UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUIZ HENRIQUE KULTZAK

MANIPULAÇÃO DE DADOS GEOGRÁFICOS UTILIZANDO A BIBLIOTECA GEOTOOLS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2017

LUIZ HENRIQUE KULTZAK

MANIPULAÇÃO DE DADOS GEOGRÁFICOS UTILIZANDO A BIBLIOTECA GEOTOOLS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientadora: Prof.ª Dr.ª Simone de Almeida

PONTA GROSSA

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

MANIPULAÇÃO DE DADOS GEOGRÁFICOS UTILIZANDO A BIBLIOTECA GEOTOOLS

por

LUIZ HENRIQUE KULTZAK

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 07 de novembro de 2017 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof(a).Drª Simone de Almeida
Orientadora

_____
Prof. Esp. Marcos Vinicius Fidelis
Membro titular

_____
Prof. Dr. Richard Duarte Ribeiro
Membro titular

_____
Prof(a). Drª Helyane Bronoski Borges
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. MSc Saulo Jorge Beltrão de Queiroz
Coordenador do curso

- A Folha de Aprovação assinada encontra-se arquivada na Secretaria Acadêmica -

Dedico este trabalho à minha família, que sempre me apoiou e incentivou na aquisição e evolução dos meus conhecimentos acadêmicos.

AGRADECIMENTOS

Agradeço à minha orientadora Prof.ª Dr.ª Simone de Almeida, pela sabedoria com que me guiou nesta trajetória.

A Secretaria do Curso e a todos os funcionários da UTFPR – Campus Ponta Grossa pela cooperação e por promoverem um ambiente propício ao enriquecimento intelectual de todos os alunos que aqui estudam.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização deste trabalho.

A reação mais comum da mente humana a uma conquista não é satisfação, e sim o anseio por mais. (HARARI, Yuval, 2015)

RESUMO

KULTZAK, Luiz Henrique. **Manipulação de Dados Geográficos Utilizando a Biblioteca Geotools**. 2017. 72 f. Trabalho de Conclusão de Curso em Bacharelado em Ciência da Computação - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

Bancos de Dados Geográficos são estruturas de armazenamento que suportam a representação de dados espaciais em tabelas. O trabalho apresenta um estudo sobre as funcionalidades obtidas a partir da utilização de uma biblioteca de funções geográficas, o GeoTools, assim como o gerenciador de banco de dados geográfico, o PostGIS. O referencial teórico sintetiza as diferenças entre um banco de dados convencional e um banco de dados geográfico, e como funcionam as estruturas de armazenamento e manipulação de dados geográficos. O resultado obtido foi a implementação de um Sistema de Informação Geográfico capaz de prover a aquisição, manipulação e visualização de informações presentes em uma base de dados, assim como realizar a inserção de novos dados para seu armazenamento.

Palavras-chave: GeoTools. PostGIS. Banco de dados Geográfico. Sistema de Informação Geográfica.

ABSTRACT

KULTZAK, Luiz Henrique. **Geographic Data Manipulation Using the Geotools Library**. 2017. 72 p. Work of Conclusion Course Graduation in Computer Science - Federal Technology University - Paraná. Ponta Grossa, 2017.

Geographic Databases are storage structures that support the representation of geometric features in tables. This work aims to present a study about the functionalities obtained from the use of a library of geographic functions, GeoTools, together with a geographic database, the PostGIS. Theoretical reference synthesizes the differences between a conventional database and a geographic database, and how geographic data storage and manipulation structures work. The result obtained was the implementation of a Geographic Information System capable of providing the acquisition, manipulation and visualization of information present in a database, as well as performing the insertion of new data for storage.

Keywords: GeoTools. PostGIS. Spatial Database. Geographic Information System.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

LISTA DE SIGLAS

BDG	Banco de Dados Geográfico
IBGE	Instituto Brasileiro de Geografia e Estatística
OMT-G	*Object Modeling Technique for Geographic Applications*
SGBD	Sistema Gerenciador de Banco de Dados
UML	*Universal Modeling Language*
IDE	*Integrated Development Environment*
SLD	*Styled Layer Descriptor*

LISTA DE ACRÔNIMOS

CAD	*Computer Aided Design*
SIG	Sistema de Informação Geográfica
JAR	*Java Archive*

SUMÁRIO

1 INTRODUÇÃO

Há algumas décadas atrás, mapas em papel eram a única forma de se representar informações geográficas. Manipular essas informações demandava esforço manual em um processo não interativo. Desde então, o rápido avanço de novas tecnologias computacionais para a coleta e digitalização de dados geográficos, acompanhado de uma crescente demanda por aplicações que realizem a manipulação e a análise destes, gerou a necessidade da criação de sistemas dedicados a estas tarefas, são os chamados Sistemas de Informação Geográficas (SIGs) (RIGAUX; SCHOLL; VOISARD, 2002).

Os dados, que são analisados por um SIG, devem primeiramente estar armazenados em uma base de dados. É reconhecido que Sistemas Gerenciadores de Banco de Dados (SGBDs) convencionais não são adequados para a manipulação de dados complexos não-tradicionais, tais como dados presentes em *computer-aided design* (CAD) e em SIGs (WORBOYS, 2003).

Os métodos de acesso à informação, a linguagem de consulta e os modelos de dados presentes nos SGBDs convencionais são projetados para tratar dados estruturados. Entretanto, em um Sistema Gerenciador de Banco de Dados Geográfico (SGBDG), tem-se a necessidade do armazenamento e manipulação de dados como pontos, linhas, retângulos e polígonos em um espaço multidimensional. Portanto, um SGBDG deve conter uma linguagem de consulta espacial e métodos de acesso dinâmicos para possibilitar a rápida recuperação de itens que correspondam as localizações espaciais pesquisadas (HWANG; BYUN; MOON, 1994).

Para se desenvolver um sistema de informação geográfico confiável, é necessário a criação de um modelo conceitual que represente os objetos do mundo real que possuam alguma função no sistema de informação, e num processo de abstração, obter um conjunto de características para cada objeto que será representado no sistema. Para tal, é de suma importância que as representações dos objetos sejam capazes de incorporar suas características geométricas.

O modelo *Object Modelling Technique for Geographic Applications* (OMT-G) parte das primitivas de representação definidas para o diagrama de classes da *Unified Modelling Language* (*UML*), acrescentando a ele primitivas geográficas para aumentar a capacidade semântica do modelo representado (DAVIS JR.; LAENDER, 2000).

Dentro deste contexto, o trabalho proposto consiste na modelagem de um problema do mundo real, que contenha dados geográficos, utilizando a linguagem de modelagem de dados OMT-G para sua representação, bem como a implementação do modelo gerado, em um sistema gerenciador de banco de dados geográfico PostGIS[1], possuindo como foco principal a utilização da biblioteca GeoTools[2] para realizar análises sobre os dados armazenados no Banco de Dados Geográfico (BDG).

PostGIS é uma extensão para o banco de dados relacional PostgreSQL. Ela oferece suporte para a representação de objetos geográficos e adiciona também a possibilidade da realização de consultas de localização a serem executadas em SQL (*Structure Query Language*) (OBE, 2011).

Além das ferramentas de modelagem e do SGBDG, essa pesquisa utilizará a ferramenta GeoTools, que é uma biblioteca de classes escritas na linguagem de programação Java disponível para comunidade com licença de software livre para serem utilizadas na construção de SIGs. As funções disponibilizadas pelo GeoTools compreendem estruturas de dados espaço-temporais, onde se procura armazenar informações geográfica de fenômenos que variam tanto no espaço como também no tempo, além de algoritmos para a decodificação de dados e análise espacial. As funções citadas atuam sobre dados armazenados em um SGBDG.

1.1 OBJETIVOS

Seguem definidos nas seções seguintes o objetivo geral e os objetivos específicos do presente trabalho.

1.1.1 Objetivo Geral

Integrar um modelo de banco de dados geográfico com a biblioteca GeoTools, utilizando seus principais recursos de manipulação e análise de dados espaciais.

[1] http://postgis.net/
[2] http://geotools.org/

1.1.2 Objetivos Específicos

- Modelar uma base de dados baseada em um problema do mundo real, contendo diferentes tipos de dados georreferenciados;
- Implementar o modelo analisado no banco geográfico PostGIS;
- Implementar a conectividade entre o banco de dados contendo a base modelada e a biblioteca GeoTools;
- Utilizar as funções disponíveis na biblioteca GeoTools no banco geográfico PostGIS.

1.2 JUSTIFICATIVA

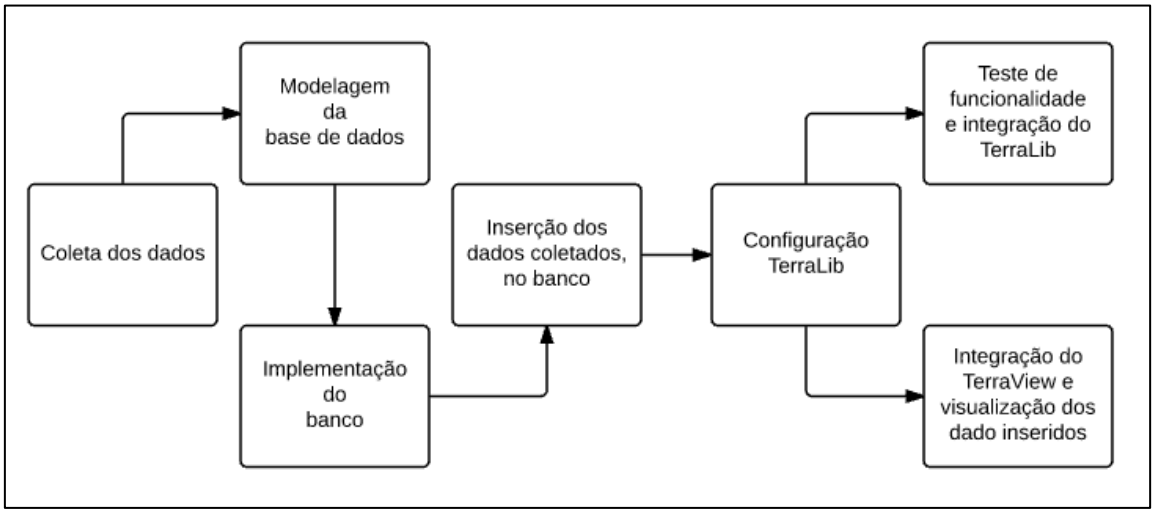
Entre os sistemas de gerenciamento de dados especializados, os de dados espaciais tem se destacado como um componente central para muitas aplicações (WORBOYS, 2003). São exemplos de ferramentas: SIGs, CADs, processamento de imagens, entre outros. Esses sistemas possuem objetos espaciais que devem ser mantidos, consultados e exibidos. Facilitar a integração entre o armazenamento de dados e prover a análise e visualização dos dados se torna uma atividade necessária. A biblioteca GeoTools pode ser utilizada para realizar essa integração (GEOTOOLS, 2017).

A fim de se aprofundar na utilização prática da ferramenta GeoTools, torna-se necessária a descrição das funcionalidades disponíveis através da implementação da biblioteca, elencando assim possíveis vantagens decorrentes da integração de um SGBD espacial com a GeoTools.

1.3 METODOLOGIA

O desenvolvimento do trabalho se deu, por meio de consultas bibliográficas, realizando o experimento prático, conforme representado na Figura 1 e discriminado a seguir:

Figura 1 - Metodologia proposta para desenvolvimento do trabalho



Fonte: Autoria própria (2017)

- Foi concebido um cenário de um problema real, do qual foi feita a modelagem conceitual dos dados referentes a este problema utilizando uma extensão do *software* de modelagem StarUML denominado OMT-G (DAVIS JR.; LAENDER, 2000). Este cenário possibilitou a implementação dos diferentes tipos de dados espaciais, como o ponto, linha e polígono;
- Após a modelagem foi realizada a implementação física da base de dados no PostGIS;
- Dados devem ser coletados e inseridos no banco para efetiva manipulação destes. A coleta dos dados ocorreu na base disponível no *site* Geoportal Ponta Grossa[3];
- Instalação e configuração da biblioteca GeoTools;
- Teste das funcionalidades adquiridas a partir da integração do SGBDG com o GeoTools;
- Instalação e integração do aplicativo TerraView para visualização dos dados inseridos.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em 6 capítulos, sendo o primeiro dedicado à introdução. No segundo capítulo são levantados os principais conceitos de uma

[3] http://geo.pg.pr.gov.br/portal/

aplicação de banco de dados, como as principais diferenças entre bancos convencionais e bancos geográficos, bem como os diferentes tipos de dados que podem ser representados por um SGBDG.

No capítulo 3 é realizada a introdução aos conceitos utilizados por um SIG para manipulação de dados geográficos. Assim como um levantamento e descrição das ferramentas utilizadas no decorrer do projeto, sendo apresentada a base de dados que serviu como base experimental para os conceitos estudados.

O capítulo 4 foi destinado ao desenvolvimento do projeto, sendo apresentados os passos necessários para a utilização das ferramentas GeoTools e PostGIS em um ambiente Windows.

No capítulo 5 é realizada a descrição das funcionalidades obtidas através da utilização da ferramenta GeoTools, apresentando exemplos de resultados obtidos, assim como detalhes da implementação destas funcionalidades.

No capítulo 6 encontram-se as considerações finais do trabalho, levantando as conclusões acerca do projeto e apresentando opções de complemento ao trabalho realizado.

2 BANCO DE DADOS

Este capítulo está dividido em 6 seções. Na seção 2.1 são apresentados os conceitos básicos em relação a um BDG. A seção 2.2 discorre sobre os tipos de arquiteturas existentes na integração entre um BDG e um SIG. Na seção 2.3 são descritos os tipos de dados que são tratados por um BDG.

A seção 2.4 traz uma comparação entre os dois tipos de representação geográfica, dados do tipo *vector* ou do tipo *raster.* Na seção 2.5 encontra-se os conceitos básicos da modelagem de dados OMT-G. Na seção 2.6 são apresentados os principais aspectos técnicos da biblioteca PostGIS (OBE, 2011). A última seção destina-se às considerações do capítulo.

2.1 CONCEITOS

Um banco de dados é uma coleção unificada de dados interconectados que se encontram armazenados em dispositivos não-voláteis para serem recuperados e utilizados em possíveis compartilhamento futuros. Os usuários detêm a capacidade de definição, acesso, recuperação, manipulação e apresentação dos dados mantidos. São inúmeros os tipos de dados que podem ser armazenados em um banco de dados, tais como: números, cadeias de caracteres, textos, imagens, sons e configurações espaciais (WORBOYS, 2003).

Um banco de dados possui a característica de ser logicamente centralizado, porém, podendo ser fisicamente distribuído por diversas localidades espaciais, caracterizando um banco de dados distribuído (WORBOYS, 2003).

Ainda segundo Worboys (2003), existem algumas peculiaridades na configuração de um banco de dados que podem levar a diferentes categorizações:

- Banco de dados para um único usuário: geralmente opera em uma única máquina. Por não precisar se preocupar com acesso concorrente, o SGBD é bastante simples e barato;
- Banco de dados distribuído: logicamente centralizado, porém, fisicamente distribuído por diversas localidades espaciais;

- Banco de dados corporativo: os elementos essenciais presentes nesta categoria são segurança e integridade dos dados, desempenho e proteção para acesso concorrente.
- Banco de dados para engenharia: armazena principalmente dados gráficos; gerenciamento de múltiplas versões de projetos são geralmente necessárias; transações geralmente complexas e demoradas;
- Banco de dados científico: contém estruturas de dados complexas (Exemplo: projeto genoma humano); análises complexas são executadas nos dados armazenados;
- Banco de dados geográfico: armazenando uma combinação de dados espaciais e não espaciais; tal categoria de banco de dados contém uma estrutura complexa de dados e de análise.

Outra importante definição que deve ser introduzida é a de um SIG, sendo um sistema de informação computacional que realiza o armazenamento de dados geográficos, bem como a recuperação desses, podendo prover a combinação dos mesmos para criar novas representações geográficas. Outra característica relevante que deve ser encontrada em um SIG é a de poder realizar simulações para auxiliar usuários a organizar trabalhos em outras áreas, incluindo administração pública, rede de transportes, aplicações militares e sistemas de informações ambientais (RIGAUX; SCHOLL; VOISARD, 2002).

Levando em consideração as definições de BDG e SIG apresentadas, a próxima seção introduz os tipos de arquiteturas que podem ser implementadas a partir da integração entre tais tecnologias.

2.2 TIPOS DE ARQUITETURAS

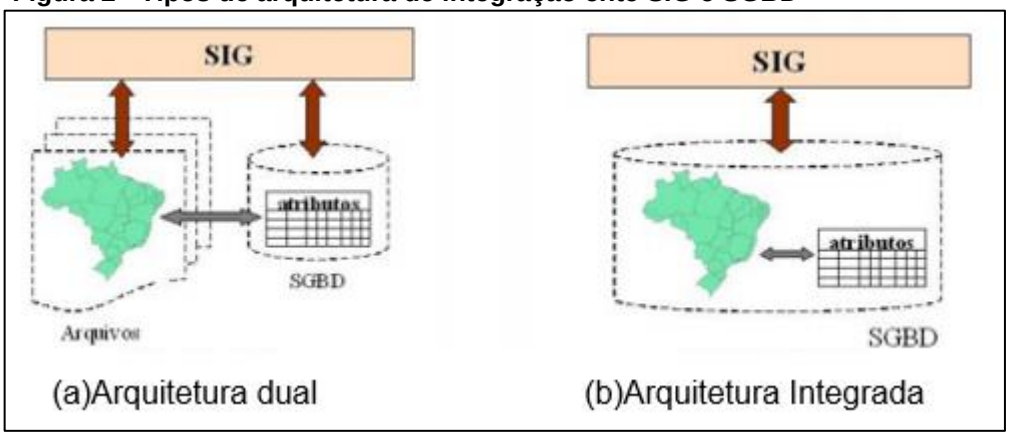
De acordo com Casanova et al. (2005) existem duas maneiras de se integrar um SIG e um SGBDG:

a) Arquitetura dual: componentes convencionais são armazenados em um SGBDG relacional e componentes de caráter espacial são armazenados em arquivos separados do SGBDG, dificultando o controle e a manutenção dos

componentes espaciais. Além disso, a integridade da relação entre os dados convencionais e os dados espaciais também são um problema (Figura 2a).

b) Arquitetura integrada: todos os dados são armazenados em um SGBDG que detém recursos para armazenamento e manipulação de dados espaciais (Figura 2b).

Figura 2 - Tipos de arquitetura de integração ente SIG e SGBD



Fonte: Adaptado de Casanova et al. (2005)

Para Casanova et al. (2005) os serviços básicos que devem ser providos em um SIG integrado a um BDG são:

- Inclusão e verificação de dados;
- Armazenamento e administração dos dados;
- Apresentação dos dados armazenados;
- Transformação de dados;
- Interação com usuários finais.

O conceito de dado geográfico apresentado nesta seção colabora para a compreensão dos tipos de dados que são utilizados para representar um objeto geográfico e suas classificações.

2.3 TIPOS DE DADOS

Os atributos que representam as características de um geo-objeto não correspondem a nenhum tipo de dado que pode ser armazenado em um banco de dados convencional, tais como *integer, date, float* ou *string.* A representação de um objeto geográfico deve obedecer a uma representação fiel de uma abstração do

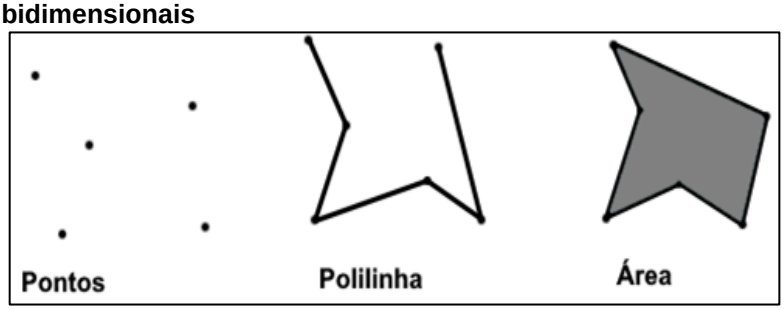
ambiente, levando a necessidade de utilização de um novo modelo de dados (CASANOVA et al., 2005).

Os tipos de dados que são utilizados para representação de um modelo de dados geográfico, segundo Rigaux, Scholl e Voisard (2002), são:

- Objetos zero-dimensionais ou pontos: pontos são utilizados para representar a localização de objetos cuja forma é considerada de pouca relevância para o contexto da aplicação, ou sua área é demasiadamente pequena em relação ao tamanho total da representação. Cidades, igrejas e cruzamentos, são exemplos de entidades que podem ter seu tamanho reduzido à representação de um ponto no espaço.
- Objetos unidimensionais ou objetos lineares: tais objetos são geralmente utilizados para representação de redes (Exemplos: estradas, bacias hidrográficas, malha ferroviária, etc.). A forma mais utilizada para representação de um objeto linear é a polilinha. Uma polilinha é definida como um conjunto finito de segmentos de reta, onde cada segmento tem suas pontas compartilhadas por dois outros segmentos de reta, exceto pelos pontos extremos, que possuem compartilhamento de ponta apenas com um único segmento de reta.
- Objetos bidimensionais ou objetos de superfície: são objetos comumente utilizados para identificar entidades que representam áreas grandes no contexto do espaço geográfico armazenado. Pode-se citar como um exemplo as unidades administrativas de uma cidade, onde cada unidade representa uma larga parcela da área da cidade. Um polígono é uma região do plano que está delimitada por uma polilinha chamada fronteira.

A Figura 3, apresenta graficamente os conceitos explanados no texto.

Figura 3 - Representação gráfica de objetos zerodimensionais, unidimensionais e bidimensionais



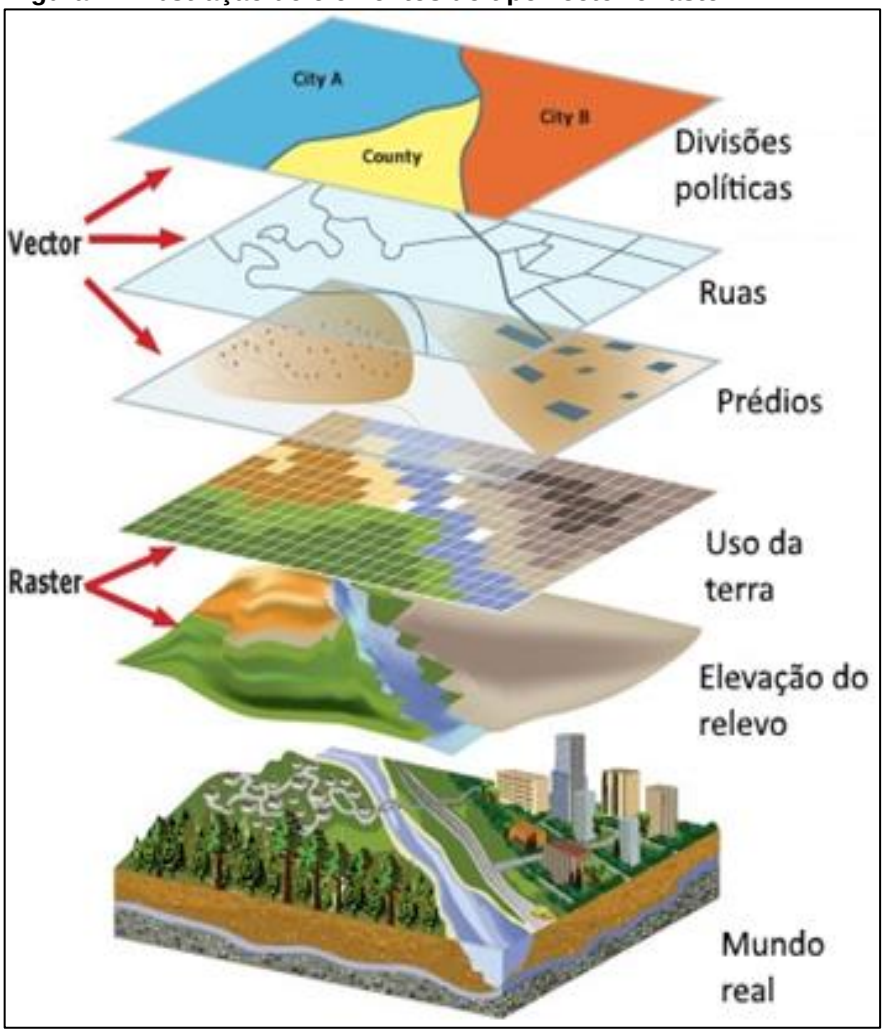
Fonte: Autoria própria (2017)

2.4 RASTER X VECTOR

Além da representação vetorial descrita na seção 2.3, existe uma outra forma para representação de dados geográficos em um plano computacional, chamada *raster*. Como descrevem Rigaux, Scholl e Voisard (2002) neste modelo, que também pode ser chamado de modelo matricial, o terreno é representado por uma matriz M (i, j), composta por i colunas e j linhas, que definem células, denominadas *pixels*. Cada *pixel* apresenta um ou mais valores referentes aos atributos do sistema. Um arquivo georreferenciado está associado a um par de coordenadas x e y que se encontra dentro da área abrangida pelos *pixels*.

A Figura 4 traz uma representação dos conceitos apresentados, onde é possível visualizar as diferenças entre dados do tipo *raster* e do tipo *vector,* tendo como base informações provenientes de um ambiente do mundo real.

Figura 4 - Abstração de elementos do tipo *vector* e raster



Fonte: Adaptado de Shaletrak (2016)

2.5 MODELAGEM OMT-G

O modelo OMT-G adiciona primitivas geográficas ao diagrama de classes definido na UML, possibilitando um aumento da capacidade de representação semântica do modelo desejado (DAVIS JR.; LAENDER, 2000).

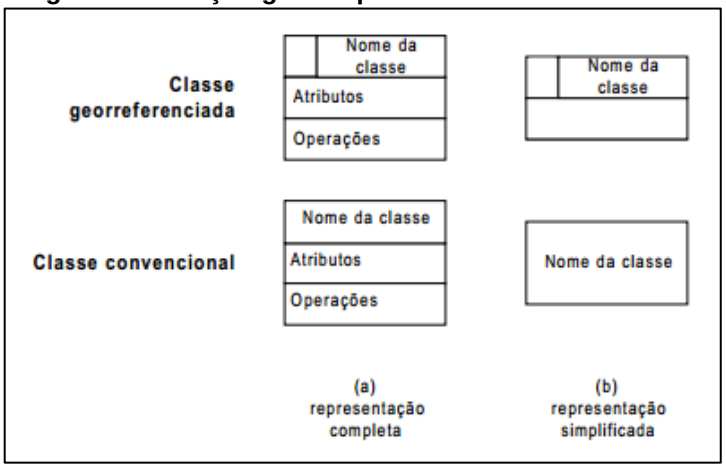
Segundo Davis Jr. e Laender (2000), por ser um modelo adaptado da UML convencional, o padrão OMT-G mantém valores de sua representação tradicional, podendo assim representar classes convencionais e também classes georreferenciadas. Uma classe convencional representa um objeto que não contém propriedades geométricas, possuindo relação com objetos espaciais sendo representada exatamente como na UML convencional. Uma classe georreferenciada é utilizada para representar um objeto que possui representação espacial e está associada a uma região terrestre.

Ainda segundo Davis Jr. e Laender (2000) a classificação de um objeto geográfico se dá de duas maneiras:

- Geo-campo: correspondendo a objetos e fenômenos que são continuamente distribuídos no espaço representado computacionalmente. Pode-se citar como exemplo de geo-campo o tipo de solo, vegetação ou relevo. Descendem de uma classe geo-campo do modelo OMT-G cinco classes: isolinhas, subdivisão planar, tesselação, amostragem e malha triangular.
- Geo-objeto: representam objetos geográficos particulares, individualizáveis e que estão diretamente associados a elementos do mundo real, tal como a reprodução computacional de um edifício, rio ou de uma rodovia. Partindo da definição de um geo-objeto é possível derivar outras duas classes, são elas: geo-objeto com geometria e geo-objeto com geometria e topologia.

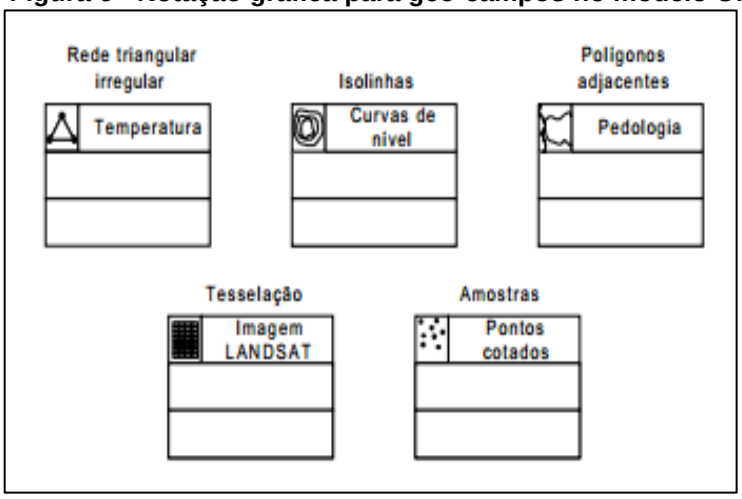
As Figuras 5, 6 e 7 apresentam a notação gráfica para a representação das classes geográficas e não-geográficas (convencionais) em diagramas no modelo OMT-G. Também são apresentadas as diferentes classes que podem ser utilizadas em um diagrama OMT-G.

Figura 5 - Notação gráfica para classes do modelo OMT-G



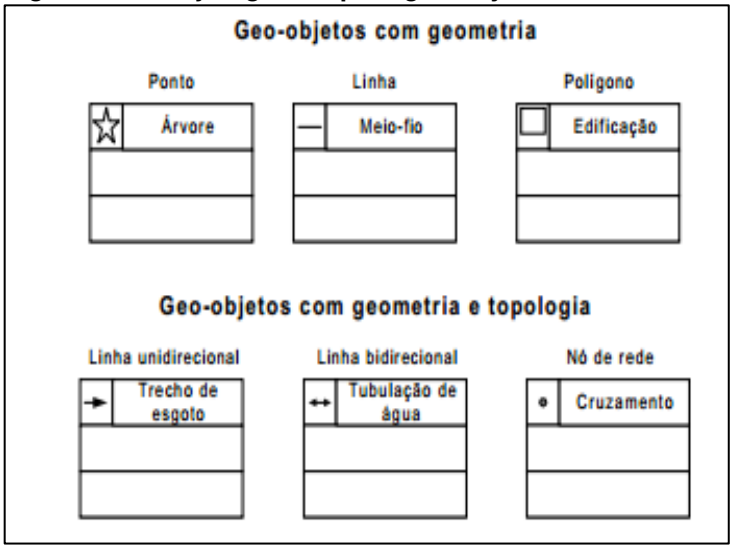
Fonte: Davis Jr. e Laender (2000)

Figura 6 - Notação gráfica para geo-campos no modelo OMT-G



Fonte: Davis Jr. e Laender (2000)

Figura 7 - Notação gráfica para geo-objetos no modelo OMT-G



Fonte: Davis Jr. e Laender (2000)

2.6 POSTGIS

A ferramenta PostGIS (OBE, 2011) é uma extensão para o banco de dados *open source* PostgreSQL[4], e começou a ser desenvolvida no início dos anos 2000 pela empresa canadense *Refraction Research Inc.*[5] O PostGIS adiciona suporte à representação de objetos geográficos e permite a construção de consultas de localização a serem executadas na linguagem SQL (OBE, 2011). A versão mais recente do *software* é a 2.2, e trabalha em modo de compatibilidade com a versão 9.1 do PostgreSQL e superiores (POSTGIS, 2017).

As representações geográficas presentes na ferramenta são baseadas e moldadas por meio dos padrões impostos pela OGC (*Open Geospatial Consortium*[6]). As formas de representação de dados geométricos são: WKB (*Well Known Binary*) e WKT (*Well Known Text*). Ambas as formas de representação incluem informações sobre o tipo do objeto e as coordenadas que o formam (OBE, 2011).

De acordo com Obe e Hsu (2011), a ferramenta PostGIS estende o SGBD PostgreSQL introduzindo um tipo de dado chamado “geometria”. PostGIS utiliza uma tabela chamada *geometry_columns* para armazenar os metadados associados as colunas geométricas no banco de dados. Tal tabela é formada por sete colunas, das quais quatro são utilizadas para armazenar o nome do banco de dados (*f_table_catalog*), o nome do esquema (*f_table_schema*), o nome da tabela (*f_table_name*) e o nome da coluna geométrica (*f_geometry_column*).

Segundo Obe e Hsu (2011) as outras 3 colunas que estão presentes na tabela *geometry_columns* são:

- *coord_dimension*: esta coluna armazena a dimensão da coluna geométrica, os valores aceitos são 2, 3 e 4. Podendo assim serem armazenados dados em 4 dimensões. Onde a quarta dimensão pode ser utilizada para a representação de tempo ou um *index* para qualquer outra informação relevante;

[4] Sítio para obtenção de informações sobre o Projeto PostGreSQL <https://www.postgresql.org/>
[5] Sítio contendo informações sobre o projeto PostGIS <http://refractions.net/products/postgis/history/>
[6] Maiores informações sobre os padrões podem ser encontradas no sítio <http://www.opengeospatial.org/>

- *srid (spatial reference identifier)*: traduzido para identificador de referência espacial, armazena um inteiro que representa a chave primária da metatabela *spatial_ref_sys. O* PostGIS utiliza esta tabela para todos os sistemas de referência espacial disponíveis no banco de dados, contendo o nome do sistema de referência espacial, os parâmetros necessários para reprojetar para outro sistema e sob qual autoridade o sistema foi definido;
- *type*: esta coluna armazena o tipo geométrico como uma cadeia de caracteres. A descrição de cada valor possível neste campo será dada na subseção 2.6.1.

2.6.1 Geometrias no PostGIS

PostGIS possui uma grande variedade de tipos geométricos para proporcionar a modelagem do mundo real. Todas as geometrias armazenadas no PostGIS são baseadas no sistema de coordenadas cartesianas.

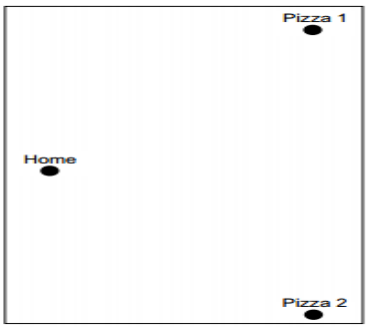
Pontos: um ponto em um espaço 2D é especificado por suas coordenadas X e Y. No espaço 3D um ponto deve conter as coordenadas X, Y e Z. Um exemplo de execução:

Quadro 1 - Exemplo de código para inserção do campo geométrico ponto seguido pela inserção de 3 pontos distintos

```
SELECT AddGeometryColumn('public','my_geometries'), 'my_points', -1, 'POINT',2);

INSERT INTO my_geometries(name,my_points) VALUES('Home',ST_geomFromText('POINT(0 0)'));

INSERT INTO my_geometries(name,my_points) VALUES('Pizza 1',ST_geomFromText('POINT(1 1)'));

INSERT INTO my_geometries(name,my_points) VALUES('Pizza 2',ST_geomFromText('POINT(1 -1)'));
```

Fonte: Adaptado de Obe e Hsu (2011)

Figura 8 - Representação gráfica dos dados inseridos no Quadro 1



Fonte: Obe e Hsu (2011)

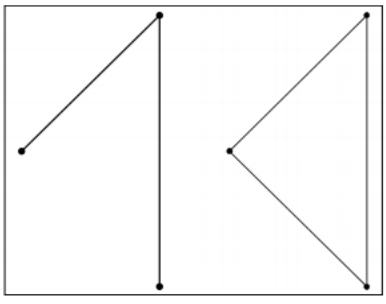
Polilinha: são definidas por pelo menos dois pontos distintos, cada coordenada adicional representa mais um vértice da polilinha.

Quadro 2 - Exemplo de código para inserção do campo polilinha seguido pela inserção de 2 linhas distintas

```
SELECT AddGeometryColumn('public','my_geometries'), 'my_linestrings', -1, 'LINESTRINGS',2);

INSERT INTO my_geometries(name,my_linestrings) VALUES('Linestring Open',ST_geomFromText('LINESTRING(0 0,1 1, 1 -1)'));

INSERT INTO my_geometries(name,my_linestrings) VALUES('Linestring Closed',ST_geomFromText('LINESTRING(0 0, 1 1, 1 -1, 0 0)'));
```

Fonte: Adaptado de Obe e Hsu (2011)

Figura 9 - Representação gráfica dos elementos inseridos no Quadro 2



Fonte: Obe e Hsu (2011)

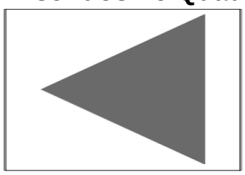
Polígono: definido por uma polilinha fechada, ou seja, o primeiro elemento é também o último.

Quadro 3 - Código executado para inserção do campo polígono seguido pela inserção das informações de um polígono

```
SELECT AddGeometryColumn('public','my_geometries'), 'my_polygons', -1, 'POLYGON',2);

INSERT INTO my_geometries(name,my_polygons) VALUES('Triangle',ST_geomFromText('POLYGON(0 0,1 1, 1 -1,0 0)'));
```

Fonte: Adaptado de Obe e Hsu (2011)

Figura 10 -Representação gráfica dos elementos inseridos no Quadro 3



Fonte: Obe e Hsu (2011)

Os outros tipos de dados geométricos representam um conjunto dos elementos citados anteriormente.

- *Multipoint*: representando um conjunto de pontos executados num mesmo comando - MULTIPOINT (-1 1, 0 0, 2 3)
- *Multilinestring*: representa uma coleção de polilinhas - MULTILINESTRING ((0 0,0 1,1 1), (-1 1, -1 -1))
- *Multipolygon*: representa uma coleção de polígonos - MULTIPOLYGON (((2.25 0, 1.25 1, 1.25 -1, 2.25 0)), ((1 -1,1 1,0 0,1 -1)))
- *GeometryCollection*: é um tipo de dado presente no PostGIS que pode armazenar tipos de geometrias de forma heterogênea, diferente dos tipos citados anteriormente, nos quais só podem ser armazenados geometrias do mesmo tipo - GEOMETRYCOLLECTION (MULTIPOINT (-1 1,0 0,2 3), MULTILINESTRING ((0 0,0 1,1 1), (-1 1, -1 -1)), POLYGON ((-0.25 -1.25, -0.25 1.25, 2.5 1.25, 2.5 -1.25, -0.25 -1.25), (2.25 0, 1.25 1, 1.25 -1, 2.25 0), (1 -1,1 1,0 0,1 -1)))

2.7 CONSIDERAÇÕES DO CAPÍTULO

Neste capítulo foram apresentados os elementos básicos que compõem a estrutura de um SIG, definindo-se as características elementares comuns aos SIGs

modernos. Como os tipos de dados e arquiteturas suportados por um SIG. A diferença entre dados do tipo Raster e Vector. A modelagem de tabelas no modelo OMT-G e finalizando com a relação dessas informações no contexto do BDG PostGIS.

3 SISTEMA DE INFORMAÇÃO GEOGRÁFICA

Este capítulo objetiva apresentar as definições de um SIG e explanar acerca da utilização da ferramenta GeoTools. Está dividido em 6 seções. A seção 3.1 apresenta os conceitos básicos de um SIG. A seção 3.2 discorre sobre a ferramenta TerraLib. Na seção 3.3 é apresentada a ferramenta GeoTools. Na seção 3.4 é apresentada a ferramenta TerraView. A seção 3.5 discorre sobre a base de dados utilizada como experimento deste trabalho. Como conclusão do capítulo, a seção 3.6 levanta as dificuldades encontradas com a utilização da ferramenta TerraLib e apresenta uma justificativa para a substituição desta pela biblioteca GeoTools.

3.1 CONCEITOS BÁSICOS

Sistema de Informação Geográfica (SIG) é o termo utilizado para designar sistemas que tratam computacionalmente de dados geográficos (WORBOYS, 2003). Comparando um SIG com um sistema de informação convencional, além de atributos descritivos, no SIG é possível armazenar também as geometrias dos diferentes tipos de dados. Por exemplo: no cadastro de um terreno, além das características como valor do IPTU e nome do proprietário, um SIG armazena também as coordenadas dos limites do terreno (CASANOVA et al., 2005).

Para Rigaux, Scholl e Voisard (2002), as principais funções que devem estar incorporadas a um sistema SIG são:

- Inserção e verificação de dados;
- Armazenamento e gerência dos dados;
- Saída e apresentação dos dados armazenados;
- Transformação de dados;
- Interação com usuários finais.

De acordo com Casanova et al. (2005) as ferramentas que devem estar presentes na formação de um SIG são:

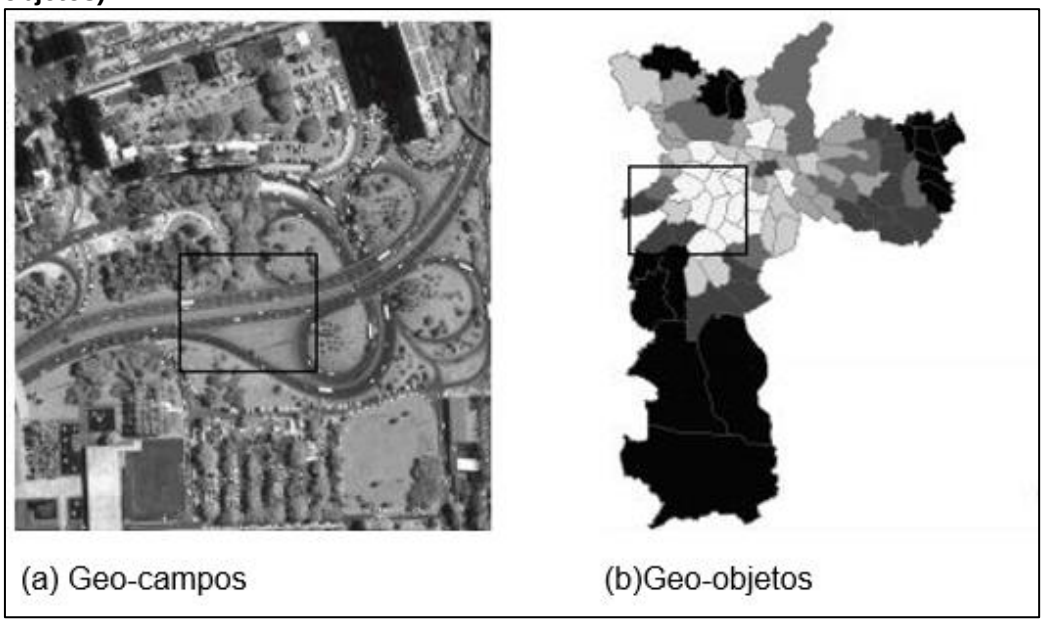
- Interface homem-máquina: define como o sistema será operado;
- Entrada e integração de dados: mecanismos para conversão de dados;

- Consulta e análise espacial: operações topológicas, álgebra de mapas, estatística espacial, modelagem numérica de terreno e processamento de imagens;
- Gerência de dados espaciais: armazenamento e recuperação dos dados;
- Visualização e plotagem: mecanismos adequados para a representação relevante dos dados pesquisados.

Ainda segundo Casanova et al. (2005), os conceitos de espaço absoluto e espaço relativo também são importantes para entender a forma de coleta e armazenamento de dados em um SIG. Esses conceitos são detalhados na sequência:

- Espaço absoluto: representação dos objetos no espaço absoluto, onde as coordenadas das fronteiras dos objetos armazenados devem corresponder as coordenadas estabelecidas. Os modelos no espaço absoluto, são:
  - Geo-campos: superfície representada é contínua e os fenômenos observados variam ao longo do espaço. Exemplo: imagens capturadas por satélite (Figura 11 a);
  - Geo-objetos: representação de objetos distintos e identificáveis, cada objeto representado contém fronteiras fechadas. Exemplo: distritos de uma cidade separados por fronteiras vetoriais (Figura 11 b).

Figura 11 - Imagem IKONOS (geo-campo), distritos com fronteiras vetoriais (geo-objetos)

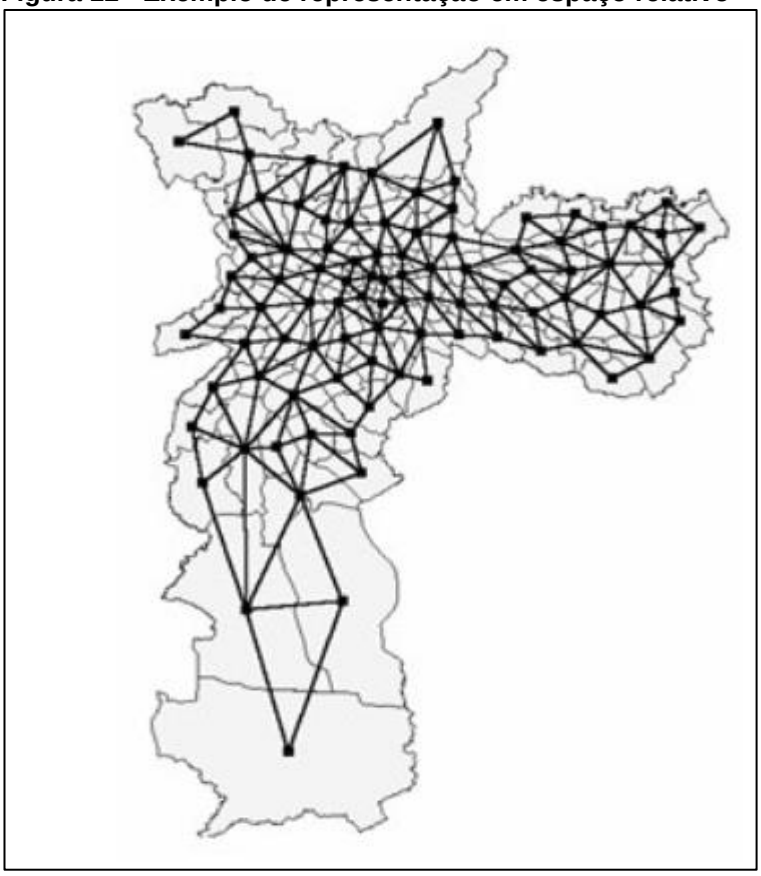


Fonte: Adaptado de Casanova et al. (2005)

A diferença primordial entre os dois modelos de representação está no papel da fronteira. Nos geo-campos, os limites da imagem determinam apenas as limitações do instrumento de captura das informações, podendo assim a imagem ser dividida e ainda manter suas propriedades essenciais. No modelo de geo-objetos, a fronteira define um elemento por completo, separando o objeto representado do mundo exterior e mantendo todas as propriedades do objeto constantes.

- Espaço relativo: representação apenas do posicionamento relativo entre as entidades, não é armazenada a localização exata de um elemento no plano, apenas as relações de adjacência entre os elementos. O modelo concebido no espaço relativo é o modelo de redes, análogo à teoria dos grafos. As representações são dadas por um conjunto de pontos no espaço (nós) e ligações entre tais pontos (arcos), onde é possível armazenar informações tanto nos nós quanto nos arcos, como exemplifica a Figura 12, onde pode-se observar os distritos da cidade de São Paulo e suas adjacências, a figura ao fundo está presente apenas para legibilidade (CASANOVA et al., 2005).

Figura 12 - Exemplo de representação em espaço relativo



Fonte: Casanova et al. (2005)

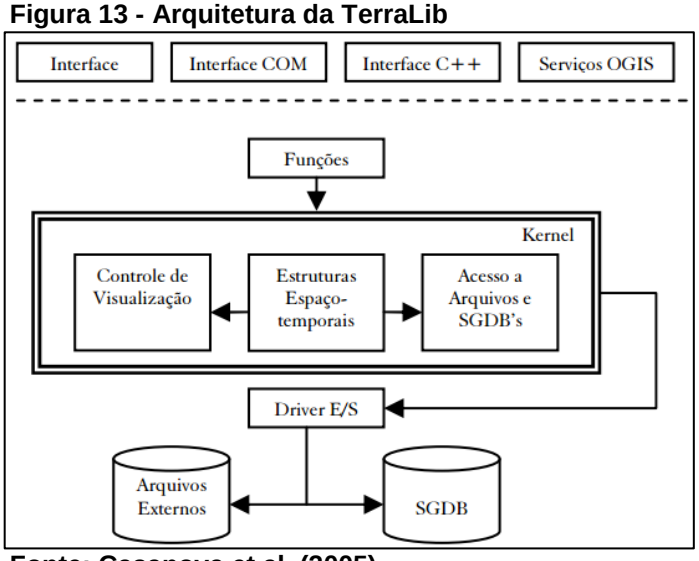
3.2 TERRALIB

De acordo com Casanova et al. (2005) as funções disponibilizadas pelo TerraLib compreendem estruturas de dados espaço-temporais, algoritmos para a decodificação de dados e análise espacial, além de propor um modelo para BDG. A estrutura do TerraLib pode ser dividida em três módulos:

- *Kernel*: é o módulo central do SIG, dotado de estrutura de dados espaço-temporais, controle de visualização, além da implementação de funções de análise espacial e suporte a projeções cartográficas.
- *Drivers* para recuperação e armazenamento de dados: contendo métodos de leitura, escrita e decodificação de dados geográficos armazenados em um banco de dados.
- Aplicativos de interface aos componentes do TerraLib: sobre esses dois módulos anteriores, pode-se construir aplicações atuando como interfaces às funcionalidades disponibilizadas pela TerraLib.

Trabalhando com uma estrutura em camadas, o TerraLib pode funcionar como uma interface entre um banco de dados e uma aplicação, onde o aplicativo se comunica com a ferramenta TerraLib e esta por sua vez se comunica com o SGBDG. Como ilustra a Figura 13, onde é possível observar o *kernel* como módulo central da representação, o módulo de *drivers* realizando a interface com o SGBD ou sistema de arquivos e na parte superior da figura as funções que provêm comunicação entre o *kernel* e outros aplicativos.

Figura 13 - Arquitetura da TerraLib



Fonte: Casanova et al. (2005)

Os algoritmos de processamento presentes na ferramenta TerraLib estão implementados levando em consideração o modelo conceitual de dados proposto pelos desenvolvedores do projeto. O *design* do projeto se deu levando em consideração requisitos de funcionalidade importantes para a comunidade desenvolvedora de aplicações georreferenciadas, como a facilidade de customização, onde a biblioteca oferece poderosas abstrações para sanar as necessidades dos desenvolvedores, e também compatibilidade com outros modelos baseados nas especificações da OGC (*Open Geospatial Consortium*) (HALL, 2008).

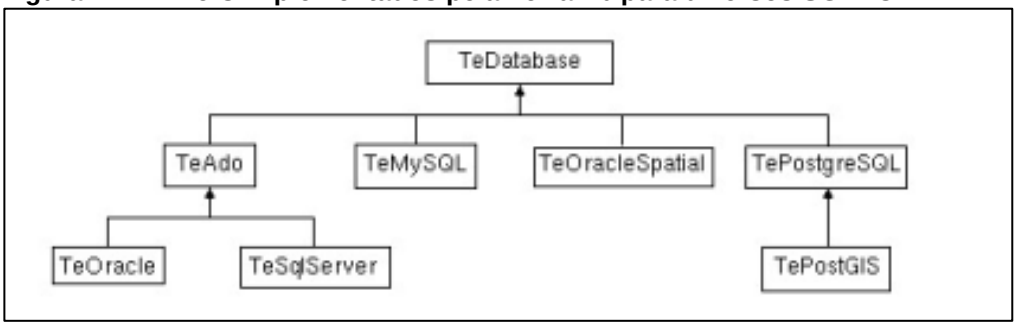
De acordo com Hall (2008), o modelo proposto pela ferramenta TerraLib oferece suporte às seguintes abstrações:

- *Database:* repositório de informações que contém dados e metadados;
- *Layer:* um agrupamento de objetos espaciais que dividem um mesmo conjunto de atributos. Exemplo: mapas cadastrais, imagens de satélite. Um *layer* é inserido ao modelo importando dados de arquivos ou outras bases de dados;
- Representação: são as informações geométricas contidas em um *layer.* Neste caso podem ser descritas das diversas formas citadas anteriormente no texto como por exemplo: pontos, linhas, células ou redes. Um mesmo dado pode ter diferentes representações dependendo da abordagem, por exemplo, uma cidade pode ser representada por um ponto ou por um polígono que descreve a sua região;
- Tema: contém um subconjunto dos objetos de um *layer*, que é produzido através de uma seleção realizada no conjunto dos elementos que formam o *layer*. Os parâmetros para esta seleção podem ser atributos, e condições espaciais e/ou temporais;
- Vista: compõe um conjunto de temas que são visualizados ou processados simultaneamente. Cada vista possui uma projeção cartográfica única, e os temas contidos nesta visão são convertidos para esta projeção;
- Visual: diz respeito aos atributos de apresentação dos elementos. Cada tema possui um visual único. Entre outras características de um visual, é possível destacar as cores e espessuras dos contornos das linhas de um tema, bem como o tamanho de cor dos elementos do tipo ponto.

De acordo com Casanova et al. (2005), o conceito de banco de dados implementado pela ferramenta TerraLib, independe do SGBD que efetivamente será utilizado para armazenamento dos dados geográficos. Para tal condição, faz-se uso da implementação de uma classe abstrata denominada *TeDatabase,* que possui métodos abstratos para criar, popular e manter um banco de dados.

Uma classe abstrata *TeDatabase* pode ser derivada para diversas classes concretas chamadas *drivers,* onde esses são as implementações dos aspectos práticos de diferentes SGBDs. Na Figura 14 pode-se visualizar os *drivers* já implementados na versão básica da instalação da TerraLib.

Figura 14 - Drivers implementados pela TerraLib para diversos SGBDs



Fonte: Casanova et al. (2005)

No Quadro 4 pode-se observar um trecho de código implementado na ferramenta TerraLib que promove a criação de um objeto do tipo *TeDatabase* sendo instanciado concretamente para de um determinado SGBD.

Quadro 4 - Exemplo de uso da classe TeDatabase

```
TeDatabase* myDB = 0;
if (op == "ado")
        myDb = new TeAdo(); //Usa ACCESS através da biblioteca ADO
else
        myDb = new TeMySQL(); //Usa MySQL
```

Fonte: Adaptado de Casanova et al. (2005)

A classe *TeLayer* é responsável por processar um dado na memória. Um exemplo desse processamento se dá pela importação de arquivos de dados do tipo MID, que contenha dados geográficos, como visualizado no Quadro 5.

Quadro 5 - Criação de um objeto do tipo *layer*

```
TeLayer* 1mun = TeImportMIF("../data/Municipios.mid",myDb,"Municipios");
```

Fonte: Adaptado de Casanova et al. (2005)

Para se criar representações, utiliza-se a classe *TeRepresentation*. Pode-se aplicar restrições ou seleções às informações sobre as representações contidas em um *layer*. O Quadro 6 mostra tal restrições.

Quadro 6- Classe *TeRepresentation*

```
TeRepresentation* munPol = 1mun->getRepresentation(TePOLYGONS);
//recupera o menor retângulo envolvente das geometrias do tipo
//polígono do layer de municípios
TeBox = munPol->box_;
```

Fonte: Casanova et al. (2005)

Para implementação de temas utiliza-se a classe *TeTheme*, e a classe *TeView* é a instanciação de visões. Pode-se ver no Quadro 7 a criação de um tema e de uma visão.

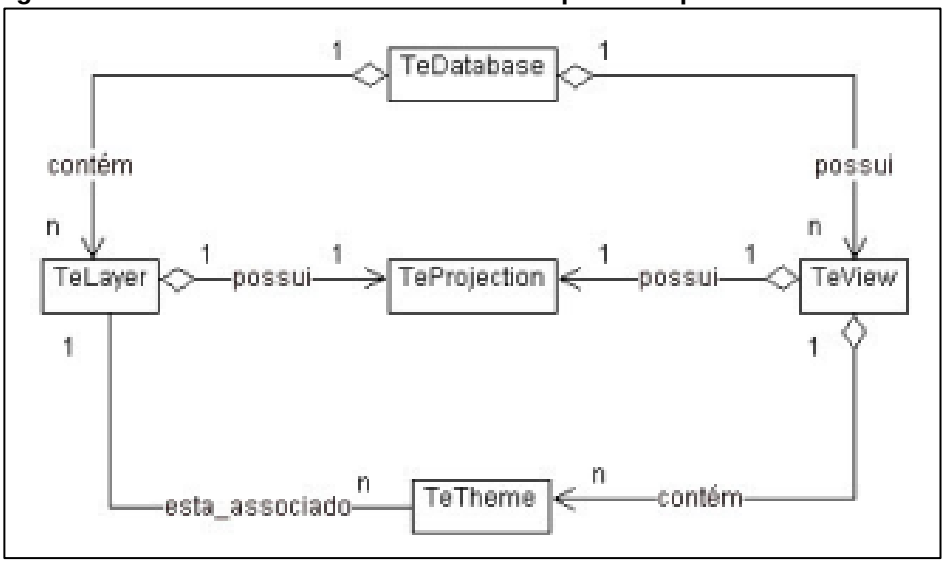
Quadro 7 - Criação de um tema e de uma vista

```
// Cria uma bista com a mesma projeção do layer
TeView* = new TeView("Municipios", user);
view->projection(munProj);
// salva a vista do banco de dados
myDb->insertView(view)
//cria um tema com todos os objetos do layerTeTheme* theme = new
TeTheme("t_municipios",1mun);
view->(theme);
```

Fonte: Casanova et al. (2005)

Para facilitar o entendimento nas relações entre as classes do modelo conceitual da ferramenta TerraLib, segue a Figura 15.

Figura 15 - Relacionamento entre as classes providas pela ferramenta TerraLib



Fonte: Casanova et al. (2005)

Exemplo de programa a ser executado utilizando a ferramenta TerraLib no Quadro 8. Este programa lê um arquivo contendo dados armazenados em formato *shapefile (*é um tipo comum de arquivo utilizado em SIGs, utilizado para armazenar localizações geométricas e informações associadas).

Um arquivo *shapefile* pode descrever geometrias, como pontos linhas e áreas, e também seus atributos. (QABAJA; BIKDASH, 2013). A primeira linha do método *main* é responsável por prover uma simples interface para visualização dos dados geográficos. As próximas duas linhas promovem a leitura do arquivo contendo os dados geográficos desejados. O comando *app.show*(ps); solicita à interface gráfica a visualização das informações solicitadas. A última linha cria uma tela de visualização.

Quadro 8 - Simples programa utilizando ferramenta TerraLib

```
#include <teapplication.h>
#include <tgeometry.h>
#include <tedataconversion.h>
int main ()
{       TApplication app;
        TePolygonSet ps;
        TeImportShape (ps, "BR.shp");
        app.Show (ps);
        app.Run(); }
```

Fonte: Adaptado de Casanova et al. (2005)

3.3 GEOTOOLS

De acordo com Hall (2008), GeoTools é uma biblioteca disponibilizada sob a licença de código aberto desenvolvida na linguagem de programação Java. As funções disponibilizadas pela biblioteca GeoTools obedecem às especificações do OGC, fornecendo métodos para facilitar a manipulação de dados geográficos armazenados por SIGs.

3.3.1 História

O desenvolvimento da biblioteca GeoTools deu início em 1996 na Universidade de Leeds, no Reino Unido, como parte de um projeto para visualização dos resultados produzidos por uma Máquina de Análise Geográfica (*Geographical Analysis Machine - GAM*). A ferramenta GeoTools foi inicialmente concebida como uma biblioteca para programadores construírem aplicações para manipulação de dados espaciais, ao invés de usuários produzirem mapas.

A primeira versão do GeoTools teve como principal característica a construção de mapas de uma maneira simples e interativa. Entretanto, esta abordagem levou o projeto a adquirir mais usuários do que desenvolvedores. Esta situação foi resolvida com o desenvolvimento de uma segunda versão da biblioteca, que possuía menos recursos de interface com o usuário. O que levou a ferramenta ser adotada por desenvolvedores, que estimavam a possibilidade de escrever mais do seu próprio código para construir um programa útil (HALL, 2008).

No ano de 2002 iniciou-se os trabalhos para o aprimoramento da biblioteca. Adicionando-se códigos provenientes do projeto *SeaGIS*, a ferramenta GeoTools passou a prover funcionalidades adicionais, tais como: transformação de coordenadas, cobertura por *grids* e renderização. Nesta etapa, com um novo *design* de desenvolvimento, tornou-se possível a ampliação da comunidade desenvolvedora (GEOTOOLS, 2009). Desde então seu código vem se desenvolvendo rapidamente com novas atualização elaboradas pela comunidade participante. Hoje a biblioteca encontra-se na sua versão 17.

3.3.2 Funcionalidades

Principais funcionalidades disponibilizadas pela biblioteca (GEOTOOLS, 2017):

- Acesso a vários arquivos de vários formatos e BDGs;
- Trabalho com uma extensa quantidade de projeções cartográficas;
- Filtro e análise de dados quanto a atributos espaciais e não espaciais;
- Geração e apresentação mapas com estilos complexos;
- Criação e análise de grafos e redes.

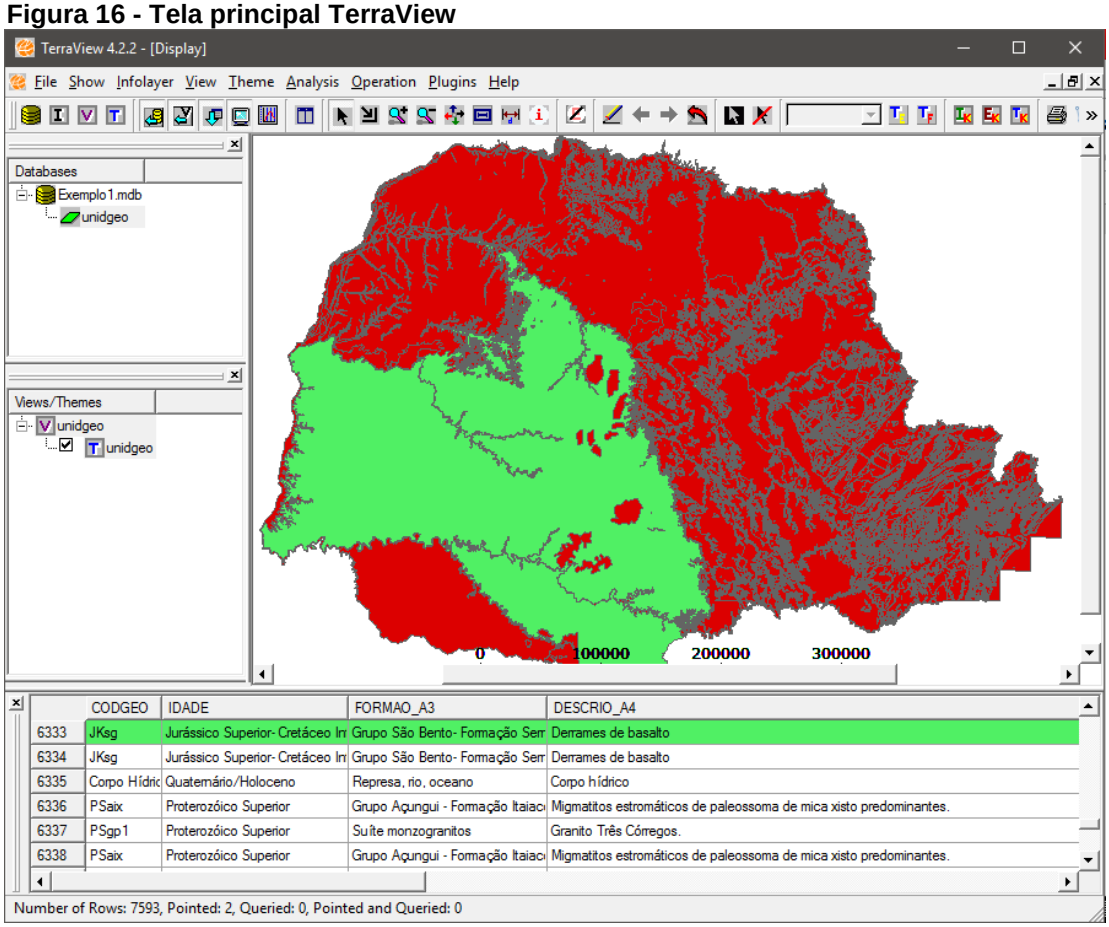
3.4 TERRAVIEW

Um exemplo de aplicação que pode usar o TerraLib para se comunicar com um BDG é a ferramenta TerraView. Essa ferramenta possui funções para interagir com um banco de dados gerido pela ferramenta TerraLib, podendo importar, visualizar e exportar dados matriciais (grades e imagens) e também dados vetoriais (pontos, linhas e polígonos) (TERRAVIEW, 2010). Ainda de acordo com TerraView (2010), os principais objetivos da ferramenta são:

- Apresentar à comunidade acadêmica uma ferramenta de fácil utilização para realização de visualização, consulta e análise de dados geográficos;
- Exemplificar a utilização da ferramenta TerraLib.

A Figura 16 mostra a tela principal do TerraLib para visualização de informações, onde, partindo de um banco de dados com informações geográficas, tem-se a possibilidade de visualiza-las graficamente. Na parte central da imagem tem-se a representação gráfica dos dados, na região inferior da janela encontram-se os dados convencionais relacionados aos dados geográficos. Na parte esquerda da janela, encontram-se as bases de dados armazenadas e suas respectivas *views* implementadas.

Figura 16 - Tela principal TerraView



Fonte: Autoria própria (2017)

3.5 BASE DE DADOS

A proposta inicial desse trabalho propunha a utilização de dados que seriam obtidos da base do IBGE, porém, dados mais relevantes para o contexto da região Pontagrossense foram encontrados no portal Geoportal Ponta Grossa, apresentando características adequadas para o projeto, como quantidade e forma dos dados geográficos.

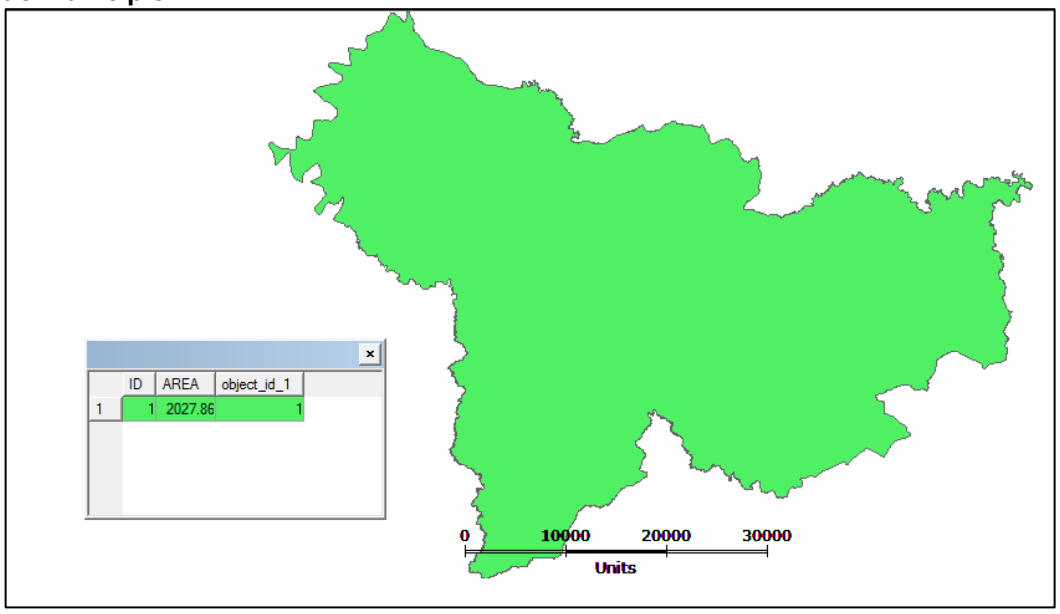
O portal Geoportal Ponta Grossa, concebido em parceria entre a Secretaria de Administração e Negócios Jurídicos e a Secretaria de Planejamento, foi desenvolvido para se tornar uma plataforma de divulgação da geoinformação do território Pontagrossense[7].

[7] O sítio do portal Geoportal Ponta Grossa é <http://geo.pg.pr.gov.br/portal/>

O portal busca oferecer acesso à comunidade interessada, arquivos contendo informações geográficas sobre vários aspectos do município, como seu tamanho, os bairros que o compõe, a hidrografia da região, compreendendo rios, arroios e cursos d'água, entre outras informações. Os dados estão disponíveis em vários formatos, sendo o mais adequado para ser utilizado nesse trabalho os dados do tipo *shapefile.*

A Figura 17 mostra a visualização gráfica das informações contidas em um arquivo adquirido da base acima descrita. Pode-se observar informações como o perímetro do município de Ponta Grossa, bem como sua área total de abrangência.

Figura 17 - Representação gráfica dos dados contidos no arquivo referente aos limites do município



Fonte: Autoria própria (2017)

Como os dados disponíveis pelo portal Geoportal Ponta Grossa são numerosos, foi realizado uma delimitação dos que foram utilizados na execução desse trabalho. Dentre os *shapefiles* disponíveis serão utilizados: limite de município, distritos, bairros, hidrografia, pontos de saúde e educação.

3.6 CONSIDERAÇÕES DO CAPÍTULO

Com o intuito de promover a instalação e utilização da biblioteca TerraLib, iniciou-se o processo para compilação e uso efetivo da mesma, porém problemas

como a falta de compatibilidade de diversos componentes necessários para sua compilação, assim como a falta de material de apoio para sanar tais dificuldades, levaram à troca do ambiente através do qual foram realizadas as análises das funcionalidades disponíveis em um SIG convencional.

Durante o processo de compilação da biblioteca TerraLib, algumas classes não foram efetivamente compiladas e a utilização do produto final da compilação estava comprometida, não sendo possível realizar o uso de todas as funcionalidades estimadas na fase de projeto. Portanto, se fez necessária a troca da ferramenta que seria utilizada para o estudo das funcionalidades de um SIG.

A biblioteca escolhida foi a GeoTools, que tal como a TerraLib, mantém integração com os conceitos do OGC e disponibiliza *plug-ins* para o controle de dados armazenados em SGBDs PostGIS. Ainda possuindo algumas facilidades em comparação à primeira escolha, como: documentação consistente para a implementação de funcionalidades básicas da ferramenta; utilização de uma ferramenta para facilitar o gerenciamento das dependências e repositórios necessários a cada projeto que utiliza GeoTools.

Dessa forma, foi possível a devida implementação das funcionalidades propostas mantendo uma compatibilidade com os conceitos explanados nos capítulos de revisão.

4 UTILIZAÇÃO DAS FERRAMENTAS

Este capítulo está dividido em 4 seções. A seção 4.1 descreve o processo para realizar a criação e configuração de uma base de dados PostGIS seguindo com o processo de importação dos arquivos no formato *shapefile,* descritos no capítulo 3.5, para a base de dados. A seção 4.2 apresenta os passos necessários para a utilização da biblioteca *GeoTools* em um ambiente *Windows* utilizando o IDE (*Integrated Development Environment*) NetBeans. A seção 4.3 destina-se a exemplificação da integração entre uma base de dados PostGIS e um projeto GeoTools. A seção 4.4 faz um resumo do assunto tratado no capítulo.

4.1 TRABALHANDO COM O AMBIENTE POSTGIS

Como descrito na seção 2.6, a biblioteca PostGIS é uma extensão aos recursos providos pelo SGBD PostgreSQL, portanto, a partir de um ambiente PostgreSQL é necessário configurar a utilização da biblioteca PostGIS para se fazer uso de suas funcionalidades referentes ao armazenamento e manutenção de dados geográficos. A seção 4.1.1 está destinada a demonstrar o processo de configuração de uma instância de banco de dados PostGIS utilizando um ambiente PostgreSQL. Na seção 4.1.2 será demonstrado o processo para importação de arquivos no formato *shapefile* para uma base PostGIS configurada.

4.1.1 Criação da Base de Dados PostGIS

A partir das instalações do ambiente PostgreSQL e da extensão PostGIS, se faz necessário realizar a criação de uma base de dados PostgreSQL convencional como exemplificado no Quadro 9.

Quadro 9 – Código Para Criação de Base de Dados PostgreSQL

```
CREATE DATABASE BaseDeDadosPostGIS;
```

Fonte: Adaptado de POSTGIS (2017)

Tendo a base de dados criada, se faz necessária realizar a execução de alguns comandos instancia para se habilitar o suporte ao armazenamento e manipulação de dados geográficos. Os comandos se encontram no site do PostGIS[8], e podem também ser visualizados no Quadro 10.

Quadro 10 - Comandos para a habilitação das funções do PostGIS

```
-- Enable PostGIS (includes raster)
CREATE EXTENSION postgis;
-- Enable Topology
CREATE EXTENSION postgis_topology;
-- Enable PostGIS Advanced 3D
-- and other geoprocessing algorithms
-- sfcgal not available with all distributions
CREATE EXTENSION postgis_sfcgal;
-- fuzzy matching needed for Tiger
CREATE EXTENSION fuzzystrmatch;
-- rule based standardizer
CREATE EXTENSION address_standardizer;
-- example rule data set
CREATE EXTENSION address_standardizer_data_us;
-- Enable US Tiger Geocoder
CREATE EXTENSION postgis_tiger_geocoder;
```

Fonte: Adaptado de POSTGIS (2017)

Após os passos acima descritos terem sido executados com sucesso, tornou-se possível a importação de arquivos no formato *shapefile* para a base de dados configurada. O processo de importação será descrito e exemplificado na seção seguinte.

4.1.2 Importação de Arquivos *Shapefile*

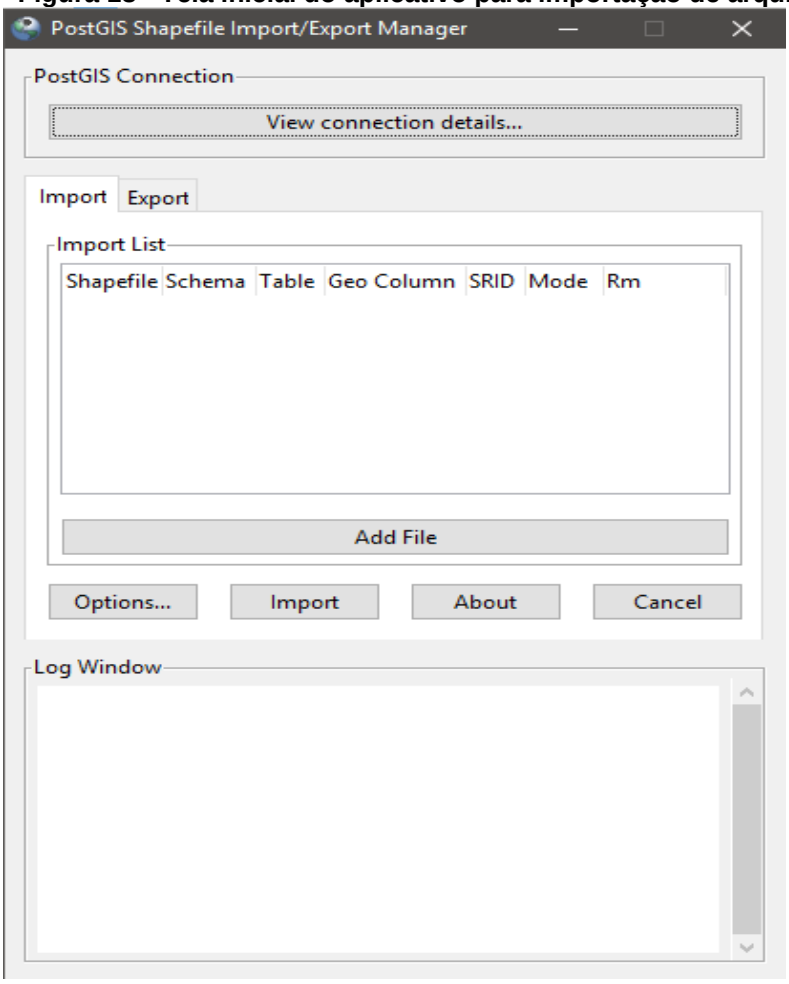
Como definido na seção 3.2, arquivos *shapefile* são utilizados para armazenamento de informações geográficas fora de BDG. Os seguintes passos são

[8] Os comandos para instalação do PostGIS estão disponíveis no sítio <http://postgis.net/install/>

necessários para a importação de arquivos no formato *shapefile* para um banco de dados PostGIS.

Acompanhando a instalação da biblioteca PostGIS no *Windows* está um utilitário que facilita a importação de arquivos no formato *shapefile* para uma base de dados PostGIS, denominado *PostGIS 2.0 Shapefile and DBF Loader Exporter*. A Figura 18 mostra a tela inicial do aplicativo. Nesta tela é possível definir as configurações de conexão com a base de dados, adicionar arquivos para a importação, informar configurações adicionais e iniciar o processo de importação de arquivos.

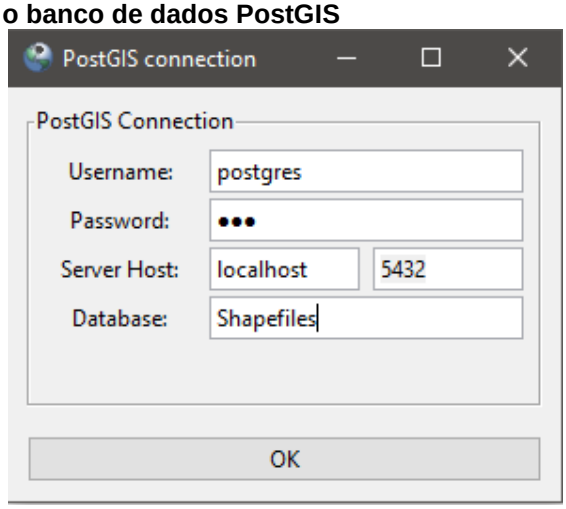
Figura 18 - Tela inicial do aplicativo para importação de arquivos Shapefile



Fonte: Autoria própria (2017)

A Figura 19 exemplifica o processo de conexão com a base de dados, onde é solicitado pela aplicação as seguintes informações: usuário, senha, servidor e nome da base de dados PostGIS.

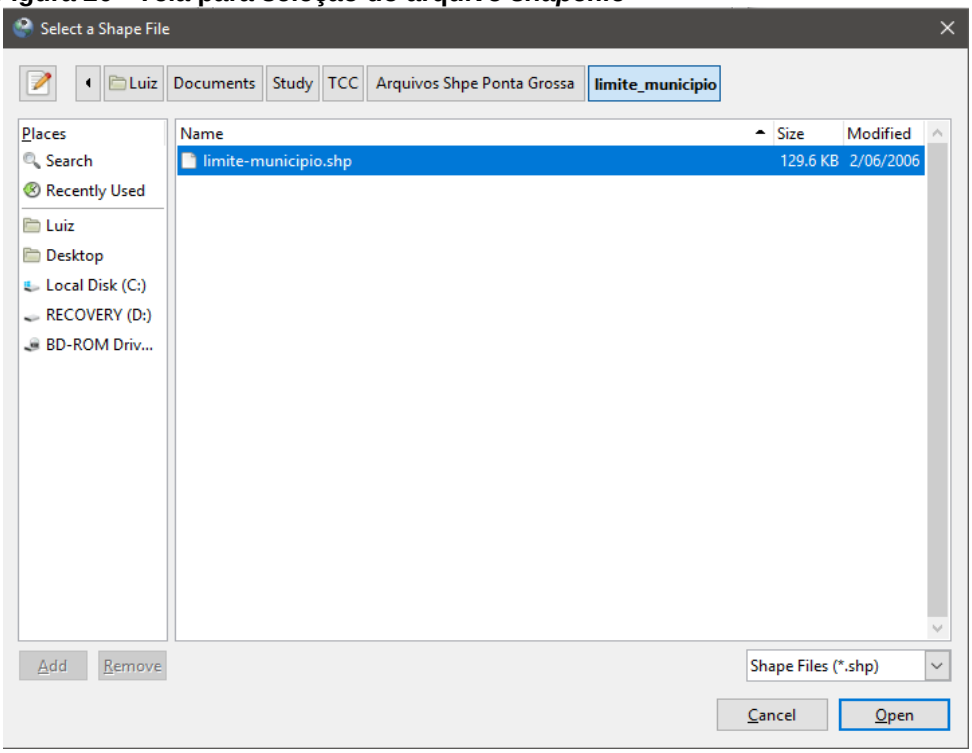
Figura 19 - Tela para configuração dos parâmetros para conexão com o banco de dados PostGIS



Fonte: Autoria própria (2017)

Na Figura 20 é possível ver como é feito o processo de escolha dos arquivos *shapefile*. Basta navegar ao local onde se encontra o arquivo desejado e selecioná-lo.

Figura 20 - Tela para seleção do arquivo *shapefile*



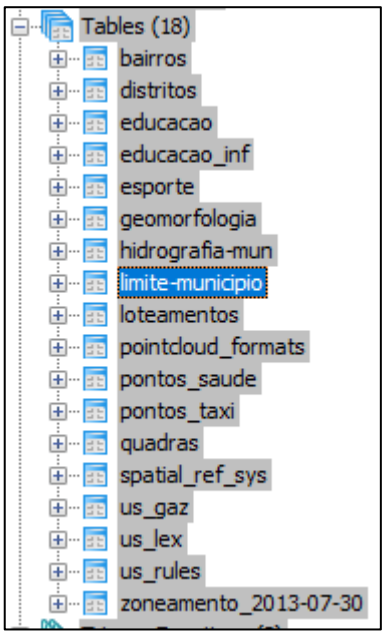
Fonte: Autoria própria (2017)

Para os arquivos definidos durante a seção 3.5 se faz necessário alterar a configuração de *DBF file character encoding* de UTF-8 para LATIN1. Em caso de

importação com sucesso, será apresentada uma mensagem na área de *Log Window* do utilitário para importação informando "*Shapefile import completed"*.

A Figura 21 demonstra uma base de dados PostgreSQL contendo os arquivos importados, onde cada arquivo importado pela ferramenta está associado a uma tabela na base de dados. Os arquivos utilizados para a importação foram os descritos no Capítulo 3.5, contendo informações geográficas da cidade de Ponta Grossa.

Figura 21 - Arquivo Shapefile importado com sucesso para a base de dados



Fonte: Autoria própria (2017)

4.2 IMPLANTAÇÃO DA BIBLIOTECA GEOTOOLS NO NETBEANS

Nas subseções seguintes serão descritos os processos envolvidos com a implantação de um ambiente GeoTools utilizando a IDE NetBeans em ambiente Windows.

4.2.1 Utilização da ferramenta Maven

De acordo com a documentação provida pelo *Apache Maven Project*[9], Maven é um projeto que foi iniciado como uma tentativa de simplificar os processos de construção no projeto Jakarta Turbine[10], que é um *framework* desenvolvido para agilizar o processo de construção de aplicações *web* seguras. Haviam vários projetos em desenvolvimento durante as fazes iniciais do projeto, cada um com seus próprios arquivos de compilação, que eram todos ligeiramente diferentes.

O que se objetivou com a criação do projeto *Maven* foi uma forma padrão de construir projetos, uma definição clara do que consistia o projeto, uma maneira fácil de publicar informações de projetos e uma maneira de compartilhar arquivos no formato JAR (*Java Archive*) entre vários projetos. Tendo assim como resultado uma ferramenta que pode ser usada para construir e gerenciar qualquer projeto baseado em Java.

A ferramenta *Maven* é utilizado na biblioteca GeoTools para o gerenciamento das bibliotecas JAR que são incorporadas aos projetos construídos através da IDE NetBeans. De acordo com (GEOTOOLS, 2017), fora escolhida a utilização da Maven para a compilação dos componentes da biblioteca pelos seguintes critérios:

- O *download* dos componentes do GeoTools será realizado conforme a necessidade de cada aplicação;
- Os arquivos com extensão .jar utilizados no projeto são baixados para um único local no diretório inicial;
- Serão utilizadas as versões específicas de todos os arquivos com extensão .jar exigidos pela aplicação. Isso ajuda a evitar erros que podem ser causados por dependências incorretas;
- A utilização de um único repositório para o armazenamento de dependências facilita a organização de projetos de código aberto.

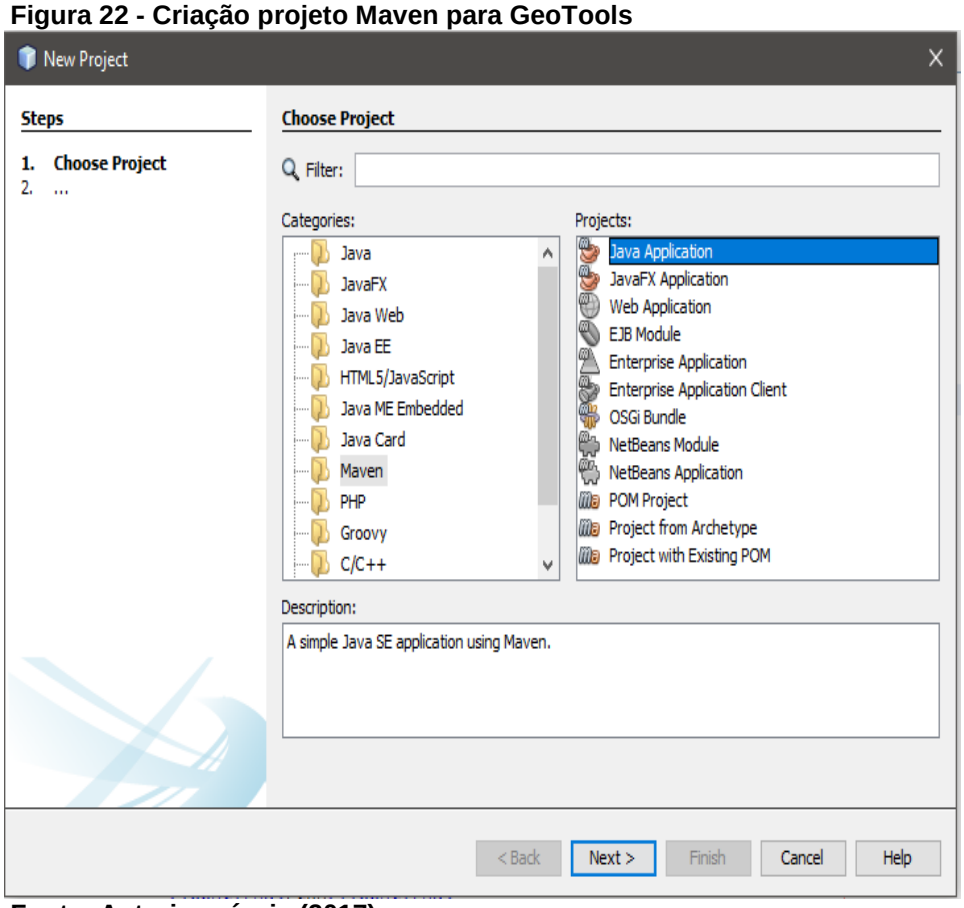
[9] A documentação do projeto Apache Maven encontra-se no sítio: <https://maven.apache.org/what-is-maven.html>

[10] Maiores informações sobre o projeto Jakarta Turbine encontram-se no sítio <https://turbine.apache.org/turbine/turbine-2.1/>

Durante o processo de construção da aplicação, o repositório local do *Maven* será utilizado para armazenar tanto os arquivos .jar baixados do repositório geral do projeto GeoTools, quanto os gerados localmente pelo NetBeans.

Para se criar um projeto Java utilizando os recursos providos pela ferramenta Maven em um ambiente NetBeans, é preciso acessar o caminho: *File -> New Project* no NetBeans, selecionar a pasta Maven e em seguida Java *Application*, como exemplificado na Figura 22.

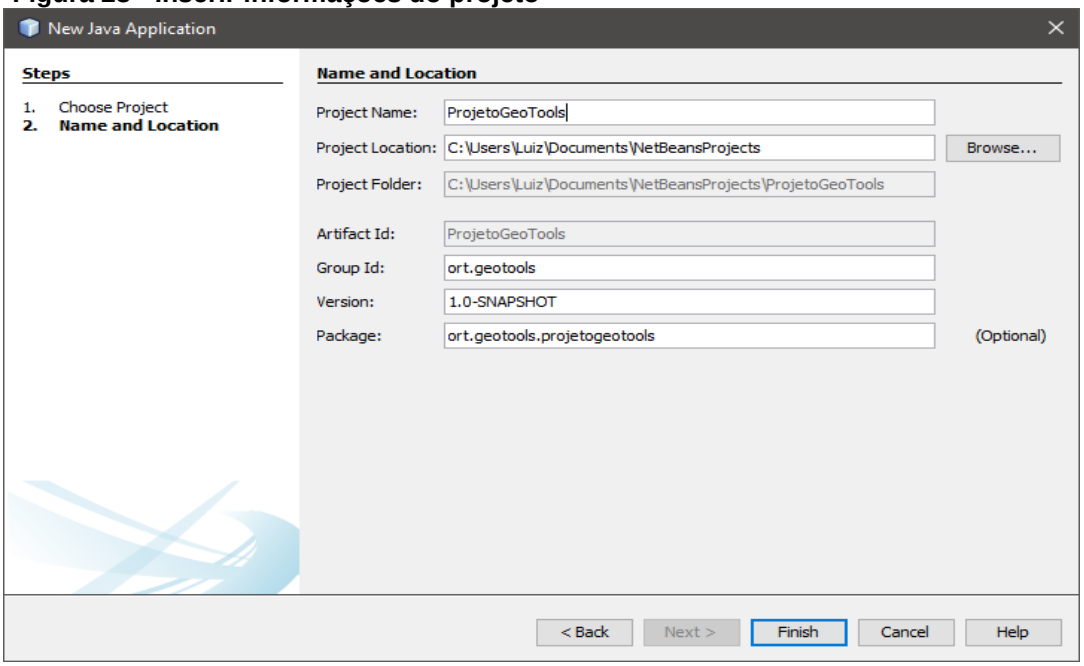
Figura 22 - Criação projeto Maven para GeoTools



Fonte: Autoria própria (2017)

Em seguida deve-se inserir as informações como nome do projeto e local para armazenamento dos arquivos, processo exemplificado na Figura 23.

Figura 23 - Inserir informações do projeto



Fonte: Autoria própria (2017)

Para controlar versões e demais dependências de repositórios da biblioteca GeoTools, o *Maven* utiliza um arquivo no formato XML denominado pom.xml, presente na pasta *Project Files* da estrutura de pastas do projeto NetBeans. As propriedades do projeto se encontram entre as *tags <properties>*, como pode ser observado no Quadro 11. Tais propriedades descrevem a versão da biblioteca GeoTools que será utilizada pela aplicação, assim como a codificação que será utilizada pelo aplicativo.

Para se alterar a versão do GeoTools, que se está utilizando para compilar o projeto, basta alterar a *tag <geotools.version>*Versão desejada *</geotools.version>* e recompilar o projeto.

Quadro 11 - Tag de propriedades arquivo pom.xml

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <geotools.version>17.2</geotools.version>
</properties>
```

Fonte: Autoria própria (2017)

As dependências do projeto entre as *tags <dependencies>,* conforme ilustrado no Quadro 12. São informadas nas dependências os recursos da biblioteca GeoTools que serão utilizados durante o projeto, como: recursos para conexão com

um banco de dados PostGIS, estruturas para a leitura, manipulação e visualização de informações geográficas como mapas.

Quadro 12 - Tag de dependencias arquivo pom.xml

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.geotools</groupId>
    <artifactId>gt-shapefile</artifactId>
    <version>${geotools.version}</version>
  </dependency>
  <dependency>
    <groupId>org.geotools</groupId>
    <artifactId>gt-swing</artifactId>
    <version>${geotools.version}</version>
  </dependency>
<dependencies>
```

Fonte: Autoria Própria (2017)

Os repositórios que serão utilizados para o *download* dos arquivos .jar se encontram entre as *tags <repositories>,* como exemplificado pelo Quadro 13.

Quadro 13 - Tag de repositórios arquivo pom.xml

```
<repositories>
 <repository>
    <id>maven2-repository.dev.java.net</id>
    <name>Java.net repository</name>
    <url>http://download.java.net/maven/2</url>
 </repository>
 <repository>
    <id>osgeo</id>
    <name>Open Source Geospatial Foundation Repository</name>
    <url>http://download.osgeo.org/webdav/geotools/</url>
 </repository>
</repositories>
```

Fonte: Autoria própria (2017)

4.2.2 Utilização dos Recursos Providos Pela Biblioteca GeoTools

Para se utilizar um recurso provido pela biblioteca, é necessário incluir na classe Java a importação do recurso, utilizando a cláusula *import* no cabeçalho do arquivo .jar, e este recurso estar vinculado a uma dependência do arquivo pom.xml. Exemplo: uma das maneiras para se realizar a visualização de mapas utilizando o GeoTools é através da classe *JMapFrame*, presente no componente *swing* da biblioteca, conforme exemplifica o Quadro 14.

Quadro 14 - Uso da classe JMapFrame presenta na biblioteca GeoTools

```
Style style = SLD.createSimpleStyle(source.getSchema());

FeatureLayer layer = new FeatureLayer(source, style);

mapsetTitle("Exibir Shape");
map.addLayer(layer);

JMapFrame.showMap(map);
```

Fonte: Autoria própria (2017)

Portanto, é necessário incluir a importação da biblioteca *swing* no cabeçalho da classe, como mostra o Quadro 15, e esta classe deve estar presente em uma dependência referenciada no arquivo pom.xml, conforme mostra o Quadro 16, onde o recurso *JMapFrame* está disponível na dependência denominada *gt-swing* da biblioteca GeoTools.

Quadro 15 - Importação da classe JMapFrame

```
import java.util.HashMap;
import org.geotools.map.FeatureLayer;
import org.geotools.map.MapContent;
import org.geotools.styling.SLD;
import org.geotools.styling.Style;
import org.geotools.swing.JMapFrame;
```

Fonte: Autoria própria (2017)

Quadro 16 - Inclusão da dependência da classe JMapFrame no arquivo pom.xml

```
<dependency>
    <groupId>org.geotools</groupId>
    <artifactId>gt-swing</artifactId>
    <version>${geotools.version}</version>
<dependency>
```

Fonte: Autoria própria (2017)

4.3 INTEGRAÇÃO DE UMA BASE DE DADOS POSTGIS COM UMA APLICAÇÃO GEOTOOLS

Para leitura e escrita de dados em um banco de dados PostGIS, é preciso configurar uma conexão com a base de dados utilizando a classe *DataStore* presente na dependência “gt-jdbc-postgis” que deve ser referenciada no arquivo pom.xml para a utilização de seus recursos. Um exemplo de configuração de conexão com uma base de dados pode ser encontrado no Quadro 17.

Quadro 17 - Configuração de Conexão com Base de Dados PostGIS

```
DataStore dataStore;
    Map<String,Object> params = new HashMap<>();
    params.put( "dbtype", "postgis");
    params.put( "host", "localhost");
    params.put( "port", 5432);
    params.put( "schema", "public");
    params.put( "database", "Shapefiles");
    params.put( "user", "postgres");
    params.put( "passwd", "123");
    dataStore = DataStoreFinder.getDataStore(params);
```

Fonte: Autoria própria (2017)

A partir da conexão estabelecida com sucesso, é possível realizar o leitura e escrita de informações no banco utilizando o objeto *DataStore* configurado. Como pode ser visualizado no Quadro 18, onde um objeto do *DataStore* é utilizado para preencher um elemento *ComboBox* com as tabelas presentes na base de dados requisitada.

Quadro 18 - Utilização do objeto DataStore

```
  ComboBoxModel cbm = new
DefaultComboBoxModel(dataStore.getTypeNames());
  cBoxTabelas.setModel(cbm);
```

Fonte: Autoria própria (2017)

4.4 CONSIDERAÇÕES DO CAPÍTULO

Neste capítulo foram exemplificados os recursos mínimos necessários para se construir uma aplicação que gerencia dados geográficos provenientes de um banco de dados PostGIS utilizando os recursos providos pela biblioteca GeoTools.

Durante o capítulo, foram utilizados trechos de código para exemplificar a implementação de recursos como: criação de uma base de dados PostGIS e importação de arquivos *shapefile;* utilização do Maven para definição das propriedades do projeto GeoTools; integração de uma aplicação GeoTools com um banco de dados PostGIS.

5 FUNCIONALIDADES OBTIDAS COM O USO DO GEOTOOLS

Esta seção está dividida em 5 seções. Cada seção demonstra uma funcionalidade obtida através da utilização de recursos providos pela biblioteca GeoTools, integrada a uma base de dados PostGIS, contendo os arquivos *shapefile* importados com informações geográficas da cidade de Ponta Grossa.

A seção 5.1 demonstra os passos necessários para se construir os estilos utilizados na visualização dos mapas. A seção 5.2 exemplifica como utilizar os recursos da biblioteca GeoTools para se realizar inserção de dados do tipo ponto, e demais informações sobre o ponto, em coordenadas específicas de determinada região.

A seção 5.3 demonstra a funcionalidade de obtenção de informações de uma determinada região partindo de uma seleção no mapa apresentado ao usuário. A seção 5.4 demonstra a obtenção de informações proveniente de interpolação de duas camadas de informações geográficas. Na última seção encontram-se as considerações do capítulo.

5.1 APLICAÇÃO DE ESTILOS NA VISUALIZAÇÃO DE MAPAS

Como descreve a documentação (GeoTools, 2017)[11], a área da geografia denominada cartografia está focada no uso de mapas para comunicar informações geográficas de maneira eficiente. Possuindo atribuições que vão desde a escolha de quais informações incluir e a forma de sua apresentação.

Ocasionalmente, organizações terão padrões cartográficos que devem ser seguidos. Por exemplo: qual espessura deve ser utilizada para uma linha que separa duas regiões, qual tom de azul deve ser utilizado para se representar água. Utilizar um padrão cartográfico é uma grande economia de tempo. Para as informações que não são compreendidas no padrão utilizado, o operador do mapa deve utilizar a criatividade de maneira produtiva.

Um padrão utilizado na biblioteca GeoTools para facilitar a utilização de estilos é o *Styled Layer Descriptor* (SLD), este documento define uma estrutura de dados

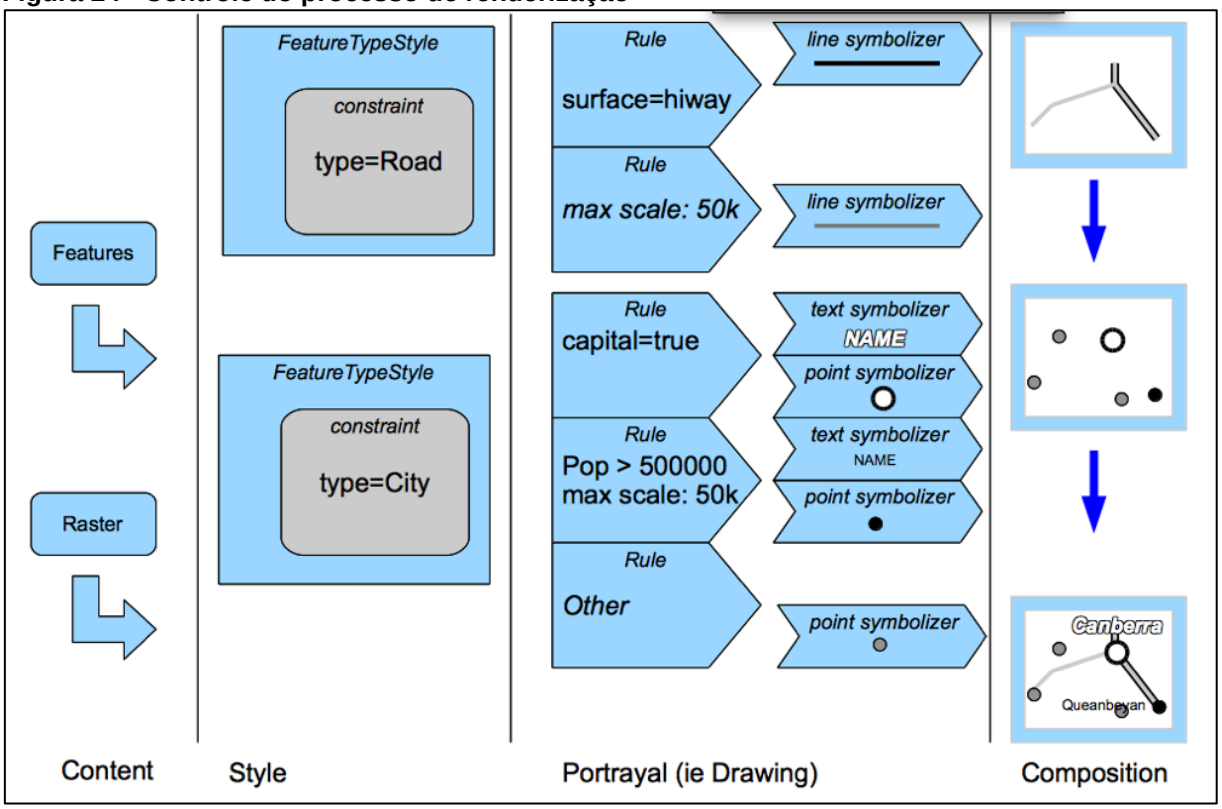
[11] http://docs.geotools.org/latest/userguide/tutorial/map/style.html

para estilos que são representados na forma de objetos Java. Sua utilização está centrada em duas abordagens:

- Descritor de camada de estilo: abrange a definição das “camadas” ou conteúdo de características;
- Codificação de simbologia: abrange a descrição de como desenhar os recursos.

A Figura 24 demonstra a forma como a biblioteca GeoTools controla o processo de renderização dos elementos solicitados para apresentação ao usuário, onde, a partir das características, regras são aplicadas com base nos valores dos recursos para definição da forma como determinada informação será representada.

Figura 24 - Controle do processo de renderização



Fonte: Geotools (2017)[12]

No Quadro 19 encontra-se o código utilizando para se criar um estilo padrão para a visualização das infomações no mapa. A classe *styleBuinder* é utilizada para

[12] Conteúdo referente a utilização de estilos no GeoTools podem ser encontrados no sítio: <http://docs.geotools.org/latest/userguide/tutorial/map/style.html>

definição de uma fonte a ser utilizada na visualização da informação de “nome” da característica visualizada.

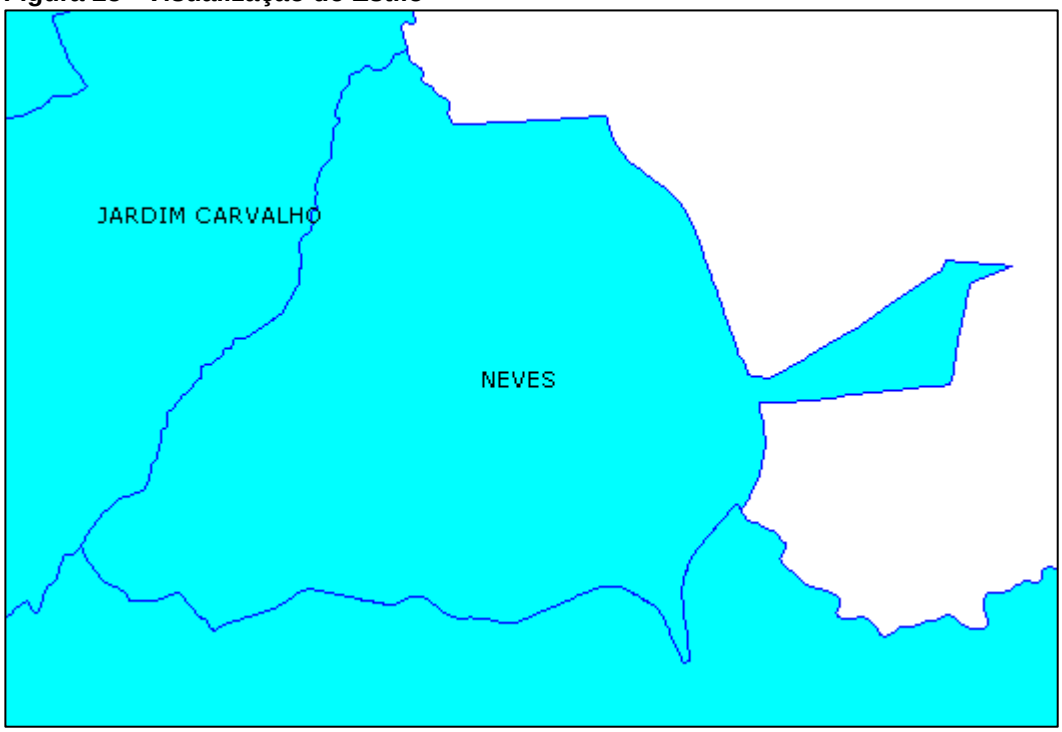
Quadro 19 - Criar Estilo

```
    private Style criarEstiloPadraoLayer1() {
        Style style = SLD.createPolygonStyle(Color.BLACK, Color.BLACK, 1f,
"nome", font);

        Rule rule = criarRegra(LINHA_COR, PREENCHIMENTO_COR);
        FeatureTypeStyle fts = sf.createFeatureTypeStyle();
        fts.rules().add(rule);

        style.featureTypeStyles().add(fts);
        return style;
    }
```

Fonte: Autoria própria (2017)

A Figura 25 mostra a visualização de um mapa com as configurações definidas pelo código do Quadro 19, onde a região nela representada possui como atributo do tipo “nome” o texto “Neves” e as cores visualizadas são as definidas na classe CriarEstiloPadrãoLayer1 definida no Quadro 19.

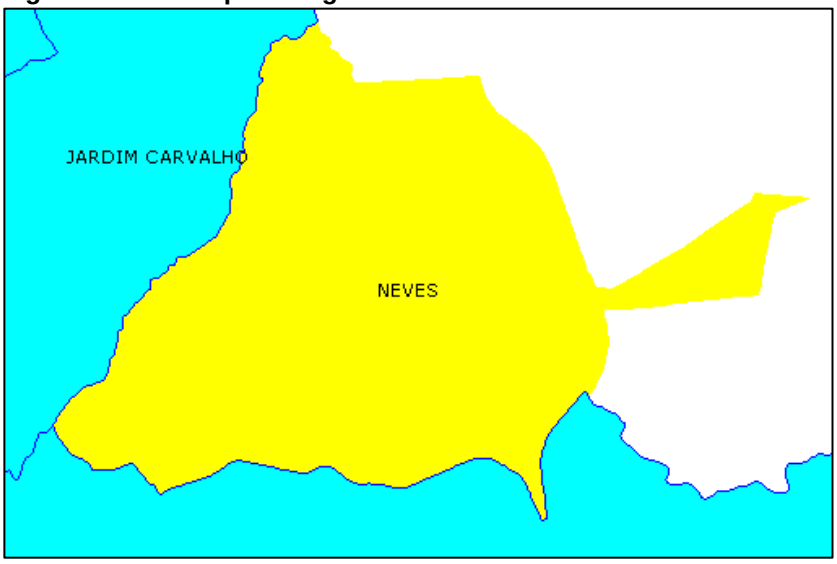
Figura 25 - Visualização de Estilo



Fonte: Autoria própria (2017)

Na tela de visalização das informações geográficas, a partir da seleção de uma determinada região, troca-se o estilo de visualização da região selecionada, como demonstra a Figura 26. Onde a região denominada Neves fora selecionada para visualização de infomrações adicionais sobre a área.

Figura 26 - Estilo para Região selecionada



Fonte: Autoria própria (2017)

O Quadro 20 domonstra o código para a definição do estilo de uma região selecionada. Como pode ser observado, cria-se uma regra definindo as cores que serão utilizadas para demonstrar uma seleção de característica, então se adiciona um filtro à regra, definindo para quais identificadores será utilizada tal característica, onde neste caso o filtro contém o código identificador da área selecionada.

Quadro 20 - Criar Estilo para Região selecionada

```
    public void visualizarCaracteristicaSelecionada(Set<FeatureId> IDs) {
        Style style;
        if (IDs.isEmpty()) {
            style = criarEstiloPadraoLayer1();
        } else {
            style = criarEstiloCaracteristicaSelecionada(IDs);
        }
        Layer layer = jMapFrame.getMapContent().layers().get(0);
        ((FeatureLayer) layer).setStyle(style);
        jMapFrame.getMapPane().repaint();
    }
```

Fonte: Autoria própria (2017)

5.2 INCLUSÃO DE INFORMAÇÕES DO TIPO PONTO NA BASE DE DADOS

Como descrito na seção 2.6 do trabalho, pontos são armazenados definindo-se um par de coordenadas (X, Y). As tabelas armazenadas no banco de dados que contém informações referentes a pontos são: educação, educação infantil, esporte, pontos de saúde e pontos de taxi. Para se adicionar um ponto na base de dados, deve-se selecionar qual informação do tipo ponto se quer adicionar, em seguida definir sobre qual região se quer visualizar os pontos. A Figura 27 demonstra a visualização de pontos do tipo educação infantil sobre uma camada contendo informações referente aos bairros.

Figura 27 - Visualizar/Adicionar Pontos

Fonte: Autoria própria (2017)

Para se adicionar um ponto clica-se no botão “Adicionar Ponto – Educação Infantil” e clica no mapa na região desejada. Serão solicitados os demais dados

referentes ao ponto. Após a informação dos dados o ponto será visualizado na região selecionada e estará armazenado na base de dados.

O Quadro 21 apresenta o código desenvolvido para a inclusão de um ponto do tipo Educação Infantil à base de dados e à visualização do mapa. As três primeiras linhas da função *adicionarPonto* adicionam à variável do tipo coordenada os valores de X, e Y do ponto selecionado no *click*.

Quadro 21 - Adicionar Ponto Educação Infantil Parte 1

```
    void adicionarPonto(MapMouseEvent ev, String typeName) throws IOException
{
        DirectPosition2D p = ev.getWorldPos();
        SimpleFeatureBuilder BLDR = null;
        Coordinate pos = new Coordinate(p.getX(), p.getY());
        featureSource = dataStore.getFeatureSource(typeName);
        Style style = SLD.createSimpleStyle(featureSource.getSchema());
        switch (typeName) {
           case "educacao_inf":
              BLDR = new SimpleFeatureBuilder(TYPEEducacao_inf);
              featureCollection = new DefaultFeatureCollection("internal",
TYPEEducacao_inf);
              break;
```

Fonte: Autoria própria (2017)

No Quadro 22 está definido o código desenvolvido para solicitação das informações referentes ao ponto que se deseja incluir, bem como a gravação dessas informações na base de dados.

Quadro 22 - Adicionar Ponto Educação Infantil Parte 2

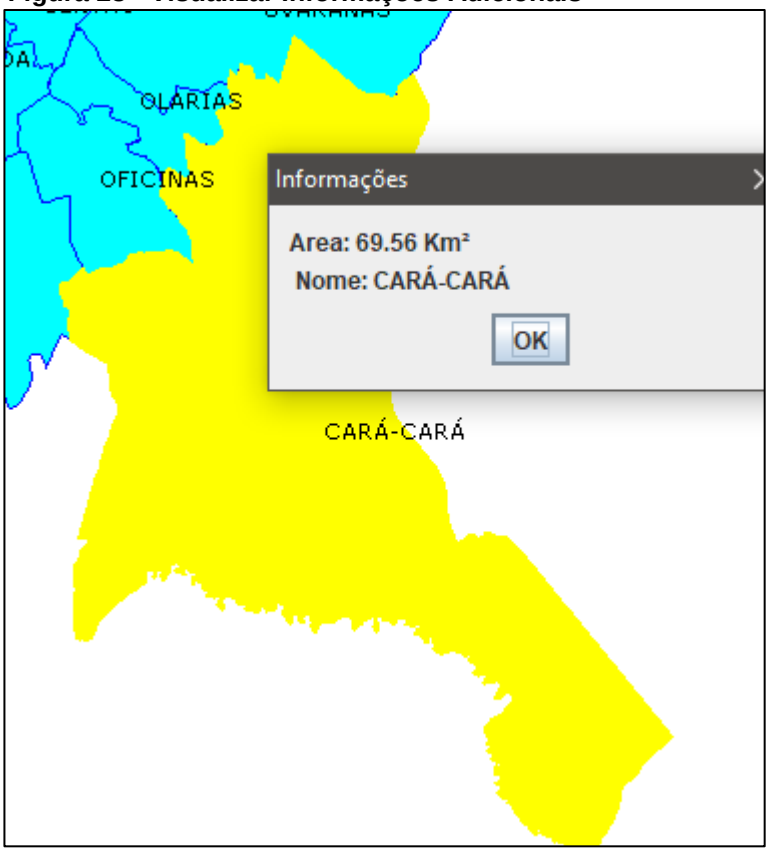
```
        featureCollection.add(criarCaracteristica(BLDR, pos, 1,typeName));
        FeatureLayer layer = new FeatureLayer(featureCollection, style);
        SimpleFeatureStore store = (SimpleFeatureStore) dataStore.getFeatureSource(
typeName );
        Transaction transaction = new DefaultTransaction("Adicionar Ponto");
        store.setTransaction( transaction );
        try {
           store.addFeatures( featureCollection );
           transaction.commit();       }
        catch( Exception eek){
           transaction.rollback();      }
        mapContent.addLayer(layer);
        jMapFrame.repaint();          }
```

Fonte: Autoria própria (2017)

5.3 SELEÇÃO DE REGIÃO E OBTENÇÃO DE INFORMAÇÕES ADICIONAIS

Como pode-se observar na Figura 28, ao se clicar em uma região a qual se deseja visualizar a informação da área total compreendida pela região, é apresentada uma janela contendo tal informação.

Figura 28 - Visualizar Informações Adicionais



Fonte: Autoria própria (2017)

Para se obter tal resultado, utilizou-se uma técnica de interpolação de regiões. Ao se clicar no mapa, gera-se um retângulo 5x5 *pixels* a partir das coordenadas do *click*. Esse retângulo é utilizado para geração de um filtro que será aplicado na seleção das características que contém o retângulo. Como pode ser verificado no Quadro 23.

Quadro 23 – Implementação do Filtro Contendo a Região Selecionada

```
    void visualizarInfo(MapMouseEvent ev) throws IOException, CQLException {
        Point coordenadasClick = ev.getPoint();
        Rectangle retanguloAux = new Rectangle(coordenadasClick.x-2,
                                    coordenadasClick.y-2, 5, 5);

        AffineTransform screenToWorld =
jMapFrame.getMapPane().getScreenToWorldTransform();
        Rectangle2D worldRect =
screenToWorld.createTransformedShape(retanguloAux).getBounds2D();
        ReferencedEnvelope bbox = new ReferencedEnvelope(worldRect,
              jMapFrame.getMapContent().getCoordinateReferenceSystem());

        Filter filtro = CQL.toFilter("BBOX(geom, " + bbox.getMinX() +
        ", " + bbox.getMinY() + ", " + bbox.getMaxX() + ", " + bbox.getMaxY()
        + ")");
```

Fonte: Autoria própria (2017)

O Quadro 24 apresenta o código utilizado para a aplicação do filtro na seleção da característica selecionada no momento do *click*. A função visualizarCaracteristicaSelecionada define um estilo diferente para a região selecionada.

Por fim a função apresenta a informação da área total compreendida pela região selecionada, através de uma função disponibilizada pela biblioteca GeoTools denominada getArea, que utiliza a geometria da característica selecionada para calcular a área.

Quadro 24 - Utilização do Filtro para Selecionar Característica

```
        try {
            SimpleFeatureCollection featureSelec =
featureSource.getFeatures(filtro);

            Set<FeatureId> IDs = new HashSet<>();
            try (SimpleFeatureIterator iter = featureSelec.features()) {
               while (iter.hasNext()) {

                  SimpleFeature feature = iter.next();
                  IDs.add(feature.getIdentifier());
                  System.out.println("   " + feature.getIdentifier());
                  Geometry geom = (Geometry) feature.getDefaultGeometry();

                  visualizarCaracteristicaSelecionada(IDs);
                  DecimalFormat df = new DecimalFormat("#.##");
                  JOptionPane.showMessageDialog(jMapFrame, "Area: "
                        + df.format(geom.getArea()/1000000)+" Km²\n Nome: "
                        +(String) feature.getAttribute("nome"), "Informações",
                        JOptionPane.PLAIN_MESSAGE);
                  break;
               }
            }
```

Fonte: Autoria própria (2017)

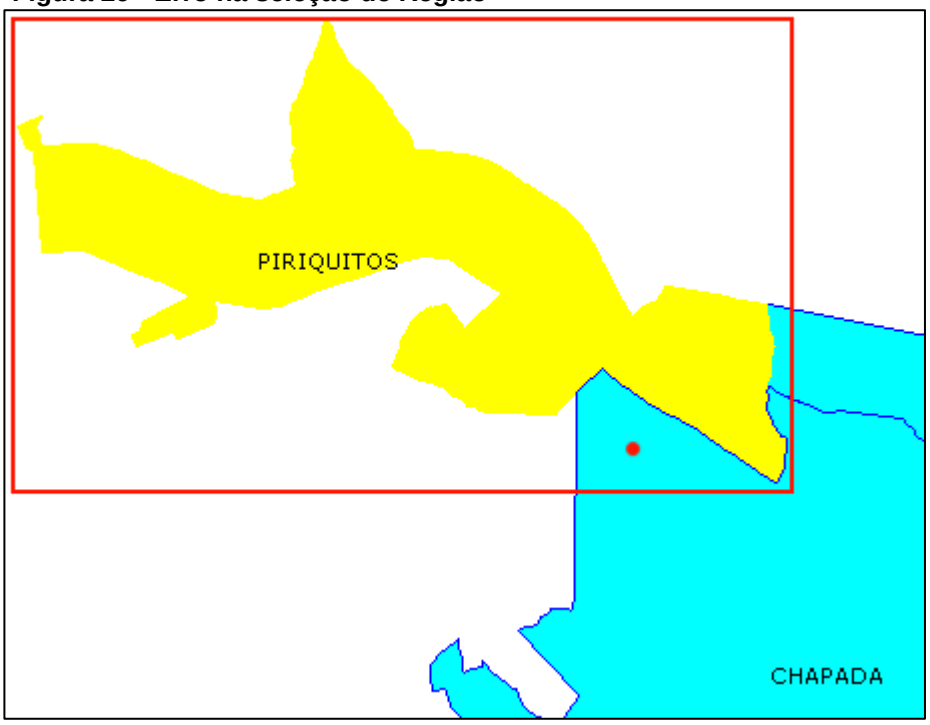
Para se desenvolver esta funcionalidade foi utilizado como base o código proveniente da seção de tutoriais do *site* da biblioteca GeoTools. Os tutoriais apresentam apenas rotinas para a implementação de funcionalidades a partir da importação de arquivos *shapefile*, se fazendo necessária uma adaptação do código para utilizar dados provenientes do banco de dados PostGIS. Tal modificação levou a uma divergência de resultados.

Para se selecionar uma área quando a origem dos dados geográficos é um arquivo *shapefile*, o *click* dentro de uma região específica leva à seleção desta área. No caso da implementação com dados provenientes do PostGIS, o *click* dentro de uma área não necessariamente leva a sua seleção. Isto ocorre quando a aplicação utiliza os dados oriundos do PostGIS. Cada elemento do mapa possui um retângulo em seu entorno que compreende a região por ele representado.

Uma ilustração do problema se encontra na Figura 29. Apesar de o *click* ter sido realizado no ponto vermelho dentro da área compreendida pela região

denominada Chapada, por estar dentro do retângulo da região Piriquitos, esta foi a região selecionada.

Figura 29 - Erro na seleção de Região



Fonte: Autoria própria (2017)

5.4 OBTENÇÃO DE INFORMAÇÕES A PARTIR DA INTERPOLAÇÃO DE CAMADAS

Adicionando-se duas camadas distintas ao modelo, se torna possível realizar operações de interpolação de camadas, buscando por exemplo a incidência de pontos em determinada região.

O Quadro 25 apresenta o código desenvolvido para determinar a quantidade de pontos incidentes sobre determinada região selecionada. Tal como na implementação da funcionalidade de seleção, o objeto *feature* contém a característica selecionada no momento do *click* do *mouse* na tela, este objeto é então utilizado para realizar filtro e subsequentemente uma *query* para determinar a quantidade de pontos presentes na área abrangida pela região selecionada.

Quadro 25 - Código Quantidade de Pontos em região

```
   SimpleFeature feature = iterator.next();
                IDs.add(feature.getIdentifier());
                visualizarCaracteristicaSelecionada(IDs);
                try {
                   Geometry geometry = (Geometry) feature.getDefaultGeometry();
                   if (!geometry.isValid()) {
                      // desconsiderar geometria inválida
                      continue;
                   }
                   Filter innerFilter = ff.intersects(ff.property(geomName2), ff.literal(geometry));
                   Query innerQuery = new Query(typeName2, innerFilter,
Query.NO_NAMES);
                   SimpleFeatureCollection join = featureSource2.getFeatures(innerQuery);
                   int size = join.size();
                   max = Math.max(max, size);
                   nomeFeature = (String) feature.getAttribute("nome");
                } catch (Exception skipBadData) {
                }
```

Fonte: Autoria própria (2017)

A Figura 30 exemplifica a utilização desta funcionalidade. Onde se é possível observar que na região denominada Neves se encontram 2 pontos do tipo Educação Infantil.

Figura 30 - Incidência de Pontos sobre Região

Fonte: Autoria própria (2017)

5.5 CONSIDERAÇÕES DO CAPÍTULO

Neste capítulo foram apresentados exemplos de recursos providos pela ferramenta GeoTools quando utilizando como fonte de dados uma instancia de um banco de dados PostGIS. Cada seção apresentou um recurso específico, apresentando trechos de código e imagens do resultado da execução destas implementações.

As funcionalidades abordadas durante o capítulo foram: aplicação de estilos às informações apresentadas na tela, seleção de características com base em clique, inclusão de informações do tipo ponto à base de dados e obtenção de informações com base em uma interpolação de duas camadas.

6 CONSIDERAÇÕES FINAIS

Este capítulo está dividido em duas seções, a primeira seção aborda as conclusões sobre o presente trabalho. A segunda seção apresenta os possíveis trabalhos futuros complementando o trabalho atual.

6.1 CONCLUSÕES

Partindo-se do objetivo de se demonstrar as funcionalidades presentes em um SIG, a biblioteca GeoTools foi escolhida para ser analisada e, com base nos estudos realizados, desenvolver a implementação de um projeto que contenha algumas funcionalidades que demonstrassem a utilização de conceitos relacionados ao armazenamento e a manipulação de dados geográficos.

Com o intuito de se demonstrar as vantagens do armazenamento das informações geográficas em uma base de dados se comparado ao armazenamento em arquivos, escolheu-se o BDG PostGIS para servir como sistema gerenciador das informações geográficas, provendo integração entre a aplicação e os dados.

Objetivando estabelecer maior integração entre o estudo realizado e um contexto relevante com a região dos Campos Gerais a partir de uma base de dados geográfica da cidade de Ponta Grossa, cujas informações foram utilizadas para serem manipuladas como experimento do projeto, tais como, a área total da cidade, a subdivisão em bairros, informações sobre os pontos de saúde e educação, definição de localização, tamanho e curso dos rios e arroios da cidade.

Os dados em questão estão disponíveis no *site* do portal de informações geográficas da cidade de Ponta Grossa em arquivos no formato *shapefile,* onde cada arquivo possuí informações a respeito de um aspecto da cidade, como por exemplo, existe um arquivo para o armazenamento dos dados sobre a área total da cidade de Ponta Grossa e outro arquivo armazena as informações sobre os pontos de saúde. Os arquivos foram então importados para uma base de dados PostGIS para atender aos objetivos propostos.

Sendo o BDG PostGIS uma extensão para o SGBD PostGreSQL, se faz necessário habilitar a operação desta extensão em uma instância de um banco de dados PostGreSQL convencional. A partir da execução dos comandos responsáveis

pela habilitação da extensão PostGIS, a base de dados foi disponibilizada para manter dados de natureza geográfica.

Partindo-se de um ambiente configurado para a implementação de soluções em linguagem Java, os passos para se construir uma aplicação que utilize a biblioteca GeoTools são claros e acessíveis, por contar com uma comunidade ativa em fóruns e sua documentação sempre atualizada com tutoriais e definições de ferramentas.

Utilizando a documentação oficial da ferramenta GeoTools, assim como ajuda da comunidade desenvolvedora em fóruns, foram desenvolvidas funções que manipulam as informações geográficas da cidade de Ponta Grossa armazenadas no banco de dados PostGIS. As funcionalidades desenvolvidas e descritas neste trabalho foram: definição de estilo para mapas; inclusão de informações do tipo ponto para visualização e armazenamento na base de dados; visualização de informações adicionais de uma região selecionada, como a área total compreendida pela região e a quantidade de pontos presentes na região.

Tendo em vista as funcionalidades desenvolvidas utilizando as ferramentas propostas, conclui-se que o objetivo para esse trabalho foi alcançado: Integrar um modelo de banco de dados geográfico com a biblioteca GeoTools, utilizando seus principais recursos de manipulação e análise de dados espaciais.

6.2 TRABALHOS FUTUROS

Com base nos resultados obtidos com o presente trabalho, apresentam-se sugestões para realização de trabalhos que podem complementa-lo.

- Maior aprofundamento na implementação da funcionalidade de seleção de características a partir de um clique no mapa utilizando o GeoTools integrado ao PostGIS. Onde ocorre que, após o clique de seleção, a área selecionada não corresponde a área compreendida pelo clique do *mouse*. Diferente da integração entre o GeoTools e arquivos no formato *shapefile,* onde a seleção acontece normalmente;
- Ampliação da personalização dos estilos dos mapas com base nas características dos elementos geográficos;

- Estudo de outras funcionalidades providas a partir da interpolação de camadas, como por exemplo a possibilidade de conseguir a quantidade de linhas presentes em determinada região, bem como seus comprimentos;
- Implementar a funcionalidade de inclusão de regiões e linhas para determinadas coordenadas.

REFERÊNCIAS

CASANOVA, Marco; CÂMARA Gilberto; DAVIS Clodoveu; QUEIROZ, Gilberto R. de; VINHAS, Lúbia. **Bancos de Dados Geográficos**. Curitiba: MundoGEO, 2005.

DAVIS JR., C. A.; LAENDER, A. H. F. Extensões ao modelo OMT-G para produção de esquemas dinâmicos e de apresentação. In: Workshop Brasileiro de GeoInformática, 2, São Paulo. **Anais…** GeoInfo, 2000, p. 29-36.

GEOPORTAL PONTA GROSSA. Ponta Grossa, 2016. Disponível em: < http://geo.pg.pr.gov.br/portal/ >. Acesso em 05 jun. 2016, 22:00.

GEOTOOLS. GeoTools The Open Source Java GIS Toolkit GeoTools v2.6 documentation; Disponível em: < http://www.geotools.org/> Acesso em: 19 ago. 2017.

HALL, Brent; LEAHY, Michael G. **Open Source Approaches to Spatial Data Handling**. Berlin: Springer-Verlag Berlin Heidelberg, 2008.

HWANG, Byungyeon; BYUN, Taekyung; MOON, Songchun. Spatial Query Processing in Geographic Database Systems. In: EUROMICRO 94. System Architecture and Integration. Proceedings of the 20th EUROMICRO Conference, 1994, Liverpool, England. **Anais…** Los Alamitos: IEEE, 1994. p. 53-60.

OBE, Regina O.; HSU, Leo S. **PostGIS in Action**. Stamford: Manning Publications, 2011.

POSTGIS. **Spatial and Geographic objects for PostgreSQL**. Disponível em: <http://postgis.net/>Acesso em: 18 ago. 2017.

QABAJA, Hamzeh; BIKDASH, Marwan. Use of GIS Data to Model the Vulnerability of the Emergency System to Attacks. In: Southeastcon, 2013 Proceedings of IEEE. **Anais…** Jacksonville: IEEE, 2013. p. 01-06.

RIGAUX, P.; SCHOLL, M.; VOISARD, A. **Spatial Databases with Application to GIS**. San Francisco: Morgan Kaufman, 2002.

SHALETRAK: ROYALTY MANAGEMENT SOLUTIONS. **Oil and Gas Mineral Management Solutions**. Disponível em: <http://www.shaletrak.com/services/> Acesso em: 29 mai. 2016.

TERRAVIEW 4.1.0. São José dos Campos, SP: INPE, 2010. Disponível em: < www.dpi.inpe.br/terraview >. Acesso em: 04 jun. 2016, 22:00.

WORBOYS, Michael F., **GIS: A Computing Perspective**. UK: Taylor & Francis, 2003.