UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
FACULDADE DE CIÊNCIAS EXATAS
CURSO DE SISTEMAS DE INFORMAÇÃO

ALBERES DO ESPIRITO SANTO

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA EXIBIR ROTAS, A PARTIR DO MAPA DO PROJETO OPENSTREETMAP

Diamantina
2016

ALBERES DO ESPIRITO SANTO

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA EXIBIR ROTAS, A PARTIR DO MAPA DO PROJETO OPENSTREETMAP

Trabalho apresentado a Faculdade de Ciências Exatas, como parte das exigências para conclusão do Curso de Bacharelado em Sistemas de Informação, da Universidade Federal dos Vales do Jequitinhonha e Mucuri.
Orientador: Prof. Dr. Alessandro Vivas Andrade

Diamantina

2016

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me dá forças para alcançar meus objetivos, meu professor orientador Alessandro Vivas pela compreensão, apoio e incentivo, a meus familiares por estarem sempre ao meu lado, em especial o meu irmão Ulisses pela parceira e a minha noiva Catharine que esta sempre ao meu lado em todos os momentos.

“Que os nossos esforços desafiem as possibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que pareciam impossível.”

Charles Chaplin

RESUMO

Os serviços de visualização de mapas são aprimorados a cada dia, a fim de proporcionar aos usuários qualidade, rapidez e exatidão nas rotas, caminhos, pontos turísticos ou áreas disponíveis nesses. Dentre esses serviços, vem crescendo e se destacando atualmente o OpenStreetMap – um serviço gratuito de mapeamento colaborativo, por ser robusto e de código aberto. Qualquer indivíduo pode editar o mapa principal, acrescendo ou denominando áreas, ruas, bairros, trilhas e disponibilizar para os demais usuários do serviço. Para interagir com esse serviço, aplicações são criadas e trabalham sob esses mapas, informando aos usuários as melhores rotas, pontos destacáveis de regiões especificas dentre outras informações. Esse projeto também objetiva cartografar (mapear) as ruas, bairros e pontos destacáveis de um município ainda não mapeado pelo OpenStreetMap, transferir para o seu mapa principal e disponibilizar essas informações para os usuários desse serviço. Simultaneamente desenvolver um aplicativo móvel para a plataforma Android que interage com esse mapa submetido e informa a melhor rota entre pontos distintos informados pelo usuário, definindo e atualizando sua localização para que a rota traçada no mapa seja seguida.

Palavras-chave: Android, Aplicativos, Dispositivos Móveis, Geolocalização, GPS, Mapa, OpenStreetMap.

ABSTRACT

Map display services are improved daily in order to provide users with quality, speed and accuracy on the routes, paths, sightseeing or other areas available. Among these growing services, an app called OpenStreetMap has been gainning much populatiry lately. It is a free service of collaborative mapping that is both robust and open source. Anyone can edit the main map, adding or naming areas, streets, neighborhoods, trails and it also makes this information available to other users of the service. To interact with this service, applications are created to work under these maps and it informs users of the best routes, points of interests of the specific regions, and other important information. This mapping project aims to map the streets, neighborhoods, and points of interest of municipalities not yet mapped by OpenStreetMap. It uses the information in your map to transfer to other users of that service. This app simultaneously develops a mobile application for the Android platform that interacts with this subject map and informs the best routes between different distinct points entered by the user, defining and updating your location so that the route drawn on the map can be followed.

Keywords: Android, Applications, Geolocation, GPS, Map, Mobile, OpenStreetMap.

LISTA DE ILUSTRAÇÕES E TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADT: Android Development Tools
API: Application Programming Interface
AVD: Android Virtual Device
HTTP: Hypertext Transfer Protocol
IDE: Integrated Development Environment
JDK: Java Development Kit
OSRM: Open Source Routing Machine
PDA: Personal Digital Assistants
SDK: Software Development Kit
SLF4J: Simple Logging Facade for Java
XML: Extensible Markup Language
OSM: OpenStreetMap

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1- Apresentação

Desde que os serviços de pesquisa e visualização de mapas foram criados e aprimorados muitas das cidades do mundo passaram a contar com essa poderosa ferramenta para identificar locais, rotas e caminhos sejam entre cidades, bairros ou ruas. No entanto as pequenas cidades ainda não contam com esses serviços totalmente implementados. A cidade de Datas em Minas Gerais, por exemplo, ainda não tem suas ruas totalmente cartografadas ou mapeadas por nenhum serviço de pesquisa e visualização de mapas, sendo os mais conhecidos: Google Maps e o OpenStreetMap. Esse último, o OpenStreetMap, vem crescendo gradativamente, por ser um projeto de código aberto e bastante robusto.

Para que essa deficiência seja sanada, tal trabalho visa mapear as ruas da cidade, transferir para o mapa principal do OpenStreetMap e simultaneamente desenvolver um aplicativo móvel, mais precisamente na plataforma Android, o qual interage com esse mapa. O aplicativo busca pontos no mapa, (origem e destino) definidos pelo usuário e exibe a melhor rota entre esses pontos, além de mostrar no mapa pontos turísticos e destacáveis da cidade.

1.2- Objetivos deste trabalho

O trabalho destaca funcionalidades do OpenStreetMap, seu sistema de mapeamento e do Sistema Operacional móvel Android, demonstra como podem ser integradas essas duas plataformas e assim cria uma aplicação que gera a rota entre dois pontos distintos de um mapa de uma determinada região. Nesse caso a região pré-estabelecida é a cidade de Datas/MG, a qual será mapeada via OpenStreetMap. Através de um satélite artificial, são coletadas informações geográficas para montagem do mapa das ruas e bairros da cidade, sendo que para denominações dessas áreas, foi disponibilizado pelo município o mapa desenhado pelo engenheiro do Município.

Segundo Teresa Gallotti (2008): “o satélite artificial, como o próprio nome diz, é um equipamento construído pelo homem e, dependendo da finalidade, desloca-se

em órbita da Terra ou de outro astro registrando imagens.” Portanto através dessas imagens registradas pelo satélite, é possível localizar a posição em termos de latitude e longitude da terra e assim consegue-se mapear pontos específicos.

> “A latitude é a distância ao Equador medida ao longo do meridiano de Greenwich. Esta distância mede-se em graus, podendo variar entre 0º e 90º para Norte(N) ou para Sul(S). A longitude é a distância ao meridiano de Greenwich medida ao longo do Equador. Esta distância mede-se em graus, podendo variar entre 0º e 180º para Leste(E) ou para Oeste(W)” (QUOOS, 2014, p. 01).

O mapa da região de Datas/MG em agosto de 2014, mostra como a cidade não era mapeada pelo OpenStreetMap (figura 01).

Figura 01 – Mapa da região de Datas no OpenStreetMap

Fonte: OPENSTREETMAP.ORG, 2014.

CAPÍTULO 2- PROJETO OPENSTREETMAP

No início do segundo milênio, as informações geográficas ficavam mantidas sob domínio dos órgãos geográficos nacionais, estaduais ou municipais, portanto esses órgãos que controlavam todo o acesso aos documentos referentes a essas informações. Com o avanço tecnológico, esse cenário foi se modificando tornando possível confeccionar esses dados geoespaciais em outros formatos.

Segundo o INDE - Infraestrutura Nacional de Dados Espaciais (2011), dados geoespaciais são dados “que se distinguem essencialmente pela componente espacial, que associa a cada entidade ou fenômeno uma localização na terra , traduzida por sistema geodésico de referência, em dado instante ou período de tempo, podendo ser derivado, entre outras fontes das tecnologias de levantamento, inclusive as associadas a sistemas globais de posicionamento apoiados por satélites, bem como de mapeamento ou de sensoriamento remoto.”, e a partir dos anos 2000 passaram a ser produzidos computacionalmente. Foi quando empresas surgiram com o propósito de produzir softwares com diversas finalidades para atender à essas novas exigências.

Segundo Tanenbaun (2002), software são instruções que controlam o hardware de modo a realizar tarefas determinadas por um algoritmo. Os softwares para essa área trabalham em conjunto com a parte física do computador a fim de aprimorar o processamento dessas informações geoespaciais o que chamamos de geoprocessamento. Contudo, isso ocasionou a impulsão das indústrias o que trouxe vários formatos de dados, causando uma dificuldade em importar e exportar dados entre softwares diferentes. Para Craglia (2008), as organizações internacionais então, passaram a promover a “interoperabilidade dos serviços para ver, acessar e integrar informações geográficas.” E então, têm-se desenvolvido tecnologias que incluem os usuários da internet no processo de produção de mapas o que fez surgir novos termos de caráter mundial: Mapeamento colaborativo e informações geográficas voluntárias.

(Goodchild e Li 2012) definem as Informações geográficas Voluntárias como “uma versão de crowdsourcing na qual pessoas do público geral criam e contribuem com feições (features) e fatos georreferenciados a respeito da superfície da Terra

para websites. Crowdsourcing refere-se ao processo de edição colaborativo a partir do público.

A criação dos vários globos virtuais ou geonavegadores como Google Earth[1], Google Maps[2], ArcGIS Explorer[3], Bing maps[4], Openstreetmap[5], entre outros, utilizam massivamente imagens de satélite e fotografias aéreas na apresentação de seus mapas, mas, não se igualam aos serviços de mapeamento modernos mais recentes (CRAGLIA et al., 2008). Contudo ambos têm a dependência de outros avanços tecnológicos que alimentem essas informações geográficas. Esses navegadores ou globo virtuais proporcionam a fácil visualização do terreno disponibilizado: ruas, estradas, trilhas, e imagens aéreas ou do próprio satélite, a qualquer cidadão, além de possibilitar a esses a colaboração e inserção de informações.

As duas principais ferramentas para a colaboração e inserção de informações (mapeamento colaborativo), são: Google Earth (earth.google.com) e o OpenStreetMap (openstreetmap.org), as quais permitem que cidadãos contribuam voluntariamente com seus conhecimentos sobre determinadas áreas.

O Projeto OpenStreetMap cria e disponibiliza gratuitamente dados geográficos como mapas de cidades, estradas que podem ser utilizados por qualquer cidadão e para qualquer fim, desde que dados os créditos ao OpenStreetMap. O projeto foi iniciado porque a maioria dos mapas que parecem ser gratuitos possui restrições legais ou técnicas, limitando a sua utilização e impedindo o uso de forma criativa e produtiva.

Em maio de 2015, a Osmfoundation (2015) divulgou que são mais de 600.000 contribuidores do OpenStreetMap, que criam dados geográficos de diversas formas. A Osmfoundation, segundo o openstreetmap, é uma fundação sem fins lucrativos cujo objetivo é apoiar e permitir o desenvolvimento de dados geoespaciais livremente reutilizáveis. Como o próprio nome sugere, ele está intimamente ligado com o projeto OpenStreetMap, embora sua constituição não impeça de apoiar outros projetos.

[1] https://www.google.com/earth/
[2] https://www.google.com.br/maps/@-18.2330007,-43.6132984,15z
[3] https://www.arcgis.com/features/index.html
[4] https://www.bing.com/mapspreview?cc=br
[5] http://www.openstreetmap.com.br/

A Microsoft, contribuiu para o projeto OpenStreetmap cedendo as imagens do satélite captadas pelo serviço Bing Maps, cuja função permite a visualização de fotos de praticamente toda a superfície do planeta, produzidas por satélites que orbitam a Terra.

O OpenStreetMap provê ferramentas online de edição dos mapas, assim as informações geográficas captadas, podem ser transferidas para o banco de dados atualizado diariamente.

Segundo Korth (1994), um banco de dados é uma coleção de dados relacionados, representando informações sobre um domínio específico. Esses dados são então processados e aparecem no mapa online em poucos minutos.

O valor principal do OpenStreetMap é que esses mapas e os dados que os geram, podem ser publicados livremente em sites, utilizados em aplicações como nesse trabalho, e impressos e copiados sem qualquer tipo de restrição, permitindo ate mesmo o uso comercial.

A essência do projeto é cartografar as vias, bairros, estradas, trilhas etc. gerando no final um mapa da região explorada. O projeto se desenrola em duas etapas. A primeira consiste em coletar dados geográficos. Para isso basta deslocar-se, seja de bicicleta, carro ou a pé, usar preferencialmente um GPS e fazer o mapeamento da região do deslocamento ou mesmo com dados em mãos, oriundos de outras fontes. Os dados levantados, esses devem ser acrescidos a uma base de dados do OpenStreetMap com uma das ferramentas de edição disponibilizadas pela própria entidade ou por pessoas ou empresas ligadas ao projeto. A próxima etapa é acrescentar as vias, os nomes de rua, códigos postais e tudo o que for interessante a ser representado no mapa. Para qualquer alteração que o mapa principal sofra, deve ser possível identificar de onde vieram essas informações, por isso um cadastro no site do openstreetmap é exigido.

2.1- Estrutura

Os serviços básicos do projeto OpenStreetMap e os direitos legais são geridos pela OpenStreetMap Foundation (OSMF), cujo capital provém da associação de novos membros, de doações, e de patrocínios. O mapa é disponibilizado sob a licença Open Data (dados abertos).

De acordo com o Opendefinition, dados são abertos quando qualquer pessoa pode livremente usá-los, reutilizá-los e redistribuí-los, estando sujeito a, no máximo, a exigência de creditar a sua autoria e compartilhar pela mesma licença. Portanto, permite-se o uso dos dados para fins comerciais desde que haja referência ao projeto OpenStreetMap e que os dados e as alterações feitas neles também sejam disponibilizadas sob a mesma licença ou outra compatível.

Segundo o OpenStreeMap (2014), os principais softwares e serviços executados em seus servidores são:

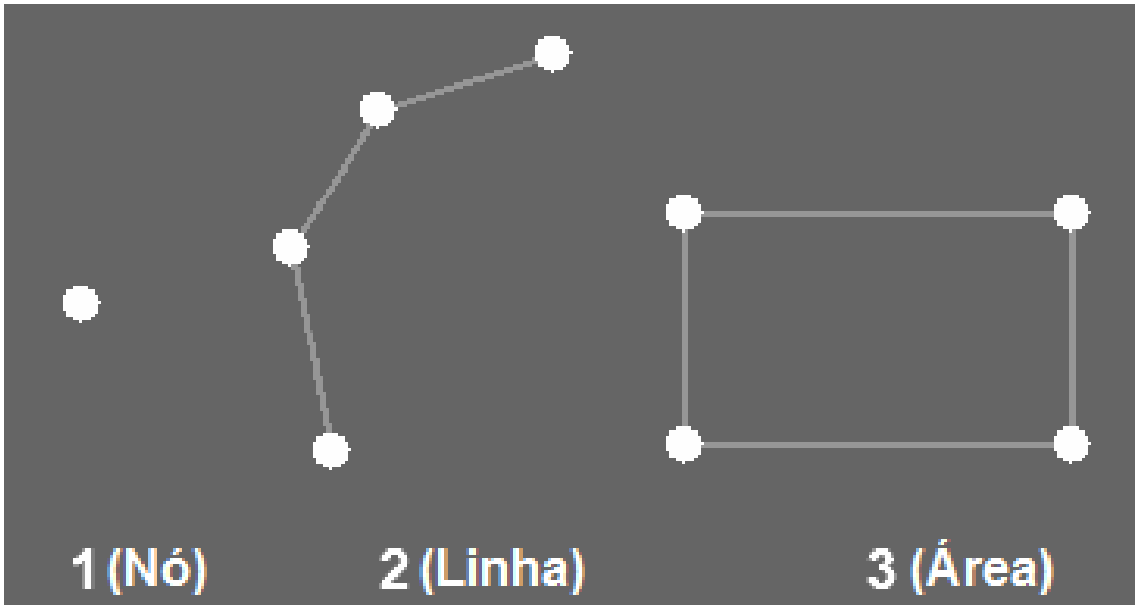
- Mapnik: programa executado nos servidores do OpenStreetMap que efetivamente desenha o mapa padrão e alguns outros básicos (mapa das rotas de transporte público, mapa ciclístico).
- ITO Map: serviço que fornece representações alternativas do mapa OpenStreetMap para diversas finalidades, geralmente expondo informações que não aparecem no Mapnik.
- Open Source Routing Machine (OSRM): programa que faz cálculo de rotas, com serviço independente atualizado diariamente e usável no Android através do OSMBonusPack.
- Nominatim: programa que encontra as coordenadas geográficas (geocoding) de endereços e pontos de interesse (POIs), usado na própria página principal do OpenStreetMap e nas buscas do OSRM.
- OpenLayers: serviço que permite integrar mapas do OpenStreetMap com outros websites
- Notes: serviço para registrar erros nos mapas (corrigidos mais tarde por mapeadores interessados).

2.2- Representação de Dados

Na Figura 02 estão representados, de forma simples, os dados mapeados, conforme a própria entidade define. Cada elemento está descrito por um conjunto de tags, sendo cada uma associada a um valor, e, há elementos de apenas 4 tipos, sendo eles os nós; as linhas; as áreas; e as relações. Os nós servem para marcar localizações específicas, são caracterizados pelos pontos individuais. As linhas simbolizam as estradas, caminhos, rios, enfim, são os percursos. Cada linha é

composta por no mínimo 2 nós, podendo chegar a centenas ou milhares deles. As áreas são definidas pelas linhas, cujo segmento estão todos ligados, dando uma noção de “espaço fechado”. Simbolizam, por exemplo, parques de campismo, áreas reservadas, lagos, entre outros. Já as relações, agregam uma série de elementos, podendo estes, ser nós, linhas e/ou linhas fechadas. Simbolizam as rotas, restrições de viragem e áreas que não são contíguas.

Figura 02 – Representação de dados



Fonte: OPENSTREETMAP.ORG, 2014.

As tags são usadas para descrever informações como o tipo de objeto (restaurante, rua, lagoa, casa, trilha etc.) e seus detalhes mais relevantes (o endereço, se o acesso é restrito, se é iluminado, etc.). O nome e o valor de cada tag não têm qualquer restrição imposta pelo sistema, mas o OpenStreetMap usa um conjunto de convenções propostas e votadas através deste wiki, o qual serve de referência para aplicação das novas práticas. Isso permite a evolução do mapa assim que surgirem novas necessidades.

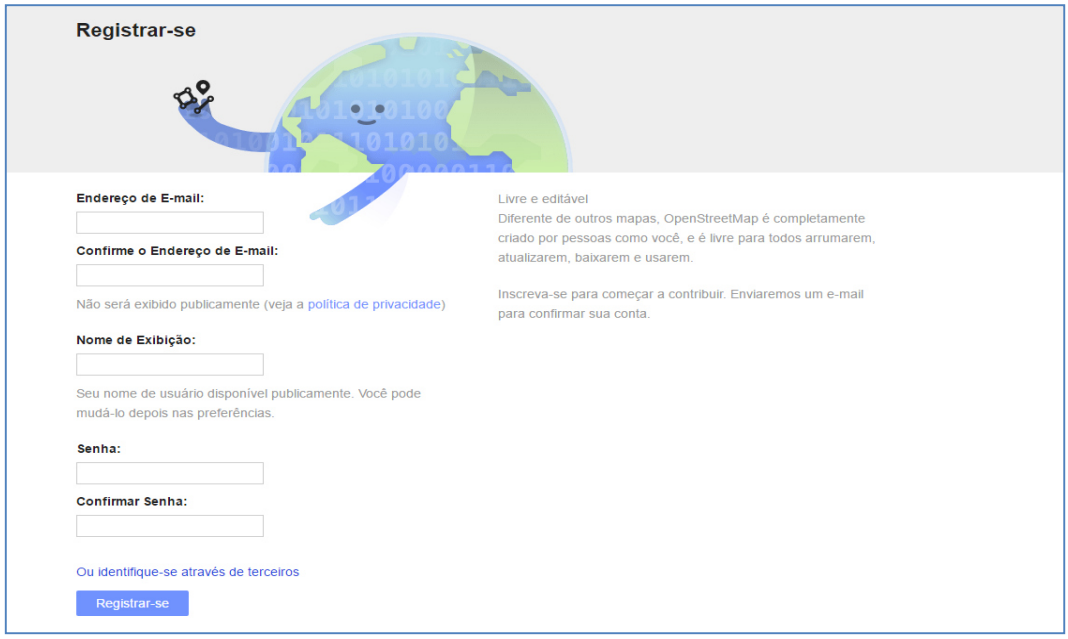
2.3- Editores

Os editores adotam as novas convenções gradualmente. Devido a isso muitos são os editores listados pelo OpenStreetMap, porém, dentre os mais utilizados, a organização aponta uma lista de 8 (oito) editores.

De acordo com o banco de dados do OpenStreetMap (2016), o editor mais fácil de utilizar é o Potlatch, que consiste em um Flash encontrado no ‘Edit’ tab do mapa principal, este, contribui na realização de edições rápidas, podendo salvar as mudanças para o servidor OpenStreetMap instantaneamente no modo ‘Live’. O editor mais popular é o JOSM, sendo ele uma aplicação Java ‘stand-alone, permitindo ‘fluid zooming’, panning’, e edição dos dados armazenados localmente. A Figua 04, localizada na página 22 deste trabalho, mostra a interface o JOSM. O OSM2Go, atualmente suportado pela ‘Nokia Internet Tablet’, é uma solução de ‘mobile mapping’. Como editor para dispositivos Android têm-se o Vespucci, muito simples de ser utilizado. O Amenity Editor é um editor ‘online’ usado para criar e editar ‘nodes’ e POIs. O Mapzen POI Collector é um editor para iPhones, este, está em uso por um grupo fechado de teste-alfa. Temos ainda, o iPhone Little OpenStreetMap Editor (iLOE), outro editor para iPhones, este, se encontra disponível no ‘App Store’.

É importante salientar que antes de qualquer alteração deve-se registrar no Projeto OpenStreetMap através do site da entidade. A figura 03 indica os campos a preencher no cadastro inicial do site.

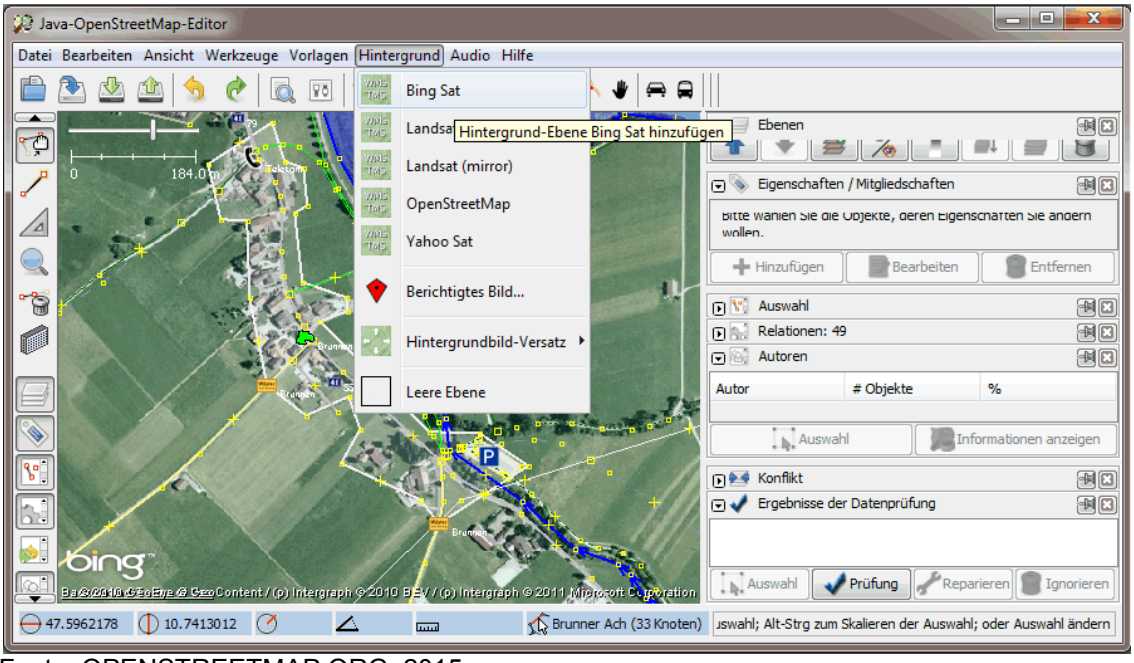
Figura 03 – Cadastro no site



Fonte: OPENSTREETMAP.ORG, 2014.

Para melhor entender como funciona cada um dos editores e como submeter os mapas usando-os, vide ao site do OpenStreetMap, que detalha como cada ferramenta pode ser usada, tutorial de instalação, dentre outras informações, através dos editores de mapa. Além desses softwares destacados na tabela 01, ainda existe a versão online, via browser disponível no próprio site da organização. Para efetuar o mapeamento nesse projeto, foi usada a forma online, via browser, uma vez que essa é a forma mais rápida e interativa para tal processo porque é possível acessar e alterar o mapa principal em tempo real. O passo a passo de como efetuar as alterações usando essa modalidade é descrito no capitulo 4.

Figura 04 – JOSM (Java OpenStreetMap Editor)



Fonte: OPENSTREETMAP.ORG, 2015.

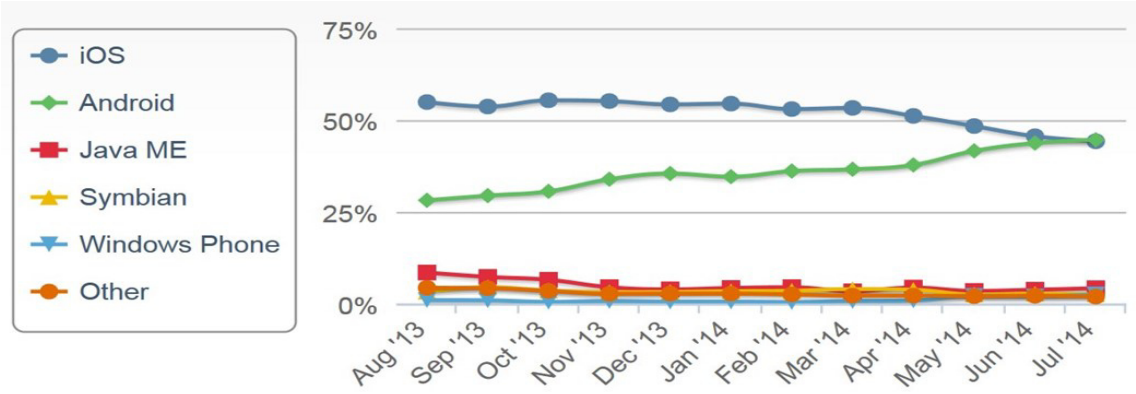
CAPÍTULO 3 - DISPOSITIVOS MÓVEIS

Cada vez mais as pessoas de todo o mundo necessitam de acesso a informações pessoais e corporativas, independentemente do momento ou lugar. Para que essas informações sejam acessadas com maior facilidade as empresas investem constantemente em desenvolvimentos tecnológicos. Dentre as novas tecnologias, uma vive uma ascensão contínua, o Smartphone.

> “Smartphone é, em tradução literal, "um telefone inteligente". Ele é a evolução do celular. A capacidade de realizar e receber chamadas é “apenas um detalhe” para este aparelho, que permite uma infinidade de possibilidades. Os modelos são muitos, com os mais diversos tipos e funções que você pode imaginar.” (BARROS, 2012, p. 01)

Os smartphones se baseiam em funções executadas por computadores. Não têm o hardware potente de um computador, mas são muito mais completos que um celular comum, apresentando inúmeras outras funções não executadas por esses celulares. As principais tecnologias de comunicação em somente um local, essa é a proposta do smartphone; internet, GPS, e-mail, SMS, mensageiro instantâneo e aplicativos para muitos fins. As principais marcas de smartphones mundialmente conhecidas são: Samsung, Apple, Motorola, LG, Nokia e BlackBerry. A figura 05 mostra as principais plataformas para dispositivos móveis, os líderes são o Android, do Google; iOS, da Apple; e o Windows Phone, da Microsoft.

Figura 05- Principais plataformas para aplicativos móveis

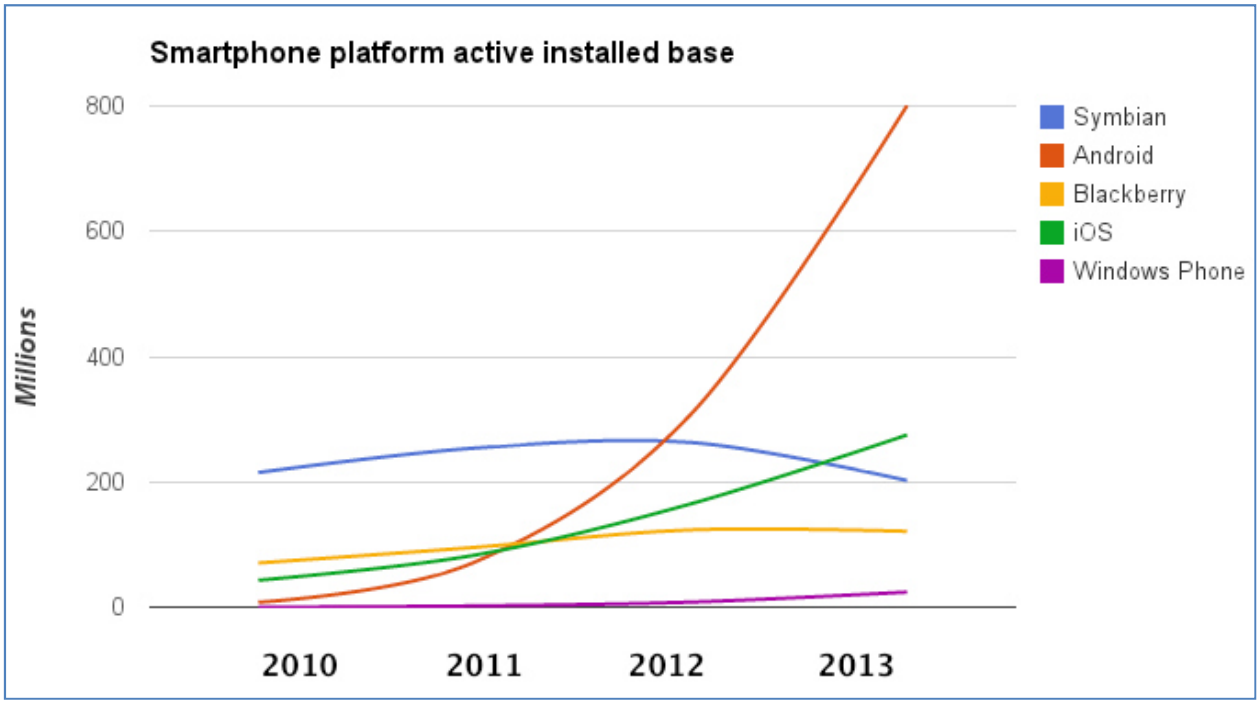


Fonte: TECMUNDO, 2014.

Destaca-se a ascensão da plataforma Android, passando a ser o sistema operacional móvel mais usado nos aparelhos mundiais.

A Figura 06 mostra os principais sistemas operacionais instalados (2010 - 2013), evidenciando o crescimento do Android em relação aos demais. Atualmente o Android e o IOS são os principais sistemas operacionais mobile, dominando mais de 80% do mercado mundial.

Figura 06 – Comparativo do crescimento dos sistemas operacionais



Fonte: Tecmundo, 2014. Adaptado.

3.1 – Sistema Operacional Android

O projeto Android foi desenvolvido pela Android Inc, uma startup americana situada atualmente no vale do silício. A Android Inc foi adquirida pela Google no ano de 2005 e o projeto tornou-se público no ano de 2007, com o objetivo de apresentar a primeira plataforma de código aberto (open source) de desenvolvimento para dispositivos móveis. Hoje o Android é mantido por um grupo denominado Open Handset Alliance (OHA), para Lecheta (2013), é um grupo formado por empresas lideres em tecnologia móvel, com a intenção de criar padrões abertos para telefonia móvel.

Entre as empresas participantes da OHA além da Google, estão, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile e Nvidia.

3.1.1- A Plataforma Android

O Android é um sistema operacional de desenvolvimento de aplicativos para dispositivos móveis. As aplicações geradas são escritas na linguagem Java, que consiste em uma linguagem orientada a objetos, compiladas em bytecodes Dalvik e executadas em uma máquina virtual, denominada Máquina Virtual Dalvik.

Segundo Laureano (2006), uma máquina virtual é uma duplicata eficiente e isolada de uma máquina real. Uma máquina virtual deve compatibilizar diferentes plataformas oferecendo às aplicações, uma duplicação isolada dos recursos de hardware do aparelho.

Para que essas aplicações móveis sejam criadas um kit de desenvolvimento contendo as APIs *(*Application Programming Interface ou, em português, Interface de Programação de Aplicativos*)* e ferramentas necessárias é disponibilizado, denominado Android SDK.

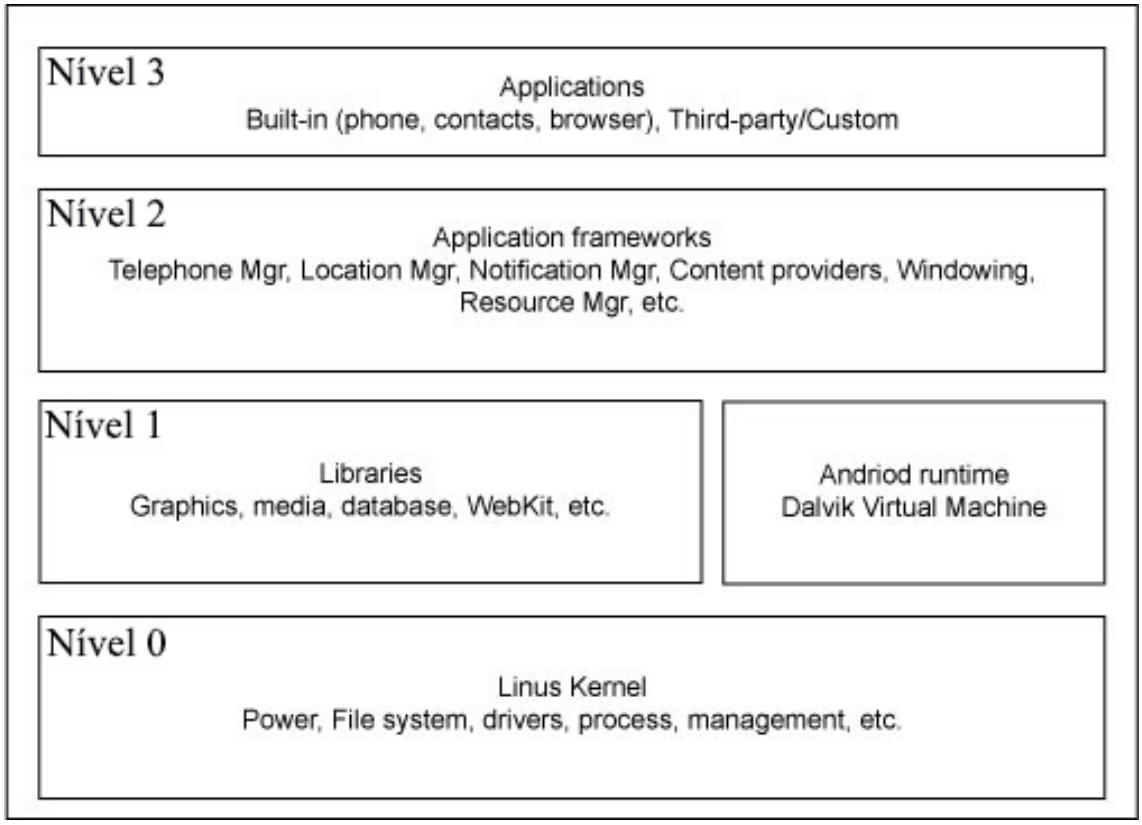
API é definida por Inácio Jr. (2007) como “especificação em linguagem de programação de um módulo de software onde outros módulos podem ou não depender”. Ou seja, um conjunto de rotinas e padrões de programação para acesso a um aplicativo. Normalmente uma API é fornecida num Kit de Desenvolvimento de Software (SDK) de uma linguagem de programação.

O SQLite é um banco autocontido, compacto, com suporte nativo no Android e sem necessidade de configuração ou instalação. Isto torna-o a escolha natural para um ambiente em que devemos prezar por desempenho, disponibilidade de memória e praticidade de uso. (LUZZI, 2014)

3.1.2- Arquitetura

Em sua arquitetura, conforme demonstra a Figura 07, temos programas agrupados em camadas, essas camadas, por sua vez, são divididas em níveis, vejamos a seguir:

Figura 07 - Arquitetura Android



Fonte: IBM, 2016. Adaptado.

De acordo com Lecheta (2010):

- No nível zero, temos a base da pilha. Localiza-se o sistema operacional da plataforma, responsável por serviços denominados de baixo nível e é utilizada pelas aplicações instaladas.
- No nível um, temos as camadas de bibliotecas (Libraries) e tempo de execução (Android RunTime). A camada de biblioteca é um conjunto de instruções que dizem ao dispositivo como lidar com diferentes tipos de dados. A camada de tempo de execução inclui um conjunto de bibliotecas do núcleo Java.
- No nível dois, temos a camada de framework de aplicação (Application Framework), programas que gerenciam as aplicações básicas do telefone.
- Já no nível três, temos a camada de aplicativos (Applications), as aplicações que são executadas sobre a plataforma, ou aplicações desenvolvidas por terceiros como é o caso do protótipo que será desenvolvido neste trabalho.

3.1.3- Visão Geral do SDK

O kit de desenvolvimento para Android SDK apresenta facilidades para o desenvolvedor, visto que dentro do pacote SDK vem um ambiente integrado para desenvolvimento de software denominado Eclipse. A versão do Eclipse disponibilizada pelo SDK possui um plugin do pacote ADT instalado, que inclui os componentes fundamentais para colaborar com o desenvolvimento ágil de aplicativos Android.

Esse kit é disponível pelo Google para, principalmente difundir o desenvolvimento de aplicativos para a plataforma e consequentemente aumentar ainda mais o número de desenvolvedores e aplicativos para o sistema e promover o crescimento da comunidade.

3.1.4- Android Runtime

O Runtime do Android possui um mecanismo baseado em dois grupos fundamentais que são as suas bibliotecas centrais e sua máquina virtual criada para que cada dispositivo móvel possa executar múltiplas Maquinas Virtuais.

Cada aplicação desenvolvida em Android executa em um único processo o qual é uma instância desta máquina virtual, executando arquivos no formato .dex (abreviação de Dalvik Executable), com um rápido carregamento, consumindo o mínimo de memória.

Segundo Alcântara (1996), Thread é um fluxo de controle sequencial isolado dentro de um programa. Como um programa sequencial qualquer, um thread tem um começo, um fim, e uma seqüência de comandos. Entretanto, um thread em Java não é um programa, não roda sozinho, roda dentro de um programa.

3.1.5- Componentes das Aplicações Android

Na Figura abaixo, temos a demonstração dos componentes de uma aplicação Android, no entanto, nem toda aplicação deve utilizar esses componentes. As Intents e as Views são itens importantes, pois, são elas quem fazem os outros quatro componentes funcionarem, vejamos:

Figura 08. Componentes de uma aplicação Android

Fonte: UNIVERSIDADE FEDERAL FLUMINENSE, 2014. Adaptado

De acordo com Gonçalves (2011):

- Activies ou Atividades são as representações das telas de aplicação. Aqui, se produz e elabora a interação visual da aplicação. Junto a uma activy normalmente têm-se uma view, onde será definida a exibição visual.
- Services ou Serviços servem para efetuar tarefas potencialmente demoradas, estes, são códigos executados em segundo plano, portanto, não apresentam uma interface visual.
- Broadcast Receivers ou Receptores da Transmissão são componentes responsáveis por receber e tratar eventos.
- Content Providres ou Provedores de Conteúdo disponibilizam um conjunto de dados específicos para outras aplicações.
- AndoidManifest.xml é o arquivo de manifesto escrito em XML, obrigatório e único para a aplicação. Este é o arquivo principal, onde localizam-se todas as configurações.

- Intents ou Intenções são mensagens utilizadas para solicitar uma ação de outro componente de aplicativo, por exemplo, Service, Activy e Broadcast Receiver.
- Views ou Visualizações possuem o objetivo de prover a interação com os usuários, estes, são os elementos que definem os objetos gráficos exibidos na tela.

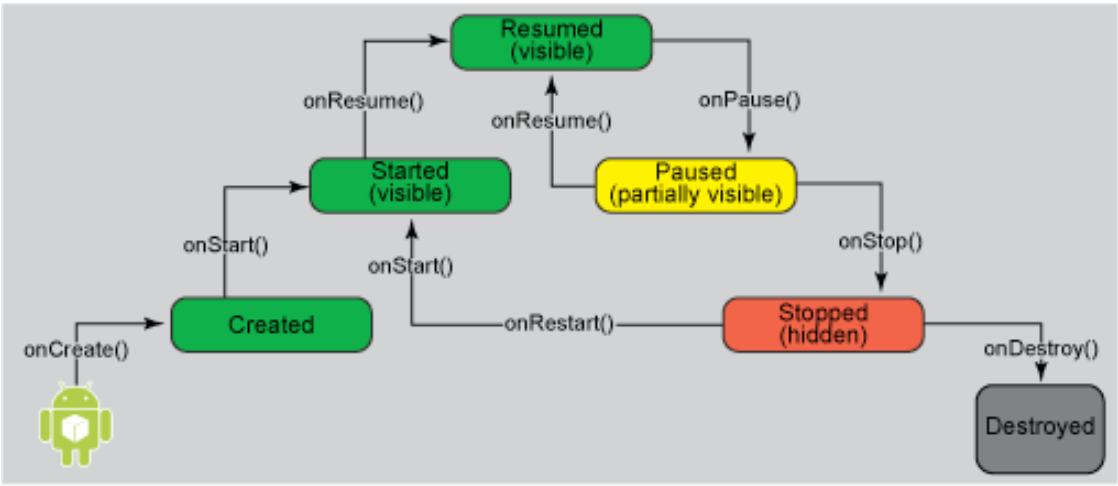
3.1.6- Ciclo de Vida

Uma aplicação android, assim como quaisquer aplicações apresentam um ciclo de vida que são definidos pelos estados de execução, temporariamente interrompida em segundo plano ou completamente destruída.

O ciclo de vida de uma aplicação android é dividida em três subníveis, que ficam se repetindo durante a execução de uma aplicação, são eles: a Entire lifetime, representando um ciclo de vida completo; a Visible lifetime, quando a activity da aplicação está visível para o usuário, ou, está em segundo plano; e, a Foreground lifetime, nessa fase, a activity está no topo da pilha, interagindo com o usuário.

O ciclo de vida completo de uma activity da aplicação é mostrado na figura 09, exibindo os estados possíveis e a chamada de cada método.

Figura 09 - Ciclo de Vida de uma Aplicação no Android



Fonte: IBM, 2015.

Cada um destes ciclos se inicia durante a chamada de um dos métodos controladores e termina quando algum outro método é chamado. A tabela 01 exibe as descrições de cada um destes métodos.

Tabela 01 – Descrições dos métodos[6]

Método	Descrição
onCreate()	É a primeira função a ser executada em uma Activity. Geralmente é a responsável por carregar os layouts XML e outras operações de inicialização. É executada apenas uma vez.
onStart()	É chamada imediatamente após a onCreate() – e também quando uma Activity que estava em background volta a ter foco.
onResume()	Assim como a onStart(), é chamada na inicialização da Activity e também quando uma Activity volta a ter foco. Qual a diferença entre as duas? A onStart() só é chamada quando a Activity não estava mais visível e volta a ter o foco, a onResume() é chamada nas "retomadas de foco".
onPause()	É a primeira função a ser invocada quando a Activity perde o foco (isso ocorre quando uma nova Activity é iniciada).
onStop()	*S*ó é chamada quando a Activity fica completamente encoberta por outra Activity.
onDestroy()	A última função a ser executada. Depois dela, a Activity é considerada "morta" – ou seja, nao pode mais ser relançada. Se o usuário voltar a requisitar essa Activity, um novo objeto será contruído.
onRestart()	Método chamado quando a activity está ficando visível ao usuário, dependendo do estado da aplicação pode ser chamado depois do método onCreate( ) ou onRestart( ).

Fonte: IBM, 2015. Adaptado.

[6] http://www.devmedia.com.br/entendendo-o-ciclo-de-vida-de-uma-aplicacao-android/22922

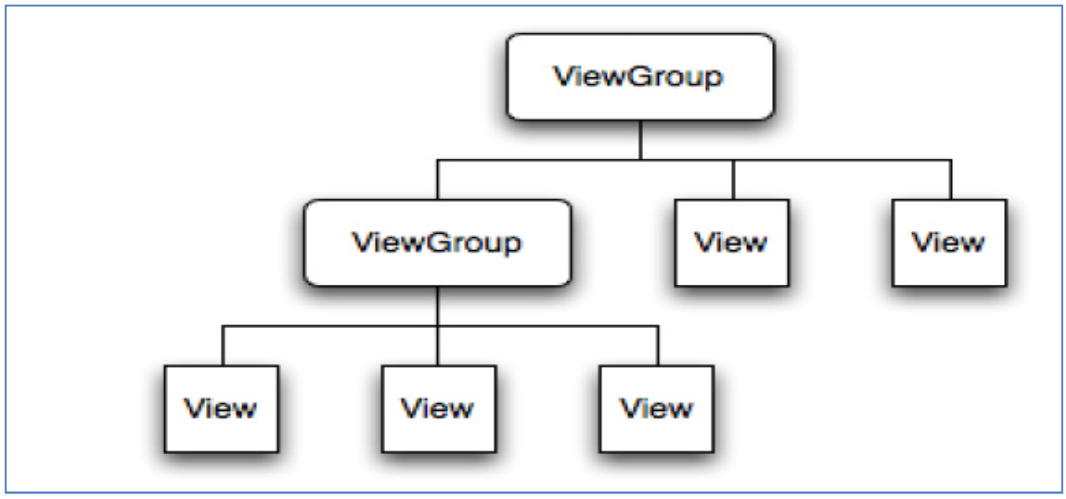
3.1.7- Interface com o Usuário

O Android fornece um modelo alternativo de construção da interface do usuário, os arquivos de layout são baseados em XML. A estrutura geral de um arquivo de layout android XML é relativamente simples.

> "Consiste em uma árvore de elementos XML, onde cada nó é o nome de uma classe View. Essa é uma classe mãe de todos os componentes visuais apresentados ao usuário. Suas subclasses irão compor uma interface com o usuário no Android. Estas subclasses implementam o método onDraw(Canvas), responsável por desenhar os componentes na tela. Tais componentes se dividem em dois grupos, os widgets e os gerenciadores de layout. Os widgets são componentes simples que herdam diretamente da classe View, como exemplos temos as classes Button, ImageView e TextView, já os gerenciadores de layout são subclasses de android.view.ViewGroup." (LECHETA, 2015)

A figura 10 exibe uma estrutura da árvore de objetos View e ViewGroup. Existe uma ampla variedade destes componentes disponíveis na plataforma Android.

Figura 10: Árvore de objetos View e ViewGroup



Fonte: DEVMEDIA, 2015. Adaptado.

Segundo Gonçalves (2011) os componentes mais básicos de interface são: TextView, representando um texto ou rótulo na tela; EditText, utilizada para que o usuário possa digitar informações em um campo de texto; Button e ImageButton, são componentes utilizados para criar um botão na tela; RadioButton, é um componente que permite ao usuário selecionar apenas uma opção de uma determinada lista de opções; CheckBox, um componente que permite ao usuário selecionar uma ou mais opções de uma determinada lista de opções; ListView, um componente que permite exibir uma lista de conteúdos.

CAPÍTULO 4 - PROJETO E DESENVOLVIMENTO DA APLICAÇÃO

4.1 – Requisitos

O protótipo da aplicação foi desenvolvido para funcionar em smartphones com a plataforma Android, sendo capaz de informar rotas entre dois pontos distintos de determinada cidade ou até mesmo de cidades diferentes. Ao informar os pontos e selecionar “enviar”, a aplicação deve buscar no mapa a menor rota entre esses pontos e a medida que o usuário desloca em direção ao segundo ponto informado, então, é feita uma atualização da sua localização para que o usuário saiba exatamente onde está.

Contudo para que essa aplicação funcione perfeitamente nos pequenos municípios, é necessário que as suas ruas estejam cartografadas e inseridas no mapa principal do OpenStreetMap. Para isso é necessário mostrar como realizar esse mapeamento e a inserção de dados nesse mapa.

4.2 - Construir Mapa da cidade de Datas MG

A forma adotada para adicionar ruas, bairros, pontos turísticos e outros pontos destacáveis da cidade de Datas/MG, foram através dos dados já exibidos por satélite que foram cedidos ao OpenStreetMap. Dessa forma, o trabalho é feito de forma interativa e online. A figura 11 exibe o mapa inicial da região de Datas antes da edição.

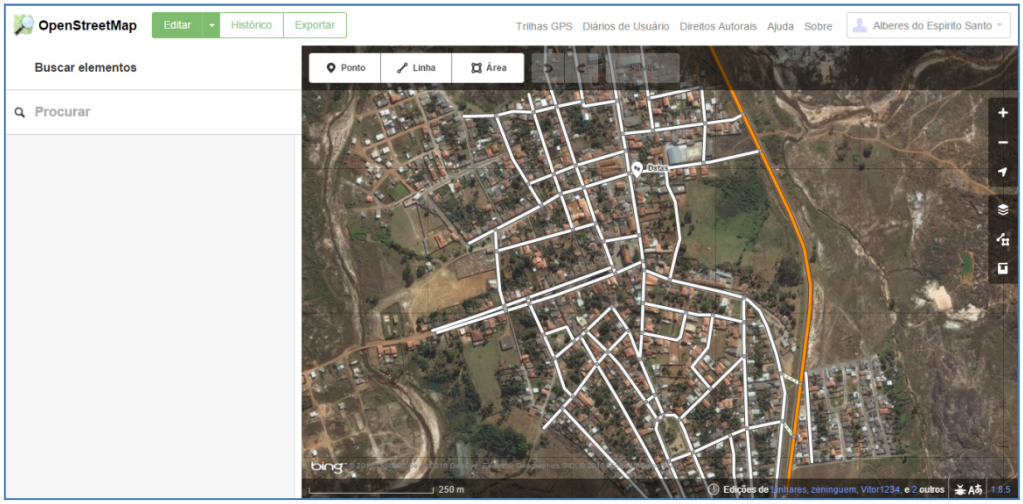
Figura 11: Mapa inicial da região de Datas para edição

Fonte: OPENSTREETMAP.ORG,2015.

Acessando a área do editor online é possível realizar as alterações desejadas. A busca por Datas/MG mostra o mapa principal da cidade, com

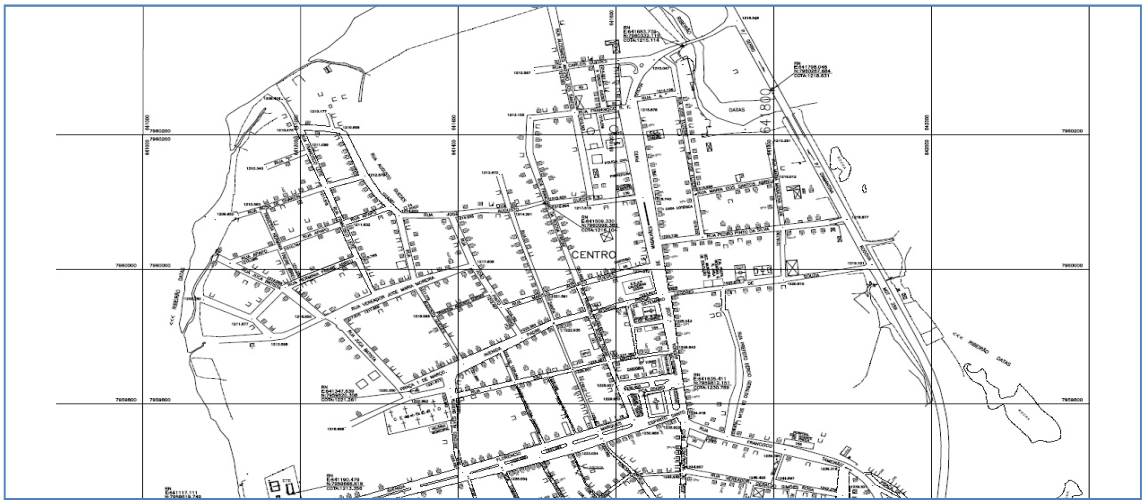
pouquíssimas ruas ou bairros nomeados, para dar inicio à edição conforme a figura 12. Com as imagens captadas por satélite a edição se torna bem simples e imediata, é preciso saber os dados e informações de onde serão realizadas as edições. Essas informações foram encontradas em um mapa desenhado por um engenheiro civil, no site da Prefeitura Municipal de Datas, conforme a figura 13 mostra, o que facilitou o desenvolvimento desse trabalho.

Figura 12: Mapa da cidade de Datas no OpenStreetMap, inicio da edição



Fonte: OPENSTREETMAP.ORG,2015.

Figura 13: Mapa parcial da cidade de Datas em desenho



Fonte: Prefeitura Municipal de Datas, 2014.

A edição se tornou um trabalho lento, porém efetivo. Todas as ruas e bairros começaram a ser traçados e nomeados conforme o mapa disponível no site do

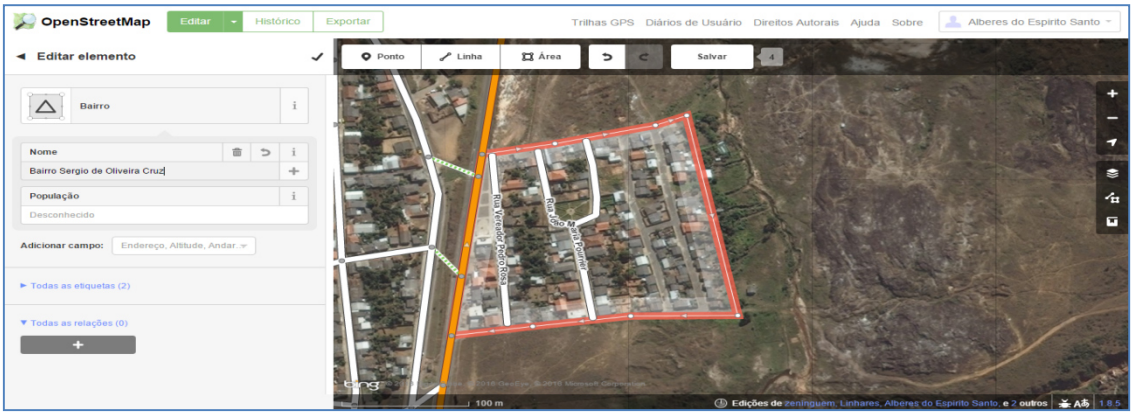
município. Cada rua ou avenida é traçada por uma linha e depois de ser demarcada, são informados os dados referentes a ela, como nome, velocidade máxima permitida, se é via de mão única, se veículo que pode transitar naquele local, dentre outras informações. A Figura 14 mostra como essa edição é simples.

Já os bairros ou outras áreas maiores, são informados pela ferramenta “áreas disponíveis” no OpenStreetMap (Figura 15). A ferramenta demarca toda a área desejada, arrastando-a sob aquela região. Depois de demarcada, deve-se adicionar informações a respeito daquela área, nome, extensão, dentre outros.

Figura 14: Edição de rua usando a ferramenta linha

Fonte: OPENSTREETMAP.ORG,2015.

Figura 15: Edição de bairro usando a ferramenta área

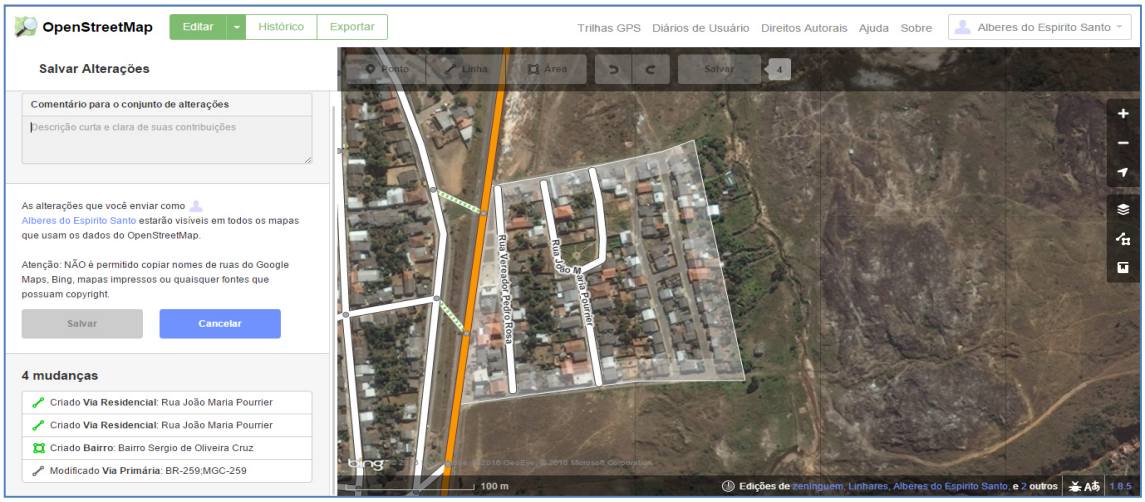


Fonte: OPENSTREETMAP.ORG,2015.

Quando as edições forem revisadas e concluídas, deve-se enviá-las ao mapa principal do OpenStreetMap (Figura 16), para que as informações editadas estejam

disponíveis para todos os usuários do serviço. Isso é possível ao salvar e comentar o que foi editado.

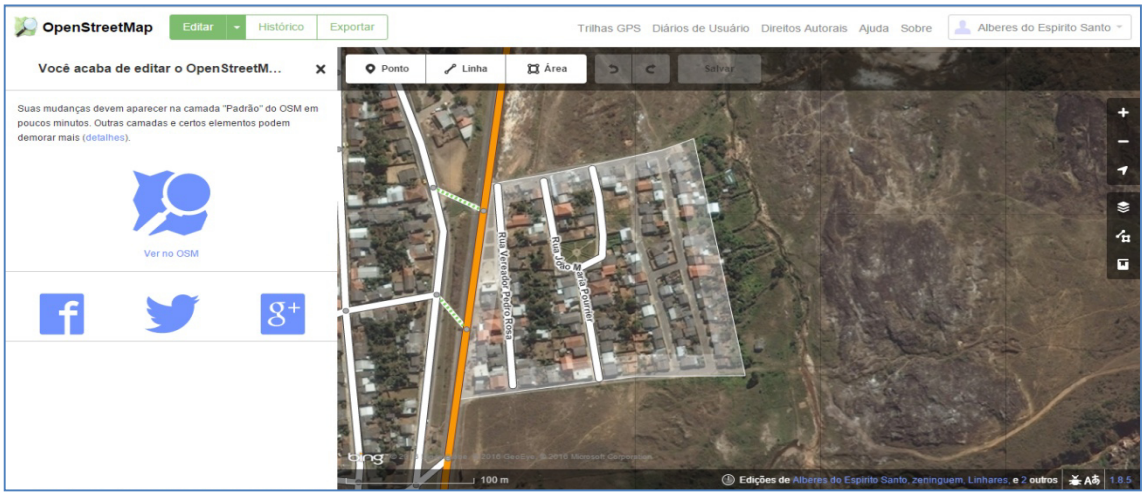
Figura 16: Salvar e enviar dados ao OpenStreetMap



Fonte: OPENSTREETMAP.ORG,2015.

Os servidores validam as informações e publicam no mapa principal em minutos (Figura 17). Os dados dos editores ficam disponíveis, de forma que, qualquer informação equivocada sabe-se quem foi o responsável por publicá-las.

Figura 17: Dados salvos e enviados ao mapa principal

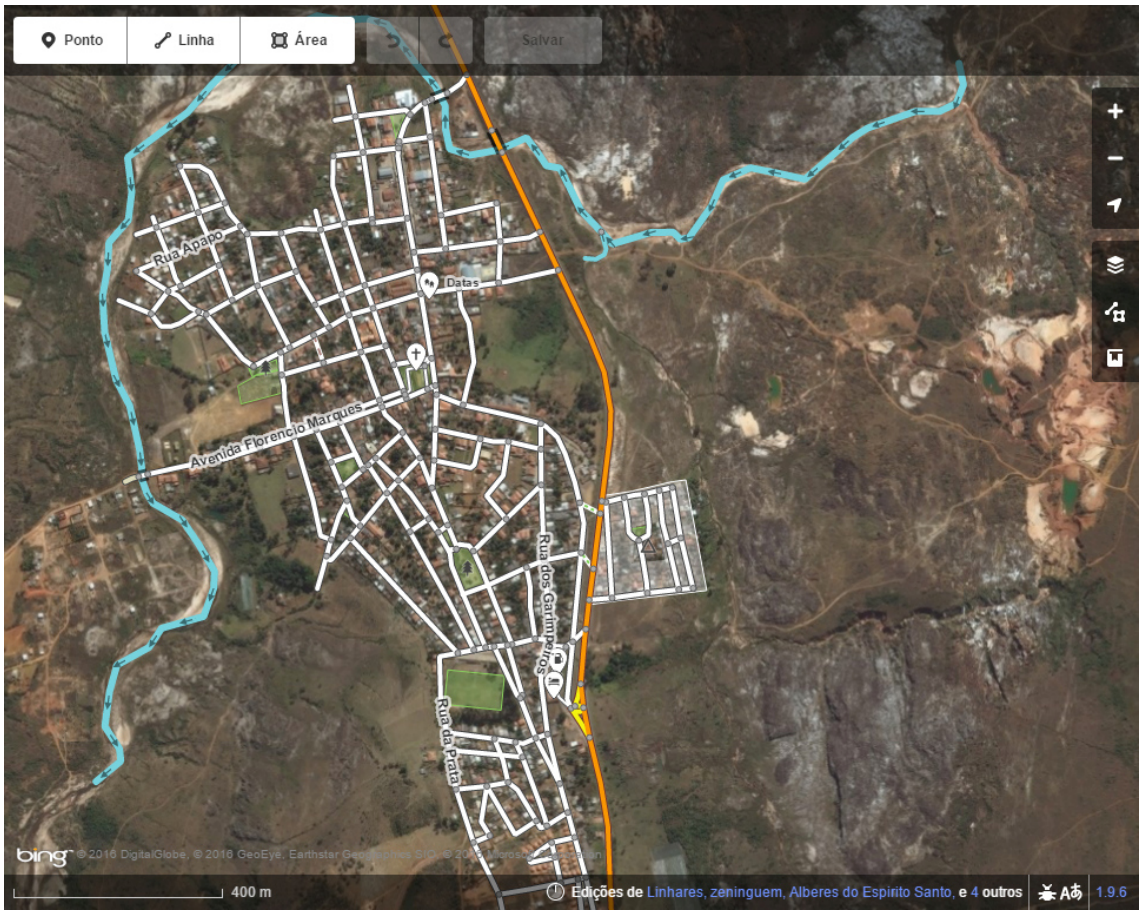


Fonte: OPENSTREETMAP.ORG,2015.

Dessa forma, foram feitas as edições no mapa completo da cidade de Datas/MG, para todas as outras áreas, ruas, usando os mesmos conceitos. A Figura 18 mostra o mapa totalmente editado e pronto para ser integrado ao mapa principal.

Quaisquer serviços de geolocalização podem ser trabalhados sob esse mapa. Inclusive o aplicativo desenvolvido nesse trabalho.

Figura 18: Mapa da cidade editado por completo



Fonte: OPENSTREETMAP.ORG,2015.

4.3 - Ambiente de desenvolvimento

O desenvolvimento de aplicativos na plataforma Android é possível por várias formas e frameworks diferentes. A forma adotada foi configurar um ambiente contendo a última versão do Java Development Kit (JDK) juntamente com o Android SDK e uma IDE para a codificação, ambos disponiveis no site do desenvolvedor Android. Para o protótipo desenvolvido neste trabalho, optou-se pela utilização da IDE Eclipse com o plugin ADT. O ADT é um plugin desenvolvido para estender as funcionalidades do Eclipse possibilitando a criação de projetos voltados para o

sistema operacional Android. O plugin utiliza bibliotecas presentes no SDK, e faz a ligação entre o Android SDK e o Eclipse.

4.3.1 AVD – Android Virtual Devices

Com o ambiente (IDE + ADT + SDK) instalado e configurado, é preciso adicionar um dispositivo virtual (AVD) para executar e testar a aplicação desenvolvida.

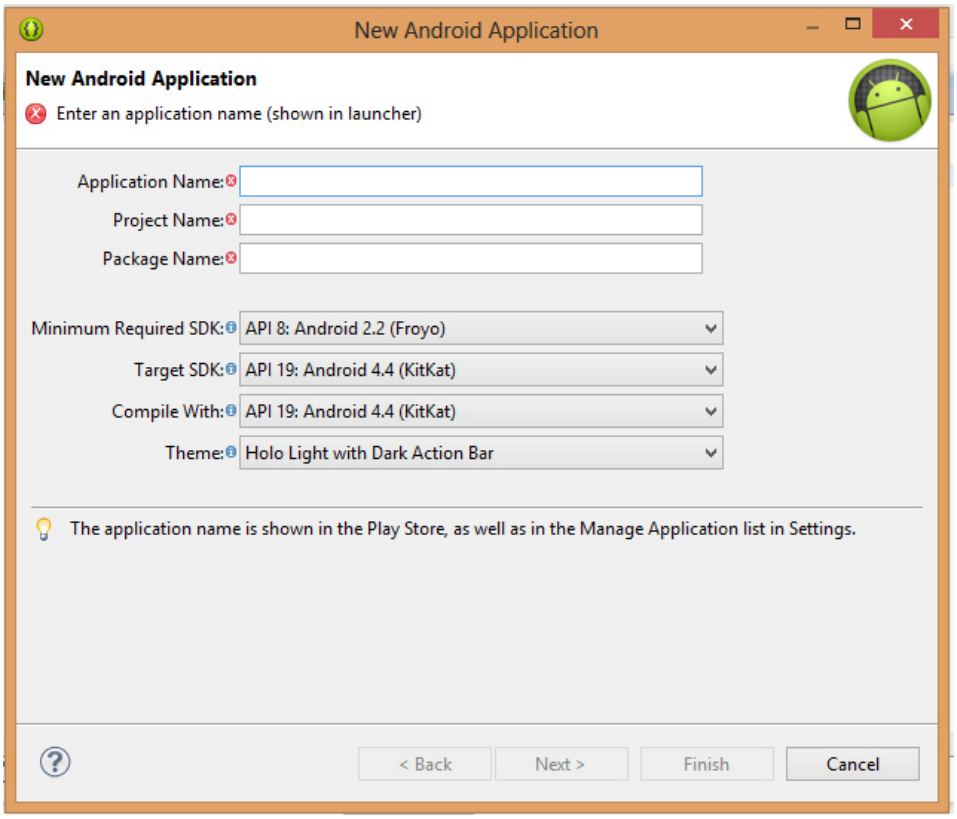
Segundo Lecheta (2015), o AVD é um dispositivo virtual do sistema operacional do Android que simula um aparelho no qual poderá executar e testar suas aplicações. Vários AVDs podem ser criados de acordo com o tipo de aparelho que for necessário ou julgar interessante emular a versão da plataforma Android na qual é desenvolvida a aplicação. Um AVD é basicamente composto de: um perfil de hardware e seus componentes; a versão da plataforma Android a qual a aplicação irá executar; permite a utilização do cartão SD; área de armazenamento na máquina de desenvolvimento, dentre outras componentes.

4.3.2 Criação e Estrutura do Projeto

Com o ambiente de desenvolvimento configurado e a AVD criada, já é possível criar o projeto da aplicação. Utilizando a IDE Eclipse segue-se o caminho File -> New -> Other, opção Android -> Android Project e clicar no botão Next. A figura 18 mostra a tela de configuração do projeto no Eclipse. Existe uma serie de itens a serem observados nessa seção, os principais são:

- Target SDK: Selecionar a versão da plataforma na qual será desenvolvido o aplicativo.
- Application Name: Especificar o nome do aplicativo, que será exibido na lista de aplicativos do dispositivo.
- Package Name: O nome do pacote que deve conter as classes e subclasses que se estendem de Activity.
- Minimum required. SDK: Especificar qual o número da versão mínima compatível com a aplicação a ser desenvolvida.

Figura 19: Tela de configuração de um projeto no Eclipse.



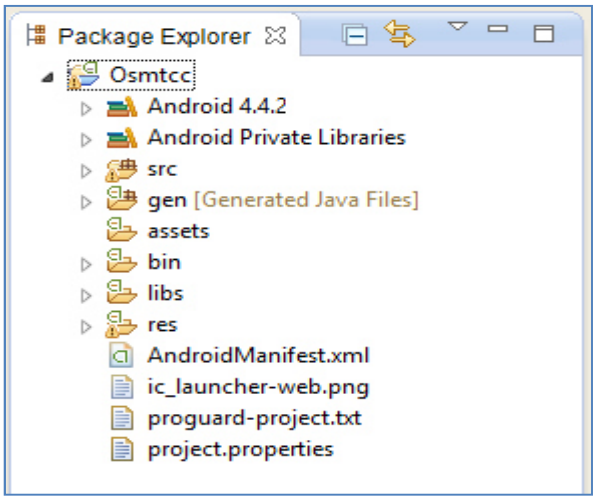
Fonte: IDE ECLIPSE, 2015.

A estrutura do projeto depois que as configurações foram criadas é mostrada na figura 19, cada item desta estrutura tem um significado, dentre os quais, os principais são descritos a seguir:

- src: pasta onde a Activity principal fica definida na criação do projeto.
- gen: pasta na qual é criada a classe R.java.
- assets: pasta auxiliar da aplicação.
- res: localização dos recursos gráficos da aplicação como arquivos de imagem, internacionalização e layout.
- drawable: pasta na qual ficam localizadas as imagens da aplicação, sendo possível categorizá-las em três resoluções distintas utilizando as sub-pastas: drawable-ldpi, drawable-mdpi e drawable-hdpi.
- layout: Os arquivos XML que representam recursos gráficos, como telas da aplicação.
- Libs: São bibliotecas Android.

- values: pasta na qual ficam arquivos XML que guardam Strings que podem ser usadas na aplicação.
- AndroidManifest.xml: é o arquivo de manifesto.
- default.propierties: arquivo que contém configurações do projeto e é utilizado para controle de revisão de código.

Figura 20: Estrutura de um projeto Android no Eclipse.



Fonte: IDE ECLIPSE, 2015.

4.4 Interface

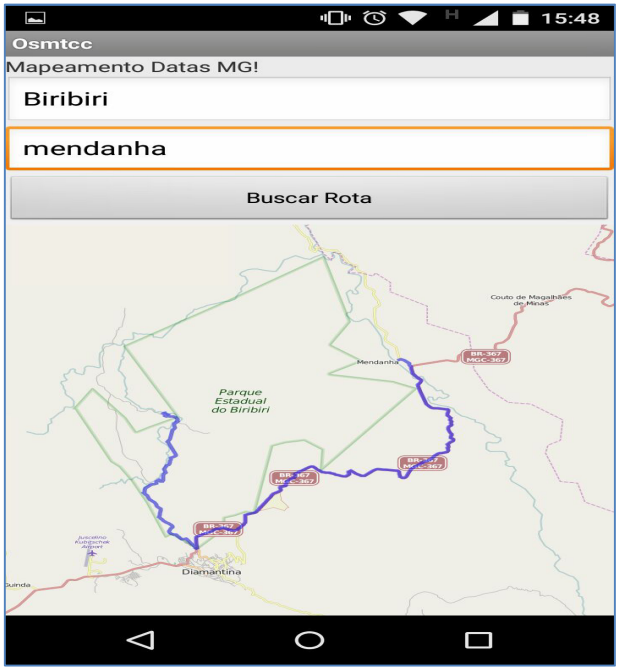
O aplicativo desenvolvido é relativamente simples, contudo bastante funcional. Apresenta apenas uma activity (atividade), este componente é descrito no AndroidManifest.xml (arquivo de manifesto), onde consta as informações, requisitos e permissões que a aplicação exige ao ser instalada em qualquer dispositivo.

Essa tela da aplicação contém dois campos de texto, onde são inseridas as informações de origem e destino, fornecidas pelo usuário para que a melhor rota entre esse dois pontos fornecidos seja exibida no mapa principal. Esse mapa principal esta disposto logo abaixo desses dois campos de textos e do botão buscar rotas e vem pelo padrão default com a cidade de Datas no centro, conforme mostra a figura 21, visto que a finalidade do aplicativo é exibir rotas nessa cidade. Contudo, o mapa muda a forma de visualização conforme as informações inseridas e após o botão “buscar rota” ser tocado como pode ser visto na figura 22, mostra rotas para outra região pesquisada no mapa.

Figura 21 - Interface do aplicativo mapeamento Datas

Fonte: PRINTSCREEN MOTOROLA MOTO G, 2015.

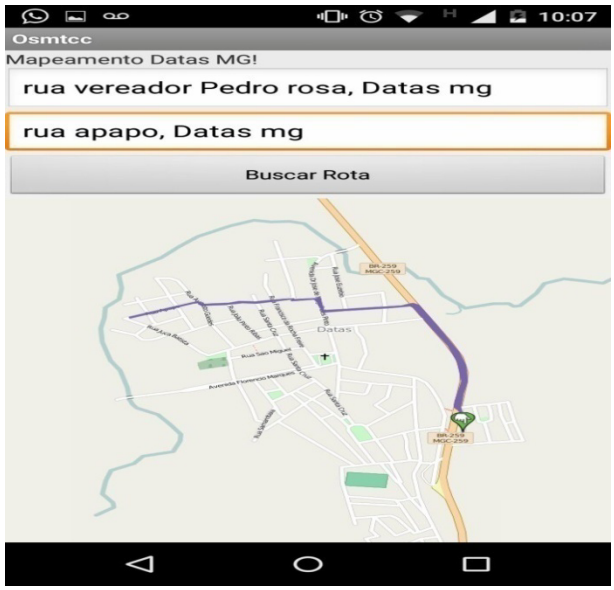
Figura 22 – Tela principal do app buscando outras regiões



Fonte: PRINTSCREEN MOTOROLA MOTO G, 2015.

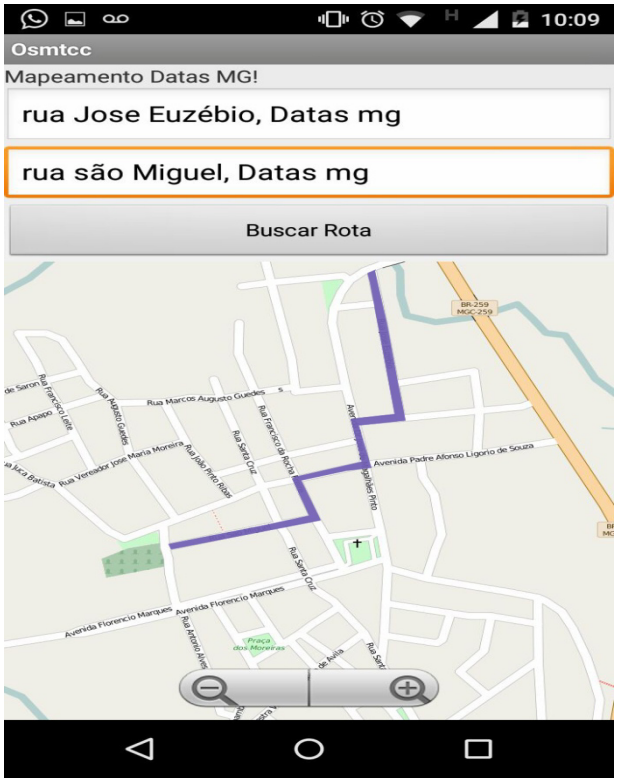
As figuras 23 e 24 mostram as rotas exibidas na região cartografada. A figura 23 exibe o resultado da busca, sendo o ponto de origem: rua vereador Pedro Rosa, Datas MG e o destino: rua Apapo, Datas MG. O ponto inicial é exatamente onde se encontra o usuário. Já a figura 24 mostra o melhor caminho entre as ruas José Euzébio e São Miguel.

Figura 23 – Rota entre ruas



Fonte: PRINTSCREEN MOTOROLA MOTO G, 2015.

Figura 24 – Rota entre ruas

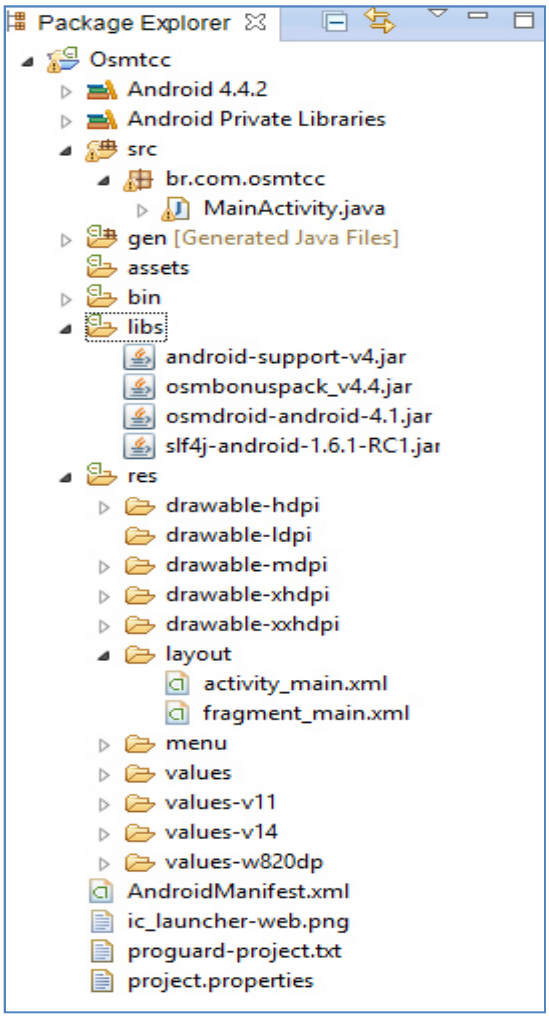


Fonte: PRINTSCREEN MOTOROLA MOTO G, 2015.

4.5 Implementação

A implementação do aplicativo foi desenvolvida na linguagem Java, a linguagem adotada para o desenvolvimento de aplicações nativas na plataforma Android pela maioria dos desenvolvedores. A IDE usada foi Eclipse, acrescida do plugin ADT. A estrutura de desenvolvimento é relativamente simples conforme a figura 25.

Figura 25 – Estrutura do aplicativo



Fonte: PRINTSCREEM DA IDE ECLIPSE, 2015.

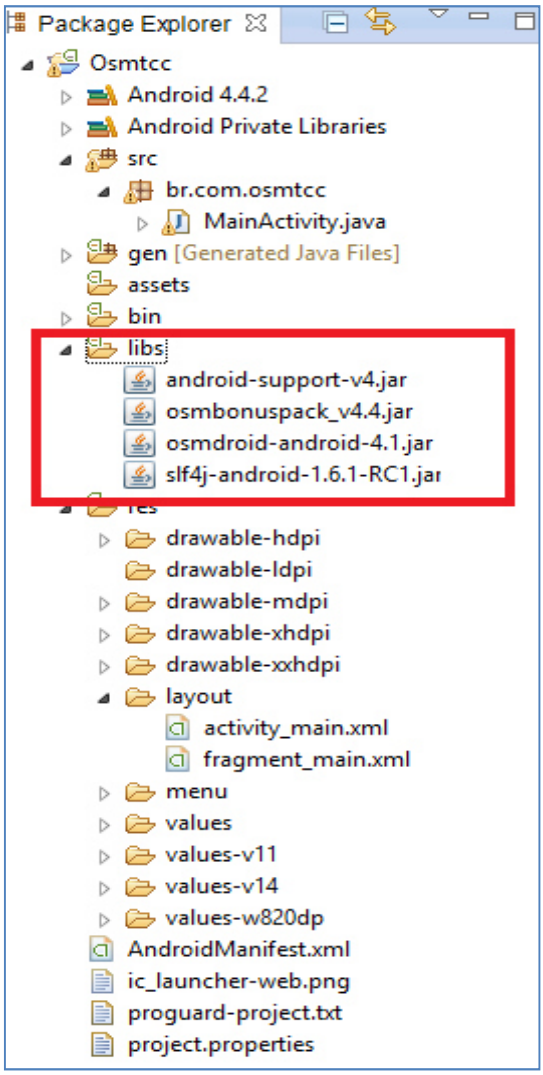
É importante codificar os elementos diretamente nas classes, contudo é recomendável a construção das interfaces utilizando descritores XML de forma declarativa, dessa forma proporciona-se a separação dos elementos de layout (componentes gráficos de interface) do código propriamente dito, facilitando futuras manutenções no código da aplicação.

A organização OpenStreetMap, oferece uma serie de Api's e arquivos JAR's para facilitar a integração do aplicativo com o mapa ou outro dispositivo fornecidos pela entidade. Esta interface é o conjunto de padrões de programação que permite a construção de aplicativos e a sua utilização de maneira não tão evidente para os usuários. O arquivo JAR (Java ARchive) é um arquivo compactado no formato ZIP que contém um conjunto de classes (arquivos ".class") e arquivos de configuração. O

formato JAR é utilizado para permitir a distribuição de bibliotecas ou, até mesmo, sistemas inteiros em Java (uma vez que é possível montar arquivos JAR formados por centenas de classes). (GOLÇALVES, 2014)

A figura 26, mostra os arquivos.JAR disponíveis para o projeto os quais estão disponíveis no Google code e na organização Slf4j. Depois de baixados, os arquivos são incorporados ao projeto dentro da pasta *libs* e assim ficam disponíveis para uso no projeto.

Figura 26 – Arquivos .jar



Fonte: IDE ECLIPSE

Dentre os arquivos apresentados, o essencial para o projeto é o OsmBonusPack, visto que esse contem classes responsáveis por exibir no mapa rotas, pontos dentre outras funcionalidades.

Para utilizar o mapa do OpenStreetMap é necessário declarar a view de layout de mapa <org.osmdroid.views.MapView em Activity_main.xml (arquivo de manifesto), de forma que ocupe toda a tela. A figura 27 mostra como é declarado.

Figura 27 – declaração *<org.osmdroid.views.MapView*

```
<org.osmdroid.views.MapView
    android:id="@+id/mapView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="0.87" >

</org.osmdroid.views.MapView>
```

Fonte: PRINTSCREEM DA ACTIVITY_MAIN.XML, 2015.

Ainda na Activity_main.xml, temos os dois campos de texto, “Origem” e “Destino” que são fornecidos pelo usuário e um botão “Buscar Rota”, cujo chama o método *getRoute*. Esse método é o responsável por buscar numa lista de endereços os dois pontos definidos pelo usuário nos campos de origem e destino, montar a rota e exibir na *view*. Na figura 28 percebe-se os campos declarados conforme mencionado.

Figura 28 – Elementos de texto e botão

```
<EditText
    android:id="@+id/origem"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Origem: Rua, Cidade, Estado, País" >

    <requestFocus />
</EditText>

<EditText
    android:id="@+id/destino"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Destino: Rua, Cidade, Estado, País" />

<Button
    android:id="@+id/buscar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Buscar Rota"
    android:onClick="getRoute"/>
```

Fonte: PRINTSCREEM ACTIVITY_MAIN.XML, 2015.

Depois de definidos os campos e botão de busca no descritor xml, a classe MainActivity é implementada.

São três as principais funções declarados como: *Localizaçãogps* do tipo *LocationManager* que permite, a localização do usuário via gps; *osm* do tipo *MapView* para visualização do mapa; e *mapa* do tipo *MapController,* permite o controle total sob o mapa exibido, todas essas classes são importadas de org.osmdroid.views, como na figura 27.

São liberadas as configurações de visualização de mapa, de pontos e zoom no mapa exibido.

É importante destacar o GeoPoint, também importado de *org.osmdroid.views,* fornece a localização no mapa através das referências latitude e longitude. Nesse caso foram passadas como parâmetro a latitude e longitude de uma rua da cidade de Datas (figura 29), logo esse é o ponto central do mapa exibido no dispositivo.

Figura 29 – Classe MainActivity.java

MainActivity.java

```
package br.com.osmtcc;

import java.io.IOException;

public class MainActivity extends Activity implements LocationListener

    private LocationManager localizacaogps;
    private MapView osm;
    private MapController mapa;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        osm = (MapView) findViewById(R.id.mapView);
        osm.setTileSource(TileSourceFactory.MAPNIK);
        osm.setBuiltInZoomControls(true);
        osm.setMultiTouchControls(true);

        mapa = (MapController) osm.getController();
        mapa.setZoom(12);

        GeoPoint center = new GeoPoint(-18.4650,-43.6835);
        mapa.animateTo(center);
        addMarker(center);
```

Fonte: PRINTSCREEM CLASSE MAINACTIVITY.JAVA, 2015.

Para que o usuário tenha a sua localização real via GPS, a localizacaogps é implementada de forma que atualiza a posição em tempo igual a 0 e localização igual a 0, dessa forma se o usuário se deslocar a posição é atualizada instantaneamente (figura 30).

Figura 30 – Função para localização

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    osm = (MapView) findViewById(R.id.mapView);
    osm.setTileSource(TileSourceFactory.MAPNIK);
    osm.setBuiltInZoomControls(true);
    osm.setMultiTouchControls(true);

    mapa = (MapController) osm.getController();
    mapa.setZoom(12);

    GeoPoint center = new GeoPoint(-18.4650,-43.6835);
    mapa.animateTo(center);
    addMarker(center);

    localizacaogps = (LocationManager) getSystemService(Context.LOCATION_SERV
    localizacaogps.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
}
```

Fonte: PRINTSCREEM DA CLASSE MAINACTIVITY.JAVA, 2015.

Para exibir no mapa a rota que o usuário deseja, a função getRoute recebe como parâmetros os campos de origem e destino fornecidos. Se os campos estiverem vazios uma mensagem de falha é exibida ao usuário se não, a função start é chamada com tais parâmetros (figura 31).

Figura 31 – Funções para exibir rota

```
// EXIBIR ROTA
    public void getRoute(View view){
        EditText etOrigem = (EditText) findViewById(R.id.origem);
        EditText etDestino = (EditText) findViewById(R.id.destino);
        final String o = etOrigem.getText().toString();
        final String d = etDestino.getText().toString();

        new Thread(){
            public void run(){
                GeoPoint start = getLocation(o);
                GeoPoint end = getLocation(d);

                if(start != null && end != null){
                    drawRoute(start, end);
                }
                else{
                    Toast.makeText(MainActivity.this, "Falha ao exibir rota!"
                }
            }
        }.start();
    }
```

Fonte: PRINTSCREEM DA CLASSE MAINACTIVITY.JAVA, 2015.

GeocoderNominatim oferece uma lista de endereços imensa, na qual trabalha-se o script para identificar nomes de ruas, cidades, estados e países (figura 32). Dessa forma, já temos os dados de latitude e longitude dos pontos informados pelo usuário, agora é desenhar a rota.

Figura 32 – Função GeoPoint, lista de endereços

```
public GeoPoint getLocation(String location){
    GeocoderNominatim gn = new GeocoderNominatim(MainActivity.this);
    GeoPoint gp = null;
    List<Address> al = new ArrayList<Address>();

    try{
        al = gn.getFromLocationName(location, 1);

        if(al != null && al.size() > 0){
            Log.i("Script", "Rua: "+al.get(0).getThoroughfare());
            Log.i("Script", "Cidade: "+al.get(0).getSubAdminArea());
            Log.i("Script", "Estado: "+al.get(0).getAdminArea());
            Log.i("Script", "País: "+al.get(0).getCountryName());

            gp = new GeoPoint(al.get(0).getLatitude(), al.get(0).getLongit
        }
    }
    catch(IOException e){ e.printStackTrace(); }

    return(gp);
}
```

Fonte: PRINTSCREEM IDE ECLIPSE, 2015.

Para que a função getRoute funcione, a drawRoute tem que receber os parâmetros e a roadManager desenha a rota usando Polyline conforme mostra a figura 33.

Figura 33 – Função drawRoute

```
public void drawRoute(GeoPoint start, GeoPoint end){
    RoadManager roadManager = new OSRMRoadManager();
    ArrayList<GeoPoint> points = new ArrayList<GeoPoint>();
    points.add(start);
    points.add(end);
    Road road = roadManager.getRoad(points);
    final Polyline roadOverlay = RoadManager.buildRoadOverlay(road, MainActivity.this

    runOnUiThread(new Runnable(){
        public void run(){
            osm.getOverlays().add(roadOverlay);
        }
    });
}
```

Fonte: PRINTSCREEM IDE ECLIPSE, 2015.

4.6 Testes realizados

Os testes foram realizados em dispositivos reais, os quais mostram o funcionamento mais claro do aplicativo. Foi possível verificar que todas as funções implementadas atuam de forma objetiva, fazendo o que foi proposto na aplicação. Vale uma atenção especial à localização via GPS, essa atualiza automaticamente a cada deslocamento seja por qualquer distância.

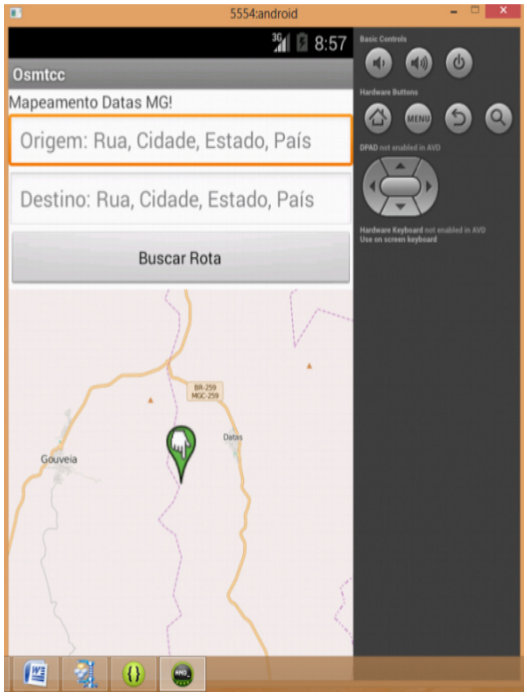
Os primeiros testes foram realizados no dispositivo simulado (AVD), a medida que o aplicativo foi sendo criado. O dispositivo virtual criado executa o sistema Android na versão 2.2, API level 8, que identifica a versão do Android e cartão de memória de 500 MB. Na figura 34 pode ser observado o ícone do aplicativo OSMTCC gerado na área de iniciação rápida do dispositivo virtual.

Figura 34 – Ícone do aplicativo no dispositivo virtual

Fonte: Prinstscreem do AVD

O dispositivo virtual se comporta na maneira esperada para exibir o layout especificado no código, principalmente no que diz respeito ao *mapview* para exibir o mapa da região explorada, conforme a figura 35.

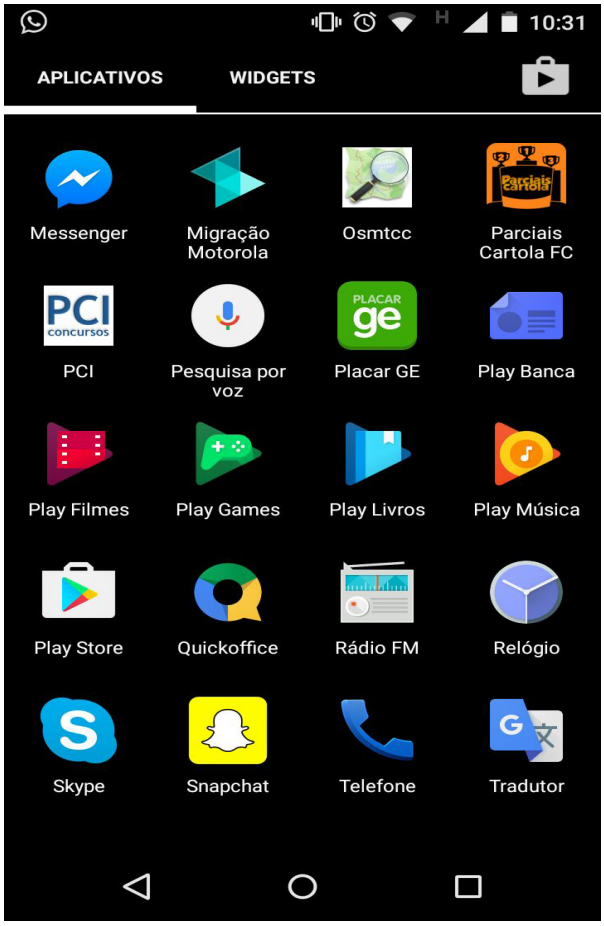
Figura 35 - Teste no dispositivo virtual



Fonte: Prinstscreem do AVD

Após os testes realizados no dispositivo virtual, é necessário testar a aplicação em um dispositivo real. Para isso foi usado um smartphone Motorola Moto G 1ª geração. O aparelho deve receber o arquivo de instalação na extensão APK o qual é gerado pela IDE Eclipse no diretório \BIN. A figura 36 exibe o ícone do aplicativo instalado no aparelho.

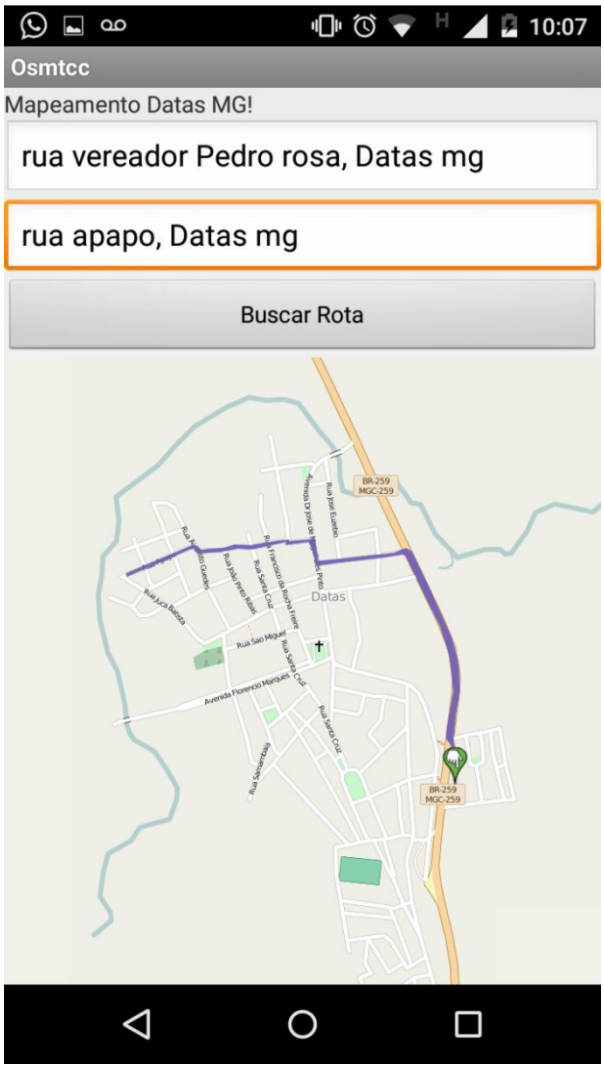
Figura 36 – Ícone do aplicativo OSMTCC no smartphone



Fonte: Prinstscreem do smartphone Motorola Moto G.

Depois de exportado e instalado no dispositivo os demais testes foram realizados e exibiram resultados satisfatórios. Os resultados das buscas por rotas, exibição do mapa, zoom, localização do usuário foram bem precisos conforme os resultados apresentados na figura 37.

Figura 37 – Teste realizado em dispositivo real



Fonte: Prinstscreem do smartphone Motorola Moto G

5- CONCLUSÃO

Através dos estudos aqui apresentados, conclui-se que o OpenStreetmap é uma excelente opção ante ao GoogleMaps. No caso das pequenas cidades, trilhas ou áreas não mapeadas pelo GoogleMaps, o OpenStreetmap é sem duvida a melhor opção, visto que qualquer pessoa pode mapear a área desejada e enviar ao mapa principal, de forma que esse conteúdo fica disponível para acesso a todos que usam a plataforma.

Ao unir uma aplicação móvel com o mapa oferecido pelo OpenStreetMap, temos uma perfeita interação de forma a proporcionar ao usuário melhores rotas entre pontos definidos.

Portanto a aplicação desenvolvida será de grande valia aos portadores de smartphones que necessitam de se locomover nas pequenas cidades ou áreas isoladas, desde que essas estejam cartografadas no mapa principal do OpenStreetMap, caso não estejam ainda há a opção de o próprio usuário, de forma colaborativa, enviar dados ao mapa auxiliando novos usuários que venham a navegar por aquela mesma área.

5.1 Trabalhos futuros

Para um trabalho posterior pode-se programar o cálculo de distância entre os dois pontos, definir se a rota fornecida é para veículos ou ciclistas ou ainda se a via possui contramão, para que os usuários não acabem se envolvendo em acidentes ou multas por exemplo.

6 - REFERÊNCIAS BIBLIOGRÁFICAS

ALCÂNTARA, Andreia A. **O que são threads?** Curdo de verão de Java. 22 de janeiro de 1996.
Disponível em:< http://www.di.ufpe.br/~java/verao/aula8/definicao.html> acesso em: 16 mar. 2016.

BARROS, Thiago. O que é Smartphone e para que serve?. **Informática Sistemas Operacionais**. São Paulo. 2012.
Disponível em: <http://www.techtudo.com.br/artigos/noticia/2011/12/o-que-e-smartphone-e-para-que-serve.html > acesso em: 12 jan. 2016.

DATE, C.J.; Int. a Sistemas de Bancos de Dados, tradução da 4a.edição norte-americana, Editora Campus, 1991.

FAEBER, N., HILZINGER, M. **Sistemas Operacionais Móveis**. Linux Magazine. São Paulo: LNMB, fev. 2011.

GONÇALVES, Julio Cesar. Uso da plataforma Android em um protótipo de aplicativo coletor de consumo de gás natural. Universidade Tecnológica Federal do Paraná. Curitiba. 2011.

HAKLAY, Mordechai; WEBER, Patrick. Openstreetmap: User-generated street maps. **IEEE Pervasive Computing**, v. 7, n. 4, p. 12-18, 2008.

Infraestrutura Nacional de Dados Espaciais. **SIG Brasil- O Portal Brasileiro de Dados Geoespaciais.**
Disponível em:< http://www.inde.gov.br/> acesso em: 07 mar. 2015.

KORTH, Henry F.; SILBERSCHATZ, Abraham. **Sistemas de Bancos de Dados**. Makron Books, 2a. edição revisada, 1994.

LECHETA, Ricardo. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. Novatec, 2ª edição, 2010.

LECHETA, Ricardo. R. **Google Android 4 : Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. Novatec, 4ª edição, 2015.

LEITE, Beatriz Paula; POMPERMAYER, Guilherme Andretta; WERNECK, Marcelo Gaioso. **Uso de imagens de satélite e do sistema Openstreetmap no ensino universitário para produção e atualização de mapas digitais livres e abertos na Internet**. *Simpósio Brasileiro de Sensoriamento Remoto-SBSR, XVI* (2013).

LUZZI, Luciano**. Android – persistência de dados usando sq-lite**, 2014.

MEDEIROS, Anderson. **OpenStreetMap**. Edição gratuita, 2014.

MENDES, António José. **Fundamentos de programação em JAVA 2**. 2004.

OpenStreetMap Brasil
Disponível em: <http://www.openstreetmap.org/#map=5/51.509/-0.088> acesso em: 07 mar. 2015.

QUOOS, João Henrique. **O que é latitude e longitude?** Laboratório de Cartografia. UFSM. Rio Grande do Sul. 2014.
Disponível em:
<http://coral.ufsm.br/cartografia/index.php?option=com_content&view=article&id=43&Itemid=39 > acesso em: 10 out. 2015.

RAMM, Frederik; TOPF, Jochen; CHILTON, Steve. **OpenStreetMap: using and enhancing the free map of the world**. Cambridge: UIT Cambridge, 2011.

TANENBAUM, Andrew S. **Sistemas operacionais: projeto e implementação** / Andrews S. Tanenbaum e Albert S. Woodhull; trad. Edson Furmankiewicz. 2. Ed. Porto Alegre; Bookman, 2000.