FACULDADES INTEGRADAS DE CARATINGA

FACULDADE DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UM APLICATIVO DE SISTEMA DE INFORMAÇÃO GEOGRÁFICA USANDO ANDROID

VICTOR ALEXANDRE PEREIRA DE CARVALHO

CARATINGA
2011

Victor Alexandre Pereira de Carvalho

DESENVOLVIMENTO DE UM APLICATIVO DE SISTEMA DE INFORMAÇÃO GEOGRÁFICA USANDO ANDROID

Monografia apresentada ao Curso de Ciência da Computação das Faculdades Integradas de Caratinga como requisito parcial para obtenção do título de Bacharel em Ciência da Computação orientado pelo Professor Glauber Luiz Costa.

Caratinga
2011

Victor Alexandre Pereira de Carvalho

DESENVOLVIMENTO DE UM APLICATIVO DE SISTEMA DE INFORMAÇÃO GEOGRÁFICA USANDO ANDROID

Monografia apresentada ao Curso de Ciência da Computação das Faculdades Integradas de Caratinga como requisito parcial para obtenção do título de Bacharel em Ciência da Computação orientado pelo Professor Glauber Luiz Costa.

Prof. Glauber Luiz Costa
Faculdades Integradas de Caratinga
_____
Prof. Jacson Rodrigues Correia da Silva
Faculdades Integradas de Caratinga
_____
Prof. Fabricia Pires Tiola
Faculdades Integradas de Caratinga
_____

Caratinga, 07/12/2011

AGRADECIMENTOS

Agradeço muito a Deus que me deu força e sabedoria para o desenvolvimento deste trabalho.

Gostaria de agradecer a meus amigos e ao Portal Android que me ajudaram e contribuíram para realização deste trabalho. Ao professor Glauber Luiz Costa pela orientação, pela grande ajuda nas partes mais difíceis e pelas várias idéias que me ajudaram a finalizar o trabalho.

Agradeço muito a minha família, que me deu muito apoio. Aos meus pais que ajudaram a realizar esse sonho de me formar.

Agradeço muito a minha esposa Andréia que me deu força e apoio para concluir o trabalho e por acreditar que eu seria capaz de finalizar este curso.

RESUMO

Com a evolução tecnológica, os Sistemas de Informação Geográfica (SIG) vem se tornando de grande importância na obtenção de dados geográficos com finalidade de manipular mapas e armazenar informações geográficas em diferentes bancos de dados. Esta tecnologia pode ser aplicada em várias áreas: *Global Position System* (GPS), fazer previsão do tempo, fazer medições de terrenos via satélites, aplicativos para dispositivos moveis e entre outros.

O SIG pode contar com tecnologias do tipo de dispositivos móveis que estão cada vez menores e com alto poder de processamento. Assim o campo da computação móvel vem ganhando espaço com grande crescimento neste setor como é o caso da plataforma Android, que possui recursos para trabalhar com manipulação de dados geográficos usando Google Maps. O propósito deste trabalho foi implementar um aplicativo RotaOnibus, que tem por finalidade mostrar ao usuário as rotas por onde passa o ônibus, as paradas e as ruas que pertencem as rotas, com isso, será possível através de um dispositivo móvel saber quais ônibus pegar para chegar ao destino desejado. O aplicativo foi implementado utilizando um emulador de aparelho móvel contido no kit de desenvolvimento do *Android*.

Os testes realizados na aplicação forneceram rotas compatíveis para as consultas efetuadas. Com o desenvolvimento deste trabalho foi analisado que é possível criar aplicativos capazes de fornecer um serviço de grande valor para o usuário. Por ter sido desenvolvido em um ambiente de software livre é possível implementar em órgãos púbicos para dar acesso livre a população.

PALAVRAS-CHAVE: SIG, Android, GPS, dispositivos movéis, geoprocessamento.

ABSTRACT

With technological developments, the Geographic Information Systems (GIS) has become of great importance in obtaining geographic data with the purpose of manipulating maps and geographic information stored in different databases. This technology can be applied in several areas: Global Position System (GPS), to weather, make measurements of land-satellite, mobile applications and others.

GIS technologies can count on the type of mobile devices are getting smaller and high processing power. And the field of mobile computing is gaining space with great growth in this sector as in the case of the Android platform, which has features for working with geographic data manipulation using Google Maps. The purpose of this study was to implement the application RotaOnibus, which aims to show the user the route through which the bus stops and streets that belong to the routes, thus, be possible through a mobile device to know which bus to catch reach the desired destination. The application was implemented using a mobile device emulator contained in the Android Development Kit.

The tests provided correct results in the application for the queries made. With the development of this work was reviewed that you can create applications that can provide a valuable service to the user. Because it was developed in an environment of free software can be implemented in pubic bodies to give free access to the population.

Keywords: GIS, Android, GPS, mobile devices.

LISTA DE FIGURAS

LISTA DE SIGLAS

SIG	Sistemas de Informação Geográfica
API	Application Programming Interface
XML	Extensible Markup Language
INPE	Instituto Nacional de Pesquisas Espaciais
GPS	Global Position System
UTM	Universal Transverse Mercator
SGBD	Sistemas Gerenciadores de Banco de Dados
IBGE	Instituto Brasileiro de Geografia e Estatística
DXF	Drawing Exchange Format
OHA	Open Handset Alliance
JVM	Java Virtual Machine
VM	Virtual Machine
IDE	Integrated Development Environment
SMS	Safety Management System
APK	Android Package File
MD5	Message-Digest algorithm 5
UML	Unified Modeling Language

SUMÁRIO

1 – INTRODUÇÃO

Os sistemas de informação geográfica são uma grande ferramenta tecnológica por permitir armazenamento, disponibilização e análise espacial usando mapas, cartografia e sensoriamento remoto que utiliza bancos de dados geográficos. A necessidade de se usar os métodos de Sistemas de Informação Geográfica (SIG) fica mais visível a cada dia. Hoje é possível saber o clima de uma determinada região, localizar um estabelecimento escolhido, saber a distância de uma cidade a outra, mapear uma região, entre outras coisas.

Com o passar do tempo foram surgindo tecnologias portáteis mais acessíveis à população, como é o caso do *Global Position System* (GPS) e os dispositivos móveis que podem ajudar muito no dia a dia, hoje é possível ir para qualquer lugar sem se perder, localizar um ponto turístico em determinada cidade, saber onde se encontra um usuário e até mesmo traçar uma rota de como chegar a um destino desejado.

Em diversas cidades é visto a dificuldade da população em utilizar o transporte público. Afinal, existem diversos ônibus que transitam nos mesmos pontos e com destinos diferentes, podendo confundir os usuários das linhas de ônibus, principalmente turistas e outros visitantes. A utilização desse meio de transporte é de vital importância para a locomoção da população, principalmente como forma de acesso ao local de trabalho. A falta de informação em relação às rotas de ônibus faz com que as pessoas muitas vezes cheguem atrasadas aos seus locais de trabalho.

O trabalho abordou e analisou os recursos das técnicas de SIG e a tecnologia *Android* e projetou um aplicativo capaz de mostrar um tipo de serviço prático para a população e facilitar suas vidas. O objetivo é informar as rotas dos ônibus, mostrar as paradas e as ruas que pertencem à linha escolhida, tendo o diferencial de poder ser sempre atualizado conforme as alterações das rotas e paradas.

Neste trabalho foi elaborado um estudo para a apresentação do SIG dividindo-se em cartografia e sensoriamento remoto onde serão apresentados os mapas, escalas, sistemas de coordenadas e onde é aplicado o sensoriamento remoto, em seguida os seus componentes: os recursos humanos, hardware,

software, dados e metodologias. Serão também definidos os projetos de banco de dados geográficos. E por fim a plataforma *Android* que é o sistema operacional para dispositivos móveis da Google e todos os seus recursos para um entendimento mais amplo dessas tecnologias.

Na metodologia foi desenvolvido um banco de dados para armazenar as informações geográficas das rotas e um aplicativo para interpretar essas informações, criar o mapa e posicionar cada coordenada para demonstrar sua funcionalidade. Foi feita uma representação do banco de dados com entidade de relacionamento e do aplicativo usando *Unified Modeling Language* (UML) de caso de uso e diagrama de classe. Na parte final do trabalho são apresentados os resultados das rotas completas, as ruas e as paradas do ônibus que o aplicativo proporciona para o usuário com todos os recursos proposto.

2 - REFERECIAL TEÓRICO

Neste capitulo serão abordados os assuntos sobre Sistemas de Informação Geográfica (SIG) e tecnologia *Android* que servirão como base para elaboração do sistema. Serão apresentadas algumas definições sobre SIG, abordagens sobre cartografia e sensoriamento remoto que são de grande importância para o funcionamento e entendimento do SIG, os componentes do SIG e o projeto do Banco de Dados Geográficos.

2.1 – SISTEMAS DE INFORMAÇÃO GEOGRÁFICA

O termo Sistemas de Informação Geográfica (SIG) caracteriza os sistemas de informação que envolvem análises espaciais envolvendo dados geográficos. De forma geral, se torna possível a captura, modelagem, manipulação, recuperação análise e apresentação de dados referenciados geograficamente (LISBOA, 2001).

Em um SIG é possível o usuário coletar, manusear e analisar dados georreferenciados. Ele consiste em uma combinação de *hardware*, *software*, dados, metodologia e recursos humanos, que se denominam componente de um SIG, que operam de forma harmônica para a produção e análise da informação geográfica (TEIXEIRA; CHRISTOFOLETTI, 1997).

Apresentação dos componentes do SIG (FIG. 1):

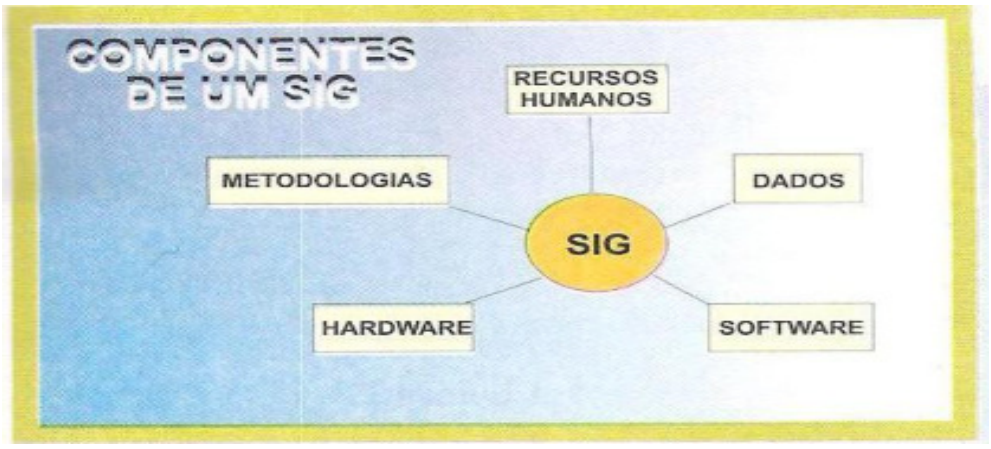


FIG. 1 – Componentes de um SIG
FONTE – TEIXEIRA, A. L. de A; CHRISTOFOLETTI, A. (1997, p. 119)

O SIG é aplicado para sistemas com tratamento em dados geográficos computacionais. A principal diferença de um SIG para um sistema de informação

convencional é a capacidade para armazenar atributos descritivos para diferentes dados geográficos. Com isso um SIG guarda informações geométricas com as coordenadas (QUEIROZ; RIBEIRO, 2006).

É possível indicar as principais características de um SIG, sendo elas: inserir e integrar, em uma única base de dados, informações espaciais provenientes ao meio físico-biótico, cadastros urbanos e rurais, imagens de satélites e GPS. Mecanismos para combinar várias informações, usando algoritmos de manipulação e análise, consultas, recuperar e visualizar o conteúdo da base de dados geográfica (QUEIROZ; RIBEIRO, 2006).

> Um *software* de SIG é composto por quatro componentes: Componente de Captura de Dados, componente de Armazenamento de Dados, componente de Análise de Dados e componente de Gerenciamento de Metadados. O componente de Captura de Dados tem por objetivo a coleta, processamento e agregação dos dados em uma forma bruta, armazenando-os em uma base de dados. Para o Componente Armazenamento de Dados é preciso ter um banco de dados bem modelado e estrutura de armazenamento apropriada para os dados coletados. Para o Componente Análise de Dados é necessário um acesso simultâneo aos dados que estão divididos geograficamente e armazenados em varios modelos de dados. Componente Gerenciamento de Metadados são coletados e armazenados nos três componentes descritos anteriormente. Os dados são armazenados, fornecendo ajuda necessária para as operações de pesquisa, navegação e transferência de dados durante a fase de Análise de dados. Metadados representam um módulo de extrema relevância numa arquitetura de extração de dados (PEREZ; MOURA; TANAKA, 2000).

Uma das principais características de um SIG é a capacidade para manipular dados gráficos (cartográficos) e não-gráficos (descritivos) de forma integrada. Pode-se fazer conexões entre diferentes fenômenos com base em relacionamentos espaciais (PEREZ; MOURA; TANAKA, 2000). Quatro aspectos caracterizam um dado georreferenciado, a saber:

- A descrição do fenômeno geográfico;
- Sua posição (ou localização) geográfica;
- Relacionamentos espaciais com outros fenômeno geográfico;
- Instante ou intervalo de tempo em que o fenômeno existe ou é válido;

Estes aspectos podem ser divididos em duas categorias: dados convencionais - atributos alfanuméricos para armazenar dados descritivos e temporais; e dados espaciais – atributos que descrevem a geometria da localização geográfica e os relacionamentos espaciais. Um SIG pode armazenar imagens sobre regiões geográficas (LISBOA, 2001).

O SIG realiza tratamento computacional de dados geográficos e recuperação de informação não apenas baseadas em suas características alfanuméricas, mas através de sua localização espacial. A condição de armazenar a geometria dos dados geográficos e seus atributos para a dualidade básica para um Sistema de Informações Geográficas (CÂMARA *et al*., 2002).

Esses sistemas oferecem em seu ambiente de trabalho informações disponíveis, interligadas com base comum a sua localização geográfica. Então, a geométrica e os atributos de dados em um SIG devem ser georeferenciados, ou seja, localizadas na superfície terrestre e representadas por uma projeção cartográficas (CÂMARA *et al*., 2002).

2.2 – CARTOGRAFIA E SENSORIAMENTO REMOTO

Para o funcionamento do SIG é necessário a utilização da cartografia e sensoriamento remoto, que são de grande importância para o entendimento do processo de funcionamento. Uma cartografia seria uma ciência para um objeto em organizar, apresentar e utilizar a informação geográfica nas formas visuais, digital ou táctil, com todos os processos de aquisição, preparação e apresentação de dados. Para isso, se utiliza informações de representações necessárias para uma cartografia sendo elas: mapas, escalas, sistemas de coordenadas e projeção cartográfica (TEIXEIRA; CHRISTOFOLETTI, 1997).

Um mapa é de grande importância para uma fonte de dados de um SIG. Pois, seria uma apresentação em escalas e superfícies planas, que seria uma seleção de características ou uma relação à superfície da terra (LISBOA, 2001). Um mapa é um instrumento para representação de informação geográfica nas modalidades digital, analógica ou táctil para uma representação ou abstração da realidade geográfica (TEIXEIRA; CHRISTOFOLETTI, 1997). Pode-se usar um mapa para vários propósitos sendo algumas delas em exibição e armazenamento de dados, como índices espaciais, como ferramenta de análise de dados ou mesmo como objetos decorativos (LISBOA, 2001).

Existem diversos tipos de mapas, dentre os quais se podem citar os mapas topográficos e temáticos. Mapas topográficos é uma representação dos acidentes planialtimétricos, permitindo efetuar medidas horizontais e verticais. Mapas temáticos são a base cartográfica com informações sobre temas diversos (geologia,

uso do solo, uma representação do relevo etc.). O objetivo principal de um mapa temático é representar fenômenos de certo tema (TEIXEIRA; CHRISTOFOLETTI, 1997).

Mapas temáticos possuem uma grande distribuição espacial geográfica, representando de forma qualitativa. Estes dados são recebidos de forma por digitação ou por forma automatizada, por meios de classificação de imagens (RAMPELOTI, 2002).

Raster são estruturas de representação de dados espaciais em que os elementos são codificados na forma de uma matriz (grid). Quanto menor for esse tamanho das quadrículas da matriz, maior será a semelhança com a representação vetorial do elemento (TEIXEIRA; CHRISTOFOLETTI, 1997). Esse tamanho da quadrícula é a função da escala de trabalho e do nível de detalhe desejado (FIG. 2).

FIG. 2 – Raster
FONTE – SILVA, E. de O. da

A escala seria a razão entre as distâncias representadas no mapa e sua correspondente distância no mundo real. Em um mapa de escala de 1:500.000 por exemplo seria que a cada 1cm no mapa corresponderia a 500.000 cm (5000m ou 5km) na superfície terrestre (LISBOA, 2001).

A escala é adimensional em relação as medidas num mapa e as respectivas medidas na superfície da terra (TEIXEIRA; CRHISTOFOLETTI, 1997).

O sistema de coordenadas define na localização de qualquer elemento sobre a superfície da terra. Os sistemas de coordenadas mais utilizados são as coordenadas geográficas (ou terrestre) e as coordenadas planas (LISBOA, 2001).

As coordenadas geográficas (FIG. 3) é um sistema de coordenadas esféricas (latitude e longitude) que permite a localização de pontos na superfície (TEIXEIRA; CHRISTOFOLETTI, 1997).

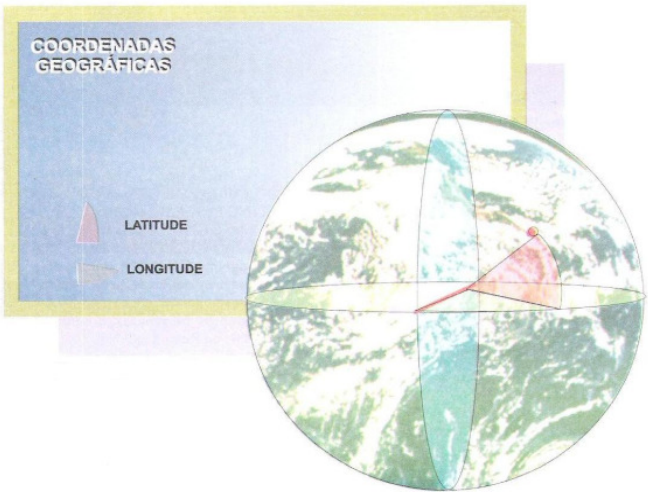


FIG. 3 – Coordenadas Geográficas
FONTE – TEIXEIRA, A. L. de A; CHRISTOFOLETTI, A. (1997, p. 119)

Coordenadas planas são baseadas em um par de eixos perpendiculares. As coordenadas são representadas por um par de valores (x,y) representando cada um dos eixos. Normalmente o eixo horizontal é a medida de longitude enquanto o eixo vertical é a medida da latitude (LISBOA, 2001).

Uma projeção cartográfica (FIG. 4) é uma técnica utilizada para representar em uma superfície plana, por meio de processos de transformação geométrica, os fenômenos que ocorrem na superfície da Terra (TEIXEIRA; CHRISTOFOLETTI, 1997).

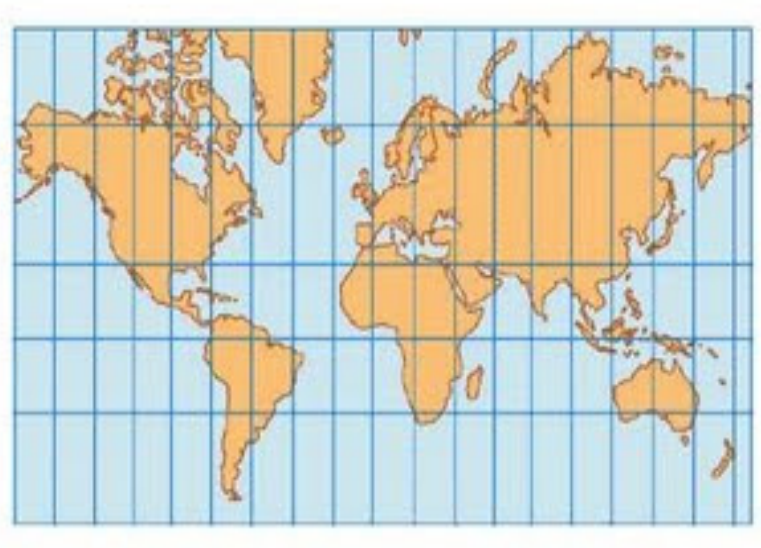


FIG. 4 – Projeção cartográfica
FONTE – mundo vestibular

Existem diversos tipos de projeções utilizadas em mapas. Algumas projeções mais utilizadas em uma projeção cartográfica são: projeção cônica de Lambert, projeção UTM (“Universal Transverse Mercator”) e projeção plana (LISBOA, 2001).

Sensoriamento Remoto é a técnica que utiliza sensores na captação e no registro da energia refletida ou emitida por elementos na superfície terrestre ou por outros astros (TEIXEIRA; CHRISTOFOLETTI, 1997).

O processamento de dados por um sensoriamento remoto é a utilização de sensores de forma remota. Existem diversas formas para capturar dados através por sensores remotos, sendo uma delas, distribuição de ondas acústicas. Estes sensores são operados por veículos aeroespaciais, ex.: satélites em órbita terrestre (LISBOA, 2001).

A superfície terrestre reflete energias eletromagnéticas por fontes naturais, (ex.: Sol) ou fontes artificiais (ex.: lâmpadas) captadas por dispositivos sensores. Os sensores eletromagnéticos são divididos em ativo ou passivo. Sensores ativos medem energias existentes no ambiente (ex.: fotografia aérea), enquanto sensores passivos geram sua própria energia (LISBOA, 2001).

Pode-se utilizar sensoriamento remoto em várias áreas de aplicação (MORAES, 1999):

- Meteorologia: previsão do tempo a um determinado prazo, mapeamento climático, etc.
- Geologia: estudos de aproveitamento do solo, procura de jazidas minerais.
- Agricultura: previsão da safra, estudo de contaminação de pragas, etc.
- Militar: usado em espionagem, mísseis teleguiados, controle de trafego aéreo e marítimo.
- Indústrias: inventário e projeção de recursos híbridos pesca e salinas.
- Ecologia: pesquisas sobre o equilíbrio ecológico do planeta
- Demografia: inventário e planejamento para controle do aumento demográfico, cidades, etc.

Os sistemas de radares emitem energias de microondas de espectros eletromagnéticos e capta a energia refletida pelos materiais que estão sobre a superfície terrestre. Um espectro eletromagnético é muito amplo e nem todos os

comprimentos de ondas são adequadas para o sensoriamento remoto (LISBOA, 2001).

Espectros eletromagnéticos são faixas de comprimentos de ondas de radiação eletromagnética, incluindo o ultravioleta, o visível, o infravermelho e as microondas (TEIXEIRA; CHRISTOFOLETTI, 1997).

A maioria das imagens produzidas via sensoriamento remoto para estas aplicações de um SIG são obtidas nestes intervalos, e existem diversas propriedades para um sensor eletromagnético que são captadas por sensor: Resolução espectral número de bandas do espectro eletromagnético que são captadas pelo sensor; resolução espacial área da superfície terrestre observada pelo sensor; resolução temporal intervalo de tempo entre duas tomadas de imagens. Os sistemas de processamento de imagens são softwares com a finalidade de resolver problemas específicos de tratamento de imagens obtidas remotamente (LISBOA, 2001).

2.3 – COMPONENTES DO SIG

Os SIGs necessitam de uma grande quantidade de informações de dados para executar operações de consulta e análise. Sistemas Gerenciadores de Banco de Dados (SGBD) são ferramentas fundamentais para um SIG.

Os Sistemas Gerenciadores de Banco de Dados são um conjuntos de módulos para organização, armazenamento, acesso, a segurança e integridade dos dados em uma estrutura de banco de dados, atuando como interface para programas de aplicação e o sistema operacional (TEIXEIRA; CHRISTOFOLETTI, 1997).

O objetivo é encontrar soluções adequadas para o problema do gerenciamento de banco de dados georreferenciados, sob os bancos de dados espaciais e geográficos. As tendências para os novos SIGs estão incorporadas com características para sistemas orientados para objetos (LISBOA, 2001).

Os SIGs constituem em três aspectos distintos da tecnologia computacional: BDGeo(sistemas de gerenciamento de banco de dados geográficos); Interface que são procedimentos para obtenção, manipulação, exibição e impressão de dados de forma gráfica; Ferramentas: algoritmos e técnicas para análise de dados espaciais (LISBOA, 2001).

Um SIG pode ser indicado pelas seguintes arquiteturas (FIG. 5): interface com o usuário; entrada e integração de dados; funções de processamento; visualização e plotagem; armazenamento e recuperação; e banco de dados geográficos (CÂMARA; ORTIZ, 2007).

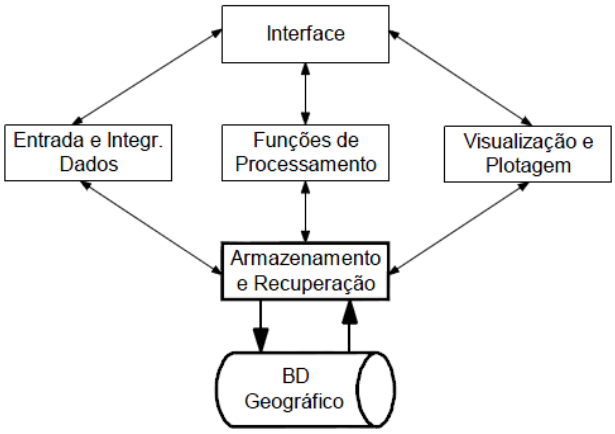


FIG. 5 – Arquitetura de um SIG
FONTE – LISBOA, J. F. (2001)

Estes componentes se relacionam de forma hierárquica. No nível mais próximo ao usuário do SIG a interface homem-máquina define como o sistema é operado e controlado. Em um nível de SIG intermediário, deve ter mecanismos de entrada, processamento, visualização e saída de dados espaciais (CÂMARA; ORTIZ, 2007).

2.3.1 – Interface com o Usuário

O primeiro tipo de interface a ser utilizado nos vários sistemas existentes foi à linguagem de comandos. De forma que a funcionalidade dos sistemas cresce, as linguagens ficam mais complexas e o nível de dificuldade aumenta. A disponibilidade dos ambientes computacionais interativos deu à origem as interfaces baseadas em menus. Estas interfaces são mais fáceis de operar e com um poder expressivo menor. A vantagem é que o usuário não precisa aprender uma linguagem para operar, pois já esta pronta. O problema dessa interface é a dificuldade para personalizar o ambiente e podendo gerar soluções incompletas ou de uso restrito (CÂMARA; ORTIZ, 2007).

2.3.2 – Entrada e Integração de Dados

A entrada de dados para aplicações de geoprocessamento é um processo muito complexo quando comparado com a maioria das aplicações. O fato dessa aplicação é quando as entradas dos dados não são apenas de operações de inserção. Estas dificuldades ocorrem por duas razões. Primeiro, por se tratar com informações gráficas, pois é uma tarefa mais complexa do que entrada de dados alfanuméricos. Segundo, é devido à natureza dos dados dessas aplicações (LISBOA, 2001).

Os processos de coleta dados são baseados em tecnologias como fotometrias, sensoriamento remoto e levantamento de campo. Existem quatro modos principais de entrada de dados: digitação em mesa, digitação ótica, entrada de dados via caderneta de campo e importação de dados digitais. A digitação em mesa é um processo mais demorado que envolve vários passos como: ajuste de nós, digitação de linhas, geração da topologia e rotulação de um objeto geográfico. A digitação ótica são instrumentos de varreduras de imagens (*scanner*) que é a forma mais usada em um ambiente de produção. É necessária a utilização de dispositivos de alta qualidade para obter resultados mais aceitáveis e algoritmos de conversão de formato matricial. A caderneta de campo é fundamental para poder inserir no sistema com as devidas checagens e correções.

O advento do GPS, os sistemas de posicionamento global são baseados em uma rede de satélites. Os SIGs atualmente no mercado possuem interfaces para importar dados oriundos de sistemas GPS. No caso da importação de dados digitais a coleta de dados foram feitas pelas principais fontes de dados, sendo elas, as fitas INPE, os dados digitais do IBGE e do centro de Cartografia Automatizada do Exército e os dados disponíveis no formato DXF (*AutoCad*) (CÂMARA; ORTIZ, 2007).

2.3.3 – Visualização e Plotagem

A visualização para ambientes são conseqüências de escolhas feitas para a interface. O uso de interfaces com janelas, onde uma ou mais telas estão a disposição do usuário (CÂMARA; ORTIZ, 2007).

No caso da plotagem, em alguns SIGs se utiliza ferramentas para produção de cartas, que utiliza recursos altamente sofisticados para uma apresentação gráfica. Estas ferramentas ajudam a indicar uma área de plotagem, legendas, textos explicativos e notas de crédito (CÂMARA; ORTIZ, 2007).

Visualização e plotagem devem oferecer suporte adequado para a apreensão cognitivas dos aspectos relevantes dos dados (QUEIROZ; RIBEIRO, 2006).

2.3.4 – Banco de Dados Geográficos

Um banco de dados geográfico armazena e recupera dados geográficos de acordo com suas geometrias (imagem, vetores, grades), bem como as informações descritivas (atributos não-espaciais). Em SIG, um conjunto de dados geográficos e seus atributos, organizados de forma adequada para operações de inserção, busca, edição e analise espacial (TEIXEIRA; CHRISTOFOLETTI, 1997).

Os SIG's anteriormente armazenavam os dados geográficos e seus atributos em arquivos internos. Então os bancos de dados, foram substituindo estes arquivos internos pelos Sistemas de Gerenciamento de Banco de Dados (SGBD), para satisfazer a demanda para um tratamento mais eficiente de base de dados espaciais cada vez maiores (CÂMARA; ORTIZ, 2007).

Um SGBD apresenta três requisitos importantes: eficiência, que é o acesso e modificações de um grande volume de dados; integridade, controle de acesso para múltiplos usuários; e persistência, que é manutenção de dados por longo tempo, independente dos aplicativos que acessem os dados. Um SGBD interliga banco de dados já existentes a um sistema de Geoprocessamento. Essa interligação de um SGBD com um SIG da origem a um ambiente "dual": os atributos convencionais são guardados em um banco de dados e os dados espaciais são tratados por sistemas dedicados (CÂMARA; ORTIZ, 2007).

A organização de bando de dados geográficos mais usados é o modelo geo-relacional (ou arquitetura dual), que utiliza o Sistema Gerenciador de Bancos de Dados relacional. A principal vantagem do modelo geo-relacional é poder utilizar os SGBDs relacionadas ao mercado. Então um SGBD relacional não conhece a estrutura gráfica, que pode provocar uma grande inconsistência no banco de dados geográficos. Assim, o acesso a atributos alfanuméricos de dados geográficos só

pode ser feito de maneira criteriosa em um controle rígido que precisa ser implementado pela aplicação (CÂMARA et al., 2002). Representação das geometrias associadas a ponto, amostra e polígono (FIG. 6).

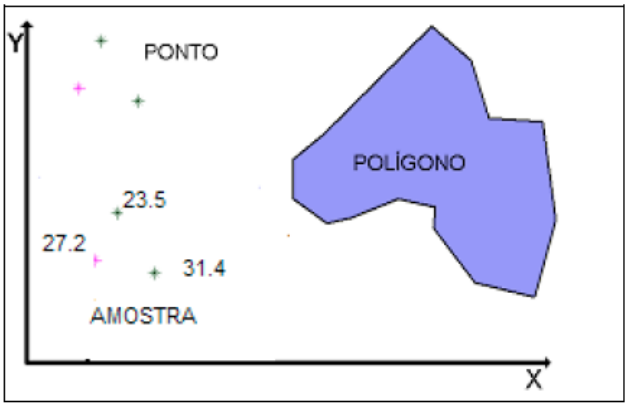


FIG. 6 – Geometrias: Ponto 2D, Amostra e Poligonos
FONTE – CÂMARA *et al*. (2002)

As representações geométricas utilizadas incluem as seguintes alternativas:

- Ponto 2D: é um par ordenado (x, y) de coordenadas espaciais. Um ponto mostra o local da ocorrência de um evento.
- Polígonos: é um conjunto de pares ordenados {(x, y)} de coordenadas espaciais, onde o último ponto é igual ao primeiro ponto, formando uma região fechada do plano. Cada polígono delimita um objeto individual, no caso mais geral uma região individual pode ser delimitada por vários polígonos.
- Amostra: consiste em pares ordenados {(x, y, z)} nos quais os pares (x, y) indicam coordenadas geográficas e (z) indica o valor do fenômeno estudado para essa localização. A amostra pode ser generalizado para o acesso de múltiplas medidas em uma mesma localidade
- Grade Regular: é uma matriz onde cada elemento possui um valor numérico. Esta matriz esta associada a uma região da superfície terrestre, normalmente referida ao canto inferior esquerda da matriz e de espaçamentos regulares na direção horizontal e vertical.
- Imagem: é uma matriz que este associado a um valor inteiro que é utilizado para visualização. Esta matriz é apresentação gráfica de uma grade regular, onde os valores numéricos da grade são escalonados para o intervalo de apresentação de imagens.

Estas referências geográficas dos dados (FIG. 6) estão associadas a uma projeção de cartografia planar ou a valores de latitude e longitude.

2.3.5 – Funções de Processamento e Análise

Um SIG deve fornecer operações para a recuperação de informações em critério de natureza espacial e não-espacial. Os SGBD mais convencionais foram projetados para recuperar informações para critérios não-espaciais. Já o SIG, deverá ser capaz em manipular e recuperar dados espaciais (LISBOA, 2001).

Existem várias funções de manipulação e análise de dados disponíveis nos sistemas atuais. Essas funções estão agrupadas em quatro categorias principais. São elas: manutenção de dados espaciais; manutenção e análise de atributos descritivos; análise integrada de dados espaciais e descritivos; e formatação de saída (LISBOA, 2001).

As funções de manutenção de dados espaciais incluem funções na fase de pré-processamento dos dados espaciais, usadas na preparação ou reorganização dos dados em operações de analise e consultas (LISBOA, 2001).

As funções de manutenção e análise de atributos descritivos, a manipulação dos atributos descritivos (não-gráficos) é realizada por linguagens de manipulação/consulta de dados disponíveis nos SGBD. Diversas operações de análise podem ser resolvidas sem consultas espaciais (LISBOA, 2001).

As funções de análise integrada de dados espaciais e descritivos esta na capacidade de realizar operações de análise espaciais envolvendo atributos espaciais e descritivos de forma conjunta (LISBOA, 2001). Existem varias funções que enquadra nesta categoria, sendo elas:

- Recuperação de dados: que envolvem busca seletiva, manipulação e geração de resultados. Processo pelo qual os dados são selecionados e extraídos de arquivos segundo (TEIXEIRA; CHRISTOFOLETTI, 1997).
- Funções de medidas: são executados em objetos espaciais e incluem funções como distância entre pontos, comprimento de linhas, perímetro de área, etc.
- Funções de sobreposição de camadas: relaciona duas ou mais camadas de dados. Esta função pode executar operações lógicas ou aritméticas.

- Funções de interpolação: valores não definidos em um método matemático para cálculos em base de estimativas feitas a partir de valores conhecidos em localizações vizinhas.
- Geração de contorno: são usadas para representar superfícies e gerar linhas de contorno para construir mapas topográficos a partir de um conjunto de pontos conhecidos.
- Funções de aproximidade: funções para análise de proximidades para geração de zonas de buffer, que é uma área de extensão regular gerada ao redor de objetos espaciais (FIG. 7).

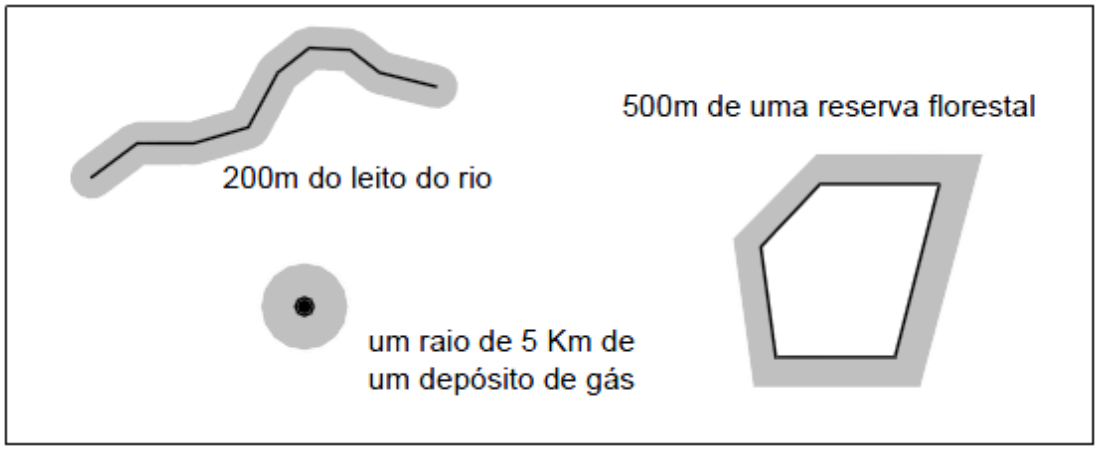


FIG. 7 – Exemplo de zonas de buffer.
FONTE – LISBOA, J. F (2001)

Como se pode observar a FIG. 7, as zonas de buffers são as partes cinza sobre as demarcações espaciais no mapa.

As funções de formatação de saída os resultados de análise espacial podem ser gerados na forma de relatórios, gráficos ou formas de mapas. Para melhorar a aparências dos mapas, existem funções como, por exemplo, anotações em mapas que permite adicionar informações como títulos, barra de escalas, orientação norte-sul, legendas, etc; posicionamento de rótulos, que são símbolos gráficos para apresentar fenômenos nos mapas; padrões de texturas e estilos de linhas, textos com tipo de fonte, tamanho, cor e estilo; símbolos gráficos, que são usados para representar classes de entidades em um mapa (LISBOA, 2001).

2.4 – PROJETO DE BANCO DE DADOS GEOGRÁFICOS

2.4.1 – Dados, Informação e Fenômeno Geográfico

Uma informação é obtida a partir do processamento ou da contextualização de dados brutos. Um dado bruto corresponde a um valor para uma medida observada (LISBOA, 2001).

De modo equivalente, informação geográfica é um tipo de informação relativa a um fenômeno ou elemento que possa ser georeferenciado. Um fenômeno geográfico é qualquer ocorrência que possa ser: natural, antrópica, de fatos ou mesmo de objetos ainda inexistentes (TEIXEIRA; CHRISTOFOLETTI, 1997).

2.4.2 – Componentes de Informação Geográfica

Uma informação geográfica possui três componentes: atributo, espaço e tempo (FIG. 8). Estes componentes possibilitam fazer três perguntas: o quê? Onde? E quando? Estes componentes determinam uma categoria ao longo da qual os valores são medidos (LISBOA, 2001).

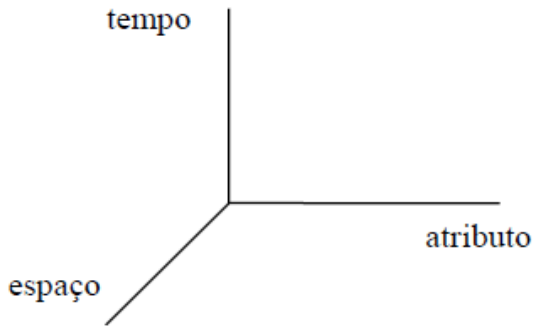


FIG. 8 – Categorias de dimensões da informação geográfica.
FONTE – LISBOA, J. F. (2001)

Componente espaço descreve a localização geográfica e a forma geométrica do fenômeno descrito pela informação geográfica. Como uma principal função de um SIG possibilita realizar operações de análise espacial, o componente espacial é o mais importante no contexto SIG. Componente atributo possui características qualitativas e quantitativas em um fenômeno geográfico descrito de

forma textual e/ou numérica O componente atributo é também chamado de atributo descritivo ou atributo não-espacial, que descreve as características não espaciais de um fenômeno geográfico. Atributos descritivos são mais freqüentes na maioria dos sistemas de informação geral (LISBOA, 2001).

Componente tempo esta relacionado há um instante ou intervalo de tempo em que este ocorre ou em que é observado. O componente tempo pode ser indispensável para a informação geográfica, dependendo do tipo de fenômeno e do tipo de aplicação em que esta sendo utilizado. Este componente vem sido tratado como uma informação complementar. A maioria dos sistemas são projetados para fornecer informação atual sobre fenômenos geográficos (LISBOA, 2001).

2.4.3 – Modelagem para Aplicação do SIG

As aplicações de SIG possui alguns requisitos especiais para modelagem pelos modelos conceituais no projeto de banco de dados para estas aplicações. Estes requisitos especiais estão representados na seguinte forma (LISBOA, 2001):

- Fenômeno Geográfico e Objeto Convencional: existem outros tipos de dados referentes ao fenômeno geográficos presente na maioria dos sistemas de informação. Uma fazenda é um fenômeno geográfico quando suas informações espaciais estão armazenadas em um banco de dados.
- Visões de Campo e de Objetos: na visão de campo a realidade é modelada por variáveis que possuem uma distribuição continua no espaço. Na visão de objetos, a realidade consiste em fenômenos individuais bem definidos e identificáveis.
- Aspectos Temáticos: a localização e a forma dos fenômenos geográficos são representadas através de objetos espaciais a um sistema de coordenadas.
- Aspectos Espaciais: os fenômenos geográficos podem ser implementados através de objetos espaciais. Este objeto representa a forma espacial do fenômeno, sendo registradas através de um sistema de coordenada e um sistema de projeção.
- Múltiplas Representações: existência de várias representações para um mesmo fenômeno geográfico. Um fenômeno geográfico pode ser

representado em diferentes escalas e projeções, inclusive por objetos espaciais diferentes.

- Relacionamento Espacial: a maioria dos SIG fornece estruturas especiais para armazenamento explicito para alguns tipos de relacionamento espacial. Somente os relacionamentos de adjacência ou de conectividade são mantidos, deixando os demais tipos de relacionamentos espaciais para serem calculados a partir de coordenadas de relacionamentos espaciais.
- Aspectos Temporais: a maioria dos SIG considera os fenômenos somente no presente. Os dados geográficos são modificados por um grande período de tempo, mais o histórico não é mantido no banco de dados. É necessário adicionar aos SIG as potencialidades dos sistemas de bancos de dados temporais para a evolução dos fenômenos geográficos.

2.5 – ANDROID

Android é um conjunto de softwares para dispositivos móveis para aplicações de um sistema operacional baseado no *Linux*. São fornecidas ferramentas e APIs necessárias a desenvolver aplicações para *Android* usando a linguagem de programação *Java* (GOOGLE, 2011).

O *Android* foi criado pela Google e possui uma parceria com um grupo de empresas gigantes no mercado conhecidas como *Open Handset Alliance* (OHA). Foi criado para padronizar a plataforma de código aberto e livre para celulares. A OHA são empresas de telefonia celulares lideradas pela Google. Alguns desses integrantes se destacam: LG, Samsung, HTC, Sony Ericsson, Motorola, Intel e entre outros (LECHETA, 2010).

O *Android* possui características como estrutura de aplicativos para permitir reutilização e substituição de componentes, Dalvik VM que é uma máquina virtual própria para dispositivos portáteis, gráficos otimizados alimentado por biblioteca de gráfico 2D personalizado e 3D que é baseado no *OpenGL ES* 1.0, suporte de mídia para áudio, vídeo e imagens estáticas(MPEG4, H.264, MP3, JSP, BMP, PNG, GIF), *SQLite* que é o banco de dados nativo para armazenamento de informações, *Bluethoot*, 3G, *Wi-Fi*, câmera, GPS, bussola e outros que são dependentes do

hardware, rico ambiente de desenvolvimento que inclui um emulador, memória e perfil de desempenho, e um plugin para *Eclipse IDE* (GOOGLE, 2011).

Para os fabricantes por existir uma plataforma única e consolidada é uma grande vantagem para criar novos dispositivos e a grande vantagem porque a plataforma é livre e de código aberto. A licença do *Android* é flexível e permite que cada fabricante possa modificar o código fonte para customizar seus produtos e não pagar nada por isso. O fato de o *Android* ser de código aberto contribui no seu aperfeiçoamento e correções de falhas por vários desenvolvedores de todo o mundo é possível criar novas funcionalidades e até mesmo novas versões da plataforma Android (LECHETA, 2010).

O Android é uma plataforma voltada e desenvolvida em uma filosofia de softwares livres. A plataforma do *Android* é baseada na linguagem *Java* e possui uma máquina virtual otimizada para dispositivos portáteis, conhecida como Dalvik VM (MACHADO, 2010).

2.5.1 – Arquitetura do Android

O Android possui aplicações, *framework* de aplicativos, bibliotecas, *Android Runtime* e *Linux Kernel* (SILVA; BRACHT, 2010). O diagrama da FIG. 9 apresenta os principais componentes do sistema operacional do Android:

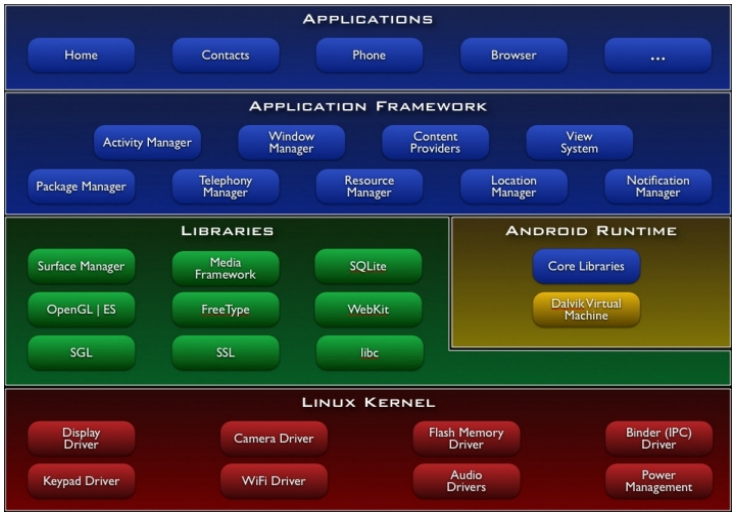


FIG. 9 – Arquitetura Android
FONTE – Google Android

2.5.1.1 – Aplicações

As aplicações são escritas usando a linguagem de programação *Java* e possui compilada no sistema operacional como sistema de email da Google, SMS, calendário, mapas, contatos, navegador de internet entre outros. (GOOGLE, 2011).

2.5.1.2 – Framework de Aplicativos

O *framework* de aplicativos fornece uma plataforma de desenvolvimento aberto e oferece aos desenvolvedores a capacidade de criar aplicações ricas e inovadoras usando a API do *Android*. Os desenvolvedores são livres para utilizar o hardware de localização, definir alarmes, notificações para adicionar na barra de *status*, *web services*, sms, banco de dados e etc. Os desenvolvedores têm acesso completo às APIs do mesmo quadro das aplicações core (GOOGLE, 2011).

O fundamento de todas as aplicações do *framework* é um conjunto de serviço e sistemas que pode definir como:

- *Activity Manager* (Gerenciador de atividade): Gerencia o ciclo de vida das aplicações.
- *Package Manager* (Gerenciador de pacotes): Mantém quais aplicações estão instaladas no dispositivo.
- *Window Manager* (Gerenciador de janelas): Gerencia as janelas das aplicações.
- *Telephony Manager* (Gerenciador de telefonia): Componentes para acesso aos recursos de telefonia.
- *Content Providers* (Provedores de conteúdo): Permitem que as aplicações acessem os dados de outras aplicações (como contatos) ou compartilhem os seus próprios dados.
- *Resource Manager* (Gerenciador de recursos): Fornece acesso a recursos gráficos e arquivos de *layout*.
- *View System* (Visão do sistema): Um conjunto rico e extensível de componentes de interface de usuário. As visões podem ser usadas para construir uma aplicação, elas incluem listas, grids, caixas de texto, botões, dentre outras.
- *Location Manager* (Gerenciador de localização): Gerencia a localização do dispositivo.
- *Notification Manager* (Gerenciador de notificações): Permite que todas as aplicações exibam alertas na barra de status. (AQUINO, 2007, p. 7)

2.5.1.3 – Runtime Android

No *Runtime Android* uma aplicação é gerada em código Dalvik Executáveis (dex) e não aos *byte-code* do *Java* (.class). Esses arquivos são interpretados pela máquina virtual Dalvik e após a compilação, todos os arquivos.dex são compactados

para o formato .apk (*Android Package File*) que é a aplicação finalizada e pronta para ser instalada em qualquer dispositivo que rode *Android* (SILVA; BRACHT, 2010).

A máquina virtual Dalvik usa o *kernel* do *GNU/Linux* para funcionalidade de múltiplos *threads* e gerenciamento de memória de baixo nível. Uma das diferenças da máquina virtual Dalvik em relação a maquina virtual *Java* (JVM) é que Dalvik é baseado em registradores e a JVM é baseada em pilhas. As máquinas virtuais baseadas em registradores executam os processos mais rápido do que as máquinas baseada em pilhas (AQUINO, 2007).

2.5.1.4 – Bibliotecas

O *Android* inclui um conjunto de bibliotecas C/C++ usadas por diversos componentes do sistema *Android*. Para os desenvolvedores estas estruturas são expostas na aplicação do *Android* (GOOGLE, 2011). Algumas das principais bibliotecas do Android são listadas abaixo:

- *System C Library*: uma implementação derivada da biblioteca padrão C(libc), sintonizado para dispositivos baseados em *Linux*.
- *Media Libraries:* são as bibliotecas para reprodução e gravação de áudios e vários formatos de vídeos, incluindo MPEG4, MP3, H.264, AAC, AMR, JPG, PNG.
- *Surface Manager:* gerencia o acesso ao subsistema de *display* e compõe as camadas 2D e 3D de múltiplas aplicações.
- *LibWebCore:* biblioteca para o navegador do *Android* e visões da web embutida.
- *SGL:* biblioteca responsável para compor gráficos 2D.
- *FreeTypes:* biblioteca responsável para renderizar fontes e bitmap.
- *3D libraries:* uma aplicação baseada em OpenGL ES 1.0.
- *SQLite:* um banco de dados relacional e leve disponível para todos os aplicativos. O *SQLite* não é uma biblioteca de cliente usada para conectar um servidor, o *SQLite* lê e escreve diretamente no banco de dados no disco (AQUINO, 2007).

2.5.1.5 – Linux Kernel

O *kernel* do Android foi baseada na versão 2.6 do Linux. Inclui os programas de gerenciamento de memória, conFIG.ções de segurança, gerenciamento de energia, *driver* da câmera, *Wi-Fi* e entre outros (MACK, 2010).

2.5.2 – Componentes do Android

No *Android* os componentes podem ser compartilhados e é possível usar os recursos de outras aplicações. Com *Android* é permitido que processos de aplicação sejam iniciados somente quando forem necessários (MACK, 2010).

Cada componente possui pontos diferentes e nem todos são pontos de entrada real para o usuário e alguns dependem um dos outros. Cada componente possui uma entidade própria e desempenha um papel específico, onde cada bloco de construção singular ajuda a definir o comportamento da aplicação global. Existem quatro tipos de componentes de aplicações. Cada tipo possui uma característica distinta e tem um ciclo de vida que define como o componente é criado ou destruído. Esses componentes são: *activity, services, broadcast receivers e content providers* (GOOGLE, 2011).

Nem toda a aplicação necessita de ter os quatro tipos de componentes, mais é necessário que tenha um ou mais desses componentes para criar o ciclo de vida da aplicação. Quando é definido os componentes a serem usados na aplicação é necessário que liste em um arquivo do Android chamado AndroidManifest.xml (AQUINO, 2007).

2.5.2.1 – Activity

A classe *activity* representa uma tela na aplicação. Cada *activity* é responsável para controlar os eventos de tela e definir qual *View* será responsável para desenhar a interface gráfica para o usuário. Cada *activity* deve implementar o método onCreate(bundle), que é obrigatório para inicializar a aplicação, como por exemplo, setContentView(View) para definir a interface de usuário, que é o método responsável por desenhar os elementos gráficos na tela (LECHETA, 2010).

A maioria das aplicações possui múltiplas telas e a mudança para outra tela é feita com uma nova *activity*. O android possui uma classe especial chamada *intent* que descreve o que o que a aplicação vai fazer (AQUINO, 2007).

Uma *intent* está presente em toda parte da aplicação e representa uma mensagem para o sistema operacional. É uma classe muito importante para arquitetura do Android. A tradução da palavra *intent* significa intenção, que representa a intenção para o sistema operacional executar uma determinada tarefa solicitada. Uma *intent* pode ser usada, por exemplo, quando solicitado ao sistema operacional que faça uma ligação, que exiba algum endereço, localização ou rota no Google *Maps*, envie uma mensagem para outra aplicação executar uma determinada tarefa ou abra o *browser* em um determinado endereço (LECHETA, 2010).

2.5.2.1.1 – Ciclo de Vida de uma Activity

Cada componente tem seus ciclos de vida. São diferentes estados em que um componente assume desde sua criação até a sua destruição (MACK, 2010).

Uma *activity* tem ciclo de vida bem definido. O sistema operacional cuida desse ciclo de vida e possui métodos importantes em relação este como (LECHETA, 2010):

- *onCreate*: método obrigatório que é chamado uma vez.
- *onStart*: é chamado quando fica visível e que já tenha uma *view*.
- *onRestart*: quando o aplicativo foi parado temporariamente e iniciado outra vez.
- *onResume*: quando uma *activity* está no topo da pilha.
- *onPause*: método que salva o estado da aplicação quando é parado temporariamente.
- *onStop*: método que é chamado quando uma *activity* esta sendo encerrada.
- *onDestroy*: método que encerra a execução de uma *activity*.

2.5.2.2 – Service

O *service* é um componente que executa em segundo plano para operações de logo duração ou para realizar processos de controle remoto. Um *service* não fornece uma interface gráfica (GOOGLE, 2011).

Um *service* tem um alto consumo de recursos, memória e CPU do dispositivo. A classe *service* sempre é executada em segundo plano, e o próprio sistema operacional cuida de seu processo e do gerenciamento de memória. Geralmente um *service* é iniciado apartir de um *Broadcast Receiver* (LECHETA, 2010).

2.5.2.3 – BroadCastReceivers

Este é um tipo de componente simples e funciona apenas recebendo e reagindo à transmissão de anúncios. Muitos aplicativos podem iniciar uma transmissão, com isso, os aplicativos podem se comunicar com outras aplicações no dispositivo (MACK, 2010).

Um *broadcastreceivers* sempre é executado em segundo plano e não utiliza uma interface gráfica. O seu principal objetivo é receber uma mensagem (*intent)* e processá-la sem que o usuário perceba. Para conFIG.r um *broadcastreceiver* é usado a *tag* <*receiver*> no aquivo AndroidManifest.xml em conjunto com a *tag* <*intent-filter*> para definir uma ação e categoria. (LECHETA, 2010).

Um *broadcastreceiver* por não apresentar uma interface gráfica com o usuário, eles podem criar uma notificação na barra de status para alertar o usuário quando um evento ocorre (GOOGLE, 2011).

2.5.2.4 – Content Provider

O conceito do *content provider* é para que determinada informação seja pública para outras aplicações instaladas no dispositivo. O *Android* permite o desenvolvimento de aplicações com banco de dados *SQLite* que é nativo dele, mas não é possível compartilhar entre diferentes aplicações. Usando essa classe será possível inserir, editar, excluir e consultar no banco de dados em diferentes aplicações. O *Android* tem uma série de provedores de conteúdo nativos como

consultar os contatos da agenda, visualizar arquivos, imagens e vídeos disponíveis no celular (LECHETA, 2010).

2.5.3 – MapActivity

O *MapActivity* é a integração do Google *Maps* e GPS (*Global Positioning System*), onde se usa o *MapView* para interpretar o mapa passando por coordenadas usando *GeoPoint* que é o objeto responsável para criar os pontos no mapa. Essas coordenadas usam a notação *microdegrees* que são graus de latitude e longitude multiplicados por 1E6 (mesma coisa que 1.000.000, onde o 6 significa a quantidade de zeros a esquerda) apenas numéricos, sem decimais (LECHETA, 2010).

2.5.3.1 – Biblioteca externa do Google Maps

O Google oferece o serviço de biblioteca externa do Google *Maps* usando a biblioteca dentro do pacote “com.google.android.maps”. A classe *MapActivity* é entendida para obter um *MapView* para visualizar um mapa usando coordenadas geográficas. Como o Google *Maps* não é *Open Sourse* é necessário obter a chave de acesso que é disponibilizada pelo próprio Google para utilizar esse tipo de serviço. A chave do Google *Maps* é responsável para dar acesso as APIs e ativar os elementos do *MapView* na aplicação (GOOGLE, 2011).

No referencial teórico foi apresentado informações sobre o SIG e *Android*, com todas as características e funcionalidades. No próximo capitulo será abordado a metodologia do trabalho e o funcionamento do aplicativo com a ideologia do SIG.

3 – METODOLOGIA

Neste capítulo é mostrado à estrutura do aplicativo RotaOnibus, que possibilitará visualizar rotas de ônibus cadastrados no banco de dados geográfico, usando o conceito do SIG. Após descrever sobre as duas abordagens essenciais à elaboração desse trabalho, plataforma *Android* e SIG, tem-se condição para desenvolver o aplicativo proposto.

Inicialmente será abordada a estrutura do banco de dados usando o *SQLite*, porque é o banco do dados interno padrão da plataforma *Android*. Ele é usado para armazenar as rotas do ônibus, com suas respectivas paradas, ruas, bairros e toda a rota da linha, além do diagrama de caso de uso. Em seguida, é descrito os passos para obtenção da chave de acesso ao Google *Maps*, para visualização dos mapas no sistema usando coordenadas geográficas. E por fim, uma modelagem da aplicação RotaOnibus com suas respectivas características e soluções para o problema.

Os dados foram coletados através de observação direta que "[...]é uma técnica de coleta de dados para conseguir informações e utiliza os sentidos na obtenção de determinados aspectos da realidade" (BERTUCCI, 2009, p. 67).

A coleta dos dados, rotas e pontos de parada, foi realizada dia 10 de setembro de 2011 através da realização do percurso da rota do ônibus utilizada no trabalho, feita pelo autor, que partiu do ponto situado na Rua Antônio Wellerson, na cidade de Manhuaçu-MG, perfazendo a rota completa e retornando ao mesmo ponto de partida. Com o objetivo de recolher os dados para alimentar o banco de dados do sistema.

3.1 – ESTRUTURA DO BANCO DE DADOS E TABELAS

O modelo de dados para representar as rotas de uma determinada linha de ônibus, sendo as rotas, paradas, ruas, bairros e as coordenadas, será apresentado em seguida, mas antes, é importante demonstrar uma característica principal da base de dados que é disponibilizar os acessos com informações usando funções.

O objetivo do banco de dados projetado para a aplicação é apresentar e aplicar as informações contidas, para a aplicação ler essas informações e converter

os dados para métodos de sistemas geográficos usando a API do Google *Maps*. Para a aplicação RotaOnibus os dados são um conjunto de informações contendo as rotas que são as linhas de ônibus, as paradas dos pontos onde passa cada ônibus, as ruas e bairros que pertencem a rota do ônibus por onde ele passa e por fim todas as coordenadas da linha do ônibus onde é usado as representações geométricas ponto 2D e polígonos.

A estrutura do banco de dados da aplicação RotaOnibus foi projetada usando diagrama de classe UML. Modelo das tabelas e os relacionamentos (FIG. 10):

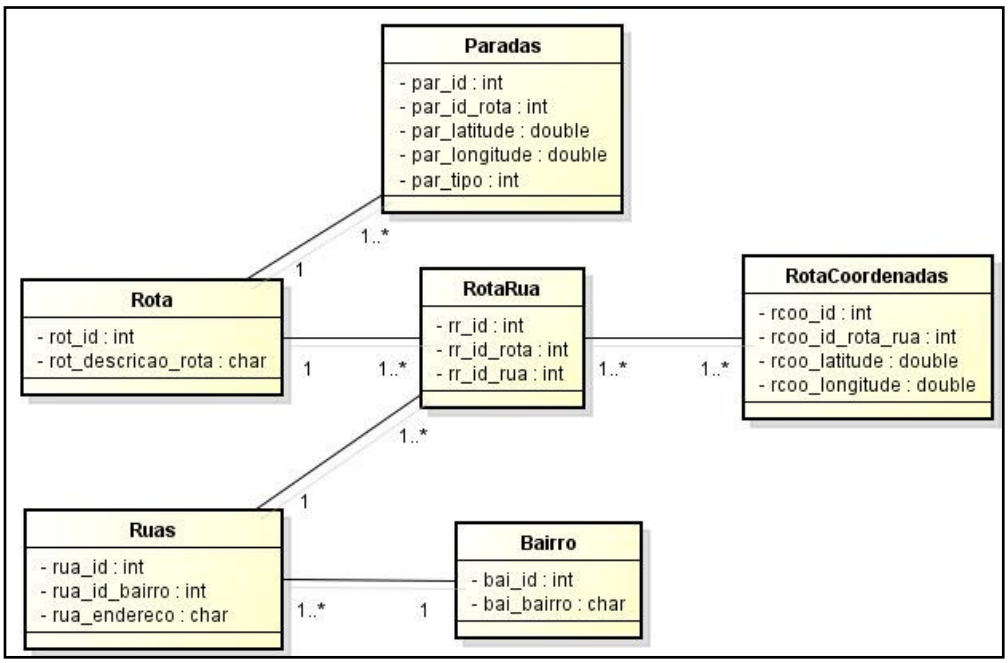


FIG. 10 – Modelo do banco de dados
FONTE – Dados da pesquisa

O diagrama na FIG. 10 apresenta o modelo do banco de dados utilizado no sistema, pois mapeia apenas o necessário para disponibilizar informações essenciais para a consulta da linha dos ônibus com duas posições geográficas sendo a latitude e a longitude.

A entidade Rota tem como finalidade em consultar as paradas, o caminho percorrido do ônibus dessa rota ou as ruas que pertence cada rota e com a possibilidade de mostrar o caminho de apenas uma rua.

Na tabela de Parada contem todos os pontos de ônibus em uma determinada rota selecionada na aplicação. Com isso, são demonstradas as

paradas por onde o ônibus irá passar em todos os pontos da cidade com os dados espaciais de latitude e longitude.

A tabela Ruas tem informações de todas as ruas que pertencem à rota selecionada. Esta tabela se relaciona a RotaRua onde fica todas as rotas e ruas para afim se relacionar com a RotaCoordenadas onde ficam todas as coordenadas das rotas dos ônibus, onde será possível mostrar as coordenadas de onde começa e termina a rua na rota especificada.

Na entidade de RotaRua tem os identificadores da rota e das ruas para se relacionar com a tabela RotaCoordenadas. A tabela Bairro está apenas ligada a tabela Ruas, pois um bairro pode ter várias ruas ligadas a ele. Então para cada bairro pode ter pelo menos um ou mais ruas ligadas a ele.

A entidade RotaCoordenadas é a principal tabela do sistema, pois é nela que estão contidas todas as informações das coordenadas de cada rota da linha de ônibus da aplicação. A tabela possui o identificador da RotaRua que é obrigatório em RotaCoordenadas, pois é dela que se pode selecionar quais serão as informações pesquisadas para compor o *GeoPoint* do Google *Maps* para rota ou para a rua. Os campos de latitude e longitude ficam as coordenadas onde combinadas tem uma informação geográfica espacial para o mapa, nesse tipo de ideologia pode-se aplicar o conceito do SIG usando ponto 2D.

O modelo proposto foi implementado usando o banco de dados SQLite, que é um motor de banco de dados SQL embutido que, com todas os aspectos da biblioteca embutidas, possui tamanho inferior a 350 kb. Estas características fazem do SQLite uma escolha bastante popular entre os produtores de aplicativos para dispositivos móveis (SQLITE, 2011).

3.2 – CHAVE PARA ACESSAR O GOOGLE MAPS

Para acessar a API do Google *Maps* é necessário obter uma chave de acesso, pois sem ela não será possível visualizar o mapa pelo *Android*. Para adquirir essa chave foi necessário ir ao *prompt* do *DOS* no *Windows* e em seguida até a pasta onde fica o *keytool* do *Java*. No *prompt* foi digitado o seguinte caminho: “cd Arquivos de programas\Java\ jdk1.6.0_24\bin”. Agora é preciso saber onde fica o arquivo debug.keystore do android. No Windows XP, que foi utilizado neste trabalho,

fica na pasta “C:\Documents and Settings\User\.android\debug.keystore”. User é o nome do usuário da máquina onde será feito o desenvolvimento no *Windows*.

Com essas informações foi executado o comando do *keytool* da seguinte forma: keytool –list –alias androiddebugkey –keystore “C:\Documents and Settings\User\.android\debug.keystore” –storepass android –keypass android. Foi obtido o seguinte certificado MD5[1]. O ANEXO II apresenta o caminho para chegar ao certificado MD5.

Após adquirir o certificado MD5 é possível obter a chave do Google *Maps* na pagina “http://code.google.com/intl/pt-BR/android/add-ons/google-apis/maps-api-signup.html”. Quando colocar o código do certificado MD5 será gerado a chave de acesso para o Google *Maps* (FIG. 11).

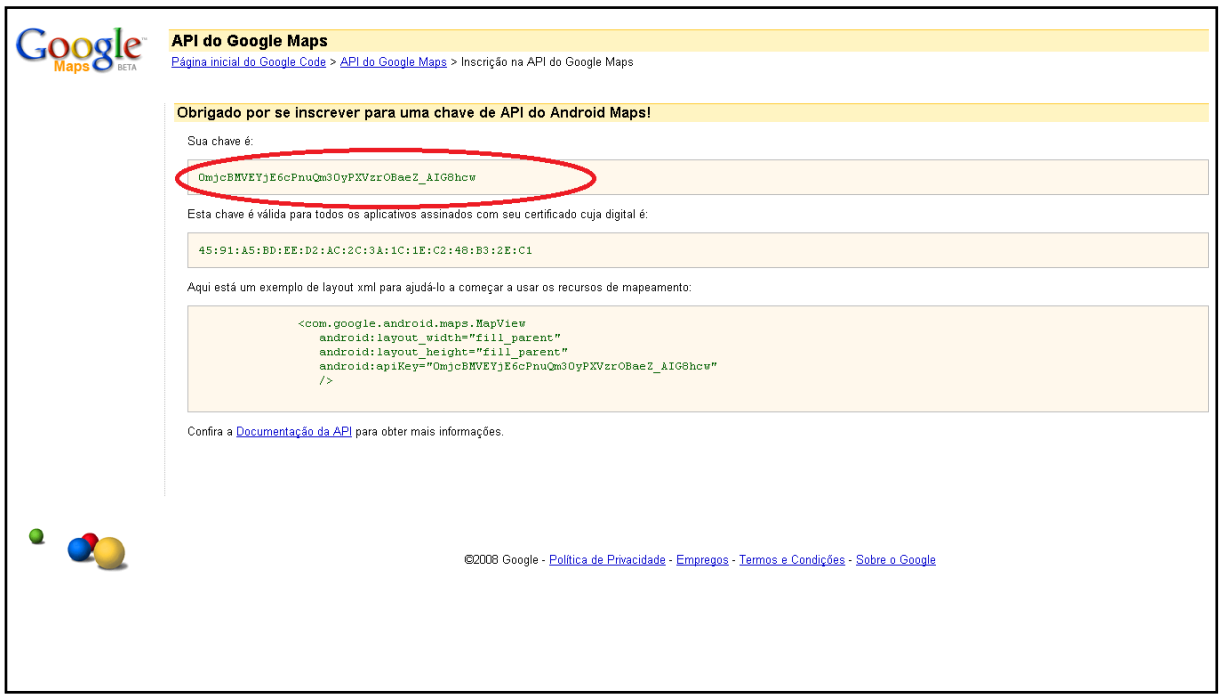


FIG. 11 – Obtenção da chave de acesso para o Google Maps
FONTE – Dados da pesquisa

[1] MD5: é uma seqüência numérica computada através da leitura dos bytes em um arquivo. Disponível em: < http://comunidade-linux-brasil.info/index2.php?option=com_content&do_pdf=1&id=44>. Acesso em: 10 set. 2011

3.3 – ARQUITETURA E IMPLEMENTAÇÃO DA APLICAÇÃO ROTAONIBUS

A arquitetura e a implementação do aplicativo tem o objetivo de demonstrar o funcionamento e abstração dos elementos que compuseram o sistema, com os relacionamentos entre eles e a fase de operação do sistema a ser desenvolvido.

3.3.1 – Arquitetura geral da aplicação

A arquitetura do aplicativo consiste em visualizar as rotas cadastradas e disponibilizar três ações para poder visualizar as rotas das linhas, as paradas e as ruas que possui uma opção para visualizar de onde começa e termina uma determinada rua. Será demonstrado a funcionalidade principal do aplicativo RotaOnibus com ajuda do diagrama de caso de uso (FIG. 12):

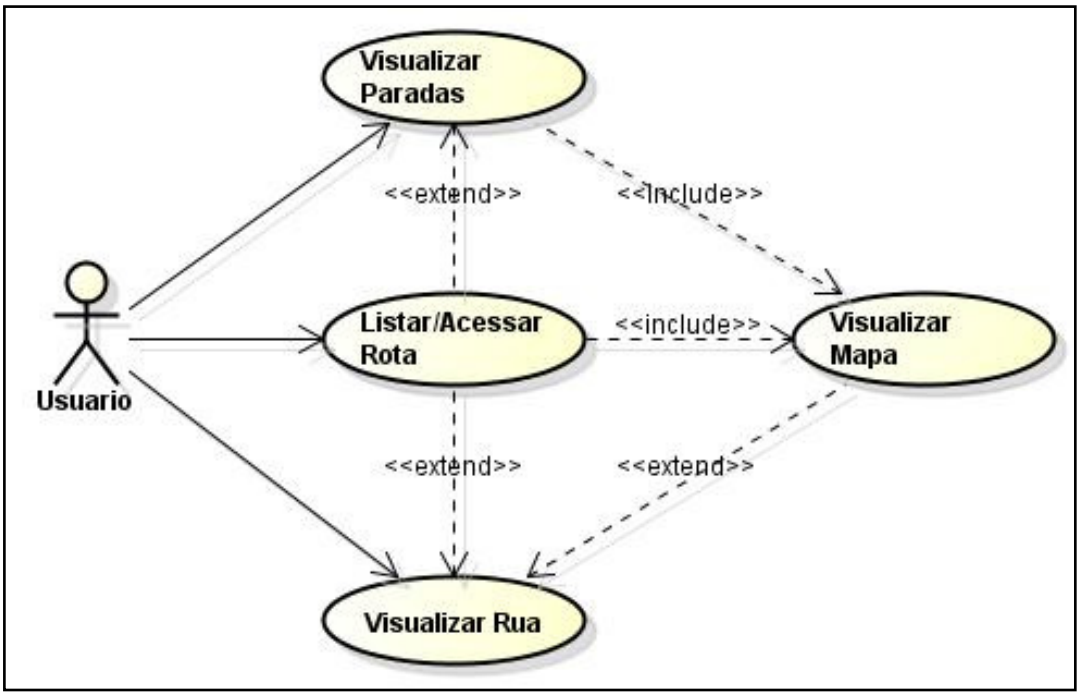


FIG. 12 – Caso de uso da aplicação RotaOnibus
FONTE – Dados da pesquisa

De acordo com a FIG. 12 o diagrama de caso de uso exibe a funcionalidade da aplicação de maneira bem simples. Então, será necessário escolher uma rota do ônibus, onde será possível escolher as opções de visualizar a rota geral, paradas ou as ruas que pertence a uma determinada rota. A seguir será detalhada cada funcionalidade que pertence ao caso de uso.

3.3.1.1 – Listar ou Acessar Rota

Neste processo serão listadas todas as rotas cadastradas na tela principal da aplicação. Nesta parte terá três formas para visualização do mapa sendo possível selecionar uma rota com apenas um clique. O processo de visualização do mapa buscará todas as coordenadas espaciais do banco de dados, latitude e longitude, onde irá criar o caminho percorrido pelo ônibus, ou seja, sua rota. Caso pressione por um determinado tempo a rota escolhida, serão disponibilizadas duas opções: ponto ônibus ou lista ruas (FIG. 13).

FIG. 13 – Tela principal do listar ou acessar rota e duas opções de menu
FONTE – Dados da pesquisa

Quando escolher uma rota com um clique o sistema interpretará a rota escolhida, fazer as buscas no banco de dados e pegar as coordenadas para acessar a funcionalidade de visualizar o mapa (FIG. 14).

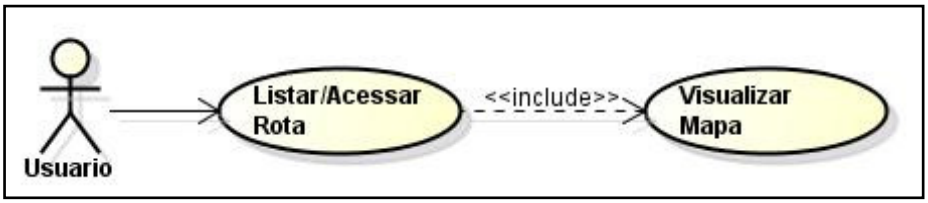


FIG. 14 – Caso de Uso Listar/Acessar Rota
FONTE – Dados da pesquisa

3.3.1.2 – Visualizar Paradas

A funcionalidade de visualizar as paradas sendo a ponto ônibus, tem como objetivo no sistema de consultar e exibir todas as coordenadas dos pontos de ônibus que pertence a uma determinada rota selecionada pelo usuário. Este processo usa a ideologia do SIG de ponto 2D, onde possui as coordenadas de latitude e longitude para a execução do processo de visualizar mapa.

Com a opção de ponto ônibus, serão processadas todas as coordenadas armazenadas no banco de dados que pertence a rota selecionada e obrigatoriamente irá mostrar o mapa (FIG. 15).

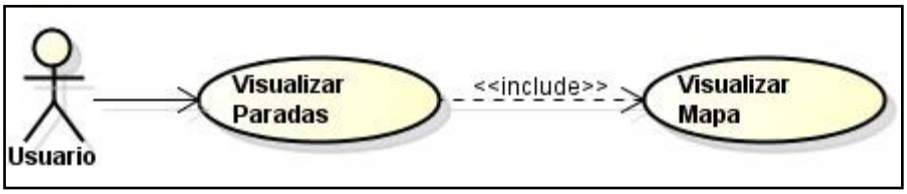


FIG. 15 – Caso de Uso Visualizar Paradas
FONTE – Dados da pesquisa

3.3.1.3 – Visualizar Ruas

O processo de visualizar as ruas tem o objetivo de listar todas as ruas que pertence à rota do ônibus por onde ele passa. Quando listado as ruas, será possível selecionar uma determinada rua e poder visualizar o mapa com a rota por onde o ônibus irá passar. Esse processo, por exemplo, possibilita saber o tamanho da rua e quais as ruas tem acesso a ela (FIG. 16).

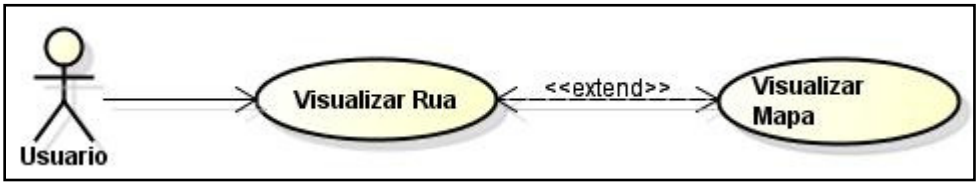


FIG. 16 – Caso de Uso Visualizar Ruas
FONTE – Dados da pesquisa

3.3.1.4 – Visualizar Mapa

Essa funcionalidade tem o principal objetivo de montar o mapa e posicionar todos os pontos das coordenadas de acordo com as três opções citadas a cima.

Todo o processo referente à ideologia do SIG será executado na parte do sistema visualizar mapa. Os recursos de localização e mapas são baseados na API do Google *Maps* no sistema do *Android* e com os métodos para tratar as manipulações das coordenadas de geoprocessamento usando o *GeoPoint*.

3.3.2 – Implementação da aplicação RotaOnibus

Nesta seção serão abordados pontos importantes para explicar o funcionamento do sistema RotaOnibus. O sistema consiste em várias classes necessárias para o desenvolvimento dos métodos e atributos. A arquitetura do diagrama de classe dos métodos para elaboração do sistema (FIG. 17).

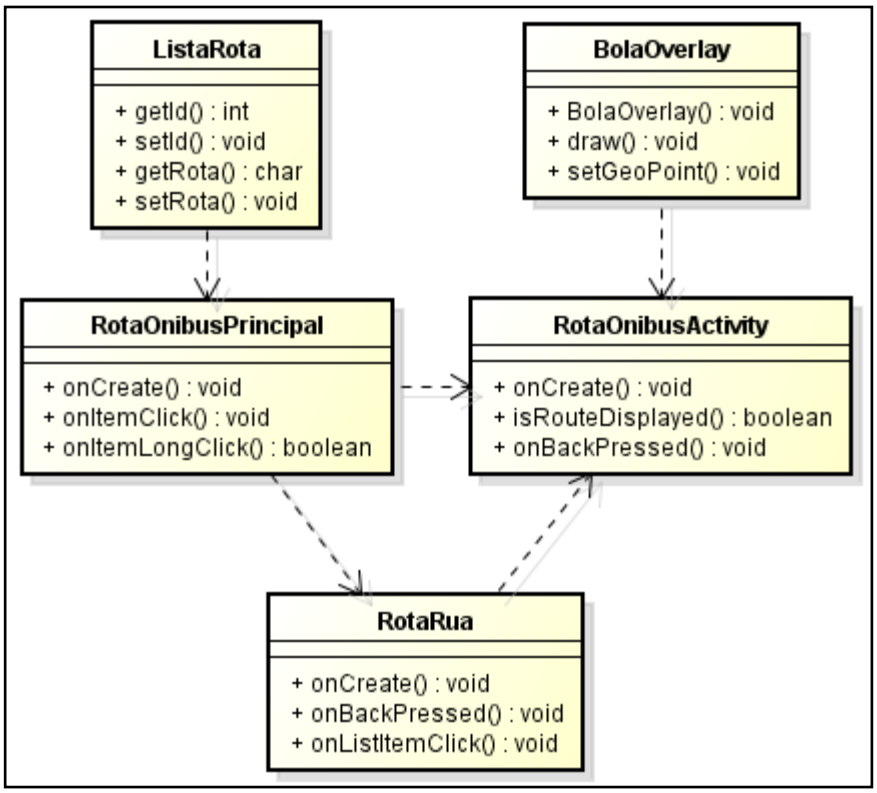


FIG. 17 – Diagrama de Classe RotaOnibus
FONTE – Dados da pesquisa

As classes e métodos que compõem o sistema são necessários para interpretar as informações do banco de dados, a criação da interface gráfica e a modelagem do mapa referente à cidade. As classes do diagrama são definidas como: ListaRota, RotaOnibusPrincipal, RotaRua, RotaOnibusActivity e BolaOverlay. A seguir a descrição de cada classe.

3.3.2.1 – Classe ListaRota

A classe ListaRota é responsável por armazenar e enviar as informações da rota em forma de lista. Os métodos que possuem esta classe irão compor e enviar as informações para lista de array na classe RotaOnibusPrincipal. A ListaRota possui os seguintes métodos: getID() e setID() com a função de retornar e enviar o id da rota para um objeto, getRota() e setRota() para retornar e enviar a descrição da rota.

3.3.2.2 – Classe RotaOnibusPrincipal

A classe RotaOnibusPrincipal é responsável por criar a primeira tela do sistema usando o método onCreate(). Este método carrega o layout de um arquivo XML, abre ou a cria o banco de dados e cria os objetos necessários para visualização das informações na tela. Esta classe é composta pelos métodos onCreate(), onItemClick(), onItemLongClick(), conforme descrições a seguir:

- Método onCreate(): é o método responsável por criar a tela da classe RotaOnibusPrincipal baseado em um arquivo XML. Foi necessária uma abertura do banco de dados para manipular os dados usando um cursor, com isso, uma lista de array do tipo ListaRota foi criada para receber as informações passadas pelos métodos e adicionadas na lista. A tabela que irá compor esta lista é a tabela rota, que passará seu id e a descrição da rota para cada linha da lista.
- Método onItemClick(): método que irá retornar a posição da lista selecionada, pegar o id e a descrição da rota. Com a rota selecionada, será possível enviar o ID da rota por parâmetro para a classe RotaOnibusActivity, que irá montar o mapa e as coordenadas passadas pela rota escolhida.

- Método onItemLongClick(): método responsável em visualizar duas opções de escolha sendo: Ponto Ônibus e Lista Rua. Nestas opções é possível pegar o id da rota selecionada e acessar as classes correspondentes a elas. Na opção Ponto Ônibus é possível acessar a classe RotaOnibusActivity e montar as paradas por onde o ônibus passa. A opção Lista Rua acessará a classe RotaRua através da qual é exibida as ruas da rota selecionada.

3.3.2.3 – Classe RotaRua

Esta classe é responsável em pesquisar no banco de dados todas as ruas que pertencem à rota selecionada. Irá receber um parâmetro da classe RotaOnibusPrincipal e exibir em uma lista as ruas. É possível selecionar uma rua específica e acessar a classe RotaOnibusActivity para criar no mapa as coordenadas da rua selecionada.

3.3.2.4 – Classe RotaOnibusActivity

A classe RotaOnibusActivity é responsável em criar a interface gráfica do mapa e posicionar as coordenadas. Esta classe recebe três parâmetros das classes RotaOnibusPrincipal que é dividido pela paradas e a rota completa e pela classe RotaRua, para assim, criar as marcações no mapa.

Nesta classe RotaOnibusActivity foi usado um objeto para ler e converter as coordenadas para aplicar no mapa. No momento em que são lidas as coordenadas é criado uma *overlay*[2] para adicionar no mapa uma imagem para visualização. Estas *overlays* são criadas por uma classe chamada BolaOverlay.

3.3.2.5 – Classe BolaOverlay

A finalidade desta classe é criar uma imagem no formato de uma bola para um ponto de coordenada no mapa. É possível definir a cor para facilitar a visualização e diferenciação do ponto.

[2] *Overlay*: é uma subclasse responsável para criar um desenho especial, que podem ser desenhados e adicionados no mapa.

4 – ANÁLISE DE RESULTADOS

Neste capítulo é mostrado o resultado das consultas das rotas no aplicativo RotaOnibus e visualizado no Google *Maps*. O aplicativo consiste em três tipos de saídas de resultados sendo: rota do ônibus completa, as paradas de uma determinada rota organizada por cores e todas as ruas que pertencem à rota com as coordenadas separadas por cada rua.

No mapa foi possível definir cada ponto de coordenada por círculos com cores em azul ou vermelho. Foi usado o conceito de ponto 2D e polígonos do SIG para aplicar no mapa referente a cidade de Manhuaçu-MG.

No resultado da rota completa da rota Cidade SUS foi adicionado círculos para cada coordenada latitude e longitude no mapa, podendo ajudar na visualização de todo o percurso que o ônibus irá fazer. Com isso, é possível saber em que parte da cidade o ônibus percorre em uma determinada rota escolhida na pesquisa (FIG. 18).

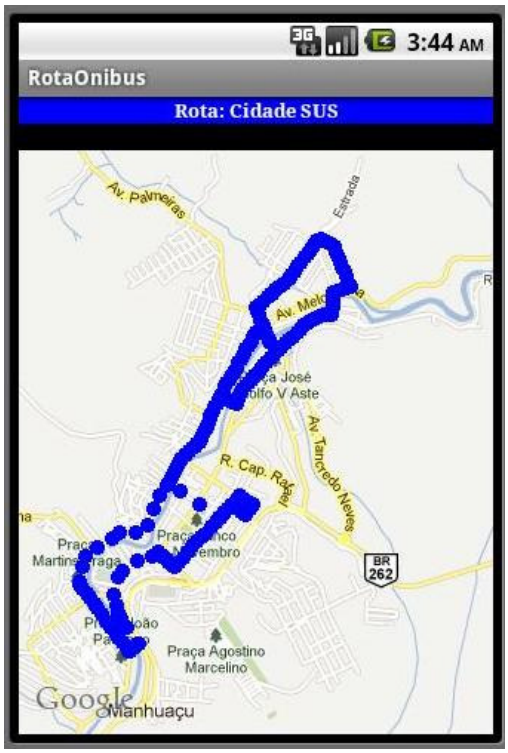


FIG. 18 – Resultado da rota completa para Cidade SUS
FONTE – Dados da pesquisa

A rota Cidade tem uma semelhança referente à Cidade SUS, onde o caminho percorrido será pela Praça Cinco de Novembro (FIG. 19).

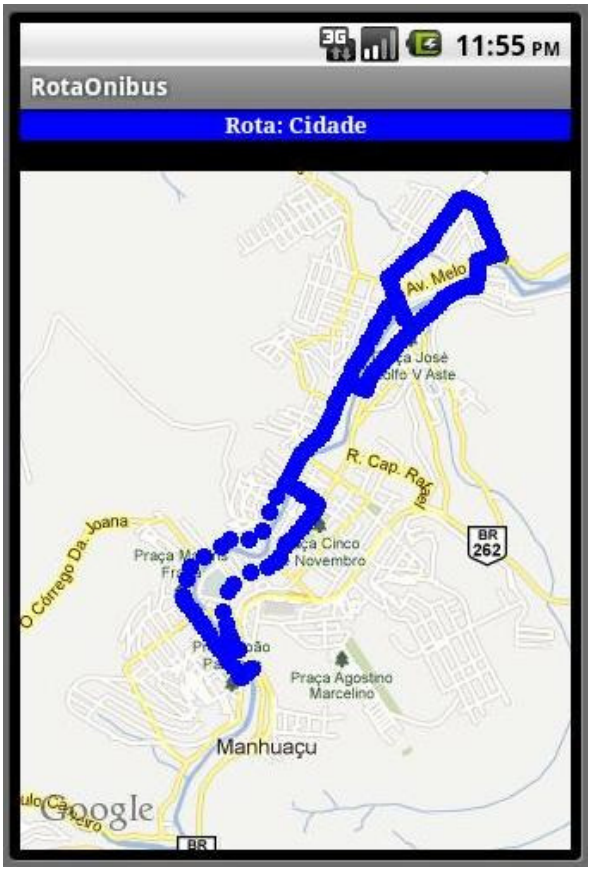


FIG. 19 – Resultado da rota completa para rota Cidade
FONTE – Dados da pesquisa

De acordo com a FIG. 20, para o caso do resultado referente às paradas de ônibus é possível obter cores diferentes para cada ponto no mapa. Estes pontos são representados por círculos azuis e vermelhos. Os círculos de cor azul são responsáveis em demonstrar os pontos de ida do ônibus e os vermelhos são os pontos de ônibus de retorno. Este tipo de resultado ajuda em entender as diversas formas de usar os pontos geográficos no mapa para visualizar um determinado local desejado.

FIG. 20 – Resultado das paradas de ônibus para rota Cidade SUS
FONTE – Dados da pesquisa

As paradas da rota Cidade passam por um ponto na rua Pç. 5 de Novembro (FIG. 21)

FIG. 21 – Resultado das paradas de ônibus para rota Cidade
FONTE – Dados da pesquisa

O resultado referente ao listar ruas tem por finalidade em uma determinada rota, listar todas as ruas que pertencem a rota escolhida no aplicativo RotaOnibus. Isso facilita na consulta de ruas em que o ônibus irá passar e se é a linha correta para o destino desejado (FIG. 22). Quando são listadas as ruas de uma determinada rota é possível visualizar o caminho que o ônibus percorre somente em uma determinada rua e visualizar no mapa este caminho específico (FIG. 23).

FIG. 22 – Resultado das listagem de ruas para rota Cidade SUS
FONTE – Dados da pesquisa

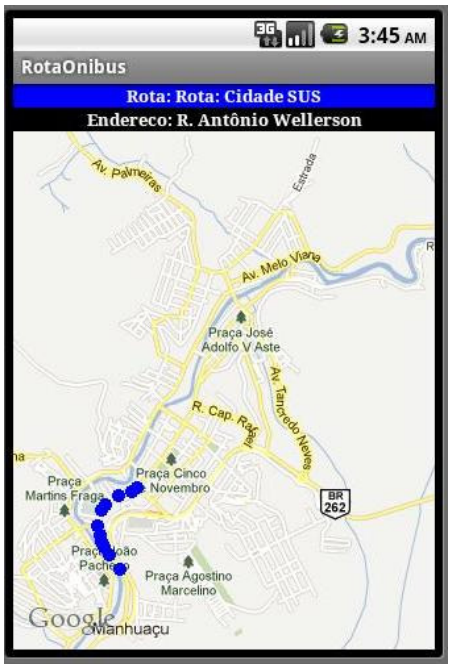


FIG. 23 – Resultado da rota da rua para Cidade SUS
FONTE – Dados da pesquisa

As ruas R. Etelvino Guimarães e R. Monsenhor Gonzalez pertence a rota Cidade SUS que não possui para a rota Cidade que passa pela Pç. 5 de Novembro (FIG. 24). Foi visualizado a rota da rua Pç. 5 de Novembro para apresentar a diferença da rota Cidade (FIG. 25).

FIG. 24 – Resultado das listagem de ruas para rota Cidade
FONTE – Dados da pesquisa

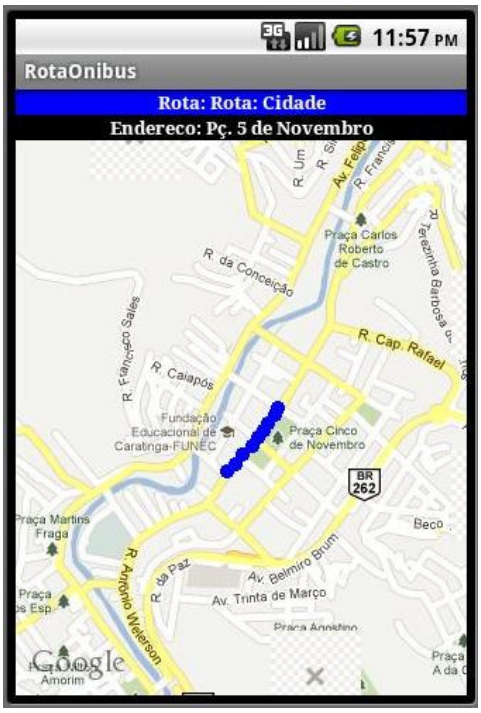


FIG. 25 – Resultado da rota da rua para Cidade
FONTE – Dados da pesquisa

O resultado do aplicativo foi baseado no conceito do SIG usando as ideologias de ponto 2D e polígonos para traçar uma rota, as paradas de ônibus e buscar as ruas referente a rota especificada.

5 – CONCLUSÃO

O presente trabalhado foi realizado com o intuito de criar um aplicativo utilizando a ideologia do SIG e suas funcionalidades. A motivação para a realização desse trabalho originou-se com a observância da grande quantidade de pessoas que utilizam o transporte coletivo para se locomoverem.

Através do aplicativo é possível obter informações de quais ruas compõe as rotas, a localização dos pontos nessas ruas e a rota completa por onde os ônibus transitam podendo ser visualizadas através de mapas. As funcionalidades do sistema são facilmente executadas através de menus e listas que compõe o aplicativo.

Foram realizados alguns testes com as informações geradas no banco de dados, onde são armazenadas as informações geográficas. Os resultados obtidos foram corretos dentro da proposta deste trabalho, proporcionando três tipos de saída de resultados, sendo: a rota completa do ônibus, todas as paradas que pertence a rota ordenadas por cores e as ruas que pertence à rota.

O acesso ao aplicativo para o público poderá ser disponibilizado através de um site da empresa de viação ou pela loja da Google conhecida como *Android Market*. Para a instalação do aplicativo é necessário obter um *smarthphone* ou um *tablet* que rode o sistema operacional *Android*.

6 – TRABALHOS FUTUROS

Após o desenvolvimento do aplicativo pode ser proposto aprimoramento e novas funcionalidades. Pode se destacar:

- A possibilidade de localizar o ônibus via GPS.
- Possibilidade de mostrar a distância restante para a chegada.
- Possibilidade de indicar estabelecimentos no mapa e qual ônibus passa por este local. Ex.: hospital, restaurante, escolas e outros.
- Informar o local de destino, traçar a rota do ponto onde o usuário estiver sendo localizado via GPS até a chegada automaticamente e indicar qual ônibus leva até o local determinado.

7 – REFERÊNCIAS

AQUINO, J. F. S. **Plataforma de desenvolvimento para dispositivos móveis.** Documento de trabalho. Rio de Janeiro: PUC/RIO. Disponível em: < http://www-di.inf.puc-rio.br/~endler/courses/Mobile/Monografias/07/Android-Juliana-Mono.pdf>. Acesso em: 3 set. 2011

BERTUCCI, J. L de O. **Metodologia básica para elaboração de trabalhos de conclusão de cursos:** ênfase na elaboração de TCC de Pós-graduação Lato Sensu. São Paulo: Atlas, 2009.

CÂMARA, G.; MONTEIRO, A. M. V.; FUCKS, S. D.; CARVALHO, M. S. **Análise espacial de dados geográficos**. Disponível em: <http://www.dpi.inpe.br/Gilberto/livro/analise/cap1-intro.pdf>. Acesso em: 11 ago. 2011

CÂMARA, G.; ORTIZ, M. J. **Sistemas de informação geográfica para aplicações ambientais e cadastrais: uma visão geral**. Disponível em: <http://www.dpi.inpe.br/gilberto/papers/analise.pdf>. Acesso em: 15 ago. 2011

GOOGLE. **TUTORIAL.** Disponível em:<http://developer.android.com/guide/basics/what-is-android.html>. Acesso 15 set. 2011

LECHETA, R. R. **Google Android** – Aprenda a criar aplicações para dispositivos móveis com o android SDK. 2. ed. São Paulo: Novatec, 2010.

LISBOA FILHO, J. **Projeto de banco de dados para sistemas de informação Geográfica.** Documento de trabalho. Disponível em: <ftp://ftp.cefetes.br/Cursos/Geomatica/Wellington/N21_SIG/ProjetodeBDparaSistemasdeInformacaoGeografica.pdf>. Acesso em: 11 ago. 2011

MACHADO, R. da S. F. **Modelagem e prototipação de uma aplicação LBS utilizando a plataforma android**. Documento de trabalho. Porto Alegre: UFRGS. Disponível em: http://www.lume.ufrgs.br/handle/10183/26356. Acesso em: 10 set. 2011

MORAES, R. M. **Sensoriamento remoto e classificação de imagens**. Documento de trabalho. Campus I – João Pessoa: UFPB. Disponível em: <http://www.de.ufpb.br/~ronei/procimagem/procimagem.htm>. Acesso em: 10 set. 2011

MACK, R. S. **Sistema de recomendação baseado na localização e perfil utilizando a plataforma android.** Documento de trabalho. Porto Alegre: UFRGS. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/28328/000767836.pdf?sequence=1>. Acesso em: 10 set. 2011

MUNDO VESTIBULAR. Disponível em: <http://www.mundovestibular.com.br/articles/23/1/PROJECOES-CARTOGRAFICAS/ Paacutegina1.html>. Acesso em: 03 set 2011

PEREZ, H. A. de M.; MOURA, A. M. de C.; TANAKA, A.K. **Extração de dados em sistemas de informações ambientais: arquitetura e esquemas de metadados.** Documento de trabalho. Disponível em: < http://mtc-m18.sid.inpe.br/col/dpi.inpe.br/vagner/2000/06.28.12.43/doc/006.pdf >. Acesso em: 09 set. 2011

QUEIROZ, G. R.; RIBEIRO, R. K. **Tutorial sobre Banco de Dados Geográficos. GeoBrasil 2006**. Instituto Nacional de Pesquisas Espaciais. Disponível em: < http://www.dpi.inpe.br/TutorialBdGeo_GeoBrasil2006.pdf>. Acesso em: 11 ago. 2011

RAMPELO, E. L. **Implementação de uma ferramenta de interface para um sistema de informações geográficas.** Documento de trabalho. Blumenau: Universidade Regional de Blumenau. Disponível em: < http://campeche.inf.furb.br/tccs/2002-II/2002-2emersonluisrampelottivf.pdf>. Acesso em: 20 ago. 2011

SILVA, E. de O. da**. Introdução a sistemas de informações geográficas.** Disponível em: <http://www.devmedia.com.br/post-1595-Introducao-a-Sistemas-de-Informacoes-Geograficas.html>. Acesso em: 20 nov. 2011

SILVA, L. E. P. da; BRACHT, E. C. Uma nova abordagem para o Cálculo de Balanço Hídrico Climatológico. **Revista Brasileira de Computação Aplicada** (ISSN 2176-6649), Passo Fundo, v. 2, n. 1, p. 2-16, mar. 2010. Disponível em: < http://www.upf.tche.br/seer/index.php/rbca/article/view/722>. Acesso em: 3 set. 2011

SQLITE. **TUTORIAL.** Disponível em:<http://www.sqlite.org/about.html>. Acesso 27 nov. 2011

TEIXEIRA, A. L. de A; CHRISTOFOLETTI, A. **Sistema de informação geográfica: Dicionário ilustrado.** São Paulo: Hucitec, 1997

ANEXO I: TABELAS DO BANCO DE DADOS

Tabela: rota

CREATE TABLE rota (rot_id INTEGER NOT NULL PRIMARY KEY ASC autoincrement, rot_descricao_rota TEXT);

Tabela: parada

CREATE TABLE parada (par_id INTEGER NOT NULL PRIMARY KEY ASC autoincrement, par_id_rota INTEGER NOT NULL, par_latitude DOUBLE(13,6), par_longitude DOUBLE(13,6), par_tipo INTEGER);
CREATE INDEX par_id_rota ON parada (par_id_rota);

Tabela: ruas

CREATE TABLE ruas (rua_id INTEGER NOT NULL PRIMARY KEY ASC autoincrement, rua_id_bairro INTEGER NOT NULL, rua_endereco TEXT);
CREATE INDEX rua_id_bairro ON ruas (rua_id_bairro);

Tabela: bairro

CREATE TABLE bairro (bai_id INTEGER NOT NULL PRIMARY KEY ASC autoincrement, bai_bairro TEXT);

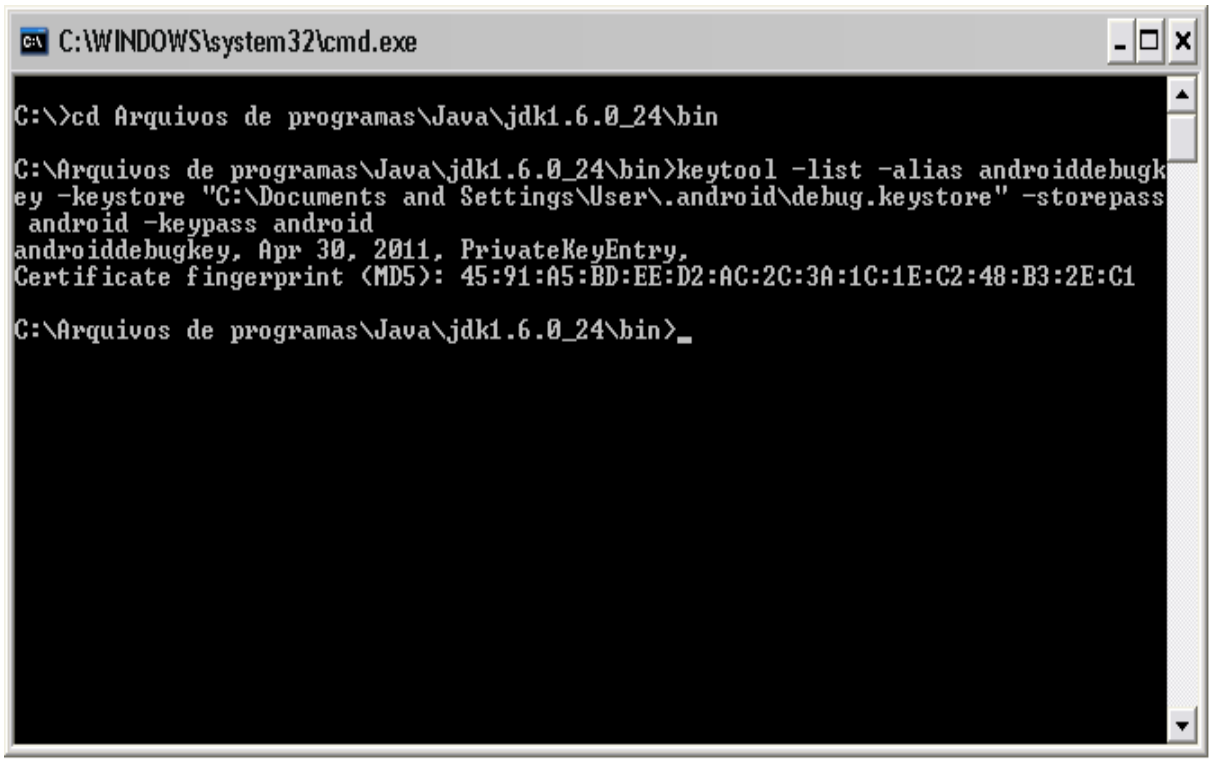
Tabela: rotarua

CREATE TABLE rotarua(rr_id INTEGER NOT NULL PRIMARY KEY ASC autoincrement, rr_id_rota INTEGER NOT NULL, rr_id_rua INTEGER NOT NULL);
CREATE INDEX rr_id_rota ON rotarua (rr_id_rota);
CREATE INDEX rr_id_rua ON rotarua (rr_id_rua);

Tabela: rotacoordenadas

```
CREATE TABLE rotacoordenadas (rcoo_id INTEGER NOT NULL PRIMARY KEY
ASC autoincrement, rcoo_id_rota_rua INTEGER NOT NULL, rcoo_latitude
DOUBLE(13,6), rcoo_longitude DOUBLE(13,6));
CREATE INDEX rcoo_id_rota_rua ON rotacoordenadas (rcoo_id_rota_rua);
```

ANEXO II: OBTENÇÃO DO MD5



ANEXO III: CÓDIGO FONTE DO ROTAONIBUS

listaRota.class

```
package br.tcc.rotaonibus;

public class listaRota {
        private int id;
    private String descricao_rota;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getRota() {
        return descricao_rota;
    }

    public void setRota(String descricao_rota) {
        this.descricao_rota = descricao_rota;
    }

    @Override
    public String toString() {
        return this.id + " - " + this.descricao_rota;
    }
}
```

RotaOnibusPrincipal.class

```
package br.tcc.rotaonibus;

import java.util.ArrayList;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
```

```
public class RotaOnibusPrincipal extends Activity {
   private final String NOME_BANCO = "rotaonibus.db";
   protected SQLiteDatabase db;
   private ListView Lista;
   private TextView Text;
   private int codLista;
   private String nomeRota;
   private boolean clicklongo = false;
   private final int DIALOG_LIST = 1;
   private final int DIALOG_SIM_NAO = 2;
   private final int Rota = 1;
   private final int Parada = 2;
   private final int RotaRua = 3;

   @Override
   protected Dialog onCreateDialog(int id) {
      switch (id) {
               case DIALOG_LIST:
                  return new AlertDialog.Builder(RotaOnibusPrincipal.this)
                     .setTitle("Menu Lista")
                     .setItems(R.array.menuListaRota, new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                           if (which == 0){
                                    Intent it = new Intent(RotaOnibusPrincipal.this,
RotaOnibusActivity.class);
                                    Bundle param = new Bundle();
                                    param.putInt("Parada", codLista);
                                    param.putInt("Tipo", Parada);
                                    param.putString("NomeRota", nomeRota);
                                    it.putExtras(param);
                                    startActivity(it);
                           } else {
                                    Intent it = new Intent(RotaOnibusPrincipal.this, RotaRua.class);
                                    Bundle param = new Bundle();
                                    param.putInt("ListaRua", codLista);
                                    param.putInt("Tipo", RotaRua);
                                    param.putString("NomeRota", nomeRota);
                                    it.putExtras(param);
                                    startActivity(it);
                           }
                        }
                     })
                     .create();
               case DIALOG_SIM_NAO:
                     return new AlertDialog.Builder(RotaOnibusPrincipal.this)
                        .setTitle("Deseja continuar?")
                        .setPositiveButton("Sim", new DialogInterface.OnClickListener() {
                           public void onClick(DialogInterface dialog, int whichButton) {

                           }
                        })
                        .setNegativeButton("Não", new DialogInterface.OnClickListener() {
                           public void onClick(DialogInterface dialog, int whichButton) {

                           }
                        })
                        .create();
               }
```

```
                return null;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        db = openOrCreateDatabase(NOME_BANCO, Context.MODE_PRIVATE, null);

        Text = (TextView) findViewById(R.id.text);
        Text.setText("ROTA DE ÔNIBUS (MANHUAÇU)");
        Lista = (ListView) findViewById(R.id.lista);

        final ArrayList<listaRota> arLista = new ArrayList<listaRota>();
        String[] colunas = new String[]{"rot_id", "rot_descricao_rota"};
        Cursor cs = null;
        cs = db.query("rota", colunas, null, null, null, null, null);

        while (cs.moveToNext()){
            listaRota lista = new listaRota();
            lista.setId(cs.getInt(cs.getColumnIndex("rot_id")));
            lista.setRota(cs.getString(cs.getColumnIndex("rot_descricao_rota")));
            arLista.add(lista);
        }
        cs.close();
        db.close();

        ArrayAdapter<listaRota> adaptadorLista = new ArrayAdapter<listaRota>(this,
android.R.layout.simple_list_item_1, arLista);
        Lista.setAdapter(adaptadorLista);

        Lista.setOnItemClickListener(new OnItemClickListener() {
                        public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                                if (!clicklongo){
                                        listaRota lista = arLista.get(position);
                                        codLista = lista.getId();
                                        nomeRota = lista.getRota();

                                        Intent it = new Intent(RotaOnibusPrincipal.this,
RotaOnibusActivity.class);
                Bundle param = new Bundle();
                param.putInt("Rota", codLista);
                param.putInt("Tipo", Rota);
                param.putString("NomeRota", nomeRota);
                it.putExtras(param);
                startActivity(it);
                                        //Toast.makeText(RotaOnibusPrincipal.this, "ID: " + codLista,
Toast.LENGTH_SHORT).show();
                                } else {
                                        clicklongo = false;
                                        onItemClick(parent, view, position, id);
                                }
                        }
                });

        Lista.setOnItemLongClickListener(new OnItemLongClickListener() {
                        public boolean onItemLongClick(AdapterView<?> parent, View view, int
position, long id) {
```

```
                                listaRota lista = arLista.get(position);
                                codLista = lista.getId();
                                nomeRota = lista.getRota();
                                showDialog(DIALOG_LIST);
                                //alertaDialogo alerta = new alertaDialogo(RotaOnibusPrincipal.this);
                                //alerta.showAlerta("Menu", new CharSequence[] {"Ponto Ônibus",
"Listas de Ruas"}).show();
                                clicklongo = true;

                                return true;
                        }
                });
    }
}
```

RotaRua.class

```
package br.tcc.rotaonibus;

import java.util.ArrayList;
import java.util.HashMap;
import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Toast;

public class RotaRua extends ListActivity{
        private final String NOME_BANCO = "rotaonibus.db";
        protected SQLiteDatabase db;
        public HashMap<String, String> map;
        public ArrayList<HashMap<String, String>> list;
        private String nomeRota;
        private int Tipo;

        @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        db = openOrCreateDatabase(NOME_BANCO, Context.MODE_PRIVATE, null);

        list = new ArrayList<HashMap<String, String>>();

        nomeRota = "Rota: " + getIntent().getExtras().getString("NomeRota");
        Tipo = getIntent().getExtras().getInt("Tipo");
        int codRota = getIntent().getExtras().getInt("ListaRua");

        String sql = "SELECT rotarua.rr_id, ruas.rua_id, ruas.rua_endereco, bairro.bai_bairro FROM ruas
INNER JOIN rotarua ON rotarua.rr_id_rua = ruas.rua_id AND rotarua.rr_id_rota = " + codRota + "
INNER JOIN bairro ON bairro.bai_id = ruas.rua_id_bairro";
        Cursor cs = db.rawQuery(sql, null);
```

```
        while (cs.moveToNext()){
            //Log.i("Ruas", "Rua: " + cs.getString(cs.getColumnIndex("rua_endereco")).toString());
            map = new HashMap<String, String>();

            String codRRid = Integer.toString(cs.getInt(cs.getColumnIndex("rr_id")));
            map.put("RRID", codRRid);
            String codRua = Integer.toString(cs.getInt(cs.getColumnIndex("rua_id")));
            map.put("ID", codRua);
            map.put("Endereco", cs.getString(cs.getColumnIndex("rua_endereco")).toString());
            map.put("Bairro", cs.getString(cs.getColumnIndex("bai_bairro")).toString());
            list.add(map);
        }
        cs.close();
        db.close();

        //ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.itemlista, rotarua);
        //setListAdapter(adapter);
        String[] from = new String[] {"Endereco", "Bairro"};
        int[] to = new int[] {android.R.id.text1, android.R.id.text2};
        setListAdapter(new SimpleAdapter(this, list, android.R.layout.two_line_list_item,  from, to));
            }

            @Override
            public void onBackPressed() {
               finish();
               Log.i("Voltar", "Foi feito a volta e o finish();");
               return;
            }

            @Override
            protected void onListItemClick(ListView l, View v, int position, long id){
                    super.onListItemClick(l, v, position, id);
                    //Modo Antigo
                    //Object o = this.getListAdapter().getItem(position);
                    //String itemlista = o.toString();
                    //HashMap<String, String> itemlista = list.get(position);
                    //Toast.makeText(this, "Você selecionou: " + itemlista.get("Bairro"),
Toast.LENGTH_LONG).show();

                    //Modo para pegar o bairro com a lista selecionada
                    //item = list.get(position);
                    //String codItem = item.get("Bairro").toString();
                    //Toast.makeText(this, "Você selecionou: " + codItem, Toast.LENGTH_LONG).show();

                    //Irá pegar o ID da lista selecionada
                    map = list.get(position);
                    int codRRid = Integer.parseInt(map.get("RRID"));
                    int codRua = Integer.parseInt(map.get("ID"));
                    String endereco = map.get("Endereco");
                    //Toast.makeText(this, "RRID: " + codRRid + " ID: " + codRua,
Toast.LENGTH_LONG).show();
                    Intent it = new Intent(RotaRua.this, RotaOnibusActivity.class);
                    Bundle param = new Bundle();
                    param.putInt("RotaRua", codRRid);
                    param.putInt("Tipo", Tipo);
                    param.putString("NomeRua", endereco);
                    param.putString("NomeRota", nomeRota);
                    it.putExtras(param);
                    startActivity(it);
            }
```

```
}
```

BolaOverlay.class

```
package br.tcc.rotaonibus;

import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;

public class BolaOverlay extends Overlay{
	private int cor;
	private Paint paint = new Paint();
	private GeoPoint geoPoint;

	public BolaOverlay(GeoPoint geoPoint, int cor) {
		this.geoPoint = geoPoint;
		this.cor = cor;
	}

	@Override
	public void draw(Canvas canvas, MapView mapView, boolean shadow) {
		super.draw(canvas, mapView, shadow);
		if (geoPoint != null) {
			paint.setColor(cor);
			Point ponto = mapView.getProjection().toPixels(geoPoint, null);
			canvas.drawCircle(ponto.x, ponto.y, 5, paint);
		}
	}

	public void setGeoPoint(GeoPoint geoPoint){
		this.geoPoint = geoPoint;
	}
}
```

RotaOnibusActivity.class

```
package br.tcc.rotaonibus;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

public class RotaOnibusActivity extends MapActivity {
```

```
/** Called when the activity is first created. */
        private final String NOME_BANCO = "rotaonibus.db";
        protected SQLiteDatabase db;
       private BolaOverlay bola;
        private int codRota;
        private String nomeRota;
        private String sql;
        private double latitude;
        private double longitude;
        private int tipoParada;
        private Cursor cs;
        private GeoPoint point;
        private int latitudeE6;
        private int longitudeE6;
        private Boolean parada = false;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);

   db = openOrCreateDatabase(NOME_BANCO, Context.MODE_PRIVATE, null);

   setContentView(R.layout.maparota);
   MapView map = (MapView) findViewById(R.id.mapa);
   map.setStreetView(true);
   map.setClickable(true);
   MapController controlador = map.getController();
   controlador.setZoom(15);

   nomeRota = "Rota: " + getIntent().getExtras().getString("NomeRota");
   TextView texto = (TextView) findViewById(R.id.texto);
   texto.setText(nomeRota);
   TextView legenda  = (TextView) findViewById(R.id.legenda);

   int Tipo = getIntent().getExtras().getInt("Tipo");
   switch (Tipo) {
                        case 1:
                                legenda.setText("");
                                parada = false;

                                codRota = getIntent().getExtras().getInt("Rota");
                                sql = "SELECT rotacoordenadas.rcoo_id_rota_rua,
rotacoordenadas.rcoo_latitude, rotacoordenadas.rcoo_longitude FROM rotacoordenadas INNER
JOIN rotarua ON (rotarua.rr_id = rotacoordenadas.rcoo_id_rota_rua AND rotarua.rr_id_rota = " +
codRota + ")";
                                Log.i("SQL", "SQL: " + sql);

                                break;
                        case 2:
                                legenda.setText("Legenda: Ponto Azul = Ida | Ponto Verm. = Volta");
                                parada = true;

                                codRota = getIntent().getExtras().getInt("Parada");
                                sql = "SELECT parada.par_id, parada.par_latitude,
parada.par_longitude, parada.par_tipo FROM parada WHERE(parada.par_id_rota = " + codRota + ")";
                                Log.i("SQL", "SQL: " + sql);

                                break;
                        case 3:
```

```
                                        String endereco = "Endereco: " +
getIntent().getExtras().getString("NomeRua");
                                        legenda.setText(endereco);
                                        parada = false;

                                        codRota = getIntent().getExtras().getInt("RotaRua");
                                        sql = "SELECT rotacoordenadas.rcoo_latitude,
rotacoordenadas.rcoo_longitude FROM rotacoordenadas WHERE(rotacoordenadas.rcoo_id_rota_rua
= " + codRota + ")";
                                        Log.i("SQL", "SQL: " + sql);
                                        break;
                                default:
                                        break;
                        }

        if (!parada) {
                        cs = db.rawQuery(sql, null);

                        while (cs.moveToNext()){
                            latitude = cs.getDouble(cs.getColumnIndex("rcoo_latitude"));
                            longitude = cs.getDouble(cs.getColumnIndex("rcoo_longitude"));

                            latitudeE6 = (int) (latitude*1E6);
                           longitudeE6 = (int) (longitude*1E6);

                           point = new GeoPoint(latitudeE6, longitudeE6);
                           bola = new BolaOverlay(point, Color.BLUE);

                           map.getOverlays().add(bola);
                        }
                        cs.close();
                        db.close();
        } else {
            cs = db.rawQuery(sql, null);

                        while (cs.moveToNext()){
                            latitude = cs.getDouble(cs.getColumnIndex("par_latitude"));
                            longitude = cs.getDouble(cs.getColumnIndex("par_longitude"));
                            tipoParada = cs.getInt(cs.getColumnIndex("par_tipo"));

                            latitudeE6 = (int) (latitude*1E6);
                           longitudeE6 = (int) (longitude*1E6);

                           point = new GeoPoint(latitudeE6, longitudeE6);
                           if (tipoParada == 0){
                            bola = new BolaOverlay(point, Color.BLUE);
                           }else{
                            bola = new BolaOverlay(point, Color.RED);
                           }
                           map.getOverlays().add(bola);
                        }
                        cs.close();
                        db.close();
        }
        //map.getController().setCenter(point);

        map.setBuiltInZoomControls(true);
    }

        @Override
```

```
        protected boolean isRouteDisplayed() {
                // TODO Auto-generated method stub
                return false;
        }

        @Override
        public void onBackPressed() {
           finish();
           Log.i("Voltar", "Foi feito a volta e o finish();");
           return;
        }
}
```

ANEXO IV: XML DO ROTAONIBUS

Itemlista.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    android:textSize="18sp">
</TextView>
```

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:id="@+id/text"
            android:padding="10dp"
            android:textSize="18sp"
            android:textStyle="italic"
            android:typeface="serif"
            android:background="@color/Blue"
            android:textColor="@color/White"
        />
        <ListView
            android:id="@+id/lista"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1">
        </ListView>
</LinearLayout>
```

maparota.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
```

```
            android:id="@+id/texto"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="13sp"
    android:textStyle="bold"
    android:typeface="serif"
    android:background="@color/Blue"
    android:textColor="@color/White"
    android:gravity="center"
/>
<TextView
            android:id="@+id/legenda"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="13sp"
    android:textStyle="bold"
    android:typeface="serif"
    android:textColor="@color/White"
    android:gravity="center"
/>
<com.google.android.maps.MapView
            android:id="@+id/mapa"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:clickable="true"
            android:apiKey="0mjcBMVEYjE6cPnuQm30yPXVzrOBaeZ_AIG8hcw"
/>
</LinearLayout>
```

array.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string-array name="menuListaRota">
      <item>Ponto Ônibus</item>
      <item>Lista Ruas</item>
   </string-array>
</resources>
```

string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="hello">Hello World, RotaOnibusActivity!</string>
        <string name="app_name">RotaOnibus</string>
        <string name="selectedEntry" />
        <color name="White">#FFFFFF</color>
        <color name="Black">#000000</color>
        <color name="Blue">#0000FF</color>
        <string name="mostraPonto">Pontos de Onibus</string>
</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.tcc.rotaonibus"
    android:versionCode="1"
    android:versionName="1.0">

        <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
        <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
        <uses-permission android:name="android.permission.INTERNET" />

  <application android:icon="@drawable/icon" android:label="@string/app_name">
        <uses-library android:name="com.google.android.maps" />

      <activity android:name=".RotaOnibusPrincipal"
            android:label="@string/app_name">
          <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
      </activity>
                  <activity android:name=".RotaOnibusActivity" />
                  <activity android:name=".RotaRua"/>
  </application>
  <uses-sdk android:minSdkVersion="8" />
</manifest>
```