UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JAMES GUSTAVO BLACK REBELATO

NAVEGADOR GPS PARA EQUIPAMENTOS MÓVEIS COM BASE NOS DADOS CARTOGRÁFICOS E ELÉTRICOS DA COPEL

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2013

JAMES GUSTAVO BLACK REBELATO

NAVEGADOR GPS PARA EQUIPAMENTOS MÓVEIS COM BASE NOS DADOS CARTOGRÁFICOS E ELÉTRICOS DA COPEL

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora:
Msc. Rúbia Eliza de Oliveira Schultz Ascari

PATO BRANCO
2013

ATA Nº: **210**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **JAMES GUSTAVO BLACK REBELATO**.

Às 14:25 hrs do dia 19 de abril de 2013, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Rúbia E. O. Schultz Ascari (Orientadora), Eliane Maria de Bortoli Fávero (Convidada) e Kathya Silvia Collazos Linares (Convidada), para avaliar o Trabalho de Diplomação do aluno James Gustavo Black Rebelato, matrícula 980412, sob o título **Navegador GPS para Equipamentos Móveis com Base nos Dados Cartográficos e Elétricos da COPEL**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 15:50 hrs foi encerrada a sessão.

Profa. Rúbia E. O. Schultz Ascari, M.Sc.
Orientadora

Profa. Eliane Maria de Bortoli Fávero, M.Sc.
Convidada

Profa. Kathya Silvia Collazos Linares, Dr.
Convidada

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

REBELATO, James Gustavo Black. **Navegador GPS para equipamentos móveis com base nos dados cartográficos e elétricos da COPEL**. 2013. 101 f. Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2013.

Este trabalho apresenta o desenvolvimento de um aplicativo navegador GPS voltado para equipamentos móveis, executado na plataforma Android. O principal objetivo do aplicativo é permitir a visualização de mapas vetoriais e a obtenção de cálculos de roteamento relacionados aos dados utilizados pela companhia COPEL. Os mapas são obtidos da base de dados da COPEL e convertidos para os padrões utilizados pelo navegador desenvolvido. Dando continuidade ao trabalho desenvolvido como estágio, foi possível idealizar e implantar um aplicativo próprio de navegação, executável em diversos dispositivos móveis, e não somente baseado em aparelhos de GPS automotivos. Para tanto foram utilizadas tecnologias que estiveram presentes na formação acadêmica, como a linguagem de programação Java, o ambiente de desenvolvimento Eclipse, conhecimentos sobre orientação a objetos, banco de dados, análise e modelagem de sistemas. Utilizando conceitos e técnicas de desenvolvimento para aplicativos móveis foi possível alcançar o objetivo final de disponibilizar uma nova ferramenta de trabalho para os empregados da COPEL.

Palavras-chave: Java. Navegador GPS. Android. Roteamento. COPEL.

ABSTRACT

REBELATO, Gustavo James Black. **GPS Navigator for mobile devices based on cartographic data and electrical Copel**. 2013. 101 f. Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Campus Pato Branco. Pato Branco, 2013.

This work presents the development of a GPS navigator application built over Android platform designed for mobile devices. The main objective of the application is to allow the visualization of vector maps and performing routing calculations related to the data used by the company Copel. The maps are obtained from company database and converted to the standards used by the navigator developed. As a consequence of the work developed during trainee, it was possible to project and implement a proper navigator, which is not based on automotive GPSs, executable on various mobile devices. To perform the study, the following technologies present on academic formation were used: Java programming language, Eclipse development environment, object-oriented programming, database knowledge, analysis and modeling systems. The final goal of deliver a work tool for the company employees was reached by using concepts and techniques for development of mobile applications.

Keywords: Java. GPS navigator. Android. Routing. Copel.

LISTA DE FIGURAS

QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

AOSP	*Android Open Source Project*
COPEL	Companhia Paranaense de Energia do Paraná
DER	Diagrama de Entidade e Relacionamento
GEDIS	Número de Cadastro de Equipamentos COPEL
GEO	Georreferenciamento da Rede Elétrica COPEL
GPS	*Global Positioning System (Sistema de Posicionamento Global)*
NIO	Número de Identificação Operacional
SDK	*Software Development Kit*
SQL	*Structured Query Language*
TR	Transformador de Potência
UC	Unidade Consumidora
UML	Linguagem de Modelagem Unificada
UTM	*Universal Transversa de Mercator*
WGS 84	*World Geodetic System 1984*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta o contexto no qual se insere a proposta do trabalho, os seus objetivos e a justificativa. Por fim está a organização do texto por meio da apresentação dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

Atualmente o GPS é uma ferramenta tecnológica muito difundida, principalmente pela sua utilização civil em diversas áreas da sociedade, sejam elas recreativas ou profissionais.

Os equipamentos móveis, como *Tablets* e *Smarthphones,* possuem cada vez mais capacidade de armazenamento e processamento, além de recursos avançados de visualização de gráficos com alta resolução, possibilitando assim a utilização de novas funcionalidades de comandos por voz e mapas em 3D.

A cada dia é maior a disponibilidade na Internet de mapas vetoriais desenvolvidos por órgão públicos, como o IBGE, ou produzido por amadores em comunidades livres, a exemplo do *OpenStreetMap*, referenciado posteriormente nesse trabalho. A COPEL possui um cadastro completo com todos os dados cartográficos e elétricos do sistema de distribuição de energia no Paraná.

Considerando o contexto favorável para a continuidade do trabalho desenvolvido no estágio e a aplicação mais efetiva dos conhecimentos de programação na plataforma de desenvolvimento Java e manipulação de dados, tornou-se viável o desenvolvimento e implantação de um Navegador GPS e a criação de mapas próprios, visando a utilização do sistema por funcionários da COPEL nos 399 municípios do Paraná.

1.2 OBJETIVOS

O objetivo geral se refere ao resultado principal obtido com a realização do trabalho. E os objetivos específicos o complementam.

1.2.1 Objetivo Geral

Desenvolver um aplicativo navegador GPS que possa ler mapas vetoriais criados a partir da base cartográfica e elétrica da COPEL.

1.2.2 Objetivos Específicos

- Desenvolver um aplicativo navegador GPS para equipamentos móveis que permita a rápida visualização e navegação em campo;
- Gerar mapas vetoriais contendo dados elétricos da empresa COPEL que possam ser lidos pelo navegador GPS desenvolvido;
- Implantar o sistema em toda a COPEL para utilização em campo;
- Analisar os resultados em campo e comprovar a efetividade do sistema.

1.3 JUSTIFICATIVA

No trabalho de estágio, foi apresentada uma alternativa para facilitar a navegação de funcionários da empresa COPEL em estradas rurais, por meio da utilização de GPS automotivos. Foram gerados mapas personalizados, a partir da base de dados da COPEL. Esses mapas permitem, além da visualização de estradas rurais - normalmente não disponíveis nos mapas padrão disponibilizados pelos aparelhos - a visualização de dados elétricos, como transformadores, fusíveis, chaves, entre outros.

O desenvolvimento de um navegador GPS para equipamentos móveis surgiu da necessidade de apresentar, para o empregado a serviço da COPEL, novas funcionalidades voltadas a visualização dos mapas utilizados e maior comunicação com a central de operações.

O GPS automotivo não permite a modificação de seus softwares de navegação, impossibilitando assim a adição de novas funcionalidades específicas às atividades cotidianas realizadas na empresa COPEL. Sendo assim, tornou-se atrativa a ideia de desenvolver um aplicativo que possa ser executado em equipamentos móveis e permita a sua personalização.

A fonte dos dados continua sendo proveniente da base cartográfica da COPEL, com a necessidade da conversão dos dados para um novo formato compatível com o navegador desenvolvido.

Esse trabalho de conclusão de curso trará diversos benefícios para a empresa COPEL, pois apresenta uma solução que poderá ser implementada sem a necessidade de aquisição de GPS automotivo. Será possível a utilização de equipamentos já existentes na companhia.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos, sendo que o capitulo 1 apresenta as considerações iniciais, o objetivo e a justificativa.

O capítulo 2 apresenta o referencial teórico centrado em navegadores GPS e mapas vetoriais.

O capitulo 3 apresenta os materiais utilizados na modelagem e no desenvolvimento do sistema e o método empregado. Os materiais se referem às tecnologias e ferramentas utilizadas. O método apresenta as principais atividades realizadas para desenvolver o trabalho.

No capítulo 4 está o resultado obtido com a realização deste trabalho, que corresponde à modelagem e a implementação do sistema. A modelagem é apresentada por meio de diagramas e explicações textuais e a implementação por meio de telas e trechos de código utilizados no sistema.

O capítulo 5 apresenta a conclusão do trabalho.

2 NAVEGADOR GPS E MAPAS VETORIAIS

Este capítulo apresenta o referencial teórico do trabalho. Esse referencial está organizado em seções relacionadas que apresentam o conteúdo que foi utilizado como base de conhecimento para o desenvolvimento do Navegador GPS e conversão dos mapas vetoriais.

2.1 SOBRE A COPEL

A companhia paraense de energia, COPEL, iniciou suas atividades em 1954, sendo uma das maiores empresas do Paraná, atuando nas áreas de geração, transmissão e distribuição de energia, além de telecomunicações. Abriu seu capital em 1994 tornando-se uma empresa de economia mista controlada pelo governo do estado e pela iniciativa privada.

A COPEL atende diretamente a 4.009.281 unidades consumidoras em 396 municípios e 1.114 localidades (distritos, vilas e povoados) paranaenses. Nesse universo incluem-se 3,1 milhões de lares, 84 mil indústrias, 326 mil estabelecimentos comerciais e 374 mil propriedades rurais. O quadro de pessoal é integrado por 9.502 empregados. (COPEL, 2013).

Opera um abrangente e eficaz sistema elétrico com parque gerador próprio de usinas, linhas de transmissão, subestações, linhas e redes elétricas do sistema de distribuição e um moderno e poderoso sistema óptico de telecomunicações que integra as principais cidades do Estado.

A Figura 1 apresenta o logotipo atual da COPEL.

Figura 1 - Logotipo da COPEL
Fonte: COPEL (2012)

Efetua em média, mais de 70 mil novas ligações a cada ano, atendendo praticamente 100% dos domicílios nas áreas urbanas e passa de 90% nas regiões rurais. (COPEL, 2013).

2.2 EQUIPAMENTOS MÓVEIS – TABLETS

Tablet é um dispositivo eletrônico pessoal em formato de prancheta, capaz de realizar as mesmas tarefas que um computador *desktop* ou *notebook*. Por meio dele é possível navegar na Internet, editar documentos, executar jogos, comunicar-se, realizar videoconferências, explorar mapas, ver TV ou fazer ligações telefônicas.

Sua característica mais marcante é a possibilidade de interagir com os aplicativos clicando na tela (*touchscreen*) sendo esse o dispositivo de entrada principal, dispensando a utilização de canetas (TABLETS, 2013).

A Figura 2 apresenta um modelo de *tablet* da fabricante Samsung.

Figura 2 - *Tablet* da Samsung
Fonte: GALAXY (2013).

A popularização deste tipo de computador se deu com o lançamento do *iPad,* pela fabricante Apple, em 2010. Após o enorme sucesso do *iPad,* outras fabricantes passaram a desenvolver *Tablets* com recursos semelhantes utilizando principalmente o sistema operacional Android da Google. (TABLETS, 2013).

2.3 SISTEMA OPERACIONAL ANDROID

Android é um sistema operacional móvel baseado no núcleo do Linux. É desenvolvido e mantido pela Google em conjunto com a *Open Handset Alliance*, uma aliança de várias empresas.

O sistema operacional Android foi desenvolvido pela Android Inc., adquirida pelo Google em julho de 2005 (DEITEL et al., 2013, p. 4).

Em 2008 o Android transformou-se em *Open Source*, com o código publicado como *Android Open Source Project* (AOSP), sendo assim de código-fonte aberto. A Figura 3 ilustra o símbolo principal do Android, o *bugdroid*.

Figura 3 - Logotipo do Android
Fonte: BUGDROID (2013)

As principais características do Android referem-se a fácil adaptação aos diferentes tipos de *layouts* dos aparelhos e diferentes resoluções e tamanhos de telas, exigindo gráficos 2D e 3D; utiliza o banco de dados SQLite para armazenamento de dados; suporta diferentes tipos de formatos de mídia; é totalmente capaz de fazer uso de câmeras de vídeo, GPS, bluetooth; inclui um emulador capaz de permitir o *debugging*, depuração de aplicativos, sendo essa uma funcionalidade vital para o desenvolvimento de aplicativos (ANDROID, 2013).

Com o lançamento do SDK (*Software Development Kit*), ou seja Kit de Desenvolvimento de Software, foi possível o desenvolvimento de aplicativos em linguagem Java.

2.4 APLICATIVOS NAVEGADORES GPS

Um navegador GPS, é um aparelho ou aplicativo com a função de auxiliar o motorista na navegação, seja traçando rotas e indicando o melhor caminho, ou simplesmente exibindo o mapa das estradas e os pontos de interesse ao longo do trecho.

A funcionalidade principal do GPS, é receber os sinais emitidos por satélites, que contém a longitude, latitude e altitude e assim converter em informações relevantes para o motorista (GPS, 2013).

Um exemplo de navegação com visualização 3D pode ser visto na Figura 4.

Figura 4 - Aplicativo navegador GPS
Fonte: APLICATIVOGPS (2013)

O simples fato de saber “onde se está” já é algo importante para uma orientação em campo, sendo possível identificar quais as saídas ou caminhos mais próximas.

O GPS deve permitir a pesquisa por endereço e/ou pontos próximos citando como exemplo, postos de combustíveis mais próximos.

Outra funcionalidade em destaque é a utilização de navegação por voz onde o usuário pode falar para onde deseja ir e escutar orientações de manobras a serem seguidas enquanto percorre o trajeto sugerido pelo GPS.

2.5 OPENSTREETMAP

OpenStreetMap é um projeto colaborativo, que possui um formato próprio de mapa livre e editável do mundo todo. Os mapas são criados e mantidos por qualquer pessoa que queira colaborar.

O projeto foi fundado em 2004 por Steve Coast, sendo que inicialmente os mapas foram desenvolvidos com a utilização de coordenadas GPS anotadas em mapas impressos e posteriormente, a partir de 2006, empresas autorizaram a utilização de forma livre de seus dados cartográficos e fotos aéreas para atualizações do mapa mundial.

OpenStreetMap tem regras rigorosas quanto a utilização de mapas comerciais para alimentar seu mapa público, exigindo liberação formal dos dados antes da inclusão, pois sua licença é a *Creative Commons 2.0* (OPENSTREETMAP, 2013).

O mapa completo do projeto pode ser visualizado no endereço eletrônico (COAST, 2013), onde áreas especificas do mapa podem ser exportadas para analise e edição offline.

A Figura 5 apresenta a página inicial do projeto juntamente com a visualização da área do Paraná em destaque.

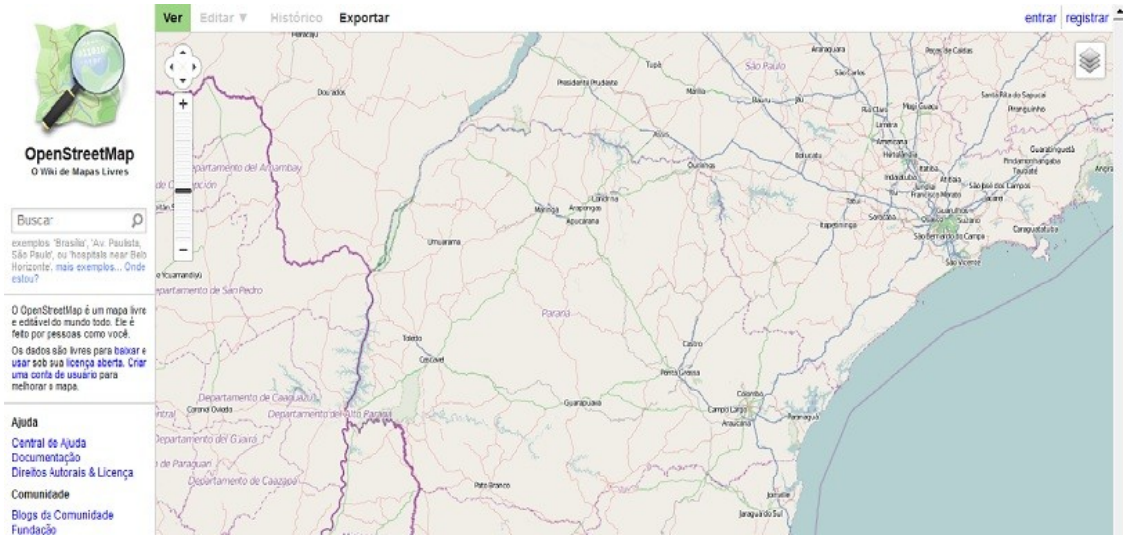

Figura 5 - Tela inicial do OpensStreetMap

Os dados mantidos no *OpenStreetMap* possuem a extensão de arquivo ".osm", sendo um formato XML, o que proporciona uma visualização rápida e legível de todo seu conteúdo, permitindo o desenvolvimento de mapas próprios.

2.6 MAPAS VETORIAIS E SEUS ELEMENTOS

Mapas vetoriais são compostos por coordenadas que representam pontos de interesse (postos de combustível, lojas, bancos), linhas (estradas de terra, rodovias, rios), polilinhas (áreas de mata, oceanos). Esses elementos em conjunto podem representar todos os objetos existentes no meio ambiente e assim proporcionar a visualização e navegação nos mapas desenvolvidos.

Os próximos itens demonstram os três formatos de mapas vetoriais utilizados para analise, conversão e criação dos mapas próprios para COPEL.

2.6.1 *Comma Separated Values* (.CSV)

Traduzido para o português, refere-se a valores separados por virgula. É um formato de arquivo que armazena dados tabelados, separados por um delimitador, a vírgula. Para separar as linhas é utilizado um caractere especial de quebra de linha.

Quando é necessário utilizar caracteres especiais (vírgula, quebra de linha, ou aspas) nos campos, então utiliza-se aspas duplas para o campo todo. Um exemplo já utilizando dados reais da COPEL, em formato ".csv", pode ser visualizado no Quadro 1.

```
NUMERO_UC,COD_LOCAL_UC,NIO_MEDIDOR,COORDENADAS,REGIONAL
51418517,2242,922807116,"[-52.1143124623513,-25.3690756322014]",SDO
81345259,5026,244733617,"[-53.8459049386436,-24.2805152768301]",SDO
51204614,4972,232200073,"[-51.9868981983845,-26.498073160458]",SDO
81529554,6184,301506523,"[-53.4825302868025,-25.8207912305414]",SDO
81176104,4444,291131763,"[-53.0320568992716,-26.1462776940661]",SDO
81371403,1810,264702160,"[-53.216389685935,-25.6348829415196]",SDO
```

Quadro 1 - Conteúdo de um arquivo ".csv" no padrão COPEL

A grande vantagem da utilização de arquivos ".csv", é que o arquivo é em formato de texto, permite uma enorme quantidade de informações e é independente de programas específicos para poder ser visualizado (CSV, 2013).

2.6.2 *Polish Map Format* (.MP)

É um formato polonês para mapas vetoriais baseados no modo texto corrido com quebra de linha após cada nova informação.

É usado principalmente como formato base para criação de mapas para aparelhos GPS Automotivos da marca GARMIN.

A formatação dos arquivos é em linhas, seguindo sequências de informações delimitadas em colchetes a cada agrupamento de característica. O sinal de igual (“=”) é utilizado para separar a chave do seu respectivo valor (MP, 2013).

2.6.2.1 Localidades

Para que seja possível a localização de endereços no GPS é necessário informar os relacionamentos entre as diferentes localidades no arquivo “.mp”.

O Quadro 2 apresenta um pequeno exemplo da formatação das localidades.

```
[Countries]
Country1=BRASIL
[END-Countries]

[Regions]
Region1=SÃO PAULO
CountryIdx1=1
Region2=PARANÁ
CountryIdx2=2
[END-Regions]

[Cities]
City3406=PATO BRANCO
RegionIdx3406=1
City1039=CASCAVEL
RegionIdx1039=1
City2351=CAMPINAS
RegionIdx2351=2
[END-Cities]
```

Quadro 2 - Representação no formato .MP para localidades

2.6.2.2 Pontos de Interesse

Para representar pontos de interesse é utilizada a chave principal [POINT]. O Quadro 3 apresenta um ponto de interesse em formato descritivo no padrão “.mp”.

```
[POINT]
Type=0x2f01
Label=Posto Kalhambeque
```

```
City=25
Data0=(-25.6706924438,-52.1251258850)
StreetDesc=Avenida Tupy
HouseNumber=675
Phone=(46)3220-2291
[END]
```

Quadro 3 - Codificação ".mp" para representar um ponto de interesse

Algumas informações precisam ser inseridas para representar o ponto de interesse (GPSMAPEDIT2, 2013):

- **Type:** É o atributo que indica o tipo do ponto, sendo inserido um valor hexadecimal. A Tabela 1 apresenta alguns tipos de pontos de interesse com sua respectiva categoria.

Tabela 1 - Significado dos tipos de pontos

Valor MP	Categoria	Descrição
0x650c	Água	Ilha
0x2a03	Alimento	Restaurante (Churrascaria)
0x2b01	Alojamento	Hotel/Motel
0x2c03	Atrações	Biblioteca
0x2f01	**Automóvel**	**Posto de combustível**
0x5905	Aviação	Aeroporto
0x3000	Cidades	Cidade Grande (2-5 milhões)
0x4000	Esportes	Campo de Golf
0x2f05	Serviços	Correios
0x2e01	Shopping	Compras/Shopping

- **Label:** É o nome do ponto de interesse que será utilizado para exibição na tela e fonte de pesquisa do local. Possui uma limitação de 80 caracteres.
- **City:** Corresponde ao número da cidade criada no início do mapa e serve para atrelar o ponto ao seu município sede possibilitando assim uma pesquisa mais precisa por proximidade.
- **Data0:** É a coordenada do ponto no mapa. O ponto de interesse deve possuir apenas 1 coordenada de latitude e longitude.
- **StreetDesc:** Nesta *tag* deve ser inserido o endereço do ponto quando existir.

- **HouseNumber:** Número do logradouro.
- **Phone:** Número do telefone.

2.6.2.3 Estradas

Para representar estradas utiliza-se a chave principal [POLYLINE] e ao término dos detalhamentos a *tag* [END]. O Quadro 4 contém um exemplo de polilinha utilizando os atributos necessários para representar uma estrada.

```
[POLYLINE]
Type=0x2
Label=PR-180
DirIndicator=0
RoadID=283
RouteParam=5,3,0,0,0,0,0,0,0,0,0,1
Data0=(-25.994,-53.0856),(-25.9958,-53.0856),(-26.0208,-53.0675),(-26.0234,-53.066)
Label2=Rodovia dos imigrantes
[END]
```

Quadro 4 - Codificação ".mp" para representar uma rodovia

Algumas informações devem ser inseridas para representar a estrada corretamente (GPSMAPEDIT2, 2013):

- **Type:** É o atributo que indica o tipo da estrada, sendo inserido um valor hexadecimal. A Tabela 2 apresenta os principais tipos de estradas.

Tabela 2 - Significado dos tipos de estradas

Valor MP	Descrição
0x0	Rodovia
0x1	Rodovia tronco/federal
0x2	**Rodovia principal/estadual**
0x3	Outras rodovias
0x4	Estrada principal
0x5	Estrada secundária
0x6	Rua residencial
0x7	Viela/Acesso particular
0x8	Rampa de rodovia, baixa velocidade
0x9	Rampa de rodovia, alta velocidade

0x0a	Rodovia sem pavimentação
0x0b	Conector da autoestrada maior
0x0c	Rotatória

- **Label:** É o nome da estrada que será utilizado para exibição na tela e fonte de dados para pesquisa de endereços. Possui uma limitação de 80 caracteres.
- **DirIndicator:** Utiliza valor binário (0 e 1) para sinalizar se a direção da estrada deve ser exibida quando se clica na tela do GPS.
- **RoadID:** É o número da estrada. Esse número é único para cada estrada.
- **RouteParam:** Para indicar atributos específicos de cada estrada são utilizados valores numéricos separados por "," (virgula). Seguindo o padrão da linha RouteParam=3,1,1,0,0,0,0,0,0,0,0,1 e analisando a codificação dos valores após o sinal de "=" (igual), destacam-se os seguintes atributos na ordem respectiva:

a. **Velocidade:** Para indicar a velocidade máxima da pista é utilizada uma escala numérica de 0 a 7. A Tabela 3 representa essa escala de conversão.

Tabela 3 - Conversão das velocidades

Valor MP	Valor real
7	128 Km/h
6	108 Km/h
5	93 Km/h
4	72 Km/h
3	**56 Km/h**
2	40 Km/h
1	20 Km/h
0	8 Km/h

b. **Classe:** O principal atributo para a realização do cálculo de roteamento é a determinação da classe da estrada. A Tabela 4 representa essa escala de conversão.

Tabela 4 - Significado das classes de estradas

Valor MP	Descrição
0	Rua Residencial/Beco/Estrada sem asfalto/Trilha
1	**Rotatória/Coletor**
2	Rua Arterial/Outras Rodovias
3	Rodovia Principal
4	Rodovia Principal/Rampa

c. **Sentido:** Para indicar que uma determinada estrada possui apenas um sentido único de trânsito é utilizado o valor numérico “1” (um) e quando a estrada permitir o trânsito em ambas as direções utiliza-se o valor “0” (zero) para representar.
d. **Pedágio:** Para estrada sem pedágio utiliza-se o valor numérico “0” (zero), já com pedágio utiliza-se “1”;
e. **Demais valores:** Referem-se a restrições de acesso para determinados meio de transporte onde segue-se o mesmo padrão de representação onde “0” (zero) para permitir o acesso e “1” (um) para estrada com restrição de acesso. O valor padrão é “0” permitindo o acesso a todos tipos de transporte. A sequência de tipos de veículos é a seguinte: ambulâncias; entregas; passeio; ônibus; táxi; pedestres; bicicletas e caminhões.

• **Data0:** Representa as coordenadas dos pontos de cada trecho. Uma estrada precisa no mínimo de 2 valores de latitude e longitude agrupados em parênteses e separados por “,” (vírgula). O padrão da coordenada é o WGS84.

• **Label2:** É o nome de referência da estrada sendo utilizado em caixas delimitadoras para dar destaque ao nome. Geralmente utilizado em rodovias federais e estaduais.

2.6.2.4 Áreas

Para representar áreas fechadas se utiliza a chave principal [POLYGON] e ao término dos detalhamentos a *tag* [END]. As áreas não são utilizadas para pesquisa, sendo apenas utilizadas como objetos de visualização.

O Quadro 5 contém um exemplo de polígono e seus atributos.

```
[POLYGON]
Type=0x20
Label=RESERVA BIOLÓGICA DO PARANÁ
Data0=(-24.94109,-53.51151),(-24.94087,-53.51202),(-24.94083,-53.51166)
[END]
```

Quadro 5 - Representação de uma área de preservação ambiental

Algumas informações devem ser inseridas para representar o polígono corretamente (GPSMAPEDIT2, 2013).

- **Type:** É o atributo que indica o tipo de área, sendo inserido um valor hexadecimal. A Tabela 5 apresenta os principais tipos de polígonos.

Tabela 5 - Significado dos tipos

Valor MP	Categoria	Descrição
0x0028	Água	Mar/Oceano
0x0046	Água	Rio principal > 1 Km
0x0018	Esporte	Campo de Golf
0x0020	**Terra**	**Parque estadual**
0x0016	Terra	Parque nacional
0x000c	Feito pelo Homem	Complexo industrial

- **Label:** É o nome da área que será utilizado para exibição na tela. Possui uma limitação de 80 caracteres.
- **Data0:** Representa as coordenadas dos pontos de cada trecho. Uma estrada precisa no minimo de 2 valores de latitude e longitude agrupados em parênteses e separados por “,” (vírgula). O padrão da coordenada é o WGS84.

2.6.3 *OpenStreetMap* (.OSM)

O formato OSM, é um sistema de marcação livre que utiliza o esquema XML como base para manter uma forma legível de estrutura transparente e independente

da máquina ou sistema operacional que vai ser executado. Pode conter um número ilimitado de dados.

Basicamente um arquivo ".osm" contém instâncias, com base em elementos primitivos (nós, linhas, polígonos e relacionamentos) para representar todas formas existentes em um mapa vetorial (OSM, 2013).

Um elemento primitivo é um componente básico do *OpenStreetMap* ao qual tudo pode ser representado e definido (ELEMENTOS, 2013).

2.6.3.1 Nó (*Node*)

Representa um único ponto geoespacial utilizando uma latitude e longitude. Podem ser utilizados para apresentar características de pontos independentes ou na definição de caminhos de uma estrada.

A Figura 6 exemplifica o ponto no padrão *OpenStreetMap*.

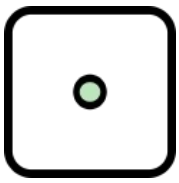

Figura 6 - Ponto OMS XML

O Quadro 6 demonstra a representação de um ponto como exemplo.

```
<node id="183" version="1" changeset="12370" lat="54.090066" lon="12.253938"
user="rebelato" uid="75625" visible="true" timestamp="2013-02-12T09:43:19Z">
   <tag k="name" v="UTFPR – Pato Branco"/>
   <tag k="amenity" v="university"/>
</node>
```

Quadro 6 - Codificação OSM XML para um ponto de interesse

2.6.3.2 Polilinha (*Polyline*)

Polilinha é usada para representar características lineares (abertas e fechadas) ou polígonos (áreas fechadas). Estradas, rios, linhas ferroviárias e trilhas são representados por polilinhas abertas. Já polilinhas fechadas, quando o ponto inicial conecta com o ponto final, representam círculos ou perímetros vazados sem

conteúdo interno. Áreas são polígonos fechados que representam, açudes, mata, construções e demais propriedades. A Figura 7 demonstra as polilinhas e área no padrão *OpenStreetMap*.

Figura 7 - Polilinha aberta, Polilinha fechada e Área OSM XML

O Quadro 7 representa uma rodovia utilizando uma polilinha aberta.

```
<way id="26659127" user="rebelato" uid="5598" visible="true" version="5" changeset="414260"
timestamp="2013-02-12T11:47:08Z">
    <nd ref="2924"/>
    <nd ref="2988"/>
    <nd ref="2617"/>
    ...
    <tag k="highway" v="trunk"/>
    <tag k="name" v="BR-277"/>
</way>
```

Quadro 7 - Codificação OSM XML para representar uma rodovia

2.6.3.3 Relacionamento (*Relation*)

Uma relação consiste de uma lista ordenada de nós, polilinhas e outras relações. É utilizada para representar relações lógicas ou geográficas entre outros elementos. Limites territoriais, linhas de ônibus e restrições de manobra são representadas com a utilização de relacionamentos. A Figura 8 exemplifica o relacionamento no padrão *OpenStreetMap*.

Figura 8 - Relação OSM XML

O Quadro 8 apresenta a utilização de uma relação para representar o itinerário de uma linha de ônibus urbano.

```
<relation id="5668" user="rebelato" uid="5619" visible="true" version="28" changeset="69476"
timestamp="2011-01-12T14:23:49Z">
   <member type="node" ref="294942" role=""/>
   <member type="way" ref="4579143" role=""/>
   ...
   <tag k="name" v="UTFPR - Alvorada"/>
   <tag k="operator" v="Motorista José"/>
   <tag k="ref" v="123"/>
   <tag k="route" v="bus"/>
   <tag k="type" v="route"/>
</relation>
```

Quadro 8 - Representação de uma relação entre pontos e polilinhas

2.6.3.4 Marcações (*Tags*)

Outra representação importante são as "*tags*". Comumente denominadas etiquetas, são pequenas unidades de dados anexadas aos elementos primitivos. Consistem em dois campos de formato livre textualmente, uma "chave" e um "valor", cada um sendo sequências de caracteres que representam características importantes dos elementos.

Tags podem representar a velocidade máxima de uma estrada ou o telefone de uma pizzaria, bastando apenas utilizar chaves sugeridas pela comunidade *OpenStreetMap*, que são interpretadas por todos os sistemas de visualização de mapas.

O Quadro 9 exemplifica a utilização de "*tags*" para incrementar as informações de uma estrada formada por uma polilinha.

```
<way id="26659127" user="rebelato" uid="5598" visible="true" version="5" changeset="414260"
timestamp="2013-02-12T11:47:08Z">
   <nd ref="2924"/>
   <nd ref="2988"/>
   <nd ref="2617"/>
   ...
   <tag k="highway" v="trunk"/> // Tipo: Rodovia Federal
   <tag k="oneway" v="no"/>     // Não é sentido único
   <tag k="lanes" v="3"/>       // Quantidade de pistas
   <tag k="name" v="BR-277"/>   // Nome da pista
   <tag k="maxspeed" v="110"/>  // Velocidade máxima
</way>
```

Quadro 9 - Várias tags representando características da rodovia

2.6.3.5 Representação OSM XML

Utilizando os elementos do padrão OSM é possível representar tudo que existe em um ambiente e incluir o maior número de atributos possíveis para que esses sejam utilizados pelo software de renderização dos mapas.

A Tabela 6 exibe alguns exemplos de elementos visuais que podem ser representados em um mapa vetorial.

Tabela 6 - Representação gráfica de alguns elementos utilizados em OSM XML

Chave	Valor	Tipo	Visual mapa	Foto real
barrier	fence	Cerca de arrame farpado. Divisa de propriedade rural.		
boundary	administrative	Limite de município, estado ou país.		
highway	trunk	Rodovia. Tipicamente utilizada para rodovias estaduais.		
sport	golf	Ponto (nó) indicando o nome do campo de golf.		

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados na realização deste trabalho. Os materiais se referem às tecnologias como linguagens e ferramentas para a modelagem e a implementação do sistema. O método contém as etapas com os principais procedimentos utilizados para o desenvolvimento do sistema, abrangendo o levantamento dos requisitos e a definição das tecnologias a serem utilizadas nos testes.

3.1 MATERIAIS

As ferramentas e as tecnologias utilizadas para o desenvolvimento do sistema foram:

a) **UML** para realizar a modelagem do sistema;
b) **Java** como a linguagem de programação;
c) **Android SDK** (*Software Development Kit*) como compilador mobile;
d) **SQLite** para o banco de dados do navegador;
e) **Eclipse Juno** como IDE de desenvolvimento;
f) **PostgreSQL** para o banco de dados;
g) **Notepad++** como visualizador dos mapas em formato texto (.csv);
h) **GPSMapEdit** como visualizador dos mapas no estágio intermediário (.mp);
i) **JOSM** para visualizador dos mapas no formato final (.osm).

A seguir são apresentadas algumas características referentes a cada material utilizado.

3.1.1 UML

A Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML) é a linguagem utilizada para representar graficamente todos os elementos envolvidos em um sistema orientado a objetos.

> "A UML não é uma metodologia, mas sim uma linguagem de modelagem gráfica para descrever um projeto de software. A UML surgiu da união dos métodos Booch e OMT em 1994, quando seus autores (Grady Booch e James Rumbaugh) juntaram forças através da *Rational Corporation* e unificaram completamente seus trabalhos. (BOOCH, 2000)".

Na UML existem vários modelos para representar a compreensão do sistema, e não um modelo único que represente a totalidade do sistema. Para representar o sistema é possível utilizar diferentes diagramas, ou seja, representações gráficas, que dividem-se em três famílias, segundo Wazlawick (2011):

a) Diagramas estruturais: diagramas de pacotes, classes, objetos, estrutura composta, componentes e distribuição;
b) Diagramas comportamentais: diagramas de caso de uso, atividades e máquina de estados;
c) Diagramas de interação: diagramas de pacotes de comunicação, sequência, tempo e visão geral de integração.

As principais vantagens na utilização da UML são a padronização; independência na linguagem utilizada para o desenvolvimento do sistema; diversidades de visões; ser extensível e adaptável.

A UML é utilizada desde sistemas simples aos mais complexos. Atualmente ela está sendo empregada nas mais diversas áreas, como por exemplo: serviços de bancos e finanças, telecomunicações, transportes, vendas, sistemas distribuídos, entre outros.

3.1.2 Linguagem Java

Java é uma linguagem de programação de alto nível orientada a objetos, cujo código-fonte gerado é organizado em classes.

Desenvolvida a partir da década de 90 por uma equipe da Sun Microsystem, chefiada por James Goling, é diferente das linguagens convencionais pois sua execução acontece em uma máquina virtual (JAVA, 2013). Uma máquina virtual permite que a aplicação seja executada em diferentes sistemas operacionais sem a necessidade de reescrever o código.

Para execução do programa o código de máquina é gerado por um compilador java, o javac. O código gerado é conhecido por *bytecode*, sendo esse traduzido pela máquina virtual java (JVM).

Classes em Java podem ser concretas ou abstratas permitindo assim a utilização dos conceitos de polimorfismo e herança, trazendo assim a reutilização,

clareza e fácil manutenção do código. Java oferece suporte a operações em Banco de Dados.

Java possui desalocação de memória automática por processo coletor de lixo (*garbage collection)*. Esse processo otimiza o uso da memória durante a execução do programa (JAVA, 2013).

Java é dividido em três grandes edições:

1. ***Java Standard Edition* (JSE) :** É a tecnologia Java para computadores pessoais, *notebooks* e arquiteturas com poder de processamento e memória consideráveis.
2. ***Java 2 Mobile Edition* (JME) :** É a tecnologia Java para dispositivos móveis. Possui APIs bem simples e leves para economizar espaço, memória e processamento.
3. ***Java 2 Enterprise Edition* (JEE) :** É a tecnologia Java para aplicações corporativas que podem estar na Internet ou não. Possui um grande número de APIs onde a segurança é a principal preocupação. É ideal para a construção de servidores de aplicação, integração de sistemas ou distribuição de serviços para terceiros.

A Figura 9 apresenta a um exemplo da linguagem Java em uma classe utilizada para representar um serviço, onde são apresentados os atributos que são referenciados em um construtor.

```
package net.osmand.plus.servicos;

import java.io.Serializable;

public class ServicoPoint implements Serializable {
    private static final long serialVersionUID = 7296543008297714666L;
    private String nome;
    private String categoria = "";
    private double latitude;
    private double longitude;
    private boolean gravado = false;


    public ServicoPoint(){
    }

    public ServicoPoint(double latitude, double longitude, String name, String category) {
        this.latitude = latitude;
        this.longitude = longitude;
        this.categoria = category;
        this.nome = name;
    }
```

Figura 9 - Exemplo da linguagem Java utilizada

Para o desenvolvimento do navegador GPS foi optado em utilizar a linguagem Java, devido as vantagens citadas acima e ao fato de o sistema operacional Android a utilizar como padrão de programação.

3.1.3 Android SDK

O SDK é um kit de desenvolvimento para o Android que proporciona ferramentas e chamadas via API na linguagem Java, para desenvolvimento de programas.

A Figura 10 apresenta a tela principal do Android SDK Manager, que possibilita que o aplicativo desenvolvido para Android seja testado em diferentes versões da API, e que as novas funcionalidades da linguagem Android sejam atualizadas de maneira prática.

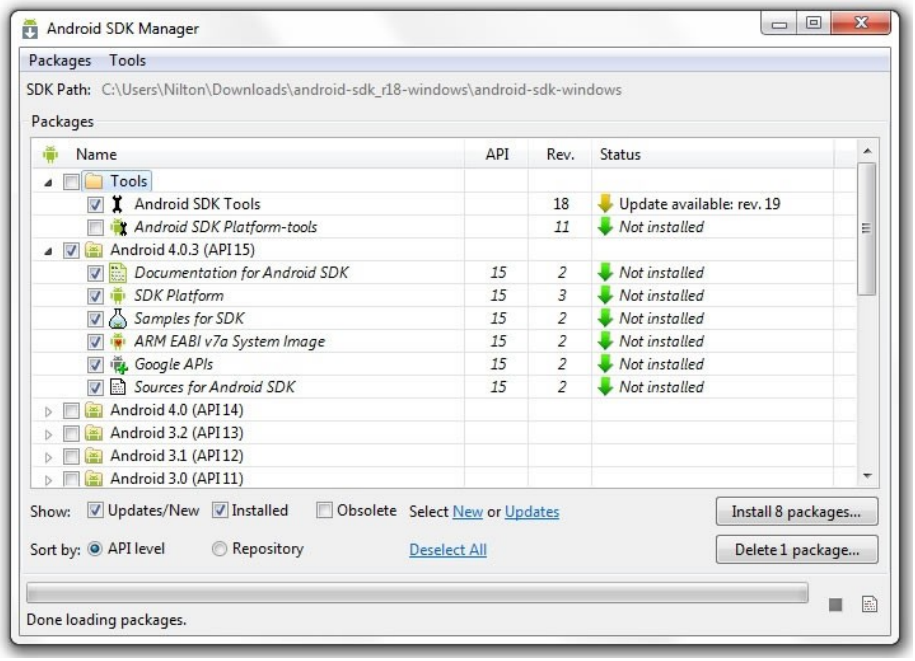

Figura 10 - Tela do Android SDK Manager

Para o desenvolvimento do navegador GPS foi utilizada a API 2.2 do Android, sendo copiadas todas as versões mais recentes para poder utilizar conjuntamente funcionalidades novas e antigas do Android.

3.1.4 SQLite

SQLite é uma ferramenta de manipulação e armazenamento de dados através de comandos SQL. É simples e eficiente, desenvolvida na linguagem C padrão (ANSI).

Na prática, o SQLite funciona como um “mini-SGB”, capaz de criar um arquivo em disco e ler e escrever diretamente sobre este arquivo. O arquivo possui a extensão “.db” e é capaz de manter diversas tabelas.

Foi utilizado como banco de dados no navegador GPS desenvolvido em Android, pois é um software gratuito, multiplataforma e não necessita instalação, configuração ou administração.

3.1.5 Eclipse Juno

O Eclipse é um ambiente integrado de desenvolvimento (IDE) que começou a ser desenvolvido no final dos anos 90 pela IBM. Essa ferramenta é *open source*, ou seja, de código livre. É um IDE comumente utilizado para desenvolvimento Java, no entanto, por meio de *plugins*, pode ser usado para desenvolver várias linguagens (ECLIPSE, 2013). A Figura 11 exibe a tela do Eclipse em utilização.

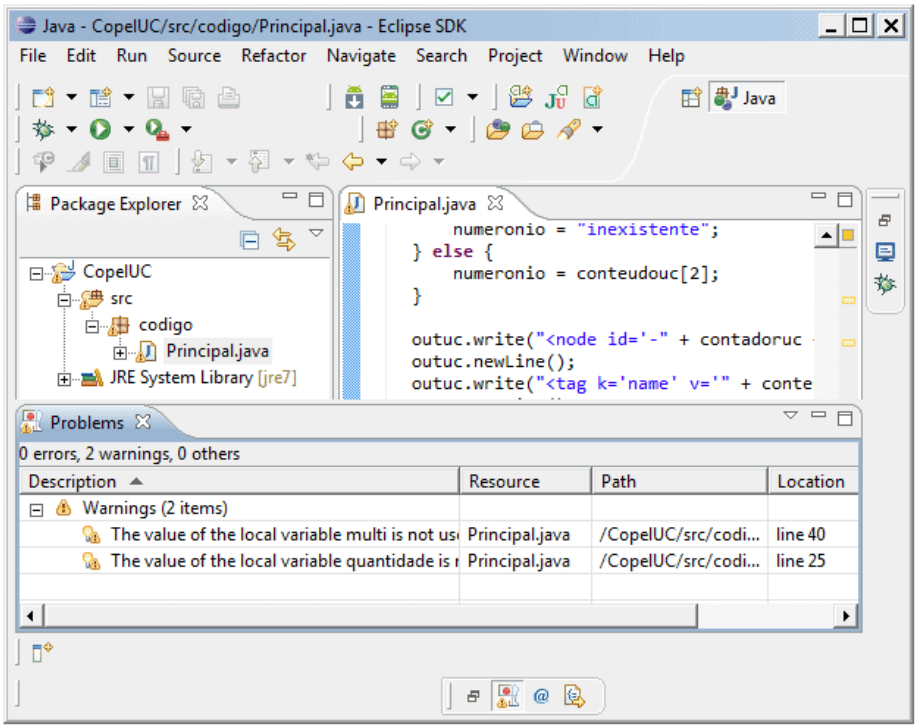

Figura 11 - Tela demonstrativa da IDE Eclipse

Como o Eclipse possui integração com Android e tem diversas ferramentas voltadas para a linguagem Java, esse aplicativo foi utilizado para o desenvolvimento dos conversores do mapas e para o navegador GPS.

3.1.6 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional (SGBDOR), desenvolvido como projeto de código livre, desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley, e possui mais de 15 anos. É um banco de dados extremamente robusto e confiável, além de extremamente flexível e rico e recursos (POSTGRESQL, 2013).

O gerenciador conta com recursos como, consultas complexas, chaves estrangeiras, integridades transacional, *triggers* e principalmente o suporte a dados espaciais com a utilização do módulo PostGIS.

A tela principal do PostgreSQL é exibida na Figura 12 abaixo.

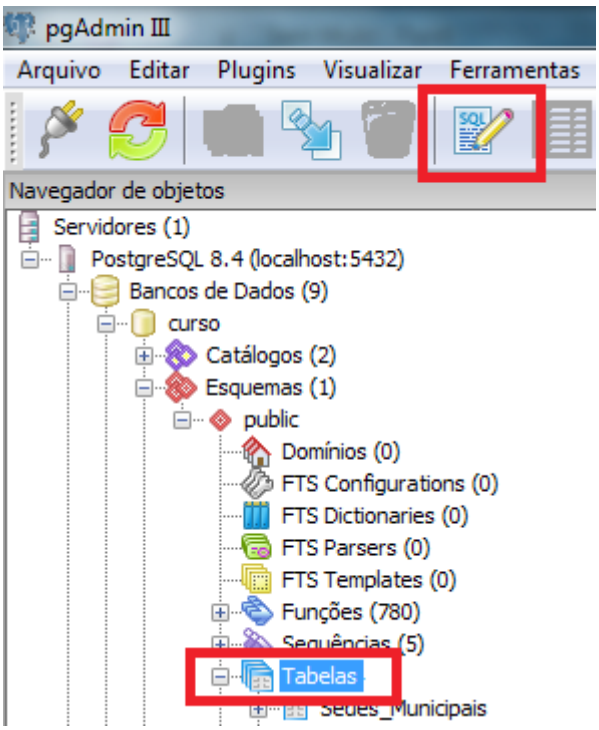

Figura 12 - Tela do PostgreSQL

PostgreSQL foi utilizado para manipulação dos dados que foram utilizados para gerar os mapas vetoriais, já que permite a utilização de dados geoespaciais (dados com coordenadas).

3.1.7 Notepad++

É um editor de textos e códigos fonte completo que suporta as mais diversas linguagens de programação (NOTEPAD, 2013).

Possibilita a abertura de arquivos com mais de 65 mil linhas e possibilita a execução de comando de localização e substituição de caracteres levando em consideração expressões complexas de pesquisa e renomeação de palavras. Essa ferramenta é executada no ambiente *Microsoft Windows* sendo baseada na licença GNU (*General Public License*).

A Figura 13 apresenta a tela principal do Notepad++.

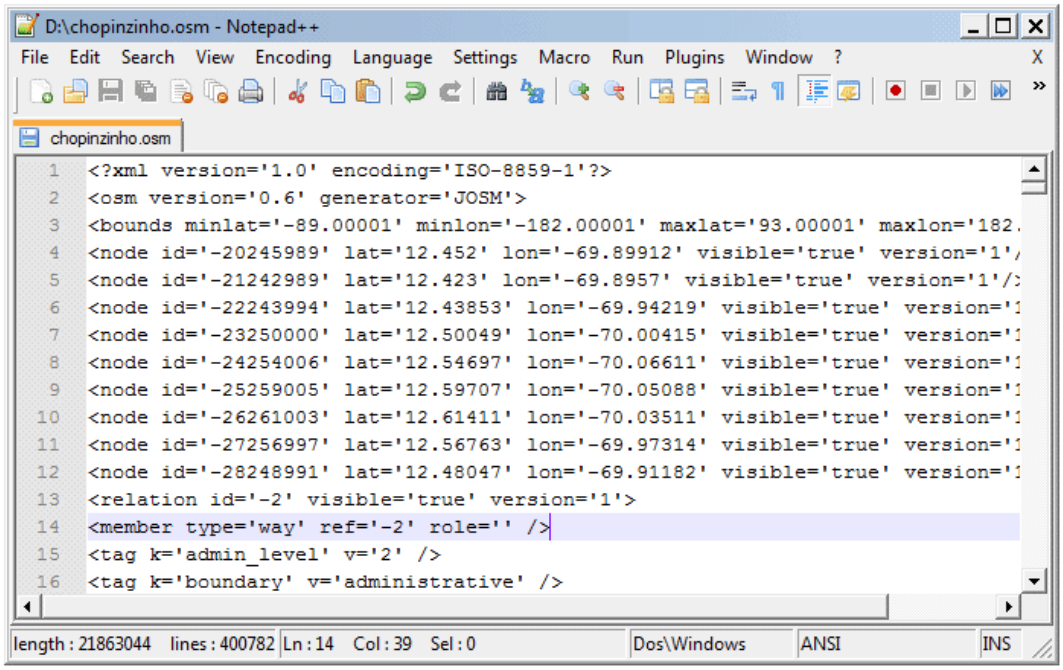

Figura 13 - Tela principal da ferramenta NotePad++

Esse aplicativo foi utilizado para visualização e manipulação dos mapas vetoriais em formato texto ou XML.

3.1.8 GPSMapEdit

O GPSMapEdit é um aplicativo que permite a visualização de mapas vetoriais e seu formato nativo é ".mp" (*polish format*). É possível editar e categorizar os eixos viários alterando sua categoria, velocidade limite, restrições de manobra e muitas outras opções (GPSMAPEDIT1, 2013).

Com o GPSMapEdit é possível realizar verificações de consistência do mapa conferindo as conexões das estradas e o caminho sugerido pelo roteamento simulado.

Abaixo um exemplo da visualização do mapa rural da COPEL na Figura 14.

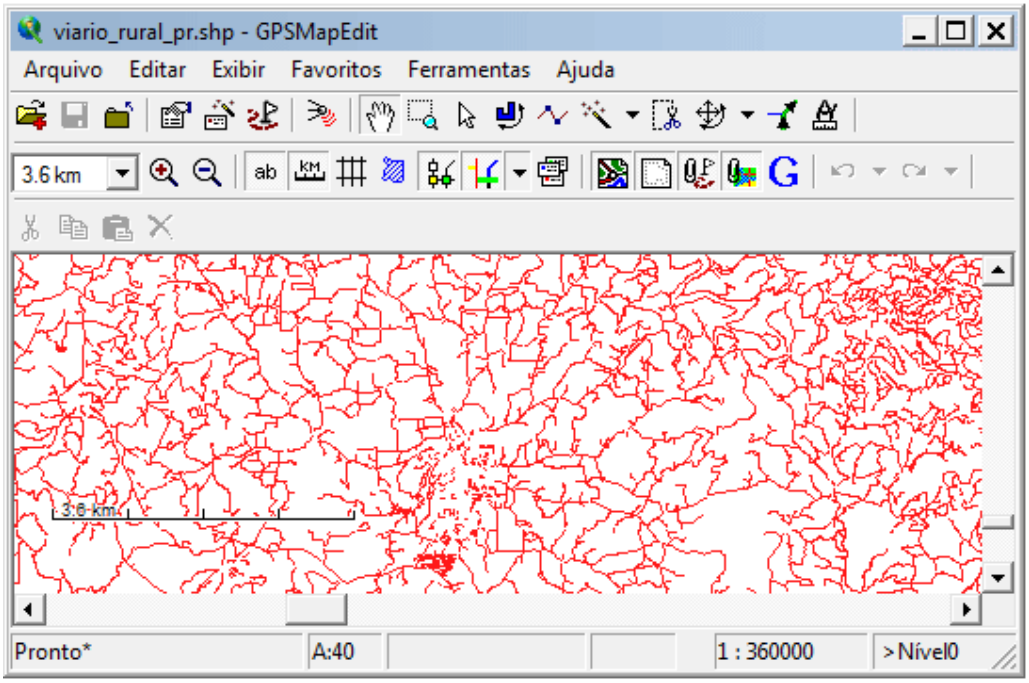

Figura 14 - GPSMapEdit com mapa vetorial aberto

Esse aplicativo foi utilizado para visualizar graficamente os mapas, no formato ".mp", no estágio intermediário de processamento, possibilitando assim um reconhecimento visual de possíveis falhas.

3.1.9 JOSM

Java OpenStreetMap Editor (JOSM), desenvolvido por Immaniel Scholz, é um editor de dados no formato ".osm", formato esse original do site "OpenStreetMap" (JOSM, 2013).

É possível carregar mapas de todos os tamanhos e visualizar possíveis falhas de conexão entre os nós de roteamento ou informações incorretas nas descrições nos pontos.

Na Figura 15 é possível evidenciar a facilidade em visualizar mapas completos.

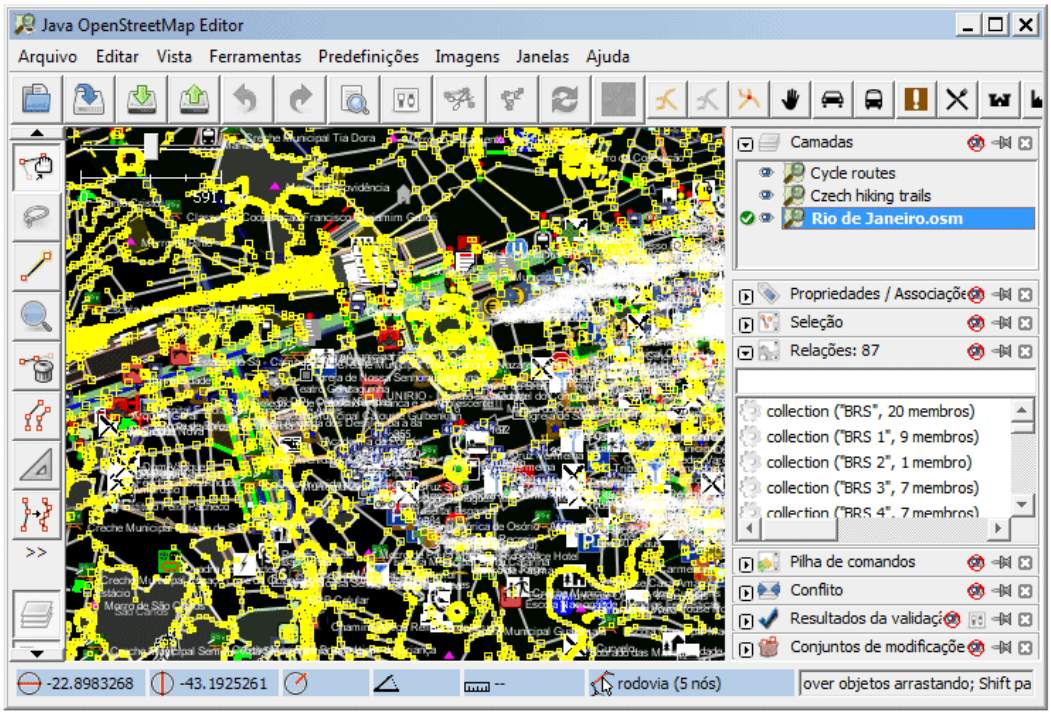

Figura 15 - JOSM exibindo o mapa vetorial do Rio de Janeiro

JOSM foi utilizado para visualizar graficamente os mapa, no formato ".osm", após serem concluídos. A visualização é muito importante para garantir a confiabilidades das informações convertidas.

3.2 MÉTODO

O método consiste nas atividades que foram realizadas para o desenvolvimento deste trabalho de conclusão. Os procedimentos seguidos foram baseados no modelo sequencial linear (PRESSMAN, 2005) sendo: definição dos requisitos, análise e projeto, implementação e testes.

a) Definição dos requisitos – Os requisitos foram definidos com base no trabalho realizado como estágio, comparando as funcionalidades do navegador GPS automotivo desenvolvido com as funcionalidades solicitadas para o aplicativo Android. Aliando a isso as experiências obtidas em campo durante os serviços realizados para a COPEL, foram definidos os requisitos funcionais e não funcionais para o Navegador GPS.

b) Análise e projeto - Para a análise foram desenvolvidos dois diagramas de casos de uso, sendo um para o Navegador GPS e outro para o Gerador de Mapas. Foram também gerados diagramas de sequencia e de classes para o

Navegador GPS. Representando a estrutura do banco de dados, foi gerado o diagrama de entidades e relacionamentos para ambos os sistemas.

c) Implementação - Nesta fase do projeto foi realizada a codificação do aplicativo navegador GPS e os testes de conversão dos mapas vetoriais.

d) Testes - Os testes foram realizados em campo, em situações reais, pelos próprios funcionários da COPEL e foram de vital importância para o aperfeiçoamento da ferramenta.

4 RESULTADO

Este capítulo apresenta o que foi obtido com o desenvolvimento do aplicativo Navegador GPS e o Gerador dos mapas vetoriais personalizados, baseados nos dados cartográficos e elétricos da COPEL.

Em um primeiro momento, serão descritas as principais funcionalidades do Navegador GPS, sendo demonstrados os requisitos levantados, Diagrama de Caso de Uso, Diagrama de Classes e Diagrama de Sequência, além do Diagrama de Entidade e Relacionamento, utilizado para modelar o banco de dados.

Na sequência será apresentado o processo de conversão, do formato ".mp" para ".osm" dos mapas vetoriais e sua conversão para o formato binário final de visualização no Navegador GPS.

Por fim será apresentada a utilização em campo do Navegador GPS devidamente instalado em um *Tablet* e sendo utilizado em conjunto com os mapas binários compilados pelo conversor de mapas desenvolvido.

4.1 APRESENTAÇÃO DO NAVEGADOR GPS

O aplicativo Navegador GPS tem como finalidade principal propiciar a rápida visualização de mapas vetoriais que também foram desenvolvidos para este trabalho. Com esse sistema os funcionários da COPEL poderão localizar-se em campo, visualizando a rede elétrica mais próxima, assim como as estradas urbanas e rurais que deve seguir, utilizando um dispositivo móvel.

No navegador GPS desenvolvido, há a possibilidade de pesquisar todos os pontos da rede elétrica do estado do Paraná bastando apenas digitar o código único de cada elemento.

A funcionalidade principal do aplicativo é a utilização dos mapas vetoriais próprios e consequentemente o cálculo do roteamento, sugerindo assim o melhor caminho a ser seguido do ponto de partida até o destino final.

4.2 MODELAGEM DO SISTEMA

O presente trabalho foi modelado baseando-se principalmente na Análise Orientada a Objetos, utilizando, portanto, os conceitos e diagramas da UML.

Segundo Booch (2000), UML é uma linguagem padrão para elaboração da estrutura de projetos de software, podendo ser empregada para visualização, especificação, construção e documentação dos artefatos de sistemas de software.

Para o desenvolvimento do sistema primeiramente foi realizado o processo de levantamento dos requisitos, sendo possível identificar as principais necessidades para o sistema em questão, conforme apresentado no item 3.2.

Após o levantamento dos requisitos seguida de sua classificação, de acordo com requisitos funcionais e não-funcionais, foram realizados os diagramas de casos de uso, diagrama de sequencia, diagrama de classes e o diagrama de entidades e relacionamentos.

4.2.1 Requisitos Funcionais e Não Funcionais

A analise de requisitos busca descrever as operações que o sistema realizará e as restrições sobre essas operações. Os requisitos funcionais são as operações que descrevem as funcionalidades do sistema e suas ações para cada entrada, ou seja, são as funcionalidades identificadas como necessárias ao sistema que será desenvolvido. As restrições sobre as funcionalidades são denominadas requisitos não-funcionais (WAZLAWICK, 2011). As tabelas de 7 a 10 apresentam os requisitos funcionais e restrições resultantes dessa análise, com base em Wazlawick (2011).

Tabela 7 - Documentação: Requisitos: Visualização

F1 Visualização do Mapa		**Oculto ( )**		
Descrição: O Navegador GPS deve permitir que o funcionário da Copel possa visualizar o mapa completo das estradas e rede elétrica da COPEL.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 1.1 Controle de acesso	A opção só pode ser acessada por equipamentos móveis autorizados pela COPEL.	Segurança	( )	(x)
NF 1.2 Coordenada inicial	O funcionário informa o ponto inicial para posicionar o mapa ou aguarda o receptor GPS captar o sinal de satélite.	Interface	(x)	( )
NF 1.3 Níveis de zoom	O funcionário deve manipular os botões de (+) e (-) para acionar o carregamento do mapa.	Interface	( )	(x)

Tabela 8 - Documentação: Requisitos: Pesquisar Pontos

F2 Pesquisar Pontos		**Oculto ( )**		
Descrição: Pesquisar pontos possibilita encontrar qualquer elemento da rede elétrica da COPEL que esteja contido nos mapas vetoriais gerados.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 2.1 Consistência dos dados	O Navegador precisa verificar a indexação a cada inicialização do sistema.	Segurança	( )	(x)
NF 2.2 Coordenada atual	O receptor GPS precisa fornecer a coordenada atual do aparelho para iniciar a pesquisa radial dos pontos próximos.	Interface	(x)	( )
NF 2.3 Últimas pesquisas	Todos as pesquisas realizadas com resultados válidos serão guardadas internamente no Navegador GPS.	Implementação	( )	(x)

Tabela 9 - Documentação: Requisitos: Pesquisar Endereços

F3 Pesquisar Endereços		**Oculto ( )**		
Descrição: O funcionário tem a possibilidade de pesquisar logradouros informando a cidade, rua e rua transversal.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF 3.1 Consistência dos dados	O Navegador precisa verificar a indexação a cada inicialização do sistema.	Segurança	( )	(x)
NF 3.2 Coordenada atual	O receptor GPS precisa fornecer a coordenada atual do aparelho para iniciar a pesquisa radial dos pontos próximos.	Interface	(x)	( )
NF 3.3 Informações detalhadas	O funcionário pode visualizar detalhes do endereço pesquisado e saber a distância prévia até o local.	Interface	(x)	( )

Tabela 10 - Documentação: Requisitos: Iniciar Navegação

F4 Iniciar Navegação		**Oculto ( )**		
Descrição: Esta opção permite que o funcionário informe o ponto de destino e ative o cálculo de roteamento, podendo assim seguir o trajeto indicado pelo Navegador GPS.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**

NF 4.1 Controle de acesso	Apenas será permitida a navegação após uma verificação se o aparelho está autorizado no cadastro da COPEL.	Segurança	( )	(x)
NF 4.2 Recálculo de rota	O Navegador GPS deve recalcular a rota conforme o funcionário siga um trajeto diferente do trajeto inicialmente sugerido pelo Navegador.	Desempenho	( )	(x)
NF 4.3 Informações de navegação	O Navegador GPS deve informar na tela as informações básicas como: - Tempo restante até o destino; - Velocidade atual; - Próxima manobra a ser executada.	Interface	( )	(x)
NF 4.4 Gravar roteiro	Durante o percurso o trajeto seguido deve ser gravado em arquivo ".gpx".	Implementação	(x)	( )

4.2.2 Diagrama de Casos de Uso

Para representar as funcionalidades disponíveis no Navegador GPS, que podem ser utilizadas pelo Funcionário da COPEL, foi criado o Diagrama de Casos de Uso conforme a Figura 16.

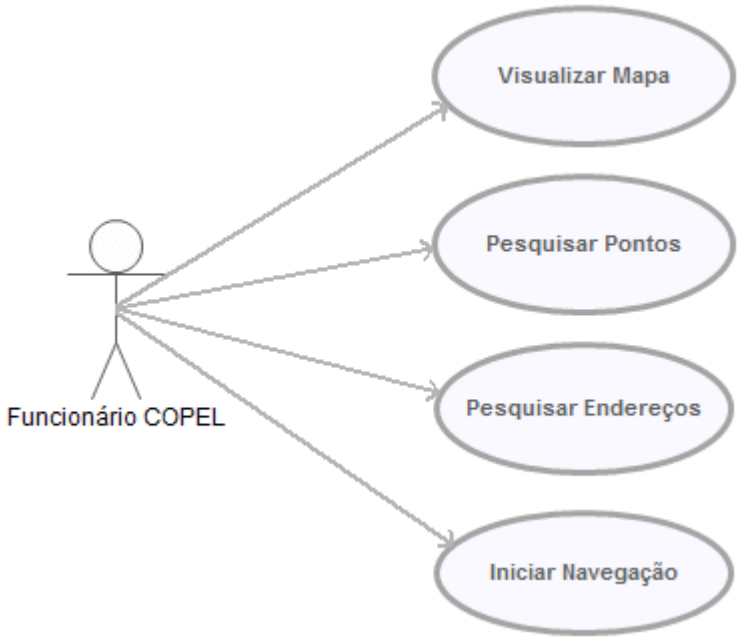

Figura 16 - Documentação: Diagrama de Casos de Uso para o Navegador GPS

Pelo diagrama é possível perceber que todas as funcionalidades são iniciadas pelo Funcionário realizando operações de consulta de informações conforme apresentado na Tabela 11.

Tabela 11 - Documentação: Descrições do Caso de Uso do Navegador GPS

CASO DE USO	DESCRIÇÃO
Visualizar Mapa	O Funcionário aciona a visualização do mapa. É carregado o mapa binário na memória interna do *Tablet* e acionado o renderizador de vetores. São exibidos os elementos cartográficos (estradas e áreas) e os pontos de interesse (rede elétrica da COPEL).
Pesquisar Pontos	Responsável pela ação de pesquisa de códigos operacionais encontrando o valor informado pelo Funcionário dentre os milhões de pontos inseridos no mapa vetorial gerado.
Pesquisar Endereços	Fornece a funcionalidade de pesquisa onde o Funcionário informa o nome da localidade e o nome do logradouro e o Navegador GPS retorna a localização na tela.
Iniciar Navegação	Quando o sistema identifica a localização atual do Funcionário e o ponto de destino já foi pesquisado e marcado na tela essa opção possibilita a inicialização do cálculo de roteamento surgindo assim o melhor trajeto a ser seguido na tela do *Tablet*.

A Figura 17 apresenta o Diagrama de Caso de Uso criado para o Gerador de Mapas, por se tratar de um sistema complementar ao Navegador GPS, mas que possui funcionalidades próprias.

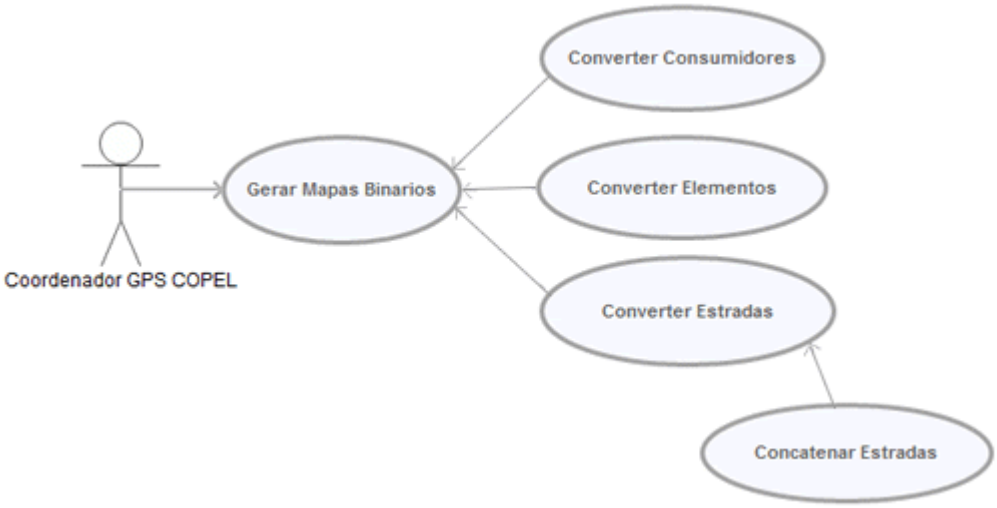

Figura 17 - Documentação: Diagrama de Casos de Uso para o Gerador de Mapas

Todo o processo de geração dos mapas baseia-se no processo de conversão de dados, realizando a leitura de informações provenientes do banco de dados da COPEL.

A Tabela 12 apresenta uma descrição sucinta dos caso de uso do Gerador de Mapas Vetoriais.

Tabela 12 - Documentação: Descrições do Caso de Uso do Gerador de Mapas

CASO DE USO	DESCRIÇÃO
Concatenar Estradas	Responsável por unir dois mapas diferentes em apenas um mapa no formato ".mp". Esse processo realiza a leitura de dados vetoriais de estradas corrigindo assim suas informações e unindo as linhas para gerar um único mapa cartográfico.
Converter Estradas	Realiza a conversão do mapa de estradas concatenadas do formato ".mp" para ".osm", analisando a consistência das informações e modificando alguns atributos para que o mapa seja exibido de maneira correta na tela do *Tablet*.
Converter Elementos	Converte os outros elementos importantes para criação do mapa como localidades, hidrografia e vegetação. Gera um mapa unificado no formato ".osm".
Converter Consumidores	O dados dos consumidores estão no formato ".csv" em milhares de linhas e precisam ser convertidos para o formato ".osm". Os dados passam por uma formatação de suas informações visando ser utilizados na pesquisa do navegador.
Gerar Mapas Binários	Realiza o processo final de leitura dos mapas no formato ".osm" e conversão para o formato final ".bin" para que os mapas possam ser interpretados e utilizados no Navegador GPS.

4.2.3 Diagrama de Sequencia

Os diagramas de sequência demonstram a ordenação das mensagens por tempo. É utilizado para mostrar as trocas de mensagens entre os objetos.

Em um diagrama de sequência, os objetos são colocados em forma de caixa na parte superior de uma linha tracejada vertical. Essa linha é chamada de linha de

vida do objeto que representa a sua duração. As flechas contidas entre as linhas pontilhadas de dois objetos são as mensagens de envio e retorno.

Segundo WAZLAWICK (2011) o Diagrama de Sequência é “útil para representar a sequência dos eventos de sistema em um cenário de um caso de uso”.

Na etapa de visualização do mapa o usuário solicita o carregamento dos dados contidos no mapa vetorial, sendo necessário para isso a informação da coordenada, em latitude e longitude, recebida pelo receptor GPS. Após é acionada a classe de pesquisa selecionando todos elementos vetoriais contidos próximos ao ponto de localização atual, exibindo assim o mapa gráfico para o funcionário. Esse processo é repetido constantemente enquanto existir movimentação do mapa.

Na Figura 18 é apresentado o Diagrama de Sequência para o processo de visualização do mapa.

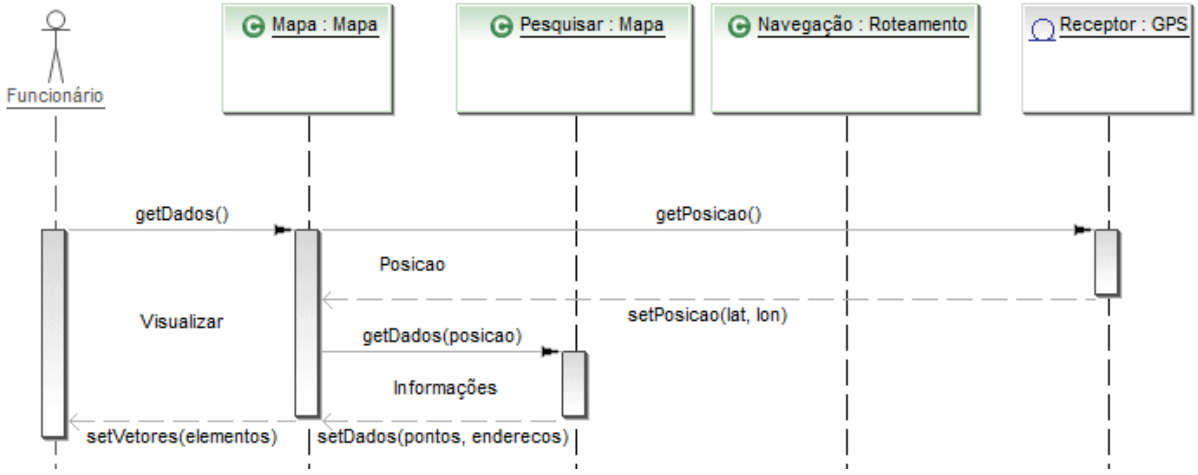

Figura 18 - Documentação: Diagrama de Sequencia: Visualização

Para realizar uma navegação, primeiramente é necessário saber a posição atual, recebida do receptor GPS, e após a verificação do ponto de destino na classe mapa para assim ser realizado o cálculo da rota.

A Figura 19 é apresenta o Diagrama de Sequência para o processo de navegação.

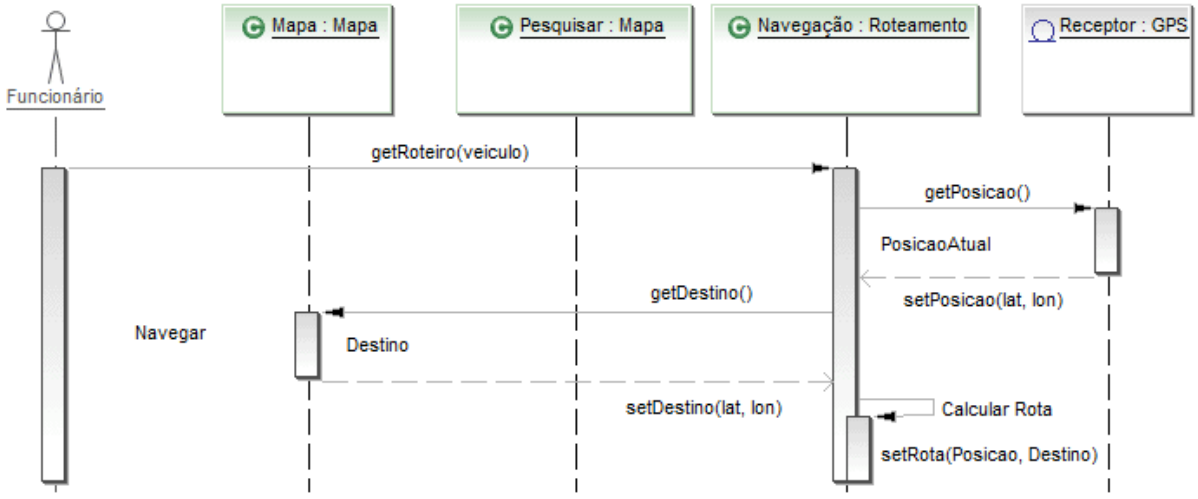

Figura 19 - Documentação: Diagrama de Sequencia: Navegação

A pesquisa também utiliza a informação da posição atual para realizar uma pesquisa gradativa circular onde serão localizados primeiramente os elementos mais próximos e coincidentes com o parâmetro de pesquisa.

É apresentado na Figura 20 o Diagrama de Sequência para a realização da pesquisa.

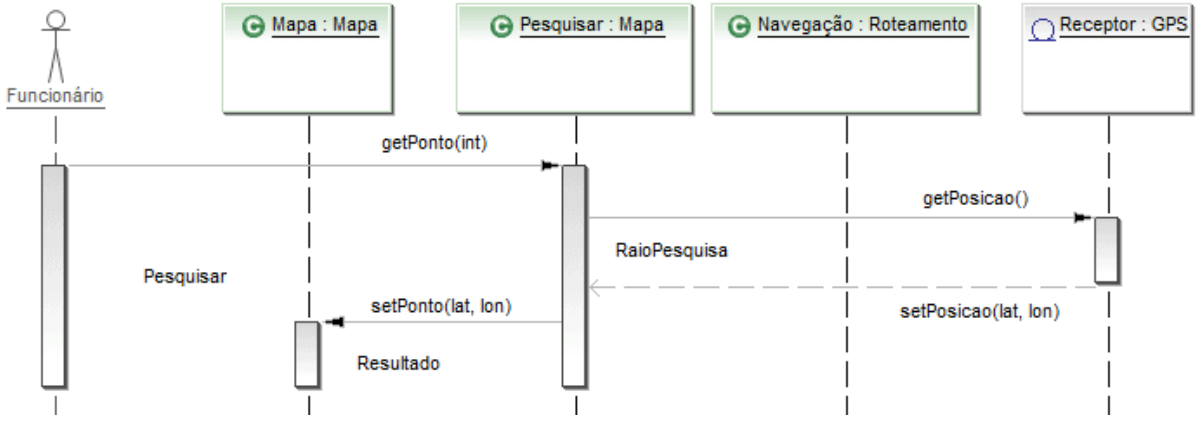

Figura 20 - Documentação: Diagrama de Sequencia: Pesquisa

4.2.4 Diagrama de Classes

O Diagrama de Classes representa a estrutura do sistema, destacando classes que servirão de base para implementação do Navegador GPS. Na Figura 21 são exibidos os atributos e métodos das classes e o relacionamento entre elas.

Figura 21 - Documentação: Diagrama de Classes

4.2.5 Diagrama de Entidade e Relacionamento

O diagrama de entidade e relacionamento (DER) representa os relacionamentos entre objetos de dados e conduzem à modelagem de dados. Este diagrama é composto por: entidades, atributos e relacionamentos.

Na Figura 22 é apresentado o DER criado para descrever o banco de dados do Navegador GPS, implementado posteriormente utilizando SQLite.

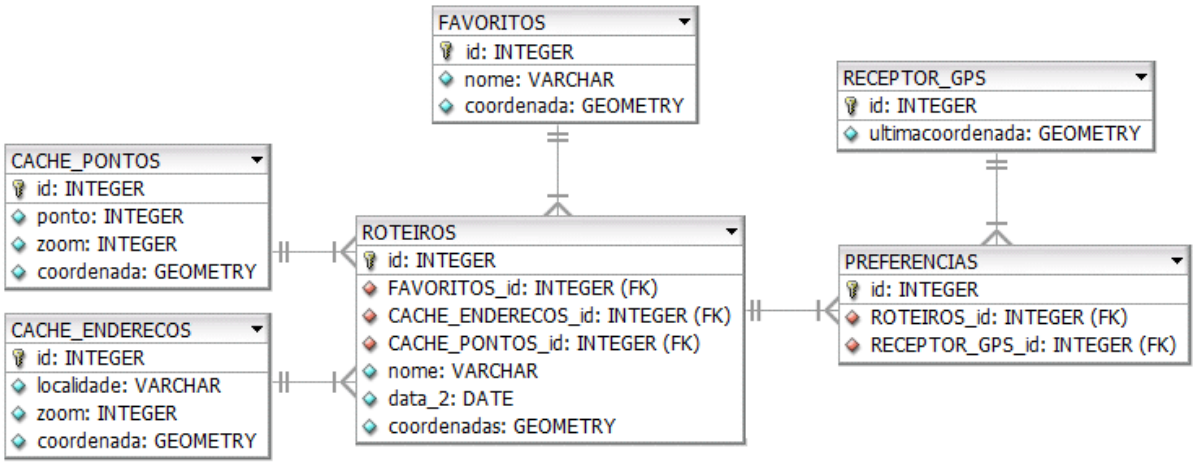

Figura 22 - Documentação: Diagrama de Relacionamento Navegador GPS.

Para a modelagem do Conversor de Mapas foi utilizado o banco de dados PostgreSQL conforme exibido na Figura 23.

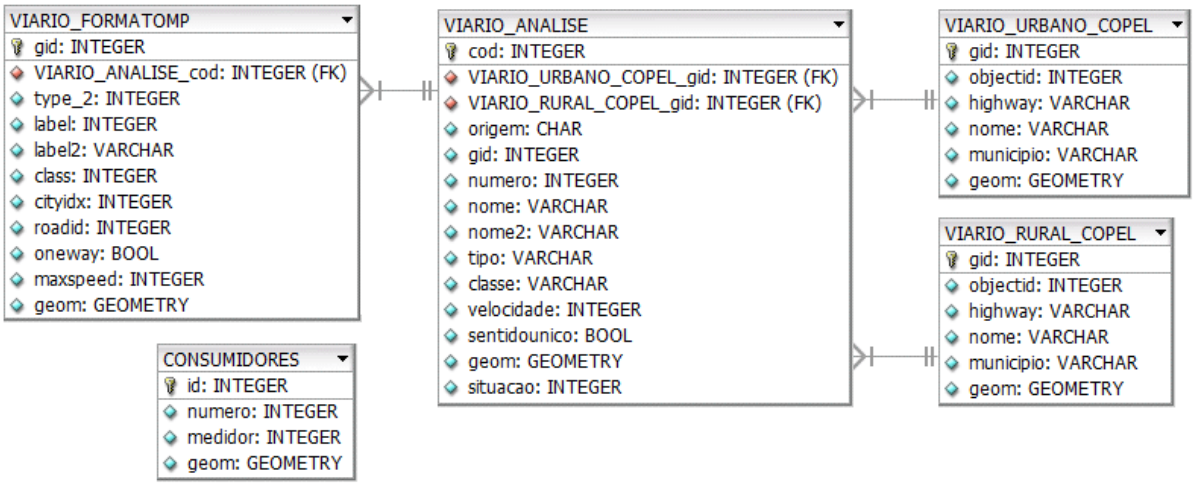

Figura 23 - Documentação: Diagrama de Relacionamento Conversor Mapas

4.3 DESENVOLVIMENTO DO NAVEGADOR

Para o desenvolvimento do Navegador GPS, foram criadas classes personalizadas, contendo comandos responsáveis por interpretar os mapas vetoriais no formato ".osm" e converter eles em imagens gráficas na tela do Tablet.

A seguir são apresentados alguns tópicos importantes relacionados ao desenvolvimento do Navegador GPS.

4.3.1 Tela Principal

Os ícones e imagens utilizados no Navegador GPS foram desenvolvidos com base em gráficos de outros aplicativos GPS para seguir uma padronização de fácil entendimento.

A Figura 24 exibe a tela principal (inicial) do Navegador GPS, onde é possível observar a disposição dos ícones principais (Mapa, Pesquisar, Alimentadores e Serviços) e dos ícones secundários (estrela de favoritos e fechar).

Figura 24 - Tela inicial do Navegador GPS COPEL

O botão "MAPA" dará acesso à visualização do mapa, o botão "PESQUISAR" acessa a página de pesquisa dos elementos e localidades. Os botões de "ALIMENTADORES" e "SERVIÇOS" são opções personalizadas para a COPEL,

que provem acesso a informações da rede de distribuição e integração com outro aplicativo próprio da COPEL de distribuição de serviços, respectivamente.

4.3.2 Visualização do Mapa

A visualização é o processo de “renderização” dos elementos gravados no mapa que está no formato “.osm”. O mapa vetorial fornece uma vista aérea superior da localidade ativa na tela, exibindo com proporcionalidade todos os elementos existentes no mapa vetorial e que sejam coincidentes com as configurações internas de renderização do navegador GPS. Um exemplo dos arquivos envolvidos no processo de renderização do mapa, é apresentado na Tabela 13.

Tabela 13 - Processo de renderização de um tipo de estrada

MAPA	
estradas.osm	<way id="74832" user="rebelato" uid="4372" visible="true" version="3"> <nd ref="98123"/> <nd ref="1233"/> <nd ref="83724"/> ... <tag k="highway" v="primary"/> <tag k="name" v="Rua Getulio Vargas"/> </way>
NAVEGADOR	
render.xml	<renderingStyle name="copel" depends="" defaultColor="#f4fbff" version="1"> <line> <group> <filter tag="highway" value="primary" maxzoom="13" color="#FFAA80"/> <groupFilter shadowRadius="1"> <filter minzoom="7" maxzoom="9" strokeWidth="4"/> <filter minzoom="10" maxzoom="13" strokeWidth="5"/> </groupFilter> </group> </line> </renderingStyle>
tipos.xml	<category name="highway"> ... <type tag="highway" value="primary" minzoom="7" nameTags="ref"/> ... </category>

A estrada possui características de exibição como cor, espessura e contorno, além da informação de níveis de zoom em cada filtro, especificando o *zoom* mínimo e máximo de cada característica, para assim exibir com clareza a estrada a cada aproximação ou afastamento do mapa na tela.

A Figura 25 exibe o mapa vetorial completo com todos os elementos cartográficos (estradas, rios, vegetação) e elétrico (rede de distribuição e transmissão de energia elétrica).

Figura 25 - Visualização completa do mapa COPEL da Região de Cascavel

O Quadro 10 apresenta o código-fonte responsável por analisar os elementos do mapa e verificar se existem ícones relacionados no Navegador GPS. Em caso afirmativo os ícones são rotacionados para obter a exibição correta na tela.

```
private boolean desenharRotacionarMapa(Canvas canvas, Bitmap bmp, Localiza bmpLocaliza) {
        boolean shown = false;
        if (bmp != null && bmpLocaliza != null) {
            float rot = bmpLocaliza.getRotate();
            float mult = (float) MapaUtil.getPowZoom(view.getZoom() - bmpLocaliza.getZoom());
            float fmult = (float) MapaUtil.getPowZoom(view.getFloatZoom() - bmpLocaliza.getZoom());
            float cos = bmpLocaliza.getRotateCos();
            float sin = bmpLocaliza.getRotateSin();
            float x1 = MapaUtil.calcDiffPixelX(sin, cos) + view.getCenterPointX();
            float y1 = MapaUtil.calcDiffPixelY(sin, cos) + view.getCenterPointY();
            canvas.rotate(-rot, view.getCenterPointX(), view.getCenterPointY());
            destImage.set(x1, y1, x1 + bmpLocaliza.getTileWidth(), y1 + bmpLocaliza.getTileHeight() * ts);

            if(!bmp.isRecycled()){
                canvas.drawBitmap(bmp, null, destImage, paintImg);
                shown = true;
            }
            canvas.rotate(rot, view.getCenterPointX(), view.getCenterPointY());
        }
        return shown;
    }
```

Quadro 10 - Código: Método responsável por rotacionar os elementos

4.3.3 Níveis de Zoom

O controle de zoom é essencial para uma experiência completa de navegação e visualização das informações, pois quanto mais se aproxima o mapa, mais informações detalhadas serão exibidas de cada elemento. O nível de zoom pode ser definido manualmente (Figura 26), utilizando o dedo indicador e o polegar em movimentos de aproximação e afastamento ou clicando nos botões de “+” e “-” posicionados no canto inferior direito da tela.

Figura 26 - Opções de zoom manual na tela de visualização de mapa

O Quadro 11 apresenta ainda a funcionalidade de controle automático do zoom, em relação a velocidade do veículo, sendo que quanto maior a velocidade, mais afastado fica o mapa, exibindo uma área maior com menos detalhamento.

```
public float definirZoomPorVelocidade(float velocidade) {
    double visivelDistancia = mapaBase.getDistance(mapaVisual.getLongitude(), mapaVisual.getLongitude());
    float tempo = 75f;
    if (velocidade < 83f) {
        tempo = 60f;
    }
    double distanciaChegada = velocidade * tempo;
    float zoomDelta = (float) (Math.log(visivelDistancia / distanciaChegada) / Math.log(2.0f));
    zoomDelta = Math.round(zoomDelta * 3) * 1/3f;
    return zoomDelta;
}
```

Quadro 11 - Código: Responsável por realizar o zoom automático

Cada elemento do mapa possui configurações próprias de renderização informando o *zoom* mínimo e máximo, informações de impressão dos rótulos como cor, tamanho e contorno. Essa configuração garante a qualidade visual do mapa impedindo o desenho de todos os elementos ao mesmo tempo. O Quadro 12 apresenta o XML existente no Navegador GPS responsável pela renderização dos elementos do mapa.

```
<group>
<filter tag="highway" value="primary" color="#788BD5" />
   <groupFilter shadowRadius="1" >
      <filter minzoom="7" maxzoom="9" strokeWidth="4"/>
      <filter minzoom="10" maxzoom="12" strokeWidth="5"/>
      <filter minzoom="13" maxzoom="16" strokeWidth="8"/>
   </groupFilter>
</group>
```

Quadro 12 - Código: XML de renderização para os elementos

O Quadro 13 apresenta o código-fonte responsável por ler o arquivo de renderização no formato XML, desenhar na tela o ícone ou formato vetorial do elemento e quando existir, o seu respectivo rótulo.

```
public static final int inicioZoom = 14;
private Paint paintIcon;
private Paint paintTextIcon;
private List<pontos> objetos = new ArrayList<pontos>();

@Override
public void desenharTela(Canvas canvas, RectF latLonLimites) {

    if (view.getZoom() >= inicioZoom) {
        objetos.clear();

        pesquisasGerais.localizarPontos(limites.topo, limites.esquerda, limites.inferior, limites.direita, view.getZoom());

        for (pontos o : objetos) {
            int x = view.getRotacionarMapaXPonto(o.getLatitude(), o.getLongitude());
            int y = view.getRotacionarMapaXPonto(o.getLatitude(), o.getLongitude());
            String id = o.getSubtitulo();
            if(id != null){
                Bitmap bmp = renderizarIcones.getIcone(view.getContext(), id);
                if(bmp != null){
                    canvas.drawBitmap(bmp, x - bmp.getComprimento() / 2, y - bmp.getAltura() / 2, paintIcon);
                }
            }
        }

        if ((view.getConfiguracoes().EXIBIR_NOMES.get()) && (view.getZoom() > 16)) {
            TIntHashSet set = new TIntHashSet();
            for (pontos o : objetos) {
                int x = view.getRotacionarMapaXPonto(o.getLatitude(), o.getLongitude());
                int y = view.getRotacionarMapaYPonto(o.getLatitude(), o.getLongitude());
```

```
                    int tx = view.getMapaXPonto(o.getLongitude());
                    int ty = view.getMapaYPonto(o.getLatitude());
                    String nome = o.getNome();
                    if (nome != null && nome.length() > 0) {
                        drawWrappedText(canvas, nome, paintTextIcon.getTextSize(), paintTextIcon.getTextSize() / 2, 3);
                    }
                }
            }
        }
    }
```

Quadro 13 - Código: Desenhar os ícones e rótulos na tela

O nível de zoom é importante para a visualização e navegação na tela devido ao fator estético do mapa, porém a principal vantagem é a utilização correta dos recursos de memória e processamento do aparelho, pois apenas os elementos relevantes em cada nível de zoom serão carregados e exibidos.

4.3.4 Pesquisas Diversas

A localização de pontos de interesse é a peça fundamental do Navegador GPS, pois são inseridos nos mapas milhões de pontos da rede elétrica da Copel. A pesquisa deve ser rápida e precisa exibindo detalhes do ponto e a sua coordenada correta no mapa. A Figura 27 apresenta a tela de pesquisa com os resultados encontrados após a digitação do código.

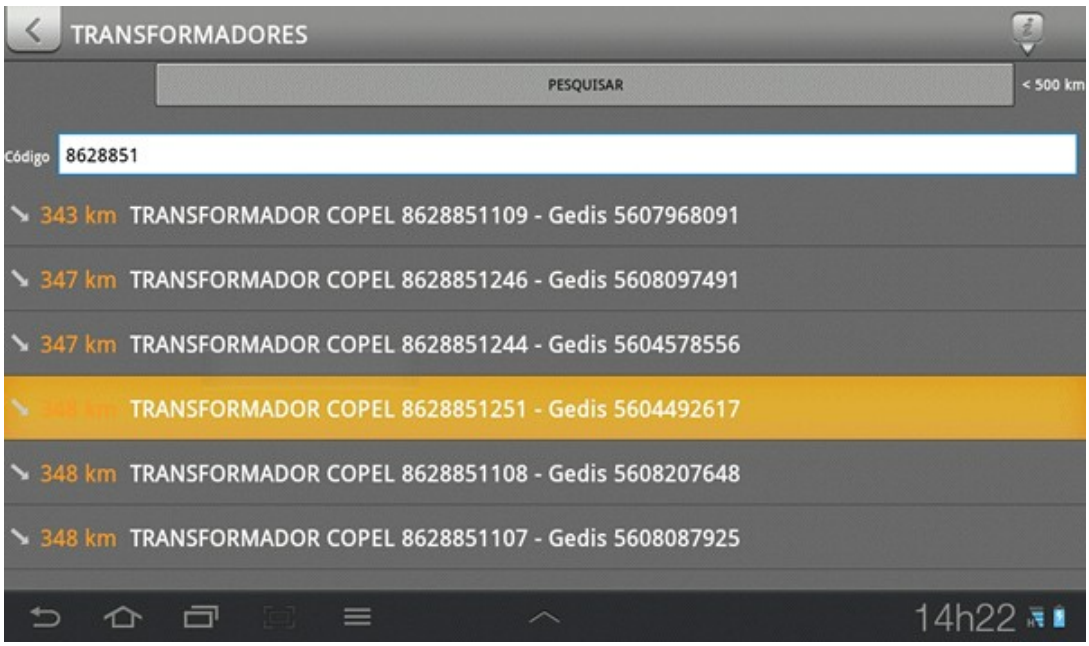

Figura 27 - Tela de pesquisa com os resultados semelhantes

A pesquisa é realizada por proximidade de localização, levando em consideração a localização atual do usuário e os pontos coincidentes mais próximos a ele. Essa metodologia é a ideal para não ocorrerem sobrecargas no sistema devido aos milhões de pontos existentes.

A Figura 28 exibe o ponto localizado após sua seleção na tela anterior.

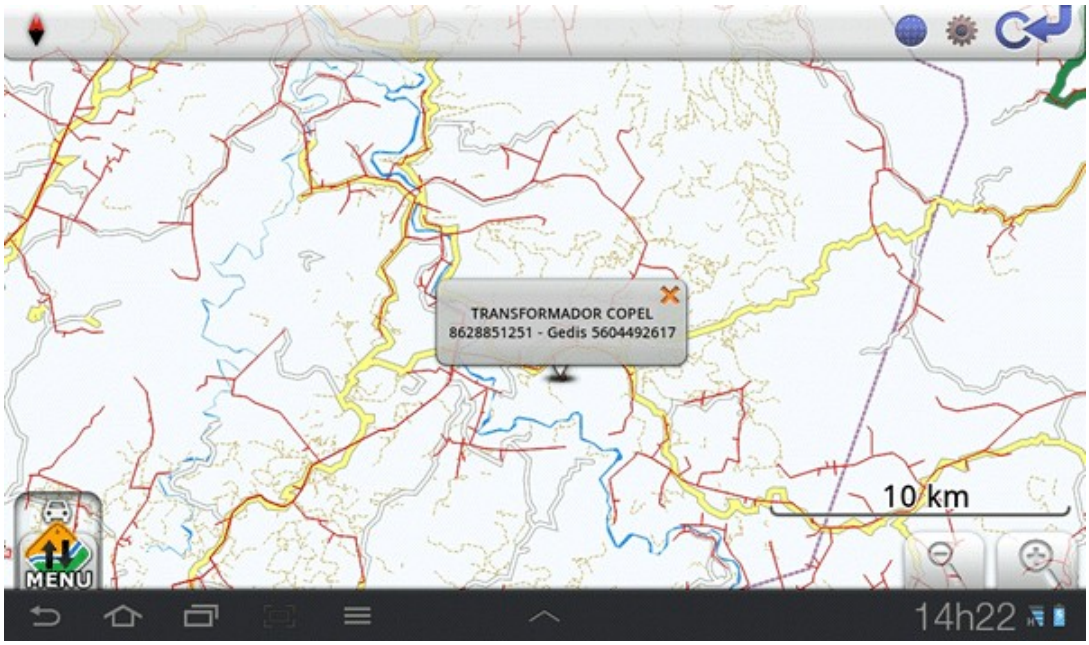

Figura 28 - Ponto localizado exibido na tela do mapa

O Quadro 14 apresenta os métodos de pesquisa do Navegador GPS.

```
private boolean leituraArea(int left31, int right31, int top31, int bottom31, int px, int py, int pzoom) throws IOException {
        int zoom = pzoom;
        int dy = py;
        int dx = px;
        int x = dx + (px << (zoom - pzoom));
        int y = dy + (py << (zoom - pzoom));
        int xL = x << (31 - zoom);
        int xR = ((x + 1) << (31 - zoom)) - 1;
        int yT = y << (31 - zoom);
        int yB = ((y + 1) << (31 - zoom)) - 1;

        if (left31 > xR || xL > right31 || bottom31 < yT || yB < top31) {
            return false;
        } else {
            return true;
        }
    }

    // ...

    public static PesquisaRequerimento<Pontos> localizarPontoR(int sleft, int sright, int stop, int sbottom, int zoom) {
        SearchResult<Pontos> request = new SearchResult<Pontos>();
        request.left = sleft;
        request.right = sright;
        request.top = stop;
```

```
        request.bottom = sbottom;
        request.zoom = zoom;
        return request;
    }

    // ...

    private static void testaPesquisaPontos(BinaryMapIndexReader reader, pontoArea pontoArea) throws IOException {
        println(pontoArea.leftLongitude + " " + pontoArea.rightLongitude + " " + pontoArea.bottomLatitude + " "
        + pontoArea.topLatitude);
        for (int i = 0; i < pontoArea.categorias.size(); i++) {
            println(pontoArea.categorias.get(i));
            println(" " + pontoArea.subcategorias.get(i));
        }
        PesquisaRequerimento<Pontos> req = localizarPonto(sleft, sright, stop, sbottom, -1, new localizaPonto() {
            @Override
            public boolean accept(PontosTipos tipo, String subcategoria) {
                return true;
            }}, null);
        List<Pontos> results = reader.localizaPonto(req);
        for (Pontos a : results) {
            println(a.getTipo() + " " + a.getSubTipo() + " " + a.getNome() + " " + a.getLocalizacao());
        }
    }

    // ...

    protected void localizaPonto(pontoArea regiao, PesquisaRequerimento<Pontos> req) throws IOException {
        TIntLongHashMap offsets = new TIntLongHashMap();
        CollatorStringMatcher matcher = new CollatorStringMatcher(req.nomeQuery);
        long time = System.currentTimeMillis();
        while (true) {
            if (req.isCancelled()) {
                return;
            }
            int t = codedIS.readTag();
            int tag = WireFormat.getTagFieldNumber(t);
            switch (tag) {
            case 0:
                return;
            case GpsCopel.PESQUISA_PONTOS:
                Integer[] offKeys = new Integer[offsets.size()];
                if (offsets.size() > 0) {
                    int[] keys = offsets.keys();
                    for (int i = 0; i < keys.length; i++) {
                        offKeys[i] = keys[i];
                    }
                    final TIntLongHashMap foffsets = offsets;
                    Arrays.sort(offKeys, new Comparator<Integer>() {
                        @Override
                        public int compare(Integer object1, Integer object2) {
                            return Double.compare(foffsets.get(object1), foffsets.get(object2));
                        }
                    });
                    int p = PESQ_NOMES * 3;
                    if (p < offKeys.length) {
                        for (int i = p + PESQ_NOMES;; i += PESQ_NOMES) {
                            if (i > offKeys.length) {
                                Arrays.sort(offKeys, p, offKeys.length);
                                break;
                            } else {
                                Arrays.sort(offKeys, p, i);
                            }
                            p = i;
                        }
                    }
                } return;
            default:
                break;
            } } }
```

Quadro 14 - Código: Métodos de pesquisa utilizados no Navegador GPS

4.3.5 Roteamento

Para realizar um deslocamento, inicialmente é preciso saber a localização atual do usuário, e em seguida saber o destino final. A informação do local inicial é obtida através do receptor GPS interno do *tablet*, exibindo assim a caminhonete da COPEL na tela do aparelho. O ponto de destino é obtido clicando na tela ou pesquisando algum ponto de interesse contido nos mapas gerados.

Com o ponto de partida e destino marcados na tela é possível iniciar a classe responsável pelo roteamento, que irá acionar o algoritmo de cálculo, e assim encontrar o caminho mais curto ou mais rápido para concluir a viajem.

A informação do melhor trajeto a ser seguido é exibida em formato gráfico com uma linha azul sobreposta ao mapa cartográfico e também com avisos sonoros acionados a cada manobra realizada.

O Quadro 15 exibe um dos métodos para o cálculo de roteamento.

```
public class RotaCalculoResultado {

  private final List<Localizacao> localizacaos;
  private final List<RotaDirecaoInfo> direcao;
  private final List<RotaSegmentosResultado> segmentos;
  private final List<AlarmeInfo> alarmeInfo;
  private final String erroMensagem;
  private final int[] listDistancia;
  private final int[] intermediariosPontos;
  private final float rotaTempo;

  public RouteCalculationResult(List<RotaSegmentosResultado> list, Localizacao inicio, LatLon fim, List<LatLon>
      intermediarios, ClientContext ctx, float rotaTempo) {

      this.rotaTempo = rotaTempo;
      List<RotaDirecaoInfo> computarDirecao = new ArrayList<RotaDirecaoInfo>();
      this.erroMensagem = null;
      this.intermediariosPontos = new int[intermediarios == null ? 0 : intermediarios.size()];

      List<Localizacao> localizacaos = new ArrayList<Localizacao>();
      ArrayList<AlarmeInfo> alarms = new ArrayList<AlarmeInfo>();
      List<RotaSegmentosResultado> segmentos = convertVectorResult(computarDirecao, localizacaos, list, alarms, ctx);
      introduzirPrimeiroUltimoPonto(localizacaos, computarDirecao, segmentos, inicio, fim);

      this.localizacaos = Collections.unmodifiableList(localizacaos);
      this.segmentos = Collections.unmodifiableList(segmentos);
      this.listDistancia = new int[localizacaos.size()];

      calcularPontosIntermediarios(ctx, intermediarios, computarDirecao);
      updateListDistanceTime();
      this.direcao = Collections.unmodifiableList(computarDirecao);
      atualizarDirecao();
      this.alarmeInfo = Collections.unmodifiableList(alarms);
  }
}
```

Quadro 15 - Código: Responsável por realizar o zoom automático

4.3.5.1 Roteamento Gráfico

A Figura 29 apresenta o roteamento gráfico sobre o mapa vetorial exibindo o caminho a ser seguido pelo usuário do Navegador GPS.

Figura 29 - Roteamento gráfico exibido pela linha azul

É possível visualizar a próxima manobra e a seguinte no canto esquerdo da tela. O nome da estrada da próxima manobra situa-se na parte superior e a distância e hora para a chegada situam-se no canto direito superior.

4.3.5.2 Roteamento Sonoro

Enquanto se percorre o roteamento gráfico, pode-se opcionalmente ouvir os comandos sonoros que representam as manobras a serem efetuadas. O Quadro 16 apresenta o método responsável por ativar os arquivos de voz sequencialmente conforme a direção, tempo e localização.

```
private void playPrepararManobra(RotaDirecaoInfo proxima, int distancia) {
        Commando play = getComandoPlayer();
        if(play != null){
            String tParam = getManobraTipo(proxima.getManobraTipo());
            if(tParam != null){
                play.preparaManobra(tParam, distancia).play();
            } else if(proxima.getManobraTipo().Rotatoria()){
```

```
                play.preparaRotatoria(distancia).play();
            } else if(proxima.getManobraTipo().getValue().equals(ManobraTipo.U)){
                play.prepareRetorno(distancia).play();
            }
        }
    }
    private String getNovaManobra(ManobraTipo t){

        if(ManobraTipo.TL.equals(t.getValue())){
            return ComandoPlayer.A_ESQUERDA
        } else if(ManobraTipo.TR.equals(t.getValue())){
            return ComandoPlayer.A_DIREITA;
        }
        return null;
    }
```

Quadro 16 - Código: Método responsável pelo roteamento sonoro

4.4 GERAÇÃO DOS MAPAS

Para a geração dos mapas vetoriais no formato “.osm”, foi necessária a leitura de dados no formato “.csv”, “.mp” e informações do banco de dados PostGreSQL da COPEL. Após a geração dos mapas é possível convertê-los para o formato final “.bin”, que possibilita ao navegador GPS exibir todas as informações de maneira rápida e precisa.

O processo de geração dos mapas segue o fluxo exibido na Figura 30.

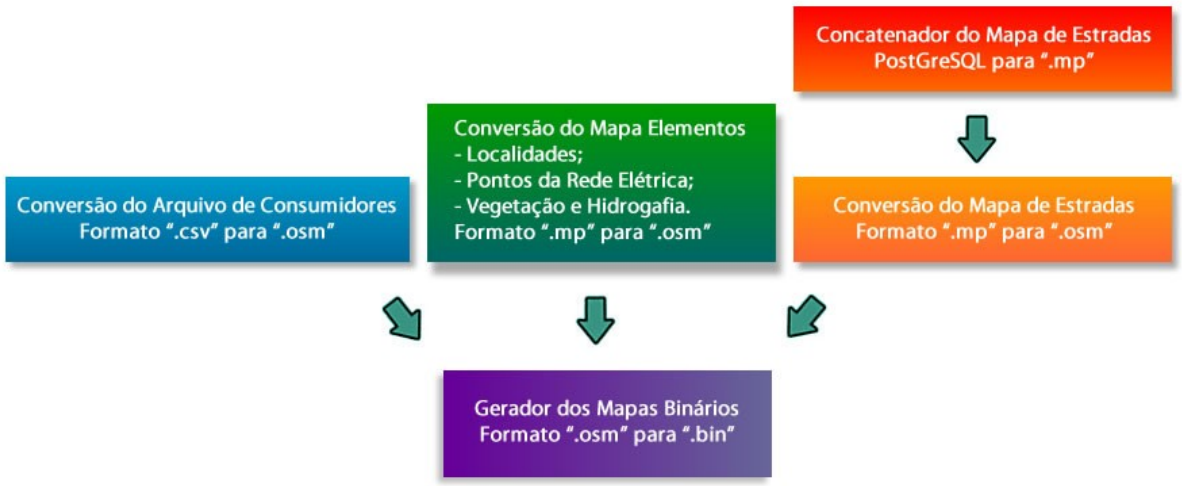

Figura 30 - Fluxo do processo de geração dos mapas binários

4.4.1 Conversão do Arquivo de Consumidores

Os clientes da COPEL são denominados “Unidades Consumidoras” sendo abreviados como “UC”. Todo consumidor possui um código numérico, aleatório e único que é utilizado com base para atrelar diversas informações importantes como

a localização e o número do medidor de energia que está sendo utilizado. A Figura 31 exibe o cabeçalho de uma fatura de energia com o número da “UC” e um medidor de consumo de energia elétrica, com seu respectivo número de identificação.

Figura 31 - Fatura e Medidor de energia com seus números de identificação

A localização é composta pela coordenada geográfica (latitude e longitude) do ponto de atendimento, e o número do medidor é denominado na COPEL pela abreviação “NIO”.

Conforme já exibido no Quadro 1, do referencial teórico deste trabalho, todas as unidades consumidoras, utilizadas para conversão do mapas, são importadas dos bancos de dados nos servidores da COPEL no formato “.csv”, contendo mais de 4 milhões de linhas que correspondem a quantidade de consumidores da COPEL (incluindo consumidores ligados e desligados). A formatação do arquivo “.csv” segue os seguintes modelo abaixo (cabeçalho e informação):

NUMERO_UC,COD_LOCAL_UC,NIO_MEDIDOR,COORDENADAS
51418517,2242,922807116,"[-52.1143124623513,-25.3690756322014]"
...

Neste trabalho foi desenvolvida em linguagem de programação Java, uma classe especialmente para a conversão dos dados de consumidores contidos no arquivo “.csv” para um arquivo no padrão “.osm”. Para esse fim foram importadas as classes *Java File* e *Buffered* que realizam a leitura e escrita de arquivos e de seu conteúdo.

O Quadro 17 contém todo o código da classe *Conversor* onde é possível verificar a conversão de arquivos de texto em um formato OSM XML.

```
package codigo;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Conversor {
  public static void main(String[] args) {

        FileReader entradaUC;
        FileWriter saidaUC;
        try {

            String linhaUC;
            String numeroNIO;
            String nomeArquivo = "Consumidores";
            int contadorUC = 1;

            // VALOR DE CORRECAO DAS COORDENADAS
            float correcaoXY = 0.00040F;

            // LEITURA DO ARQUIVO .CSV DE ENTRADA
            entradaUC = new FileReader("D:\\COPEL\\" + nomeArquivo + ".csv");
            BufferedReader intuc = new BufferedReader(entradaUC);
            linhaUC = intuc.readLine();

            while (linhaUC != null) {
                // MONTAGEM DO ARQUIVO OSM DE SAIDA
                saidaUC = new FileWriter("D:\\COPEL\\" + nomeArquivo + ".osm");
                BufferedWriter outuc = new BufferedWriter(saidaUC);

                outuc.write("<?xml version='1.0' encoding='UTF-8'?>");
                outuc.newLine();

                outuc.write("<osm version='0.6' generator='JOSM'>");
                outuc.newLine();

                while (linhaUC != null) {

                    String conteudouc[] = linhaUC.split(",");
                    String coordenadaX = conteudouc[3].replace("[", "").replace("]", "").replace("\"", "");
                    String coordenadaY = conteudouc[4].replace("[", "").replace("]", "").replace("\"", "");

                    float X = Float.parseFloat(coordenadaX);
                    float Y = Float.parseFloat(coordenadaY);
                    float X1 = X – correcaoXY;
                    float Y1 = Y – correcaoXY;

                    if (conteudouc[2].equals("")) {
                        numeroNIO = "inexistente";
                    } else {
                        numeroNIO = conteudouc[2];
                    }
                    outuc.write("<node id='-" + contadorUC + "' lat='" + Y1 + "' lon='" + X1 + "'>");
                    outuc.newLine();

                    outuc.write("<tag k='name' v='" + conteudouc[0] + " - NIO " + numeroNIO + "' />");
                    outuc.newLine();

                    outuc.write("<tag k='consumidor' v='uc' />");
                    outuc.newLine();

                    outuc.write("</node>");
                    outuc.newLine();

                    System.out.println("** " + contadorUC + " **");
                    System.out.println("UC: " + conteudouc[0]);
                    System.out.println("   ");
                    contadorUC++;
```

```
                    linhaUC = intuc.readLine();
                }
                outuc.write("</osm>");
                outuc.flush();
                outuc.close();
            }
            intuc.close();
        } catch (IOException ex) {
            Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Quadro 17 - Código: Conversor do formato “.csv” para “.osm”

O procedimento empregado na classe Conversor foi a leitura linha por linha do arquivo de texto, com valores separados por vírgula; a montagem do cabeçalho do arquivo novo “.osm”; a formatação da coordenada com sua devida correção; a inclusão das “*tags*” de identificação; exibição do andamento do processo via mensagem no console Java; fechamento do arquivo “.osm”.

Após o processamento do arquivo “.csv” é gerado o arquivo no formato “.osm” contendo todas as unidades consumidoras com seus respectivos números de medidores. O Quadro 18 exemplifica a formatação final OSM XML de duas “UC”.

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' generator='JOSM'>
...
<node id='-362691' lat='-24.46753' lon='-53.958088' >
    <tag k='name' v='55719554 - NIO 240830726' />
    <tag k='consumidor' v='uc' />
</node>
<node id='-362692' lat='-24.531803' lon='-53.742508' >
    <tag k='name' v='55758983 - NIO 241900566' />
    <tag k='consumidor' v='uc' />
</node>
...
</osm>
```

Quadro 18 - Codificação OSM XML para unidade consumidora

A *tag* “*name*” é utilizada como chave primária sendo seu valor o conteúdo utilizado para pesquisa de consumidores ou número de medidores. A segunda *tag* é utilizada para categorizar e distinguir os pontos de interesse de consumidores dos demais pontos no mapa.

4.4.2 Conversão do Mapa de Elementos

Seguindo os conhecimentos adquiridos e descritos no referencial teórico foi possível desenvolver classes em linguagem de programação Java necessárias para a conversão dos mapas do formato texto para o formato XML. Foram convertidos dados dos pontos de interesse da rede elétrica da COPEL e dados referentes a malha viária urbana e rural do estado do Paraná.

4.4.2.1 Localidades

Para representar uma localidade é preciso um ponto de interesse contendo o nome da localidade e a coordenada do ponto central da cidade, vila, bairro ou comunidade. Na Tabela 14 é realizado o comparativo da conversão de um ponto de localidade no formato ".mp" para o formato ".osm".

Tabela 14 - Comparativo do mesmo município no formato ".mp" vs. ".osm"

Formato MP	Formato OSM
[Cities] City3406=PATO BRANCO RegionIdx3406=1 **[END-Cities]** **[POI]** Type=0x800 Label=PATO BRANCO Data3=(-26.22711,-52.67258) **[END]**	<node id='-1559230' visible='true' version='1' lat='-26.22711' lon='-52.67258'> <tag k='addr:city' v='Sudoeste' /> <tag k='is_in' v='Parana' /> <tag k='name' v='PATO BRANCO' /> <tag k='place' v='town' /> </node>

O Quadro 19 contém a classe desenvolvida em linguagem Java responsável por ler o arquivo ".mp" e gravar o novo arquivo ".osm" com os respectivos pontos das localidades. É possível observar os laços de repetição para leitura linha a linha do mapa vetorial de origem e a gravação das novas linhas no arquivo de destino quando as condições são verdadeiras.

```
package beans;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import utilitarios.RemoverAcentos;
```

```
public class IconesCidades {

  public static void criarIconesCidades(BufferedReader binPontoCidade, BufferedWriter outParana) {

      String processoCidades = "nao";
      String gravaCidades = "nao";
      String itemPontoCidade = "";
      String type;
      String label = "";
      float[][] data = null;
      int contador = 0;

      try {

          String linhaPontoCidade = binPontoCidade.readLine();
          while (linhaPontoCidade != null) {

              // POI - CIDADES
              if (linhaPontoCidade.contains("[POI]")) {
                  linhaPontoCidade = binPontoCidade.readLine();
                  processoCidades = "sim";
              }

              // PROCESSA ELEMENTOS
              if (processoCidades.equals("sim") && !linhaPontoCidade.isEmpty() && !gravaCidades.equals("sim")) {

                  if (linhaPontoCidade.contains("[END]")) {
                      gravaCidades = "sim";

                  } else {
                      itemPontoCidade = linhaPontoCidade.substring(0, linhaPontoCidade.indexOf("="));
                      linhaPontoCidade = linhaPontoCidade.substring(linhaPontoCidade.indexOf("=")
                      + 1, linhaPontoCidade.length());
                      if (itemPontoCidade.equals("Type")) {
                          type = linhaPontoCidade;
                      } else if (itemPontoCidade.equals("Label")) {
                          label = linhaPontoCidade;
                          label = RemoverAcentos.remover(label).toUpperCase();
                      } else if (itemPontoCidade.substring(0, 4).equals("Data")) {
                          linhaPontoCidade = linhaPontoCidade.substring(1);
                          linhaPontoCidade = linhaPontoCidade.substring(0, linhaPontoCidade.length() - 1);
                          String aux[] = linhaPontoCidade.split("\\),\\(");
                          float[][] dataAux = new float[aux.length][2];
                          for (int i = 0; i < aux.length; i++) {
                              String[] aux2 = aux[i].split(",");
                              dataAux[i][0] = Float.parseFloat(aux2[0]);
                              dataAux[i][1] = Float.parseFloat(aux2[1]);
                          }
                          data = dataAux.clone();
                      }
                  }
              }

              if (processoCidades.equals("sim") && gravaCidades.equals("sim")) {
                  processoCidades = "nao";
                  gravaCidades = "nao";

                  if (data != null) {
                      String place = "city";

                      if (populacao > 500000) {
                          place = "city";
                      } else if (populacao > 100000) {
                          place = "town";
                      } else if (populacao > 10000) {
                          place = "village";
                      } else if (populacao > 0) {
                          place = "hamlet";
                      }

                      // MONTAGEM DO ARQUIVO .OSM
                      outParana.write("<node id='" + contador++ + "' lat='" + data[0][0]
                      + "' lon='" + data[0][1] + "' visible='true' version='1'>");
                      outParana.newLine();
```

```
                        outParana.write("<tag k='addr:city' v='" + label + "' />");
                        outParana.newLine();
                        outParana.write("<tag k='is_in' v='Parana' />");
                        outParana.newLine();
                        outParana.write("<tag k='place' v='" + place + "' />");
                        outParana.newLine();
                        outParana.write("<tag k='name' v='" + label + "' />");
                        outParana.newLine();
                        outParana.write("</node>");
                        outParana.newLine();
                        outParana.newLine();
                        label = "";
                        data = null;
                    }
                }
                linhaPontoCidade = binPontoCidade.readLine();
            }
        } catch (IndexOutOfBoundsException e) {
            System.err.println("Erro: Falha no index do array");
        } catch (Exception e) {
            System.err.println("Erro: Arquivo nao encontrado");
        }
    }
}
```

Quadro 19 - Código: Conversão das localidades em “.mp” para “.osm”

A Figura 32 apresenta os pontos de localidades com seus respectivos rótulos, em uma visualização gerada pelo software GPSMapEdit.

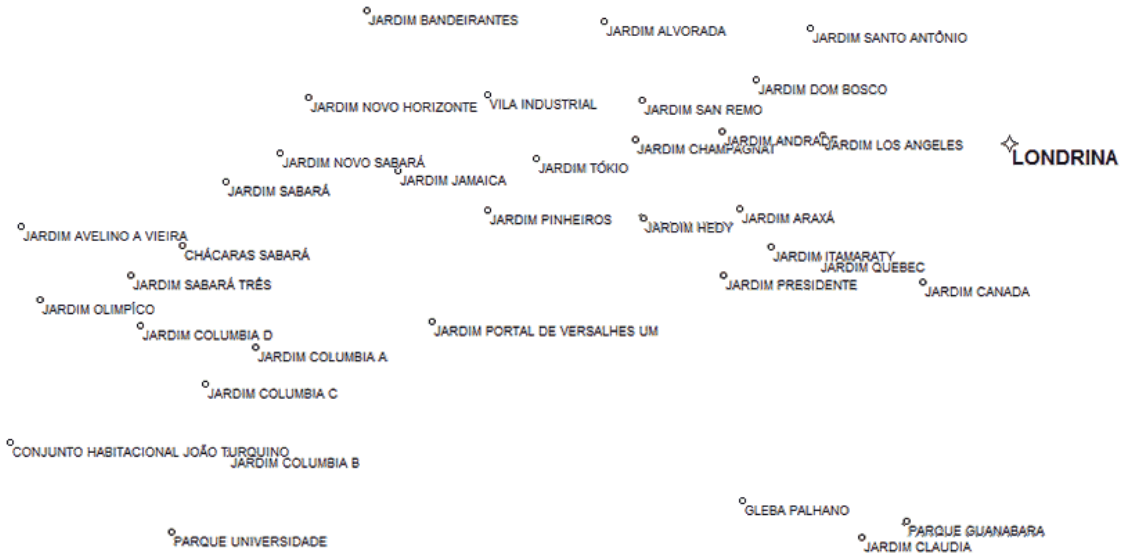

Figura 32 - Diversos pontos representando Cidades, Bairros, Comunidades, etc

4.4.2.2 Pontos da Rede Elétrica

A rede de energia elétrica da COPEL é composta por milhões de pontos de interesse, cada um contendo um número único de identificação, uma coordenada

(latitude e longitude, podendo ser coincidente com outros pontos) e uma categoria que é a fonte de informação para exibição dos ícones na tela.

A Figura 33 apresenta alguns exemplos desses pontos de interesse.

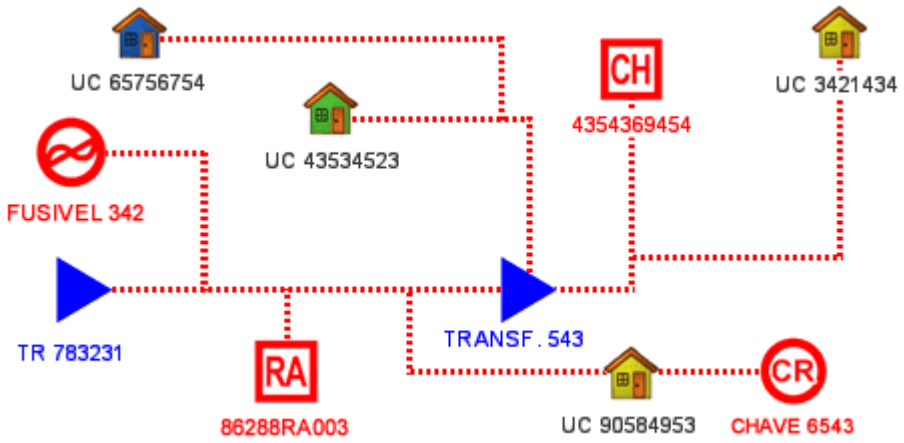

Figura 33 - Demonstração da simbologia elétrica padronizada COPEL

Seguindo como base o estudo já realizado durante o trabalho de estágio foi possível a conversão prática dos dados elétricos da rede COPEL em pontos para serem utilizados no navegador GPS.

Foram convertidos os seguintes dados:

a) **Clientes** – Unidades Consumidoras (UC): 4,15 milhões de pontos contendo o número de identificação do consumidor (UC) e o número de série do medidor (NIO). Processo de conversão descrito no item 4.4.1.
b) **Chaves** – Equipamento de proteção: 185 mil pontos contendo o número operacional e a categoria de cada equipamento.
c) **Transformadores** – Próprios e Particulares (TR): 415 mil pontos contendo o número operacional e o número de série (GEDIS).
d) **Pontos Significativos** – Postes, Divisões de circuitos e Cruzamentos: 2,96 milhões de pontos contendo o número de identificação (GEO).

Para utilização destes pontos foi necessária a geração dos mapas diretamente do banco de dados da COPEL para o formato final “.osm” ou quando possível foi realizado a conversão das informações seguindo os critérios

apresentados no item 4.4.1, referentes ao processo de conversão de dados do formato “.csv” para “.osm”.

4.4.2.3 Vegetação e Hidrografia

Para representar áreas de vegetação e reservatórios de água são utilizadas polilinhas fechadas.

Na Tabela 15 é realizado o comparativo da conversão de uma área de vegetação no formato “.mp” para o formato “.osm”.

Tabela 15 - Comparativo da mesma vegetação no formato “.mp” vs. “.osm”

Formato MP	Formato OSM

[POLYGON]
Type=0x20
Label=MATA CILIAR - PANTANAL
Data0=(-22.6358,-54.8089),(-22.63587...
[END]
	`<way id="1542" user="jr" uid="620284">`
	`  <nd ref="166810"/>`
	`  ...`
	`  <tag k='landuse' v='village_green' />`
	`  <tag k='natural' v='wood' />`
	`  <tag k="name" v="MATA CILIAR - PANTANAL"/>`
	`</way>`

O Quadro 20 contém a classe desenvolvida em linguagem Java responsável por ler o arquivo “.mp” e gravar o novo arquivo “.osm”. A leitura do arquivo segue a premissa de reconhecer a ocorrência da palavra-chave “POLYGON” para iniciar o processamento das polilinhas fechadas.

```
package utilitarios;

import java.io.BufferedReader;
import java.io.BufferedWriter;

import beans.Contadores;

import utilitarios.FormataNomeCoordenada;
import utilitarios.RemoverAcentos;

public class Vegetacao {

  public static int criarVegetacao(BufferedReader binParana, BufferedWriter outParana, Contadores contadores) {
      String processoVegetacao = "nao";
      String gravaVegetacao = "nao";
      String item = "";
      String type = "";
      String label = "";
      String[] sequenciaPoligono = null;
      float[][] dataPoligono = null;
      int vegetacao = 0;

      try {
```

```
String linhaParana = binParana.readLine();
while (linhaParana != null) {

    // PROCESSAR POLIGONOS - VEGETACAO
    if (linhaParana.contains("[POLYGON]")) {
        linhaParana = binParana.readLine();
        processoVegetacao = "sim";
    }

    // PROCESSANDO ELEMENTOS
    if (processoVegetacao.equals("sim") && !linhaParana.isEmpty() && !gravaVegetacao.equals("grava")) {
        if (linhaParana.contains("[END]")) {
            gravaVegetacao = "grava";
        } else {
            item = linhaParana.substring(0, linhaParana.indexOf("="));
            linhaParana = linhaParana.substring(linhaParana.indexOf("=") + 1, linhaParana.length());

            if (item.equals("Type")) {
                type = linhaParana;

            } else if (item.equals("Label")) {
                label = linhaParana;
                label = RemoverAcentos.remover(label);

            } else if (item.substring(0, 4).equals("Data")) {
                linhaParana = linhaParana.substring(1);
                linhaParana = linhaParana.substring(0, linhaParana.length() - 1);
                String aux[] = linhaParana.split("\\),\\(");
                String[] sequenciaAux = new String[aux.length];
                float[][] dataAux = new float[aux.length][2];

                for (int i = 0; i < aux.length; i++) {
                    String[] aux2 = aux[i].split(",");
                    dataAux[i][0] = Float.parseFloat(aux2[0]);
                    dataAux[i][1] = Float.parseFloat(aux2[1]);
                    sequenciaAux[i] = FormataNomeCoordenada.retornaNomeCoordenada(aux2);
                }
                sequenciaPoligono = sequenciaAux.clone();
                dataPoligono = dataAux.clone();
            }
        }
    } else if (processoVegetacao.equals("sim") && gravaVegetacao.equals("grava")) {

        processoVegetacao = "nao";
        gravaVegetacao = "naograva";

        if (dataPoligono != null) {

            contadores.setcontagemPolilinhasPoligonos();

            // MONTAGEM DO ARQUIVO .OSM
            outParana.write("<way id='" + contadores.getcontagemPolilinhasPoligonos()
            + "' visible='true' version='1' >");
            outParana.newLine();

            for (int i = 0; i < dataPoligono.length; i++) {
                outParana.write("<nd ref='" + contadores.getcontagemPolilinhasPoligonos() + i
                + sequenciaPoligono[i] + "' />");
                outParana.newLine();
            }
            outParana.write("<nd ref='" + contadores.getcontagemPolilinhasPoligonos() + 0
            + sequenciaPoligono[0] + "' />");
            outParana.newLine();

            outParana.write("<tag k='landuse' v='village_green' />");
            outParana.newLine();

            outParana.write("<tag k='natural' v='wood' />");
            outParana.newLine();

            outParana.write("<tag k='name' v='" + label + "' />");
            outParana.newLine();

            outParana.write("</way>");
```

```
                        outParana.newLine();

                        for (int i = 0; i < dataPoligono.length; i++) {
                            outParana.write("<node id='" + contadores.getcontagemPolilinhasPoligonos() + i
                            + sequenciaPoligono[i] + "' lat='" + dataPoligono[i][0] + "' lon='" + dataPoligono[i][1]
                            + "' visible='true' version='1' />");
                            outParana.newLine();
                        }

                        outParana.newLine();
                        type = "";
                        label = "";
                        dataPoligono = null;
                        sequenciaPoligono = null;
                        vegetacao++;
                    }
                }
                linhaParana = binParana.readLine();
            }

        } catch (IndexOutOfBoundsException e) {
            System.err.println("Erro: Falha no index do array");
        } catch (Exception e) {
            System.err.println("Erro: Arquivo nao encontrado");
        }
        return vegetacao;
    }
}
```

Quadro 20 - Código: Conversor de polígonos fechados

Por ser uma conversão linha a linha não houve a necessidade da utilização de banco de dados, sendo possível realizar a conversão das informações por meio de simples comparações do conteúdo de leitura.

4.4.3 Concatenador do Mapa de Estradas

O concatenador realiza a leitura de dois mapas vetoriais de estradas no formato “.mp”, analisando linha a linha as informações e retornando um único mapa concatenado sem estradas repetidas ou desconectadas.

A Figura 34 apresenta dois mapas vetoriais sobrepostos sem a concatenação de estradas.

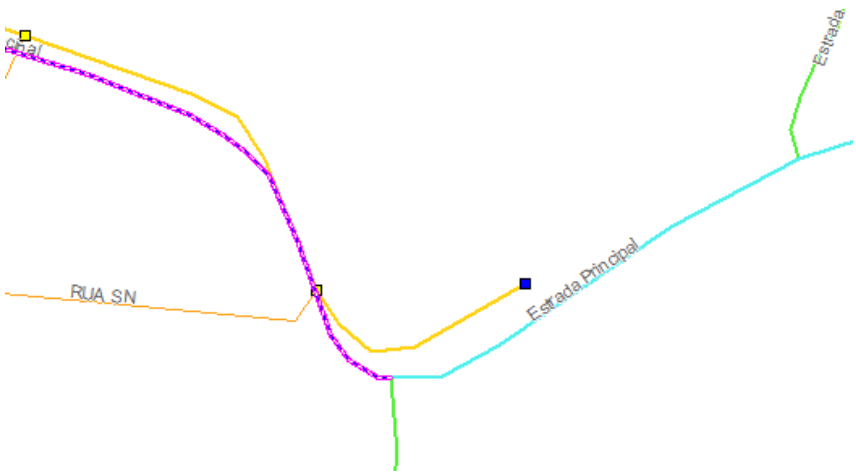

Figura 34 - Estradas sobrepostas sem nenhuma união entre si

Na Figura 35 é possível visualizar o resultado da concatenação unindo os dois mapas em um único arquivo no formato “.mp” com as estradas devidamente conectadas.

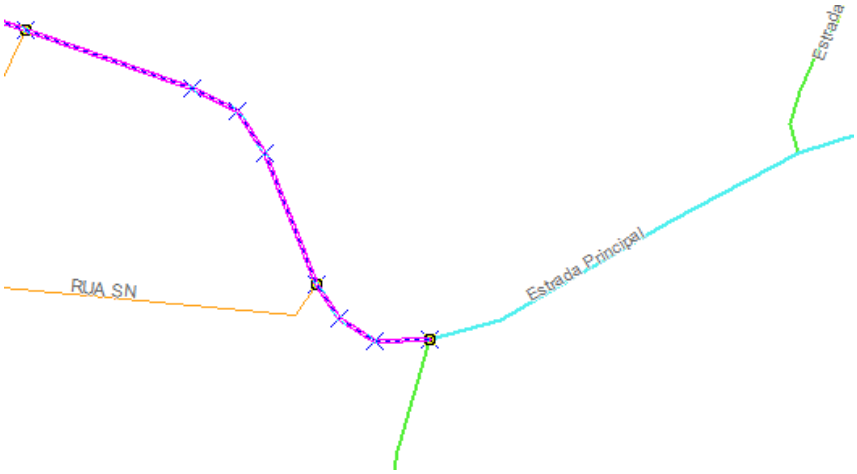

Figura 35 - Estradas unidas em um único mapa vetorial

Para a realização desse processo é preciso ler os dois mapas e gravá-los no banco de dados PostgreSQL, para assim realizar pesquisas georreferenciais (a partir de coordenadas próximas) e obter apenas as estradas necessárias para a concatenação.

A conexão com o banco de dados PostgreSQL é realizada com a classe “ConnectionFactory”, exibida no Quadro 21.

```
package conexaobd;

import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionFactory {
  public static Connection com;
  private static String IP_SERVIDOR = "127.0.0.1";
  private static String NOME_DO_BANCO = "estradas";
  private static String NOME_USUARIO_BANCO = "root";
  private static String SENHA_BANCO = "";

  public static Connection getConnection() {
      try {
          if (con == null) {
              // Carregar o Driver JDBC na memória do aplicativo
              Class.forName("com.mysql.jdbc.Driver");
              // Definir as variaveis de acesso ao banco
              String url = "jdbc:mysql://" + IP_SERVIDOR + ":3306/" + NOME_DO_BANCO;
              String user = NOME_USUARIO_BANCO;
              String pass = SENHA_BANCO;
              // Realizar a conexao
              con = DriverManager.getConnection(url, user, pass);
          }
      } catch (ClassNotFoundException e) {
          System.out.println("Erro: " + e.getMessage());
      } catch (SQLException e) {
          System.out.println("Erro: " + e.getMessage());
      }
      return com;
  }
}
```

Quadro 21 - Código: Conexão com o banco de dados PostgreSQL

A leitura dos mapas no formato “.mp”, e a consequente gravação no banco de dados se dá pelos seguintes trechos de código inseridos em classes desenvolvidas na linguagem Java (Quadro 22).

```
      private int id;
      private String type;
      private String name;
      private String ref;
      private String classe;
      private int cityidx;
      private int roadid;
      private String oneway;
      private int maxspeed;
      private String surface;
      private String junction;
      private String highway;
      private Object coordenadas;

      // ...

      estradas estradas = new estradas();
      estradas.setId(contador++);
      estradas.setType(type);
      estradas.setName(label);
      estradas.setRef(nomeRef);
      estradas.setClasse(classeosm);
      estradas.setCityidx(cityIdx);
      estradas.setRoadid(roadID);
      estradas.setOneway(sentidoosm);
      estradas.setMaxspeed(velocidadeosm);
      estradas.setSurface(surface);
```

```
estradas.setJunction(junction);
estradas.setCoordenadas(coordenadas);
dao.Inserirestradas(estradas);

// ...

public DaoViarioImportar() {
    con = ConnectionFactoryPostgreSQL.getConnection();
    try {
        stmtInserirEstradas = con.prepareStatement("INSERT INTO viario_mapa1"
        + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ST_GeomFromText(?))");
    } catch (SQLException ex) {
        Logger.getLogger(DaoViarioImportar.class.getName()).log(Level.SEVERE, null, ex);
    }
}

// ...

public boolean Inserirestradas(estradas estradas) {
    try {
        stmtInserirEstradas.setInt(1, estradas.getId());
        stmtInserirEstradas.setString(2, estradas.getType());
        stmtInserirEstradas.setString(3, estradas.getName());
        stmtInserirEstradas.setString(4, estradas.getRef());
        stmtInserirEstradas.setString(5, estradas.getClasse());
        stmtInserirEstradas.setInt(6, estradas.getCityidx());
        stmtInserirEstradas.setInt(7, estradas.getRoadid());
        stmtInserirEstradas.setString(8, estradas.getOneway());
        stmtInserirEstradas.setInt(9, estradas.getMaxspeed());
        stmtInserirEstradas.setString(10, "LINESTRING(" + estradas.getCoordenadas() + ")");
        stmtInserirEstradas.executeUpdate();
        return true;
    } catch (SQLException ex) {
        return false;
    }
}
```

Quadro 22 - Código: Trechos de códigos para leitura e gravação de estradas

Com as estradas inseridas no banco de dados PostgreSQL o próximo passo é selecionar uma estrada e gerar um *buffer* (área externa de contorno com tamanho variável) para assim poder buscar os pontos de conexão com o segundo mapa de estradas. A Figura 36 apresenta esse processo de criação do *buffer* e seleção dos pontos de conexão.

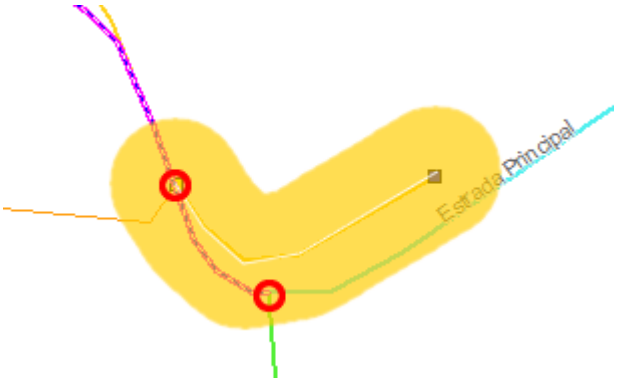

Figura 36 - Buffer e localização dos pontos de conexão

Para realização do *buffer* e seleção dos pontos, são utilizadas pesquisas nos dados das estradas gravadas anteriormente no banco da dados PostgreSQL. O Quadro 23 exibe alguns trechos da classe responsável por essa tarefa.

```
  public DaoConexoes() {

        con = ConnectionFactoryPostgreSQL.getConnection();
        try {
            stmtConexoesEstradas = con.prepareStatement("SELECT id, roadid, classe, ST_AsText(geome),
ST_AsText(dump), ST_AsText(buffer2), ST_AsText(ST_StartPoint(ST_GeometryN(geome, 1))),
ST_AsText(ST_EndPoint(ST_GeometryN(geome, 1))) FROM (SELECT *, ST_Distance(ponto, dump) as distance,
ST_DWithin(buffer2, dump, (? / 100000)) as dentro FROM (SELECT *, (ST_DumpPoints(tabela.geome)).geom as dump,
ST_GeomFromText(?) as ponto FROM (SELECT estrada2.id as id, estrada2.roadid as roadid, estrada2.classe as classe,
estrada2.geomanalise as geome, buffer.geom as buffer2 FROM (SELECT ST_Buffer(ST_GeomFromText(?), (? / 100000),
'quad_segs=8') as geom) as buffer, (SELECT * FROM viario_estradas2 WHERE geom &&
ST_Envelope(ST_Buffer(ST_GeomFromText(?), (? / 100000), 'quad_segs=8'))) as estrada2 WHERE
ST_Intersects(buffer.geom, estrada2.geom)) AS tabela) AS geral) AS final WHERE dentro = 't' ORDER BY distance ASC
LIMIT 4");
        } catch (SQLException ex) {
            Logger.getLogger(DaoConexoes.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public ArrayList<ViarioCOPELConexoes> LocalizarConexoesU(String linhaConexao, double tamanhoBuffer) {

        try {
            ArrayList<ViarioCOPELConexoes> arrayConexoes = new ArrayList<ViarioCOPELConexoes>();
            stmtConexoesEstradas.setDouble(1, tamanhoBuffer);
            stmtConexoesEstradas.setString(2, pontoConexao);
            stmtConexoesEstradas.setString(3, pontoConexao);
            stmtConexoesEstradas.setDouble(4, tamanhoBuffer);
            stmtConexoesEstradas.setString(5, pontoConexao);
            stmtConexoesEstradas.setDouble(6, tamanhoBuffer);
            ResultSet rs = stmtConexoesEstradas.executeQuery();
            while (rs.next()) {
                arrayConexoes.add(retornaConexoes(rs));
            }
            return arrayConexoes;

        } catch (SQLException ex) {
            return null;
        }
    }
```

Quadro 23 - Código: Realização do buffer e localização dos pontos de conexão

Após esse processo é possível gerar o mapa final concatenado no formato ".mp" para ser visualizado no software GPSMapEdit e assim conferir visualmente a integridade das estradas analisadas e unidas.

A Figura 37 exibe algumas estradas e seus pontos de conexão após a realização do processo de concatenação dos mapas.

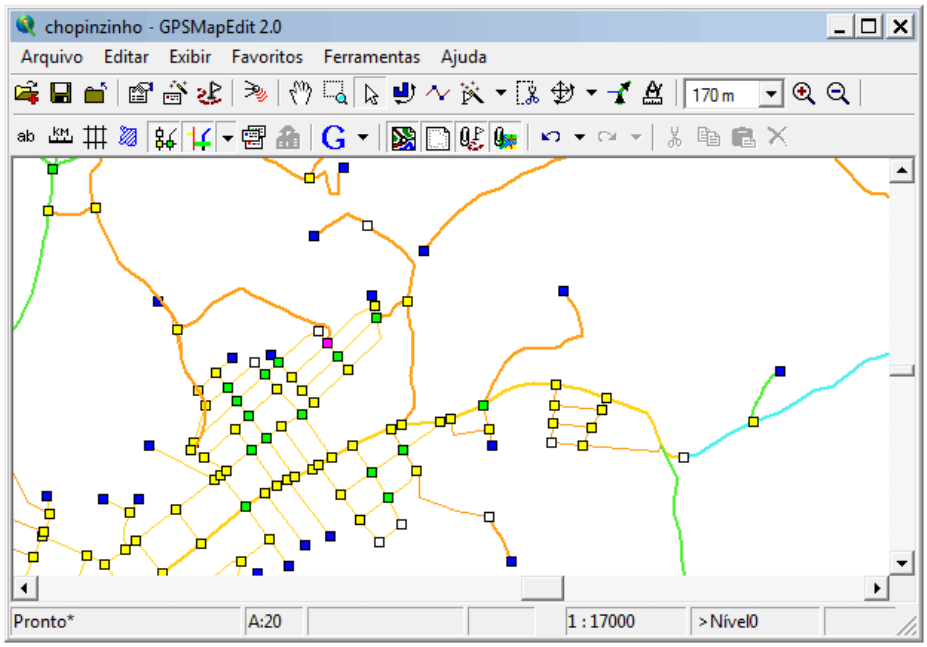

Figura 37 - Visualização das estradas na localidade de Chopinzinho - PR

4.4.4 Conversão do Mapa de Estradas

Foram convertidos, do formato ".mp" para ".osm", mais de 450 mil Km de estradas incluindo vias urbanas e rurais provenientes da base de dados da COPEL.

Para a conversão, é preciso haver a consistência dos dados, onde o mapa vetorial precisa conter pontos iniciais e finais dos segmentos precisamente conectados para que o cálculo do roteamento seja efetuado.

Cada linha no mapa é uma estrada e possui características próprias que posteriormente serão utilizadas para visualização e roteamento.

Na Tabela 16 é realizado o comparativo de uma mesma estrada antes e após a conversão.

Tabela 16 - Comparativo da mesma estrada nos formatos ".mp" e ".osm"

Formato MP	Formato OSM

[POLYLINE]
Type=0x1
Label=~[0x05]PR-490
CityIdx=4985
RoadID=609
RouteParam=6,3,1,0,0,0,0,0,0,0,0,0
Data0=(-23.79667,-53.89341),(-23.79321 ...
Label2=Rodovia do Chá
[END]
	`<way id="609" user="jr" version="3">`
	`<nd ref="258123"/>`
	`<tag k="highway" v="motorway"/>`
	`<tag k="name" v="PR-490"/>`
	`<tag k="ref" v="Rodovia do Chá"/>`
	`<tag k="maxspeed" v="108"/>`
	`<tag k="lanes" v="2"/>`
	`<tag k="oneway" v="yes"/>`
	`</way>`

É possível representar as características do elemento no mapa como a categoria, a velocidade máxima, o sentido da via entre outros atributos.

O Quadro 24 contém a classe desenvolvida em linguagem Java responsável por interpretar os valores das estradas e converter do formato “.mp” para o formato “.osm”.

```
package utilitarios;
public class Analise {
  private String nomeDaVia;
  private int pistas;
  private String classeDaVia;
  private int velocidade;

  public Analise Padronizacao(String type) {
      Analise analise = new Analise();

      if (type.equals("0x16")) {
          analise.nomeDaVia = "Trilha 4x4";
          analise.pistas = 1;
          analise.classeDaVia = "track";
          analise.velocidade = 5;

      } else if (type.equals("0xa")) {
          analise.nomeDaVia = "Estrada Precaria";
          analise.pistas = 1;
          analise.classeDaVia = "service";
          analise.velocidade = 20;

      } else if (type.equals("0x6")) {
          analise.nomeDaVia = "Viario Urbano";
          analise.pistas = 1;
          analise.classeDaVia = "residential";
          analise.velocidade = 40;

      } else if (type.equals("0x5")) {
          analise.nomeDaVia = "Estrada de Terra";
          analise.pistas = 1;
          analise.classeDaVia = "road";
          analise.velocidade = 40;

      } else if (type.equals("0x4")) {
          analise.nomeDaVia = "Estrada Pavimentada";
          analise.pistas = 1;
          analise.classeDaVia = "tertiary";
          analise.velocidade = 60;

      } else if (type.equals("0x1")) {
          analise.nomeDaVia = "Rodovia Federal";
          analise.pistas = 2;
          analise.classeDaVia = "motorway";
          analise.velocidade = 110;
      }
      return analise;
  }
}
```

Quadro 24 - Código: Conversão das estradas em “.mp” para “.osm”

É preciso ler o mapa base vetorial no formato “.mp” e percorrer linha por linha para assim identificar e converter cada característica, gerando assim um novo mapa no formato “.osm”.

No Quadro 25 é possível visualizar a classe desenvolvida em linguagem Java responsável pela leitura e gravação das linhas do mapa vetorial.

```
package beans;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.util.ArrayList;
import java.util.List;
import utilitarios.RemoverAcentos;
import utilitarios.Analise;

public class Analise {

  public static void criarRodovias(BufferedReader bin, BufferedWriter outParana) {

        String processoEstrada = "nao";
        String gravaEstrada = "nao";
        String item = "";
        String type = "";
        String label = "";
        int roadID = 0;
        int lanes = 1;
        String surface = "";
        int[] routeParam = null;
        List<String> listaCoord = new ArrayList<String>();
        List<String> listaNos = new ArrayList<String>();
        List<String> lsNode = new ArrayList<String>();
        String listadeNos = "nao";
        float[][] data = null;
        int velocidadeosm = 0;
        String nomeRef = "";
        String numRota = "";
        String sentidoosm = "no";
        String classeosm = "";

        try {
            String linhaLeitura = bin.readLine();

            while (linhaLeitura != null) {
                if (linhaLeitura.contains("[POLYLINE]")) {
                    linhaLeitura = bin.readLine();
                    processoEstrada = "sim";
                }

                if (processoEstrada.equals("sim") && !linhaLeitura.isEmpty() && gravaEstrada.equals("nao")) {
                    if (linhaLeitura.contains("[END]")) {
                        gravaEstrada = "sim";
                    } else {
                        item = linhaLeitura.substring(0, linhaLeitura.indexOf("="));
                        linhaLeitura = linhaLeitura.substring(linhaLeitura.indexOf("=") + 1, linhaLeitura.length());

                        if (item.equals("Type")) {
                            type = linhaLeitura;

                        } else if (item.equals("Label")) {
                            label = linhaLeitura;

                        } else if (item.equals("RoadID")) {
                            roadID = Integer.parseInt(linhaLeitura);

                        } else if (item.equals("RouteParam")) {
                            String[] aux = linhaLeitura.split(",");
                            int[] routeParamAux = new int[aux.length];
```

```
                    for (int i = 0; i < aux.length; i++) {
                        routeParamAux[i] = Integer.parseInt(aux[i]);
                    }
                    routeParam = routeParamAux.clone();
                } else if (item.substring(0, 4).equals("Data")) {

                    listaCoord.clear();
                    linhaLeitura = linhaLeitura.substring(1);
                    linhaLeitura = linhaLeitura.substring(0, linhaLeitura.length() - 1);
                    String aux[] = linhaLeitura.split("\\),\\(");
                    float[][] dataAux = new float[aux.length][2];

                    for (int i = 0; i < aux.length; i++) {

                        String[] aux2 = aux[i].split(",");
                        dataAux[i][0] = Float.parseFloat(aux2[0]);
                        dataAux[i][1] = Float.parseFloat(aux2[1]);
                    }
                    data = dataAux.clone();
                } else if (item.substring(0, 3).equals("Nod")) {

                    while (item.substring(0, 3).equals("Nod")) {
                        String[] aux = linhaLeitura.split(",");
                        String valor = aux[1];
                        listaNos.add(valor);
                        linhaLeitura = bin.readLine();

                    }
                    gravaEstrada = "sim";
                    listadeNos = "sim";
                }
            }
        }

        if (processoEstrada.equals("sim") && gravaEstrada.equals("sim")) {
            label = RemoverAcentos.remover(label);
            numRota = String.valueOf(roadID);
            outParana.write("<way id='" + numRota + "' visible='true' version='1'>");
            outParana.newLine();
            for (int i = 0; i < listaNos.size(); i++) {
                outParana.write("<nd ref='" + String.valueOf(listaNos.get(i)) + "' />");
                outParana.newLine();

            }
            outParana.write("<tag k='highway' v='" + classeosm + "' />");
            outParana.newLine();

            outParana.write("<tag k='lanes' v='" + lanes + "' />");
            outParana.newLine();

            outParana.write("<tag k='maxspeed' v='" + velocidadeosm + "' />");
            outParana.newLine();

            if (nomeRef != "") {
                outParana.write("<tag k='ref' v='" + nomeRef + "' />");
                outParana.newLine();

            }
            outParana.write("<tag k='oneway' v='" + sentidoosm + "' />");
            outParana.newLine();

            outParana.write("<tag k='name' v='" + label + "' />");
            outParana.newLine();

            outParana.write("</way>");
            outParana.newLine();

            for (int i = 0; i < lsNode.size(); i++) {
                outParana.write("<node id='" + lsNode.get(i) + "' visible='true' lat='" + data[i][0] + "' lon='"
                        + data[i][1] + "' version='1' />");
                outParana.newLine();

            }
            outParana.newLine();
```

```
                }
                linhaLeitura = bin.readLine();
            }
        } catch (IndexOutOfBoundsException e) {
            System.err.println("Erro: Falha no index do array");
        } catch (Exception e) {
            System.err.println("Erro: Arquivo nao encontrado");
        }
    }
}
```

Quadro 25 - Código: Leitura e gravação dos mapas vetoriais

Para a correta formatação do nome dos elementos é utilizada uma classe estática de remoção dos acentos, conforme observado no Quadro 26.

```
package utilitarios;

public class RemoverAcentos {
  static String acentuado = "çÇáéíóúýÁÉÍÓÚÝàèìòùÀÈÌÒÙãõñäëïöüÿÄËÏÖÜÃÕÑâêîôûÂÊÎÔÛ";
  static String semAcento = "cCaeiouyAEIOUYaeiouAEIOUaonaeiouyAEIOUAONaeiouAEIOU";
  static char[] tabela;
  static {
      tabela = new char[256];
      for (int i = 0; i < tabela.length; ++i) {
          tabela[i] = (char) i;
      }
      for (int i = 0; i < acentuado.length(); ++i) {
          tabela[acentuado.charAt(i)] = semAcento.charAt(i);
      }
  }
  public static String remover(final String s) {
      StringBuffer sb = new StringBuffer();
      for (int i = 0; i < s.length(); ++i) {
          char ch = s.charAt(i);
          if (ch < 256) {
              sb.append(tabela[ch]);
          } else {
              sb.append(ch);
          }
      }
      return sb.toString();
  }
}
```

Quadro 26 - Código: Classe para remover os acentos

O resultado final da conversão é apresentado na Figura 38. Pode-se visualizar uma comparação entre o mapa de estradas no formato “.mp” (visualizado no GPSMapEdit), nas cores verde e laranja, sobreposto (com deslocamento intencional para direita) o mapa já convertido para o formato “.osm” (visualizado no JOSM), em tons de azul.

Figura 38 - Mapas vetoriais sobrepostos para efeito de comparação

4.4.5 Conferência Visual dos Mapas

Para conferir se a conversão foi realizada de maneira adequada, o melhor procedimento é a visualização gráfica dos mapas no formato final ".osm". Assim todas as falhas de conexão podem ser localizadas e corrigidas em uma posterior conversão dos mapas. A visualização de mapas no formato ".osm" foi realizada com o software JOSM, conforme a Figura 39 apresenta.

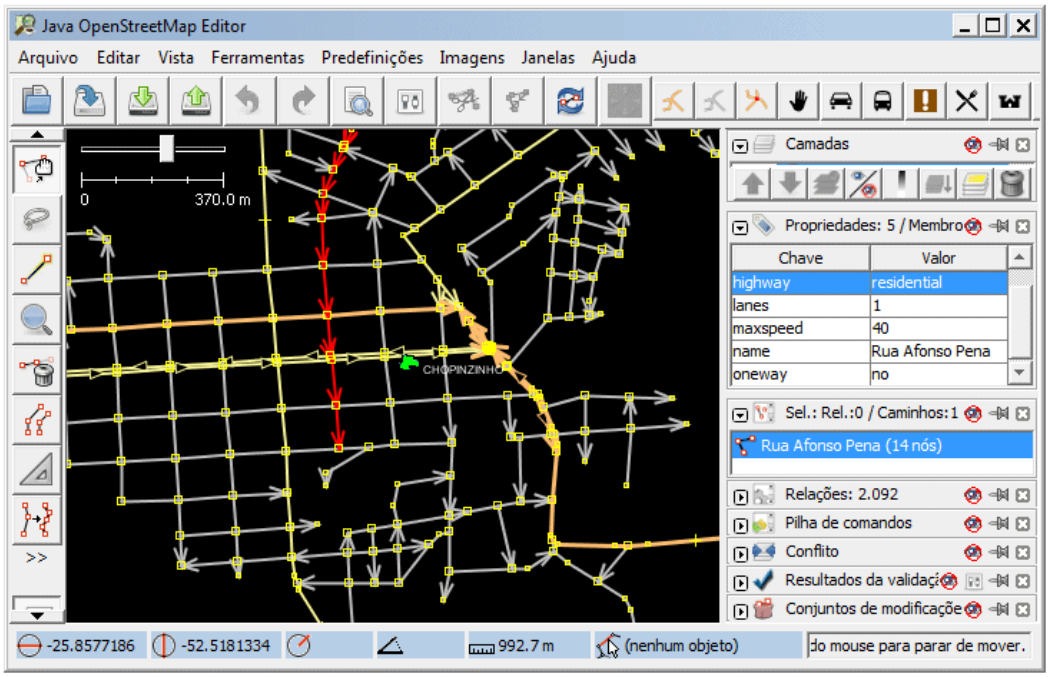

Figura 39 - Mapas convertido visualizado no JOSM

4.4.6 Gerador dos Mapas Binários

A última etapa no processo de criação dos mapas é a conversão para o formato binário com a extensão “.bin”, assim os mapas podem ser lidos pelo Navegador GPS e exibidos na tela do *Tablet*.

O Quadro 27 exibe um parte do código responsável por interpretar os mapas no formato “.osm” e convertê-los.

```
private RTree sdingTreee = null;
  private int slotLvle = 1;
  private void seedRec(Ponto sendedPonto, Ponto sdPonto, int level) throws Exception {

if (sendedPonto.getElementoType() == Ponto.LEAF_Ponto)

        throw new IllegalArgumentException("Não existe ponto");
      Ponto[] chPontos = null;

      if (level != slotLvle)

          chPontos = new Ponto[sendedPonto.getTotalElementos()];
          Elemento[] elmts = sendedPonto.getAllElementos();
          Elemento[] newElmts = null;

      if (level != slotLvle)

          newElmts = new Elemento[sendedPonto.getTotalElementos()];

      else {

          newElmts = new Elemento[1];
          newElmts[0] = new NonLeafElemento(new Rect(), Ponto.NOT_DEFINED);

      }

       for (int i = 0; i < sendedPonto.getTotalElementos(); i++) {

          if (level != slotLvle) {

              newElmts[i] = (NonLeafElemento) ((NonLeafElemento) elmts[i]).clone();
              chPontos[i] = chdPontos.getPonto(fileHdr.getFile(), fileName, sdPonto.getPontoIndex(),
                      sendedPonto.getElementoType(), fileHdr);
              newElmts[i].setPtr(chPontos[i].getPontoIndex());
              seedRec(sdingTreee.getReadPonto(elmts[i].getPtr()), chPontos[i], level + 1);

          } else {
              newElmts[0].getRect().expandToInclude(elmts[i].getRect());
          }
      }

       sdPonto.insertElemento(newElmts, false);
  }
```

Quadro 27 - Código: Trecho do conversor de mapas “.osm” para “.bin”

Após esse processo basta apenas inserir os mapas no Navegador GPS e realizar a visualização e roteamento.

4.5 RESULTADO PRÁTICO EM CAMPO

Para exemplificar a utilização, e comprovar a utilidade do Navegador GPS desenvolvido para a companhia COPEL, foi realizado um acompanhamento de campo, com uma equipe da Agência de Atendimento de Pato Branco.

A Figura 40 apresenta uma foto registrada na saída do pátio da COPEL, e é possível visualizar o Navegador GPS funcionando em um dispositivo móvel - *Tablet* da Companhia.

Figura 40 - Eletricistas Rafael Pozza Cantelli e Jurandir Antonio Biedacha
Fonte: JORNAL (2012)

Com o acompanhamento realizado, foi possível observar a utilização do aplicativo desenvolvido e assim coletar informações para a melhoria do aplicativo.

A Figura 41 apresenta ao fundo uma área urbana de Pato Branco densamente povoada, o que pode ser comprovado na visualização conjunta com a tela do Navegador GPS.

Figura 41 - Região urbana de Pato Branco e o Tablet executando o GPS COPEL.

Visualizando o mapa e todos os ícones representativos da rede elétrica, a próxima etapa foi a pesquisa de um ponto de interesse para assim ser realizada a navegação.

Em todo o trajeto foi possível acompanhar a rede elétrica e comprovar a qualidade das informações cadastrais da COPEL.

Conforme a Figura 42 apresenta, foi encontrado o poste da rede COPEL.

Figura 42 - Poste localizado na Praça Municipal de Pato Branco
Fonte: JORNAL2 (2012)

A navegação continuou agora seguindo um trajeto rural, em uma região sem acesso a Internet, celular ou rádio, o que impossibilitaria uma pesquisa e localização sem a utilização do Navegador GPS COPEL.

A Figura 43 apresenta o Navegador GPS exibindo a estrada rural.

Figura 43 - Navegação na área rural de Pato Branco

Para esse teste em campo foram realizadas diversas pesquisas de pontos da rede elétrica assim como pesquisas de localidades. O cálculo de roteamento mostrou-se eficaz exibindo uma opção de trajeto compatível com a realidade, graças aos mapas vetoriais desenvolvidos.

Comprovou-se que independente do roteamento, a grande vantagem da utilização do GPS COPEL é a visualização do viário e da rede elétrica completa, o que possibilitou uma orientação intuitiva por parte dos funcionários. Pois com o navegador foi possível saber a posição onde se estava e com as opções de *zoom* poder ampliar a área visualizada do mapa.

A Figura 44 apresenta uma imagem da rodovia BR-158 de Coronel Vivida sentido Pato Branco, em época de primavera.

Figura 44 - Bela imagem na localidade de São Roque do Chopim (Pato Branco)

Em todos os locais percorridos notou-se a satisfação dos eletricistas em poder visualizar os pontos da rede identificando cada equipamento.

Os utilizadores em geral elogiaram a praticidade do Navegador GPS COPEL.

4.6 MATÉRIAS E RECONHECIMENTO

Foram publicadas matérias que referenciam a implantação e utilização do Navegador GPS COPEL. O conteúdo e as Figuras foram copiados na íntegra.

4.6.1 Primeira Matéria na Intranet da COPEL

Matéria publicada em 17 de Janeiro de 2012.

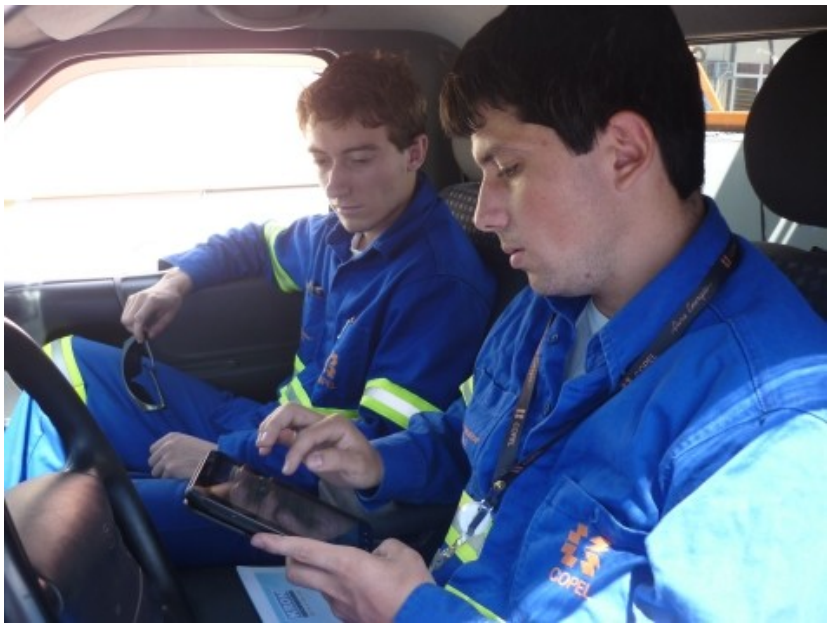

Figura 45 - Os eletricistas Guilherme e Rossinni, aprovaram a novidade.
Fonte: INTRANET1 (2012)

COPEL investe em tecnologia para o trabalho dos eletricistas - A COPEL investiu em tecnologia para facilitar e tornar mais eficaz o trabalho dos eletricistas. A Companhia adquiriu 1200 tablets que foram homologados para serem utilizados em substituição aos antigos smartphones na execução dos serviços. Há cerca de um mês, 33 equipes começaram a testar os equipamentos na Superintendência de Distribuição Leste (SDL). A reação inicial dos eletricistas é de muita satisfação.

Mesmo em período de testes do protótipo, os tablets já favorecem o trabalho das equipes. O aparelho incorporou diversas funções. Além do SOD Móvel – software em que os eletricistas recebem os serviços –, também possui navegador GPS e funciona como celular.

A tela de sete polegadas do modelo Samsung Galaxy, com sistema Android combinada com tecnologia *touchscreen* (sensível ao toque), melhorou a visualização e a navegação. “Uma das principais diferenças é para enxergar sob o sol”, conta o eletricista Rossinni Bordin, da Agência Centro. Isso porque a tela é fotossensível e o brilho se intensifica automaticamente, de acordo com a luminosidade do ambiente.

Sistemas de trabalho
Utilizar o SOD Móvel para trabalhar com as solicitações de serviço ficou muito mais fácil. É possível pesquisar dados do cliente, da unidade consumidora, da instalação, do medidor, do transformador. Tudo organizado em pastas. “A estrutura em árvore (disposição hierárquica) facilita muito o nosso trabalho”, comenta Guilherme Weigert, da Agência Centro.

A tecnologia do tablet também contribui para economia de papel. As planilhas com dados dos serviços e os manuais, que antes eram impressos, agora poderão ser consultadas em arquivos PDF, no tablet. “A atualização do SOD pode ser realizada diretamente pela rede do celular, tornando a comunicação praticamente instantânea”, explica Ana Paula Peracetta, supervisora do Setor de Distribuição de Serviços Leste.

Para complementar as melhorias, o tablet possui um navegador GPS contendo informações georreferenciadas da rede primária e secundária da COPEL. Desenvolvido pelo copeliano James Rebelato, da Regional Oeste – com apoio da Superintendência de Engenharia de Distribuição (SED) e da Superintendência de Tecnologia da Informação (STI) –, este GPS customizado integra informações da rede elétrica da COPEL, permitindo a visualização no mapa e navegação com orientações por voz até o ponto desejado.

O eletricista não precisa saber o endereço do consumidor. Basta informar o número da unidade consumidora, do medidor, da chave ou do transformador, que o software faz a navegação até o destino. “Com esse sistema, espera-se que as equipes de campo encontrem os locais de realização dos serviços com maior facilidade, melhorando os índices de atendimento”, destaca o gerente da Divisão de Operação e Estudos de Proteção da SED, Marcelo Gonçalves Santos.

Desenvolvimento e distribuição
Para que os tablets chegassem aos eletricistas, foi necessário o envolvimento das equipes da STI e da SED, que adequaram o aplicativo de gerência de serviços, o SOD Móvel, ao tablet, realizando testes e melhorias.

De acordo com Ubiraci Gomes da Silva, da Divisão de Operação e Estudos de Proteção da SED, a substituição dos antigos smartphones pelos tablets em todas as regionais será realizada em pequenos lotes, para facilitar possíveis correções no sistema. Serão também programadas visitas às regionais para treinar multiplicadores na instalação e utilização dos equipamentos. “Nossa meta é que, até o final do primeiro semestre de 2012, os tablets sejam distribuídos em todas as regionais”, afirma. (INTRANET1, 2013).

4.6.2 Segunda Matéria na Intranet da COPEL

Matéria publicada em 30 de Janeiro de 2012.

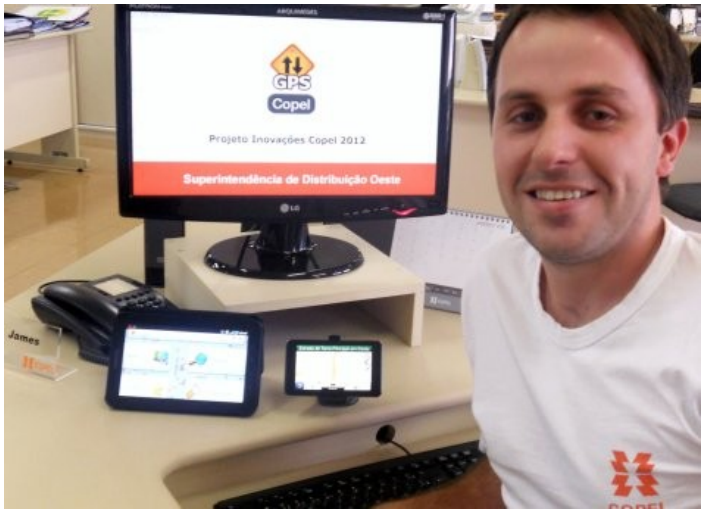

Figura 46 - James e o GPS, para ninguém mais se perder em campo
Fonte: INTRANET2 (2012)

GPS COPEL facilita a localização de consumidores - Os investimentos em ferramentas com tecnologia de ponta são mesmo uma mão na roda para facilitar a vida de centenas de profissionais da COPEL. O interessante, no entanto, é que o resultado final muitas vezes tem um começo bem prosaico, fruto do talento e da criatividade de empregados que inicialmente miram a solução de problemas da sua própria rotina de trabalho.

É o caso, por exemplo, do projeto GPS COPEL – vide COPEL Online de 18.01.2012 -, que culminou na recente aquisição de 1.200 tablets para eletricistas da Companhia. Turbinada, essa ferramenta top do mercado promete revolucionar a execução dos serviços de campo da Distribuição naquilo que há de mais precioso para um consumidor de energia elétrica: o tempo.

O mentor desse projeto chama-se James Gustavo Black Rebelato, eletrotécnico da Regional Oeste que atua na Divisão de Projetos e Obras em Pato Branco, e que simplesmente estava cansado de se perder na área rural quando a serviço da COPEL. Formado recentemente em Análise e Desenvolvimento de Sistemas pela Universidade Federal Tecnológica do Paraná (UTFPr), ele tem 28 anos de idade e dois de COPEL.

Há cerca de um ano James aproveitou o trabalho de conclusão do curso para desenvolver o embrião do GPS COPEL. Ele conseguiu sobrepor o sistema Webgeo da COPEL -- inicialmente da região de Pato Branco e posteriormente do Estado todo --, ao sistema de navegação do GPS automotivo Garmin (modelo que permite tal intervenção), e com isso pôde localizar com precisão os consumidores com todas as informações elétricas correspondentes. E nunca mais se perdeu em campo.

Depois da adoção corporativa do CIS Energia, que suprimiu o velho Local/Rota/Conta, a sacada tecnológica do James surgiu como solução interessante para economizar tempo e dinheiro na localização dos consumidores, seja em atendimento a situações de emergência ou serviços comerciais. Além de encurtar caminhos, sobretudo na área rural, o uso do GPS dispensa a impressão de mapas elétricos.

No ano passado, a alternativa ganhou a apreciação da Diretoria de

Distribuição e a ela juntaram-se outros especialistas em TI. De GPS automotivo o projeto evoluiu para os tablets, que incorporam mais soluções e serviços. Mas o James continua no desenvolvimento do projeto, fazendo esse casamento de dados e informações que englobam todo o Paraná.

Segundo ele, o GPS COPEL contém em sua primeira versão todas as estradas urbanas do Estado (as rurais virão numa segunda versão), e 4,5 milhões de unidades consumidoras com os respectivos transformadores, chaves e rede primária (AT).

"Estou muito feliz por uma ideia estar se tornando uma ferramenta de trabalho útil para tantos colegas de COPEL. A empresa está apoiando a inovação, e a maior prova disso é o reconhecimento e o apoio que estou recebendo das gerências e colegas de vários setores", afirma James com o entusiasmo que lhe é peculiar. (INTRANET2, 2012).

4.6.3 Terceira Matéria na Intranet da COPEL

Matéria publicada em 12 de Abril de 2012.

Regional Oeste começa a receber os primeiros tablets - Abastecida a Regional Leste, está chegando a Cascavel a primeira metade dos 200 tablets da cota da Regional Oeste. Ela faz parte do conjunto de 1.200 equipamentos encomendados pela Diretoria de Distribuição para se constituir na mais ágil e moderna ferramenta operacional dos eletricistas da COPEL.

Além dos recursos da comunicação móvel já conhecida, a nova ferramenta junta em um só aparelho o sistema SOD Móvel e o sistema de navegação por GPS automotivo. Acopladas, essas tecnologias permitem aos eletricistas de campo localizar com precisão os consumidores com todas as informações elétricas correspondentes. E mais: encurta caminhos, contribuindo para a redução do tempo de atendimento aos clientes, bem como dispensa a impressão de mapas elétricos.

João Henrique Gross e Ubiraci Gomes da Silva, ambos da Superintendência de Engenharia de Distribuição, estão em Cascavel ministrando o treinamento de configuração e utilização do tablet para os primeiros eletricistas e eletrotécnicos multiplicadores. A dupla de instrutores conta com o reforço do mentor do projeto GPS COPEL, James Gustavo Black Rebelato, eletrotécnico da Regional Oeste que atua na Divisão de Projetos e Obras em Pato Branco.

Os atuais smartphones, usados para receber e transmitir informações de serviços de campo, devem ser gradativamente substituídos até julho pelos novíssimos tablets em todas as cinco superintendências regionais. O aparelho top de mercado é fornecido em comodato pela operadora Vivo. (INTRANET3, 2012).

Figura 47 - João, James e Ubiraci no treinamento em Cascavel
Fonte: INTRANET3 (2012)

4.6.4 Revista COPEL

Figura 48 - Matéria de reconhecimento REVISTA COPEL
Fonte: REVISTA (2013)

Matéria publicada na Revista COPEL, Edição 301 de fevereiro 2013.

GPS COPEL conquista o Sendi - O eletrotécnico James Gustavo Black Rebelato não cabia em si de contentamento ao retornar do Rio de Janeiro, no final de outubro passado, trazendo na bagagem o troféu de melhor trabalho técnico do XX Seminário Nacional de Distribuição de Energia Elétrica (Sendi), na categoria Geoprocessamento e Integração de Sistemas Técnicos. Coube a ele o privilégio de apresentar para 1.500 especialistas do Brasil o GPS COPEL, plenamente instalado em cerca de 1.600 tablets operacionais da Companhia.

Aos recursos já conhecidos da comunicação móvel, a ferramenta agrega, em um só aparelho, o sistema SOD Móvel e o sistema de navegação por GPS com base nos dados cartográficos e elétricos da COPEL. Acopladas, essas tecnologias permitem aos eletricistas e técnicos de campo localizar com precisão os domicílios de mais de 4 milhões de consumidores com todas as informações elétricas próximas, ou seja, transformadores, chaves e redes primária (AT) e secundária (BT).

Atualmente lotado no Departamento de Projetos e Obras da Regional Oeste, em Cascavel, James trabalhou intensamente nos últimos seis meses para incluir na última versão do GPS COPEL mais de 400 mil tortuosos quilômetros de estradas rurais do Estado, acesso indispensável para atendimento a 373 mil propriedades ligadas à rede de distribuição de energia. Encurtando caminhos, a inovação tecnológica contribui para a redução do tempo de atendimento aos clientes, ao mesmo tempo em que dispensa a impressão de mapas elétricos para eventuais consultas de roteiros no papel. E nada mais será como antes.

Dos 15 trabalhos técnicos apresentados pela COPEL no XX Sendi, três foram premiados em categorias distintas:

GPS COPEL - Navegador GPS para Aplicativos Móveis com Base nos Dados Cartográficos e Elétricos da COPEL, de autoria de James Gustavo Back Rebelato (SDO), Geraldo Cezar Correa (STI) e João Henrique Gross (SED). (REVISTA, 2013).

4.6.5 Matéria para Jornal Diário do Sudoeste

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Diretoria-Geral
Assessoria de Comunicação

UTFPR

Matéria publicada no: **Diário do Sudoeste**
Data: **07/11/2012** Página: **A12**

Elaborada por: ( ) UTFPR (DECOM/ASCOM) ( ) UTFPR (ASCOM/DECOM) e Jornal
( ) UTFPR (ASCOM) (X) Próprio jornal ( ) Outro

InfoDiário

Pato-branquense cria aplicativo para GPS

DAYANNE DO NASCIMENTO
dayanne@diariodosudoeste.com.br

O objetivo é ser utilizado pelos funcionários da Copel em todo o Paraná, inclusive nos locais sem cobertura de satélite ou internet

O pato-branquense James Gustavo Black Rebelato, técnico de projeto de redes da Copel em Pato Branco, criou, há cerca de um ano, um aplicativo para GPS, o GPS Copel. O aplicativo que pode ser usado em qualquer lugar, e traz como novidade a sua usabilidade principalmente na zona rural do Paraná, já que para funcionar ele não precisa da conexão com a internet ou mesmo a cobertura de satélite, como os GPS tradicionais. Conforme Rebelato, a ideia de criar o aplicativo surgiu da necessidade de elaborar um projeto para a conclusão do curso de Análise e Desenvolvimento de Sistemas da Universidade Federal Tecnológica do Paraná (UTFPR), *campus* de Pato Branco. Então ele pensou em unir o útil ao agradável e criou o aplicativo, que além de servir para o projeto da universidade, facilitou o seu trabalho e o dos demais colegas que atuam na Copel, já que o aparelho foi criado especificamente para o uso dos eletricistas quando saem fazer algum trabalho a campo.

Para melhorar a usabilidade da equipe da Copel, o aplicativo foi instalado em tablets. Para isso, a empresa fez a aquisição de 2 mil aparelhos e distribuiu entre as equipes no Estado. O aplicativo pode ser instalado em equipamentos móveis, baseados no sistema operacional Android.

Com o uso dessa nova tecnologia, as equipes, que antes necessitavam consultar mapas impressos ou moradores locais para chegarem aos seus destinos, agora contam com uma navegação guiada por voz e conseguem encontrar os clientes com muito mais rapidez e, além de economizar tempo e dinheiro, conseguem prestar um serviço de mais qualidade, podendo até mesmo informar ao cliente o horário que a equipe chegará à sua residência.

Pato-branquense James Gustavo Black Rebelato mostra o aplicativo que tem facilitado o trabalho das equipes da Copel

Câmpus Pato Branco
Via do Conhecimento Km 1
85.503-390 – Pato Branco – Paraná – Brasil
Fone: (46) 3224-2506
www.utfpr.edu.br

Ministério da Educação
BRASIL

Figura 49 - Primeira parte da matéria
Fonte: JORNAL (2012)

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Diretoria-Geral
Assessoria de Comunicação

UTFPR

Matéria publicada no: **Diário do Sudoeste**
Data: **07/11/2012** Página: **A12**

Elaborada por: ( ) UTFPR (DECOM/ASCOM) ( ) UTFPR (ASCOM/DECOM) e Jornal
( ) UTFPR (ASCOM) (X) Próprio jornal ( ) Outro

Aplicativo não tem a necessidade da conexão com a internet porque possui a integração do banco de dados da Copel e dos mapas das estradas urbanas e rurais do Paraná

Funcionamento

O GPS Copel possui as funcionalidades típicas de um navegador GPS comercial, que somadas aos dados da rede elétrica fazem dele uma ferramenta poderosa para as equipes de campo na localização e navegação até o ponto desejado.

Para que o aplicativo funcione sem a necessidade do acesso a internet ou sinal de satélite, facilitando assim a localização dos clientes, principalmente na zona rural, Rebelato adaptou os dados do sistema Webgeo da Copel, composto pela integração de subestações, torres de transmissão e usinas com os mapas de ruas urbanas e estradas rurais do Paraná. Primeiramente ele aplicou os dados da região de Pato Branco e depois de todo o Estado ao sistema de navegação do GPS automotivo Garmin (modelo que permite tal intervenção), e com isso pôde localizar com precisão os consumidores com todas as informações elétricas correspondentes.

Contudo, ele destacou que o aplicativo não identifica os dados pessoais dos clientes, como nome ou número de documentos, pois a pesquisa dos locais só utiliza informações técnicas, que são de interesse apenas para as pessoas diretamente envolvidas com as atividades da Copel. A pesquisa pode ser feita ou com o número de identificação do consumidor, com número de série do medidor ou o endereço do cliente. Desta forma, a identificação dos clientes fica protegida. Outro detalhe que ele esclareceu é que, se o tablet for roubado ou perdido, em três meses ele bloqueia todo o sistema. Isso acontece porque a cada três meses os funcionários precisam fazer a atualização dos dados e isso não acontecendo o sistema é bloqueado automaticamente.

Além dos clientes, o aplicativo permite outros tipos de buscas, por exemplo, se a equipe precisar fazer o conserto da rede elétrica em algum poste, localizado na zona rural, mas longe de residências, o aplicativo permite localizá-lo pelo seu número de série exatamente onde ele está. O mesmo pode ser feito com transformadores, chaves (equipamentos de manobra), reguladores de tensão, subestações, usinas e torres de transmissão, pois o aplicativo mostra toda a rede da Copel em todo o Estado do Paraná.

Premiação

Pela iniciativa em criar o aplicativo, Rebelato recebeu no mês de outubro um dos mais importantes prêmios do setor de energia elétrica do Brasil, o prêmio Sendi XX, que foi entregue durante o Seminário Nacional de Distribuição de Energia Elétrica, realizado entre os dias 22 e 26 de outubro, no Rio de Janeiro.

Câmpus Pato Branco
Via do Conhecimento Km 1
85.503-390 – Pato Branco – Paraná – Brasil
Fone: (46) 3224-2506
www.utfpr.edu.br

Ministério da Educação
GOVERNO FEDERAL BRASIL PAÍS RICO É PAÍS SEM POBREZA

Figura 50 - Segunda parte da matéria
Fonte: JORNAL2 (2012)

Pato-branquense desenvolve aplicativo para GPS - Aplicativo foi criado com o objetivo de ser utilizado pelos funcionários da COPEL em todo o Paraná, inclusive nos locais sem cobertura de satélite ou Internet

O pato–branquense James Gustavo Black Rebelato, técnico de projeto de redes da COPEL em Pato Branco criou a cerca de um ano, um aplicativo para GPS, o GPS COPEL. O aplicativo que pode ser usado em qualquer lugar, trás como novidade a sua usabilidade principalmente na zona rural do Paraná, já que para funcionar ele não precisa da conexão com a Internet ou mesmo a cobertura de satélite, como os GPS tradicionais. Conforme Rebelato, a ideia de criar o aplicativo surgiu da necessidade de elaborar um projeto para a conclusão do curso de Análise e Desenvolvimento de Sistemas da Universidade Federal Tecnológica do Paraná (UTFPR), campus de Pato Branco. Então ele pensou em unir o útil ao agradável e criou o aplicativo, que além de servir para o projeto da universidade, facilitou o seu trabalho e dos demais colegas que atuam na COPEL, já que o aparelho foi criado especificamente para o uso dos eletricistas quando saem fazer algum trabalho a campo.

Para melhorar a usabilidade da equipe da COPEL, o aplicativo foi instalado em tablets. Para isso a COPEL fez a aquisição 2.000 aparelhos e distribuiu entre as equipes no Estado. O aplicativo pode ser instalado em equipamentos móveis, baseados no sistema operacional Android.

Com o uso dessa nova tecnologia as equipes que antes necessitavam consultar mapas impressos ou moradores locais para chegarem aos seus destinos, agora contam com uma navegação guiada por voz e conseguem encontrar os clientes com muito mais rapidez e além de economizar tempo e dinheiro, conseguem prestar um serviço de mais qualidade, podendo até mesmo informar ao cliente, o horário que a equipe chegará a sua residência.

Funcionamento
O GPS COPEL possui as funcionalidades típicas de um navegador GPS comercial, que somadas aos dados da rede elétrica, fazem dele uma ferramenta poderosa para as equipes de campo na localização e navegação até o ponto desejado.

Para que o aplicativo funcione sem a necessidade do acesso a Internet ou sinal de satélite, facilitando assim a localização dos clientes, principalmente na zona rural, Rebelato adaptou os dados do sistema Webgeo da COPEL, composto pela integração de subestações, torres de transmissão e usinas com os mapas de ruas urbanas e estradas rurais do Paraná. Primeiramente ele aplicou os dados da região de Pato Branco e depois de todo o Estado, ao sistema de navegação do GPS automotivo Garmin (modelo que permite tal intervenção), e com isso pôde localizar com precisão os consumidores com todas as informações elétricas correspondentes.

Contudo, ele destacou que o aplicativo não identifica os dados pessoais dos clientes, como nome ou número de documentos, pois a pesquisa dos locais só utiliza informações técnicas, que são de interesse apenas para as pessoas diretamente envolvidas com as atividades da COPEL. A pesquisa pode ser feita ou com o número de identificação do consumidor, com número de série do medidor ou o endereço do cliente. Desta forma, a identificação dos clientes fica protegida. Outro detalhe que ele esclareceu é que se o tablet for roubado ou perdido, em três meses ele bloqueia todo o sistema. Isso acontece porque a cada três meses os funcionários precisam fazer a atualização dos dados e isso não acontecendo, o sistema é bloqueado automaticamente.

Além dos clientes, o aplicativo permite outros tipos de buscas, por exemplo, se a equipe precisar fazer o concerto da rede elétrica em algum poste, localizado na zona rural, mas longe de residências, o aplicativo permite localizá-lo pelo seu número de série, exatamente onde ele está. O mesmo pode ser feito com transformadores, chaves (equipamentos de manobra), reguladores de tensão, subestações, usinas e torres de transmissão, pois o aplicativo mostra toda a rede da COPEL, em todo o Estado do Paraná.

Premiação
Pela iniciativa em criar o aplicativo, Rebelato recebeu no mês de outubro um dos mais importantes prêmios do setor de energia elétrica do Brasil, o prêmio Sendi XX, que foi entregue durante o Seminário Nacional de Distribuição de Energia Elétrica, realizado no último entre os dias 22 e 26 de outubro, no Rio de Janeiro. (JORNAL, 2012).

4.6.6 Prêmio SENDI Rio de Janeiro

E-mail recebido parabenizando pela seleção do trabalho no SENDI.

Assunto: Divulgação de Resultados SENDI 2012 - **15/08/2012 18:47**

Caro **James Gustavo Black Rebelato**,

Parabéns! Seu trabalho **"GPS COPEL - Navegador GPS para aplicativos móveis com base nos dados cartográficos e elétricos da Companhia Paranaense de Energia - COPEL"** de **tema : Geoprocessamento e integração de sistemas técnicos** foi selecionado para compor a grade de trabalhos presenciais do XX SENDI - Seminário Nacional de Distribuição de Energia Elétrica, no Rio de Janeiro.

A Comissão Técnica do XX SENDI agradece pela sua participação.

Foram mais de 700 trabalhos inscritos e realizadas mais de 2.000 avaliações com a participação de profissionais das Empresas Distribuidoras, sendo selecionados 240 trabalhos presenciais que serão apresentados durante a edição 2012 do XX SENDI.

As instruções e o modelo de apresentação estão disponíveis no site www.sendi.org.br.

Para conhecer a relação completa dos trabalhos selecionados, você pode acessar o site. Em breve, iremos divulgar a grade de programação oficial.

Com a certeza de encontrá-lo no XX SENDI, aqui na Cidade Maravilhosa!

Grande abraço.
Gustavo Alencar

Coordenador da Comissão Técnica do XX SENDI

A Tabela 17 apresenta o trabalho GPS COPEL na lista de premiados do SENDI.

Tabela 17 - Resultado das Trabalho Técnico

Tema	Título	Autores	Instituições
Geoprocessamento e integração de sistemas técnicos	GPS COPEL - Navegador GPS para aplicativos móveis com base nos dados cartográficos e elétricos da Companhia Paranaense de Energia - COPEL	**James Gustavo Black Rebelato** | Joao Henrique Gross | Geraldo Cezar Correa	Companhia Paranaense de Energia | COPEL

Na Figura 51, já em Pato Branco, o troféu de 1º lugar na categoria de Sistemas de Georreferenciamento sendo exibido para os Gerentes Antônio Anzolin e Paulo Moreira.

Figura 51 - Antônio Anzolin, James Rebelato e Paulo Moreira

5 CONCLUSÃO

O objetivo desse trabalho foi desenvolver um Navegador GPS para ser executado em dispositivos móveis, possibilitando a visualização e pesquisa de milhões de pontos da rede elétrica da COPEL. Outro objetivo compreendeu a geração dos mapas vetoriais necessários para visualização e roteamento com base nos mapas cartográficos da empresa.

O desenvolvimento desse trabalho possibilitou a continuidade dos estudos realizados no estágio e a expansão dos conhecimentos sobre utilização de algumas tecnologias, como o *SQLite* para o banco de dados do Navegador GPS, *PostgreSQL* para leitura e geração dos mapas da COPEL, além da utilização da linguagem de programação Java.

Com a utilização da plataforma Eclipse foi possível um desenvolvimento organizado e rápido. A depuração do Navegador foi realizada diretamente no Tablet, sendo possível corrigir assim erros em tempo real.

Outra etapa do trabalho foi a visualização dos dados, seja em formato de caracteres com o software Notepad++ ou graficamente com os visualizadores de mapas GPSMapEdit e JOSM.

A grande vantagem da geração de mapas próprios, é a possibilidade de total modificação dos dados, sendo possível alterações gráficas na visualização e alterações na categorização e detalhamento das informações sem a necessidade de utilizar mapas pagos bloqueados.

Dentre as vantagens no desenvolvimento de um Navegador GPS merece destaque a possibilidade da integração rápida com outros aplicativos da COPEL, como o sistema de envio de serviços via Satélite e a visualização de informações em tempo real de dados da rede.

Devido a adoção crescente de equipamentos móveis por parte da COPEL, o aplicativo desenvolvido Navegador GPS, tornou-se uma ferramenta reconhecida como oficial merecendo destaque entre os aplicativos já utilizados pela Companhia.

E por fim a conclusão é o sucesso da idealização desse projeto alcançando o feito histórico de mudar toda forma de navegação em campo utilizada na empresa COPEL, recebendo assim reconhecimento da imprensa e conquistando o prêmio do SENDI.

6 PERSPECTIVAS FUTURAS

Como perspectiva futura espera-se dar continuidade no desenvolvimento e implementação de novas funcionalidades ao trabalho, conforme abaixo:

- Implementação da funcionalidade de inspeção de rede de distribuição;
- Desenvolvimento do processo para reportar divergências de cadastro;
- Continuidade no processo de melhoria dos mapas urbanos e rurais;
- Inserção de novos elementos da rede elétrica da Copel;

Outra perspectiva é a intenção de escrever um livro em parceria com a UTFPR, Câmpus Pato Branco para exibir detalhadamente todo o processo de desenvolvimento do Navegador GPS e Gerador de Mapas Vetoriais.

7 REFERÊNCIAS

ANDROID. **About the Android**. Disponível em: <http://source.android.com/about/index.html>. Acesso em: 10 fev. 2013.

APLICATIVOGPS. **macmagazine**. Disponível em: <http://macmagazine.com.br/wp-content/uploads/2011/05/20-sygic.png>. Acesso em: 10 fev. 2013.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário**. Rio de Janeiro: Campus, 2000, 7ª edição.

BUGDROID. **Technologia**. Disponível em: <http://technologiallc.com/wp-content/uploads/2012/04/greentranparentwm1.png>. Acesso em: 10 fev. 2013.

COAST. **OpenStreetMap**. Disponível em: <http://www.openstreetmap.org>. Acesso em: 11 fev. 2013.

COPEL. **Visão Geral - A COPEL**. Novembro de 2012. Disponível em: <http://www.copel.com/hpcopel/root/nivel2.jsp?endereco=%2Fhpcopel%2Facopel %2Fpagcopel2.nsf%2Fdocs%2F01C009432D735E57032573FA00687CC4>. Acesso em: 09 fev. 2013.

CSV. **GDAL - Geospatial Data Abstraction Library** . Disponível em: <http://www.gdal.org/ogr/drv_csv.html>. Acesso em: 12 fev. 2013.

DEITEL, Paul. Uma abordagem baseada em aplicativos. In: **Android para Programadores**. Porto Alegre: bookman, 2013. cap.1, p.4.

ECLIPSE, **O IDE Eclipse - Revista easy Java Magazine 19 - Parte1**. Disponível em: <http://www.devmedia.com.br/o-ide-eclipse-revista-easy-java-magazine-19-parte1/24796>. Aceso em 12 de fev. 2013.

ELEMENTOS. **Elements - OpenStreetMaps Wiki**. Disponível em: <http://wiki.openstreetmap.org/wiki/Data_Primitives>. Acesso em: 12 fev. 2013.

GALAXY. **Tablet Galaxy: conheça o produto**. Disponível em: <http://www.assimsefaz .com.br/sabercomo/tablet-galaxy-conheca-o-produto>. Acesso em: 09 fev. 2013.

GPS. **O que é um navegador GPS automotivo?**. Disponível em: <http://www. overcar.com.br/o-que-e-um-navegador-gps-automotivo/>. Acesso em: 10 fev. 2013.

GPSMAPEDIT1, versão 2.0.76.7 : **Visualizador e editor de mapas vetoriais**, [S.l.]: geopanting, 2002-2013. Download <http://www.geopanting.com>.

GPSMAPEDIT2. **Manual básico de roteamento usando o GPSMapedit**. Disponível em: <http://tracksourcebrasil.wikispaces.com/ManualBasicoMapedit>. Acesso em: 12 fev. 2013.

INTRANET1. **COPEL investe em tecnologia para o trabalho dos eletricistas** . Disponível em: <http://webprd/intra/copelonline/not453>. Acesso em: 05 mar. 2013.

INTRANET2. **GPS COPEL facilita a localização de consumidores** . Disponível em: <http://webprd/intra/copelonline/not811>. Acesso em: 05 mar. 2013.

INTRANET3. **Regional Oeste começa a receber os primeiros tablets** . Disponível em: <http://webprd/intra/copelonline/not989>. Acesso em: 05 mar. 2013.

JAVA. **Tutorial Java : O que é Java?**. Disponível em: <http://javafree.uol.com.br /artigo/871498/Tutorial-Java-O-que-e-Java.html>. Acesso em: 11 fev. 2013.

JORNAL. Diário do Sudoeste - **Pato-branquense desenvolve aplicativo para GPS**. Disponível em: <http://www.utfpr.edu.br/patobranco/estrutura-universitaria/assessorias/ascom/noticias/clipping/2012/novembro-2012/07-11-2012/07.11-materia-pato-branquense-cria-aplicativo-para-gps/at_download/file>. Acesso em: 05 mar. 2013.

JORNAL2. Diário do Sudoeste - **Pato-branquense desenvolve aplicativo para GPS**. Disponível em: <http://www.utfpr.edu.br/patobranco/estrutura-universitaria/assessorias/ascom/noticias/clipping/2012/novembro-2012/07-11-2012/07.11-materia-pato-branquense-cria-aplicativo-para-gps-ii/at_download/file>. Acesso em: 05 mar. 2013.

JOSM. **JOSM - OpenStreetMaps Wiki.** Disponível em: <http://wiki.openstreetmap.org/wiki/JOSM>. Acesso em: 12 fev. 2013.

MP, Stanislaw Kozicki. **cGPSmapper User Manual.** Disponível em http://www.cgpsmapper.com/download/cGPSmapper-UsrMan-v02.5.pdf. Acesso em 12 fev. 2012.

NOTEPAD. **Notepad++**. Disponível em: <http://www.baixaki.com.br/download/notepad-.htm>. Acesso em: 12 fev. 2013.

OPENSTREETMAP. **About - OpenStreetMaps Wiki**. Disponível em: <http://wiki.openstreetmap.org/wiki/Pt-br:About>. Acesso em: 11 fev. 2013.

OSM. **OSM XML – OpenStreetMaps Wiki**. Disponível em: <http://wiki.openstreetmap.org/wiki/OSM_XML>. Acesso em: 12 fev. 2013.

POSTGRESQL, **O que é o PostgreSQL?.** Disponível em: <http://www.forumweb. com.br/artigo/88/postgresql/o-que-e-o-postgresql>. Acesso em: 12 fev. 2013.

PRESSMAN, R. **Engenharia de software.** Rio de Janeiro: McGraw-Hill, 2005.

SQLITE. **SQLite, Muito Prazer!**. Disponível em <http://www.devmedia.com.br/sqlite-muito-prazer/7100>. Acesso em: 12 fev. 2013.

TABLETS. **Tablet - Que bicho é esse?**. Disponível em: <http://tecnologia.ig.com.br/noticia/2010/01/14/tablet+que+bicho+e+esse+9295069.html>. Acesso em: 09 fev. 2013.

WAZLAWICK, Raul Sidnei. **Engenharia de Software**. São Paulo : Elsevier, 2011, 2ª edição.