dos recursos específicos oferecidos pelo Android para o desenvolvimento de aplicativos, o serviço baseado em localização proposto neste trabalho fez uso de diversos serviços na internet para mesclar informações e relacioná-las, criando assim um novo serviço. As próximas seções apresentam os principais *web service*s usados para o desenvolvimento da aplicação.

3.2 - Apontador

O Apontador é um serviço de busca local que oferece serviços de localização no Brasil através da internet. O serviço possui uma interface de programação aberta para desenvolvedores independentes o que possibilita a criação de aplicativos que misturam conteúdo de diversas fontes.

O *web service* Apontador é do tipo REST, todas as chamadas são feitas através de conexões HTTP no endereço “api.apontador.com.br” e prefixadas pelo identificador da versão do *web service*, a versão atual é a v1 (APONTADOR, 2011).

As chamadas da API recebem seus parâmetros através do método GET do HTTP, é possível escolher o formato do retorno entre XML (padrão) ou JSON[10]. O tipo de codificação é o UTF-8[11], e o método de autenticação é o HTTP Basic[12].

[10] JSON (JavaScript *Object Notation*) é um formato de troca de dados leve, legível por seres humanos e simples para serem lidos e gerados por máquinas (JSON, 2011).
[11] UTF-8 (8-bit *Unicode Transformation Format*) é um tipo de codificação de comprimento variável que pode representar qualquer caractere padrão do Unicode (UNICODE CONSORTIUM, 2011).
[12] *Basic Access Authentication* é um método de autenticação usado por um navegador de internet ou qualquer outro aplicativo de fornecer um usuário e senha para acessar um recurso que necessite de autenticação (THE APACHE SOFTWARE FOUNDATION, 2011).

O Apontador disponibiliza através de um *web service* REST, desde métodos para a pesquisa de locais em função de coordenadas geográficas, até a busca de ofertas de produtos em lojas de uma determinada região.

Dentre os vários métodos oferecidos, alguns se destacam e foram usados no aplicativo Traveller, dentre eles o “search/places/bypoint” e o “search/places/byaddress”.

O método “search/places/bypoint” retorna uma lista de locais em torno de um ponto representado por coordenadas geográficas e possui uma série de parâmetros, alguns deles obrigatórios, e outros opcionais, a lista de parâmetros é descrita no Quadro 3.1.

Quadro 3.1 – Parâmetros do método “search/places/bypoint”

Nome	Valor	Descrição
term	Opcional	Termo de busca (ex.: "loja").
lat	Obrigatório	Latitude do ponto, usando "." para decimal.
lng	Obrigatório	Longitude do ponto, usando "." para decimal.
radius_mt	Opcional, default=2000	Distância máxima em metros dos locais retornados a partir do ponto.
category_id	Opcional	Restringe a busca a uma categoria. O método “categories” pode ser usado para recuperar a lista dos IDs possíveis.
subcategory_id	Opcional	Restringe a busca a uma sub-categoria. O método “categories/CATEGORYID/categories” pode ser usado para recuperar a lista dos IDs possíveis.
sort_by	Opcional, default=relevance	Define como os resultados devem ser ordenados. Os valores possíveis atualmente são relevance (ordena pela relevância), rating (ordena pela nota média de avaliação) e distance (ordena pela distância do ponto).
order	Opcional, default=descending	Direção da ordenação. Pode ser ascending (ordem ascendente) ou descending (ordem

Nome	Valor	Descrição
		descendente).
rating	Opcional	Se informado, restringe os resultados àqueles com uma determinada nota média em avaliações (ou dentro de uma faixa de notas). As notas de avaliação variam de 1 a 5, mas nem todos os estabelecimentos têm nota.
page	Opcional, default=1	Número da página atual.
limit	Opcional, default=20	Número máximo de resultados por página.
user_id	Opcional	Se informado, limita a busca a pontos criados pelo usuário referenciado.
type	Opcional, default=xml	Formato de retorno. Pode ser xml,json ou kml.

Adaptado de: < http://api.apontador.com.br/pt/metodo_search_places_bypoint.html>

Quando uma requisição é feita ao *web service* Apontador via HTTP, o serviço processa o pedido consultando sua base de dados e posteriormente a resposta é enviada pelo *web service* em um arquivo no formato XML como demonstrado no trecho de código abaixo, cuja categoria de estabelecimentos passada como parâmetro na requisição é a de advogados, já as coordenadas geográficas passadas como parâmetro foram as coordenadas da cidade de Presidente Venceslau:

```
<?xml version="1.0" encoding="UTF-8"?>
<search>
	<result_count>2</result_count>
	<current_page>1</current_page>
	<places>
		<place>
			<id>J8Y82MNA</id>
			<name>SILVEIRA - Advogados</name>
			<average_rating>0</average_rating>
			<review_count>0</review_count>
			<thumbs>
				<total>0</total>
				<up>0</up>
			</thumbs>
			<category>
				<id>47</id>
				<name>CONSULTORIA</name>
				<subcategory>
					<id>66</id>
					<name>Advogados - Geral</name>
				</subcategory>
```

```
            </category>
            <address>
                <street>R Dq. De Caxias</street>
                <number>851</number>
                <district>Centro</district>
                <zipcode>19400000</zipcode>
                <complement></complement>
                <city>
                    <country>BR</country>
                    <state>SP</state>
                    <name>Presidente Venceslau</name>
                </city>
            </address>
            <phone>
                <country></country>
                <area></area>
                <number>32715300</number>
            </phone>
            <point>
                <lat>-21.87313</lat>
                <lng>-51.85074</lng>
            </point>
        </place>
        <place>
            <id>224C964Q</id>
            <name>Escritorio Contabilidade Objetivo</name>
            <average_rating>0</average_rating>
            <review_count>0</review_count>
            <thumbs>
                <total>0</total>
                <up>0</up>
            </thumbs>
            <category>
                <id>47</id>
                <name>CONSULTORIA</name>
                <subcategory>
                    <id>24813</id>
                    <name>Contabilidade - EscritÃ³rios</name>
                </subcategory>
            </category>
            <address>
                <street>Rua Carlos Gomes</street>
                <number>337</number>
                <district>Centro</district>
                <zipcode></zipcode>
                <complement></complement>
                <city>
                    <country>BR</country>
                    <state>SP</state>
                    <name>Presidente Venceslau</name>
                </city>
            </address>
            <phone>
                <country>55</country>
                <area>18</area>
                <number>32713983</number>
            </phone>
            <point>
                <lat>-21.87743</lat>
                <lng>-51.84685</lng>
            </point>
        </place>
    </places>
</search>
```


O método ‘‘search/places/byaddress*”* recupera uma lista de locais próximos a um endereço, e assim como o método ”search/places/bypoint”, possui uma série de parâmetros, alguns deles obrigatórios, e outros opcionais.

A resposta também é devolvida pelo *web service* em um arquivo no formato XML, o quadro 3.2 contém os parâmetros do método "search/places/byaddress*".*

Quadro 3.2 – Parâmetros do método "search/places/byaddress"

Nome	Valor	Descrição
term	Opcional	Termo de busca (ex.: loja).
country	Opcional, default=BR	Código de duas letras do país, no padrão ISO-3166-1 alpha 2. Atualmente apenas BR é suportado.
state	Obrigatório	Unidade Federativa (ex.: SP para o Estado de São Paulo).
city	Obrigatório	Nome da cidade (ex.: Campinas).
street	Opcional	Nome da rua.
cross_street	Opcional	Nome da rua que cruza aquela informada no parâmetro "street". Estes dois parâmetros, street e cross_street, devem ser utilizados em conjunto.
number	Opcional	Número na rua.
district	Opcional	Bairro.
radius_mt	Opcional, default=2000	Distância máxima (em metros) dos locais retornados a partir do endereço.
category_id	Opcional	Restringe a busca a uma categoria. O método "categories" pode ser usado para recuperar a lista dos IDs possíveis.
subcategory_id	Opcional	Restringe a busca a uma sub-categoria. O método "categories/CATEGORYID/categories" pode ser usado para recuperar a lista dos IDs possíveis.
sort_by	Opcional, default=relevance	Define como os resultados devem ser ordenados. Os valores possíveis atualmente são relevance (ordena pela relevância), rating

Nome	Valor	Descrição
		(ordena pela nota média de avaliação) e distance (ordena pela distância do ponto correspondente ao endereço)
order	Opcional, default=descending	Direção da ordenação. Pode ser ascending (ordem ascendente) ou descending (ordem descendente).
rating	Opcional	Se informado, restringe os resultados àqueles com uma determinada nota média em avaliações (ou dentro de uma faixa de notas).
page	Opcional, default=1	Número da página atual (começando no 1).
limit	Opcional, default=20	Número máximo de resultados por página.
user_id	Opcional	Se informado, limita a busca a pontos criados pelo usuário referenciado.
type	Opcional, default=xml	Formato de retorno. Pode ser xml, json, jsonp ou kml.

Adaptado de: <http://api.apontador.com.br/pt/metodo_search_places_byaddress.html>

3.3 – O Serviço Google Weather

O Google Weather é um serviço do Google que disponibiliza informações sobre as condições climáticas e a previsão de tempo para os próximos quatro dias de uma determinada localização. O serviço é aberto e não requer autenticação.

Trata-se de um *web service* do tipo REST, onde as mensagens são encapsuladas diretamente no protocolo HTTP (ANDDEV.ORG, 2007). As requisições são feitas diretamente no endereço “www.google.com/ig/api” e os parâmetros da localização são passados através do método GET do HTTP. A resposta é recebida através de um arquivo do tipo XML como demonstrado a seguir, cuja solicitação passou como parâmetros a cidade de Presidente Prudente:

```
<?xml version="1.0" ?>
<xml_api_reply version="1">
- <weather module_id="0" tab_id="0" mobile_row="0" mobile_zipped="1" row="0" section="0">
- <forecast_information>
<city data="Presidente Prudente, São Paulo" />
        <postal_code data="Presidente Prudente, Brazil" />
        <latitude_e6 data="" />
        <longitude_e6 data="" />
        <forecast_date data="2011-08-21" />
        <current_date_time data="2011-08-21 21:00:00 +0000" />
        <unit_system data="SI" />
</forecast_information>
- <current_conditions>
        <condition data="Nublado" />
        <temp_f data="63" />
        <temp_c data="17" />
        <humidity data="Umidade: 77%" />
        <icon data="/ig/images/weather/mostly_cloudy.gif" />
        <wind_condition data="Vento: SE a 13 km/h" />
    </current_conditions>
- <forecast_conditions>
        <day_of_week data="dom" />
        <low data="12" />
        <high data="23" />
        <icon data="/ig/images/weather/mostly_sunny.gif" />
        <condition data="Ensolarado na maioria" />
        </forecast_conditions>
- <forecast_conditions>
        <day_of_week data="seg" />
        <low data="13" />
        <high data="25" />
        <icon data="/ig/images/weather/mostly_sunny.gif" />
        <condition data="Ensolarado na maioria" />
    </forecast_conditions>
- <forecast_conditions>
        <day_of_week data="ter" />
        <low data="15" />
        <high data="24" />
        <icon data="/ig/images/weather/chance_of_rain.gif" />
        <condition data="Possibilidade de chuva" />
    </forecast_conditions>
- <forecast_conditions>
        <day_of_week data="qua" />
        <low data="17" />
        <high data="30" />
        <icon data="/ig/images/weather/mostly_sunny.gif" />
        <condition data="Ensolarado na maioria" />
    </forecast_conditions>
</weather>
</xml_api_reply>
```

Cada um dos elementos tem um significado como descrito abaixo:

- < forecast information> - armazena as informações da localidade para a qual foi solicitada a previsão do tempo;
- < city data= /> - nome da cidade e o estado;
- < postal_code data= /> - uma estrutura de exibição como "cidade, estado, país";
- < forecast_date data= /> - data da solicitação da previsão do tempo (data em que foi gerado o arquivo XML);
- < current_date_time data= /> - data e hora corrente;

- < current_conditions> - armazena as informações da previsão do tempo para a data atual;
- < condition data= /> - condição do tempo (nublado, possibilidade de chuva, etc);
- < temp_f data= /> - temperatura em graus Farenheit;
- < temp_c data= /> - temperatura em graus Celsius;
- < humidity data= /> - Umidade do ar;
- < icon data= /> - imagem que representa a condição do tempo;
- < wind_condition data= /> - velocidade do vento;
- < forecast_conditions> - armazena as informações da previsão do tempo para os próximos 4 dias;
- < day_of_week data= /> - dia da semana;
- < low data= /> - temperatura mínima;
- < high data= /> - temperatura máxima;
- < icon data= /> - imagem que representa a condição do tempo;
- < condition data= /> - condição do tempo (nublado, possibilidade de chuva, ensolarado e chuva).

3.4 – O Portal GeoNames

O banco de dados geográficos GeoNames está disponível para consulta gratuita na Internet. Ele contém mais de 10 milhões de nomes geográficos e consiste de 7,5 milhões de recursos. Todos os recursos são classificados em uma das nove classes e categorizados em um dos 645 códigos que descrevem características de determinadas classes de lugares.

Os dados são acessíveis gratuitamente através de *web services*, como o que será abordado na Seção 3.4.1. Atualmente o GeoNames responde por mais de 20 milhões de pedidos de serviços na web por dia.(GEONAMES, 2011).

O GeoNames integra dados geográficos, como nomes de lugares em várias línguas, população, altitude e outras informações de várias fontes. Os usuários

podem editar manualmente, corrigir e adicionar novos nomes usando uma interface de usuário disponibilizada pelo serviço.

3.4.1 - O método findNearbyPlaceName

O método “findNearbyPlaceName” pertencente ao portal GeoNames oferece o serviço de geocodificação. Trata-se de um web service do tipo REST que procura por um local povoado em função de coordenadas geográficas passadas como parâmetro. As requisições são feitas no endereço “ws.geonames.org/findNearbyPlaceName” e os parâmetros latitude e longitude são passados através do método GET do HTTP. A resposta da requisição é enviada pelo web service em formato XML como demonstrado abaixo:

```
<geonames>
        <geoname>
                <toponymName>Presidente Venceslau</toponymName>
                <name>Presidente Venceslau</name>
                <lat>-21.87611</lat>
                <lng>51.84389</lng>
                <geonameId>3452320</geonameId>
                <countryCode>BR</countryCode>
                <countryName>Brazil</countryName>
                <fcl>P</fcl>
                <fcode>PPL</fcode>
                <distance>0.927</distance>
        </geoname>
</geonames>
```

4 - O SERVIÇO BASEADO EM LOCALIZAÇÃO TRAVELLER

O aplicativo idealizado neste trabalho, denominado Traveller, tem por objetivo auxiliar usuários de celulares equipados com GPS em locais desconhecidos através do fornecimento de informações climáticas, de localização do usuário e informações de estabelecimentos em áreas urbanas, além de projetar no mapa a localização atual do dispositivo e de cidades de interesse sendo complementado com funções relativas à localização, como o cálculo de distâncias entre coordenadas geográficas.

O sistema operacional a que se destina o aplicativo é o Android e a linguagem de programação usada no desenvolvimento é a Java fazendo uso das ferramentas de programação do Android para o Eclipse.

As coordenadas geográficas são obtidas através do GPS do dispositivo móvel e projetadas em mapas provenientes do Google Maps. As informações de localização adicionais, de geocodificação e informações climáticas são obtidas por meio de *web services* do tipo REST ou por meio das funções presentes nas bibliotecas do Google Maps para o Android. A Figura 4.1 ilustra as interações entre o aplicativo Traveller e os *web services* usados por ele.

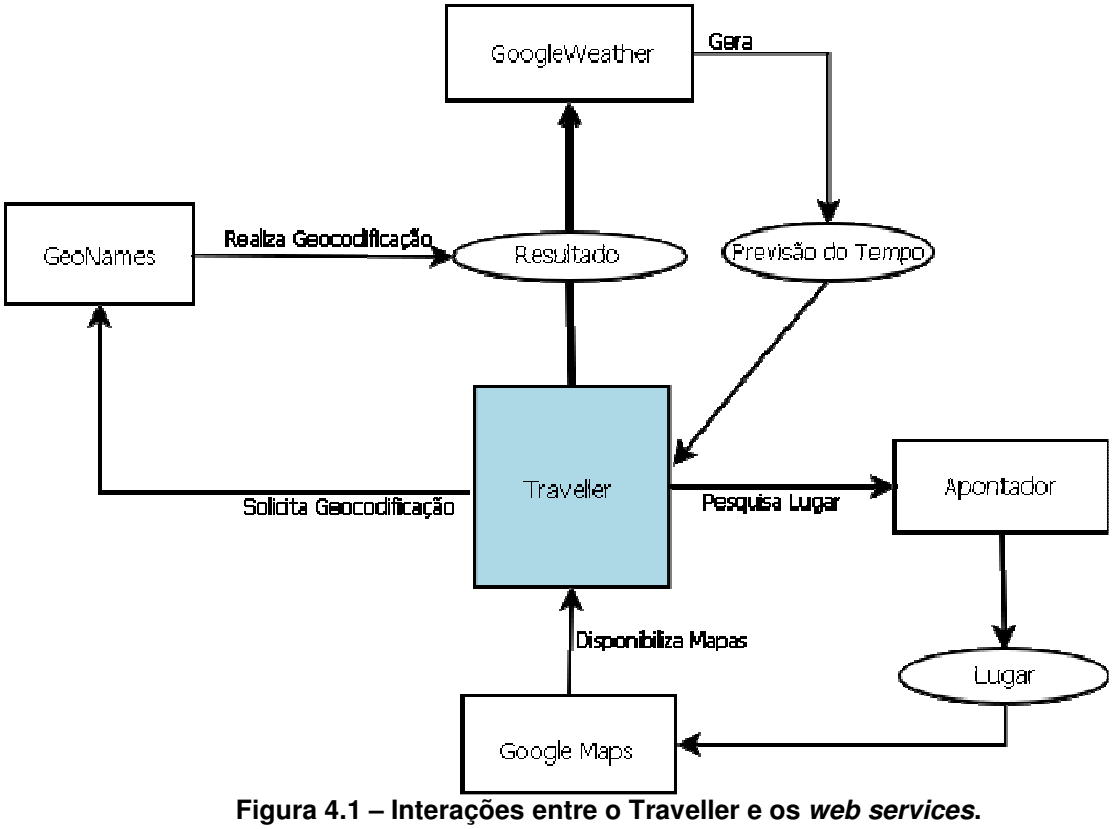

Figura 4.1 – Interações entre o Traveller e os *web services*.

O Serviço Baseado em Localização Traveller oferece quatro principais serviços ao usuário: localização, informações climáticas, pesquisa de estabelecimentos e catálogo de lugares.

O serviço de localização se dá através da obtenção das coordenadas geográficas do dispositivo móvel por meio do GPS, em princípio o programa tenta atualizar a localização atual do dispositivo, caso não seja possível é mostrada a última localização lida pelo GPS no formato decimal.

Ao clicar no botão mapa, presente na tela de abertura do programa, ilustrado pela Figura 4.2, o usuário observa a localização do dispositivo no mapa.

Através do serviço de informações climáticas, é possível obter as condições climáticas atuais e a previsão para os próximos três dias com base na localização atual do dispositivo ou em função do nome de um determinado local ou cidade.

Já o serviço de pesquisa de estabelecimentos permite que o usuário pesquise por estabelecimentos pertencentes a determinadas categorias pré-estabelecidas, contidos em um determinado raio a partir da localização atual do dispositivo, descobrir o seu endereço e posicioná-los no mapa.

Por meio do serviço de catálogo de lugares é possível fazer uma busca por uma determinada cidade ou lugar, e a partir daí utilizando geocodificação, descobrir suas coordenadas geográficas, obter informações climáticas, calcular sua distância até a localização atual, posicioná-la no mapa e posteriormente salvá-la em uma base de dados presente no aplicativo. A função de catálogo permite ainda que pontos turísticos visitados em uma determinada cidade, localizada e cadastrada pelo usuário, seja catalogada e armazenada juntamente com as respectivas informações.

4.1 – O serviço de localização e posicionamento

Através do serviço de localização o Traveller é capaz de obter, através do GPS presente no dispositivo móvel em questão, a localização do usuário representada pelas coordenadas geográficas e posicioná-lo no mapa.

O mapa utilizado é proveniente do Google Maps, a sua declaração é feita em um arquivo de recurso no formato XML responsável por gerenciar a interface gráfica

do programa e posteriormente adicionada a uma Activity que representa a tela do aplicativo responsável por mostrar o mapa.

Ao iniciar o aplicativo, é exibida a tela de abertura que contém os botões de navegação para as principais funções do Traveller. Na parte superior da tela estão presentes as coordenada geográficas atuais do dispositivo em formato decimal. Para determinar a posição do usuário, em princípio, o programa solicita ao GPS as coordenadas atuais do dispositivo, caso não seja possível são apresentadas na tela as últimas coordenadas válidas lidas pelo GPS.

A partir desse momento, o programa inicia um ciclo de solicitação das coordenadas ao GPS que tem como objetivo a contínua atualização do posicionamento para assegurar que as coordenadas apresentadas na tela são atuais mesmo que o usuário se mova, o trecho de código abaixo, presente na classe “TelaDeAbertura.java”, é responsável pela atualização da posição:

```
((LocationManager)getSystemService(TelaDeAbertura.LOCATION_SERVICE)).requestLocationUpdates
 (LocationManager.GPS_PROVIDER, 0, 0, this);
```

No canto superior direito da tela de abertura se encontra o botão “Mapa”, ilustrado na Figura 4.2, que tem como finalidade exibir o mapa com centro nas coordenadas geográficas atuais do usuário.

Figura 4.2 – Tela de Abertura

4.2 – O Serviço de Informações Climáticas

O serviço de informações climáticas presente no Traveller tem como objetivo obter as condições atuais de temperatura, umidade, vento e a previsão do tempo para os próximos três dias tendo a localização atual do dispositivo como referência.

No momento em que a solicitação é feita pelo usuário ao clicar no botão "Previsão de Tempo", ilustrado na Figura 4.3, o Traveller solicita ao GPS, através da classe "Location" a localização atual do aparelho. De posse dessas informações, o programa envia uma requisição de geocodificação reversa, através da classe "GeoCodificaçãoReversa", ao *web service* disponível no portal "GeoNames" a fim de descobrir o nome da localidade onde o usuário está, passando como parâmetro as coordenadas geográficas retornadas pelo GPS. Isso é necessário, já que o serviço de previsão de tempo, denominado GoogleWeather, recebe como parâmetro de localização, um nome de cidade e país, e não coordenadas geográficas.

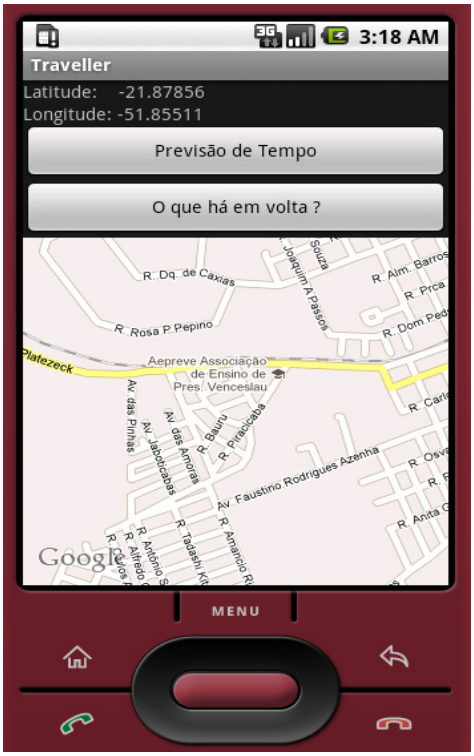

Figura 4.3 – Localização Atual

Posteriormente é feita a requisição de previsão de tempo através da classe "PrevisãoDeTempo" ao *web service* GoogleWeather, que por sua vez devolve a

resposta em um arquivo em formato XML, a partir do qual serão extraídas as informações climáticas apresentadas ao usuário como na Figura 4.4:

Figura 4.4 – Previsão de Tempo

4.3 – O Serviço de Pesquisa de Estabelecimentos

Em Serviços Baseados em Localização o posicionamento e a exibição de mapas são essenciais, porém tais serviços estão disponíveis em dispositivos GPS tradicionais que sequer se conectam a Internet. A pesquisa de estabelecimentos compreende o serviço mais importante aplicativo proposto neste trabalho. Através dele, o aplicativo é capaz de descobrir a localização do usuário e procurar por estabelecimentos de diversas categorias em um determinado raio tendo como referência a localização do dispositivo.

O serviço responsável por fornecer as informações necessárias é um *web service* do tipo REST denominado Apontador. O Apontador é um serviço de busca local que oferece serviços de localização no Brasil através da internet.

No momento em que o usuário solicita a pesquisa por lugares, o aplicativo obtém a localização atual do dispositivo através do GPS, caso não seja possível

descobrir a posição atual, o programa adotará como referência para a pesquisa a última localização válida lida pelo GPS.

Posteriormente, através do método “carregaCategorias”, presente na classe “Categoria”, o aplicativo requisita ao *web service* Apontador, a lista de categorias de estabelecimentos disponíveis para busca. Este por sua vez, responde através de um arquivo em formato XML, como o descrito abaixo, contendo as categorias disponíveis:

```
<?xml version="1.0" encoding="UTF-8"?>
<categories>
      <category>
            <id>Numero Identificador da Categoria</id>
            <name>Nome da Categoria</name>
      </category>
      <category>
            <id>Numero Identificador da Categoria</id>
            <name>Nome da Categoria</name>
      </category>
</categories>
```

O *web service* Apontador classifica os estabelecimentos presentes em sua base de dados em categorias. Através do botão Categoria, presente na tela de busca por lugares, ilustrada na Figura 4.5, o usuário tem a oportunidade de pesquisar estabelecimentos dentre as diversas categorias além de poder delimitar um raio de distância máxima entre a posição atual e os estabelecimentos encontrados.

Figura 4.5 – Tela de Busca de Lugares

Após serem escolhidos a categoria de lugares e a distância máxima, o aplicativo Traveller, através do método “buscaLugar” presente na classe “Lugar”, requisita ao *web service* Apontador uma lista de lugares que se encaixa nos parâmetros passados pelo usuário tendo como referência as coordenadas geográficas obtidas pelo GPS.

A Figura 4.6 ilustra as principais classes dentre as que são responsáveis pela função de pesquisa de estabelecimentos.

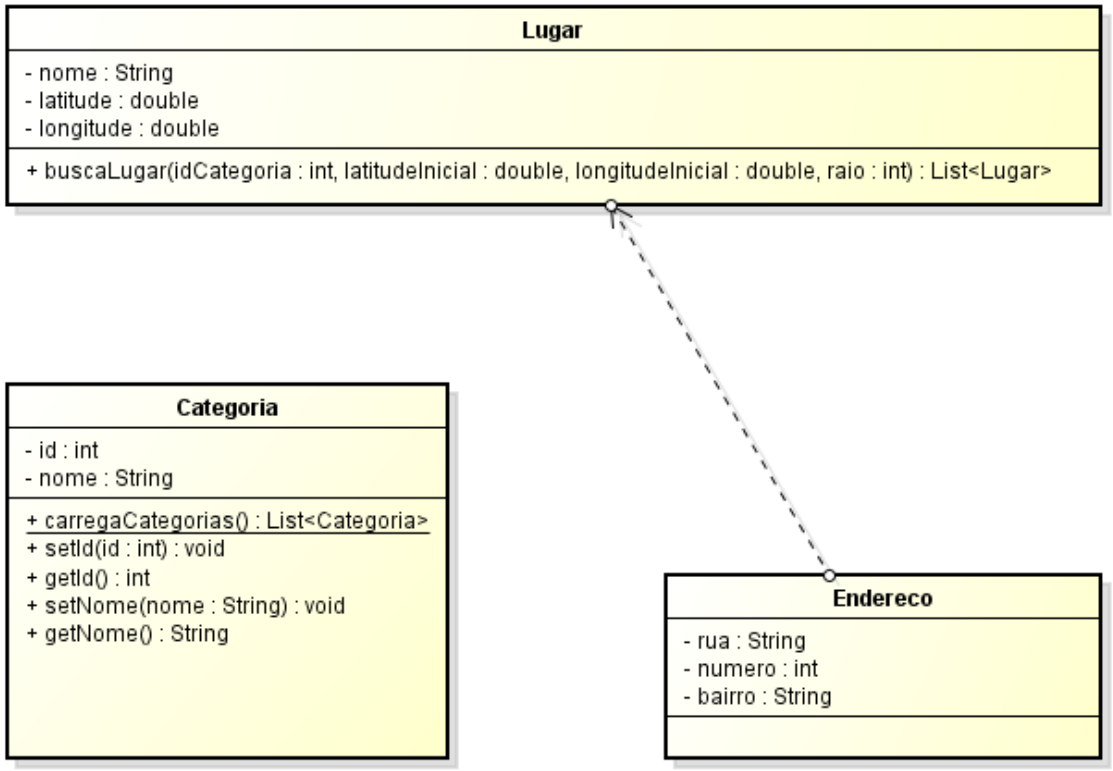

Figura 4.6 – Classes responsáveis pela busca de lugares.

O método “carregaCategoria” pertencente a classe “Categoria” é responsável por solicitar ao *web service* Apontador a lista de categorias de lugares disponíveis para busca, assim como extrair as informações do arquivo de resposta em formato XML. Já o método “buscaLugar”, pertencente a classe “Lugar”, por sua vez, tem como objetivo procurar por lugares de acordo com os parâmetros especificados, bem como a extração das informações do arquivo de resposta enviado pelo *web service* em formato XML.

4.4 – O Serviço Catálogo de Lugares

O serviço de catálogo de lugares possibilita ao usuário guardar as informações de lugares visitados em memória permanente. O módulo de catálogo é subdivido em três partes: busca e armazenamento de cidades, catálogo de pontos visitados em uma cidade e galeria de pontos visitados.

4.4.1 - Busca e Armazenamento de Cidades

Através do serviço de busca por cidades, ilustrado na figura 4.7, é possível, por meio de geocodificação, encontrar cidades e posicioná-las no mapa, além de obter informações climáticas, calcular a distância entre a localização atual e o destino pretendido, além de guardar as informações em memória persistente.

Figura 4.7 – Busca de Cidades

4.4.1.1 – Cálculo de Distâncias

Por meio do GPS integrado no dispositivo móvel, o aplicativo Traveller obtém as coordenadas geográficas da localização atual do dispositivo. Através de geocodificação, oferecida pela biblioteca do Google Maps para o Android, o programa é capaz de descobrir as coordenadas geográficas do destino pretendido pelo usuário, e partir daí é possível calcular a distância entre a posição atual e o destino.

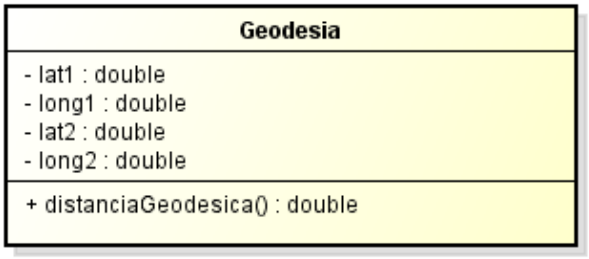

Figura 4.8 – A classe geodesia.

O método “distanciaGeodesica”, presente na classe “Geodesia”, ilustrada na Figura 4.8, é responsável pelo cálculo das distâncias entre duas coordenadas geográficas usando a lei dos cossenos , da trigonometria esférica. Os atributos lat1 e long1, do método “distanciaGeodesica” transcrito abaixo, representam a latitude e a longitude da localização atual respectivamente, já os parâmetros lat2 e long2 representam a latitude e a longitude do destino respectivamente.

```
public static double distanciaGeodesica(double lat1, double long1, double lat2, double long2)
{
        double arcoA;  /*Diferença entre as longitudes*/
        double arcoB;  /*Arco que une o ponto 2 ao pólo norte da terra*/
        double arcoC;  /*Arco que une o ponto 1 ao pólo norte da terra*/
        double cosArco;
        double arco;

        arcoA = (long2 - long1)*Math.PI/180;
        arcoB = (90 - lat2)*Math.PI/180;
        arcoC = (90 - lat1)*Math.PI/180;

        /* cos(a)= cos(b).cos(c)+sen(b).sen(c).cos(D)*/
        cosArco = Math.cos(arcoB)*Math.cos(arcoC) +
             Math.sin(arcoB)*Math.sin(arcoC)*Math.cos(arcoA);

        arco = Math.acos(cosArco);
        return arco*6371  ; /* Raio da Terra = 6.371 Km.*/
}
```

4.4.2 – Catálogo de Pontos Visitados em uma Cidade

A função de catálogo de pontos, ilustrada pela Figura 4.9, é responsável por salvar em memória persistente, locais visitados em determinadas cidades assim como informações a respeito do clima na data da visita, coordenadas geográficas e detalhes adicionais inseridos pelo próprio usuário para posterior consulta.

Figura 4.9 – Catálogo de Pontos

O Android conta com um Sistema Gerenciador de Banco de Dados integrado denominado SQLite. Através das classes “Localidade” e “PontoTuristico”, pertencentes ao pacote persistência, o aplicativo é capaz de armazenar na base de dados integrada as informações referentes aos locais visitados.

A Figura 4.10 ilustra as principais classes responsáveis pela função de catálogo de pontos visitados.

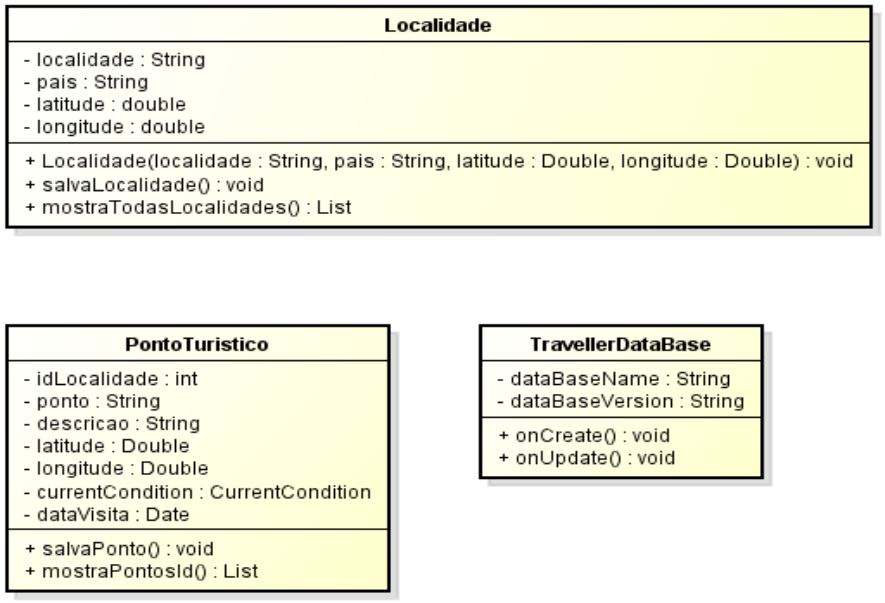

Figura 4.10 – Classes de Persistência

4.4.3 - Galeria de Pontos Visitados

Finalmente, por meio da função galeria de pontos visitados, ilustrada pela Figura 4.11, é possível consultar os pontos visitados em determinada cidade e visualizar em tela as informações referentes a cada visita além de visualizar a localização do ponto no mapa.

Figura 4.11 – Lugares Visitados

As informações referentes aos pontos visitados e as respectivas cidades são armazenados no banco de dados integrado do Android. O pacote “persistência” contém a classe “TravellerDatabase” responsável pela criação das tabelas da base de dados do aplicativo.

A tabela “Localidades”, ilustrada na Figura 4.12, armazena as informações de uma determinada cidade, uma cidade contém um ou mais pontos visitados que são representados armazenados na tabela “Pontos”.

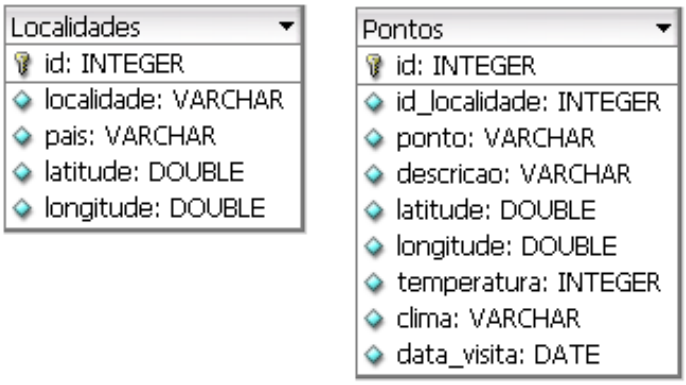

Figura 4.12 – Tabelas da Base de Dados

5 – TESTES E CONCLUSÃO

Os testes efetuados em um dispositivo móvel da marca Samsung, modelo i5700 Spica Galaxy, equipado com o sistema operacional Android na versão 1.5, kernel Linux 2.6.27, demonstraram que a plataforma Android é eficiente para o desenvolvimento de aplicativos e serviços baseados em localização, já que conta com a biblioteca do Google Maps para o Android (versão 1.5).

A biblioteca do Google Maps disponibiliza as classes e métodos necessários para a manipulação de mapas e informações de localização, como as funções de geocodificação e geocodificação reversa. Por se tratar de uma plataforma aberta (*open source*) oferece flexibilidade ao desenvolvimento de aplicativos e conta com uma vasta comunidade de desenvolvedores. O que possibilita o intercâmbio de informações, porém durante os testes, foram constatados problemas com a classe "GeoCoder" pertencente ao pacote "android.location". O serviço de geocodificação reversa oferecido pela interface de programação deixou a desejar, demonstrando instabilidade e indisponibilidade. O serviço ora retornava um resultado, ora não retornava resultado algum para a mesma requisição. O problema foi corrigido usando o serviço de geocodificação reversa oferecido pelo portal GeoNames, que demonstrou desempenho e disponibilidade, já que não deixou de responder por nenhuma das solicitações, além de ter respondido de forma quase que instantânea a todas elas.

Um dos principais serviços oferecidos pelo aplicativo Traveller é o serviço de informações sobre o clima, os dados são provenientes do *web service* GoogleWeather, que demonstrou disponibilidade e desempenho assim como o portal GeoNames. Ao comparar informações climáticas em tempo real, oferecidas pelo serviço GoogleWeather, com as informações de outros portais, foram encontradas discrepâncias nas informações. Por exemplo, no dia 22 de agosto de 2011 as 23h26min, o aplicativo Traveller marcava temperaturas de 19°C, umidade do ar em 83% com ventos na direção nordeste a 10KM/h, para a cidade do Rio de Janeiro. Já o portal de meteorologia Tempo Agora[13] marcava temperaturas de 18°C, vento leste a 11KM/h e umidade do ar em 100%, enquanto que o portal Jornal do

[13] Tempo Agora (www.tempoagora.com.br) é um portal de fornecimento de informações meteorológicas.

Tempo[14], indicava temperaturas a 18°C, umidade do ar em 88% e ventos a 12KM/h na direção leste. Julgar a eficácia e confiabilidade das informações referentes à previsão de tempo é tarefa que foge do escopo deste trabalho.

Durante as fases de desenvolvimento e testes, o sistema operacional Android e sua máquina virtual Dalvik mostraram-se eficientes no tocante ao gerenciamento dos recursos de hardware, já que conta com o kernel do Linux para funções básicas como escalonamento de processos. Em nenhum momento erros no aplicativo causaram a paralisação total do dispositivo móvel.

A base de dados do aplicativo Traveller é pequena em dimensão e complexidade, porém, o sistema gerenciador de bancos de dados relacional integrado, SQLite, mostrou-se eficiente para execução em dispositivos móveis, já que utiliza poucos recursos de *hardware* se comparado aos gerenciadores tradicionais. A ressalva fica por conta do SQLite em sua versão para Android 1.5 não garantir integridade referencial aos dados, já que não suporta chaves estrangeiras nas tabelas do banco de dados.

O GPS integrado no dispositivo móvel usado nos testes demonstrou eficácia e precisão do ponto de vista do usuário. Testes mostraram que o GPS foi capaz de distinguir estar em uma rua ou no interior das quadras, por exemplo.

O *web service* Apontador demonstrou desempenho e disponibilidade durante os testes, não houve momentos em que o serviço esteve inoperante ou demonstrou lentidão nas respostas, a ressalva fica por conta da necessidade de conexões de alta velocidade, para receber as informações de forma que não comprometa a usabilidade do aplicativo. O dispositivo móvel usado nos testes usa o sistema EDGE (*Enhanced Dara rates for GSM Evolution* - taxas de dados aperfeiçoadas para evolução do GSM) para transmissão de dados, e em alguns momentos causou lentidão a ponto de comprometer a usabilidade.

O serviço de busca por estabelecimentos oferecido pelo aplicativo Traveller é, sem dúvidas, a principal funcionalidade do programa. Ele representa o dinamismo, flexibilidade e pertinência dos serviços baseados em localização. Através dele foi possível demonstrar como um dispositivo móvel equipado com GPS e acesso a internet pôde se beneficiar de informações de localização que não tenham simplesmente o objetivo de traçar rotas ou indicar caminhos e direções.

[14] Jornal do Tempo (http://jornaldotempo.uol.com.br/previsaodotempo.html/brasil/RiodeJaneiro-RJ/) é um portal de fornecimento de informações meteorológicas.

O *web service* Apontador, oferece ainda, outras funcionalidades a respeito dos estabelecimentos cadastrados em sua base de dados, como a consulta de avaliações efetuadas por outros usuários ou a visualização de fotos dos estabelecimentos. O cruzamento de informações de diversos *web services*, aliado a portabilidade dos dispositivos móveis com acesso a internet e GPS, demonstrou ser uma alternativa eficiente para a implementação de aplicativos Mashup, já que são infinitas as possibilidades de relacionamento entre informações provenientes de *web services* na internet.

O web service Apontador disponibiliza a edição das informações a respeito dos estabelecimentos através da atribuição de notas pelos usuários, o aplicativo Traveller só provê a busca por estabelecimentos.

O cálculo e traçado de rotas, função que não está disponível para a biblioteca do Google Maps em sua versão 1.5, a consulta e atribuição de notas aos estabelecimentos pelos usuários, bem como a visualização das fotos dos estabelecimentos, são funções que podem ser projetadas e implementadas em versões futuras do aplicativo.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID DEVELOPERS. **The Developer´s Guide**. Disponível em: http://developer.android.com/guide/index.html. Acessado em 07 de julho de 2011.

APONTADOR. **Apontador API**. Disponível em: http://api.apontador.com.br/pt/ Acessado em 20 de abril de 2011.

ANDDEV.ORG. **Android Weather Forecast – Google Weather API – Description**. Disponível em: http://www.anddev.org/android_weather_forecast_-_google_weather_api_-_description-t337.html. Acessado em 10 de fevereiro de 2011.

FIELDING, Roy Thomas. **Architectural styles and the design of network–based softwares architectures**. Doctoral Dissertation, University of California, Irvine, 2000.

GEONAMES. **GeoNames WebServices Overview**. Disponível em http://www.geonames.org/export/ws-overview.html. Acessado em 24 de fevereiro de 2011.

GPS.GOV.**Space Segment**. Disponível em: http://www.gps.gov/systems/gps/space/. Acessado em 23 de março de 2011.

HURWITZ, J.; BLOOR, R.; KAUFMAN, M.; HALPER, F. **Service oriented architecture for dummies**. 2. ed. IBM Limited Edition. Wiley, 2009.

IBM. **Mashups: The New breed of Web app**. Disponível em: http://www.ibm.com/developerworks/xml/library/x_mashups/index.html. Acessado em 28 de julho de 2011.

JSON, **Introducing JSON**. Disponível em: http://www.json.org/. Acessado em 21 de agosto de 2011.

KUPPER, A. **Location-based services:** fundamentals and operation. Wiley, 2005.

LECHETA R. **Google Android:** aprenda a criar aplicações para dispositivos móveis com o Android SDK. 1 Ed. São Paulo: Novatec Editora, 2009.

MONICO, J. F. G. **Posicionamento pelo NAVSTAR-GPS:** descrição, fundamentos e aplicações. São Paulo: Editora UNESP, 2000

SCHILLER, J.; VOISARD, A. **Location-based services**. São Francisco: Elsevier, 2004.

SQLITE. **Documentation**. Disponível em: http://www.sqlite.org/docs.html. Acessado em 25 de fevereiro de 2011.

THE APACHE SOFTWARE FOUNDATION, **Authentication, Authorization, and Access Control**. Disponível em: http://httpd.apache.org/docs/1.3/howto/auth.html#intro. Acessado em: 21 de agosto de 2011.

UNICODE CONSORTIUM, **The Unicode Standard**. Mountain View, 2011. Disponível em: http://www.unicode.org/versions/Unicode6.0.0/ch02.pdf#G11165. Acessado em 21 de agosto de 2011.

W3C TECHNICAL ARCHITECTURE GROUP, **Web Services Architecture**. Disponível em: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. Acessado em 20 de abril de 2011.

W3C TECHNICAL ARCHITECTURE GROUP, **Naming and Addressing: URIs, URLs**. Disponível em: http://www.w3.org/Addressing/#background. Acessado em 21 de agosto de 2011.

XAVIER, Otávio C. **Desenvolvimento de Web Services em Java com Restlet**. Mestrado em Ciência da Computação, Universidade Federal de Goiás, 2010.