UNIVERSIDADE DO GRANDE RIO
PROF. JOSÉ DE SOUZA HERDY
ESCOLA DE CIÊNCIA E TECNOLOGIA
Bacharelado em Sistemas de Informação

Gustavo Matos de Mello

Paulo Collares Moreira Neto

Utilizando GeoTools para Manipulação de Dados Geográficos

Silva Jardim

2009

UNIVERSIDADE DO GRANDE RIO
PROF. JOSÉ DE SOUZA HERDY
ESCOLA DE CIÊNCIA E TECNOLOGIA
Bacharelado em Sistemas de Informação

Gustavo Matos de Mello

Paulo Collares Moreira Neto

Utilizando GeoTools para Manipulação de Dados Geográficos

Projeto Final de Curso apresentado como requisito final para conclusão do curso de Bacharelado em Sistemas de Informação da Universidade do Grande Rio "Prof. José de Souza Herdy" (UNIGRANRIO).

Orientador: Prof. Alessandro Cerqueira

Silva Jardim

2009

Utilizando GeoTools para Manipulação de Dados Geográficos

Gustavo Matos de Melo

Paulo Collares Moreira Neto

Projeto Final de Curso apresentado como requisito final para conclusão do curso de Bacharelado em Sistemas de Informação da Universidade do Grande Rio "Prof. José de Souza Herdy" (UNIGRANRIO).

Aprovado em dezembro de 2009.

BANCA EXAMINADORA

Prof. M.Sc. Alessandro Cerqueira- Orientador

Silva Jardim

2009

Gustavo Matos de Mello,

Paulo Collares Moreira Neto

Uma aplicação utilizando GeoTools para manipulação de dados geográficos [Silva Jardim] 2009

XII, 46 p. 29,7 cm. (Escola de Ciência e Tecnologia, 2009)

Projeto de Final de Curso - Universidade do Grande Rio, Escola de Ciência e Tecnologia.

1. Geoprocessamento
2. Programação
3. GeoTools

I. EIN/UNIGRANRIO II. Título (série)

Dedicamos este trabalho a todos aqueles que nos ajudaram de forma direta ou indiretamente, aos amigos e familiares, professores e funcionários, colegas de turma e aqueles que por motivo de força maior não nos acompanharam até o fim do curso.
A todos o nosso muito obrigado.

AGRADECIMENTOS

Primeiramente agradecemos a Deus, por nos proporcionar a oportunidade de iniciar e concluir este curso, pois sem Ele não teríamos condições nem forças para a realização desta etapa em nossas vidas.

À nossa família e amigos por nos apoiarem nesses quatro anos de estudo, seja financeiramente, emocionalmente ou com simples palavras de apoio, pois foi com muitas lutas, mas também com muitas vitórias que estamos concluindo este curso.

Aos professores que compartilharam seus conhecimentos e experiências para que, a partir de hoje, pudéssemos trilhar nossos próprios caminhos rumo a um futuro promissor.

A esta universidade, e aos seus funcionários, que estavam sempre dispostos a resolver nossos problemas e que preparavam sempre um ambiente digno de estudo.

Por fim aos colegas de classe, afinal trilhamos este caminho juntos, obrigado a todos que nos deram forças para continuar, sempre com uma palavra amiga e que aos que sempre estavam dispostos a nos ajudar nas horas das dúvidas mais cruéis, desejamos a todos muito sucesso.

A todos estes o nosso muito obrigado.

“O único lugar onde o sucesso vem antes do trabalho
é no dicionário.”

Albert Einstein

RESUMO

A partir da necessidade de controle de dados georreferenciados construir-se-á um aplicativo para atender as funcionalidades de um sistema SIG para Java *desktop*, ou seja, um aplicativo para leitura e manipulação de dados geográficos. Será possível o acesso a *shapefiles* ou a bancos de dados, pois o aplicativo oferece essas duas funcionalidades. O aplicativo desenvolvido disponibiliza um conjunto de ferramentas que auxilia na manipulação dos mapas. De acordo com os objetivos do atual projeto, conclui-se que o aplicativo será de grande utilidade na manipulação de dados geográficos.

Palavras-chave: Geoprocessamento, Shapefile, SIG.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SIG	Sistema de Informações Geográficas
GIS	Geographic Information System (Sistema de Informações Geográficas)
SAD	South American Datum (Datum da América do Sul)
GPS	Global Positioning System (Sistema de Posicionamento Global)
WGS	World Geodetic System (Sistema Geodésico Mundial)
SIRGRAS	Sistema de Referência Geocêntrico para a América do Sul
SRIDs	Spatial Reference Identifiers (Identificador de Referência Espacial)
CAD	Computer Aided Design (Desenho Assistido por Computado)
ESRI	Environmental Systems Research Institute (Instituto de Pesquisa de Sistemas Ambientais)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
CCG	Center for Computational Geography (Centro de Geografia Computacional)
GAM	Geographical Analysis Machine (Máquina de Análise Geográfica)
OGC	Open Geospatial Consortium (Consórcio Geoespacial Aberto)
WMS	Web Map Service (Serviço de Mapa para Web)
CQL	Common Query Language (Linguagem de Consulta Comum)
JAI	Java Advanced Imaging (Manipulação de imagens avançadas em Java)
API	Application Programming Interface (Interface de Programação de Aplicativos)
CEUB/ICPD	Instituto CEUB de Pesquisa e Desenvolvimento

1 -Introdução

GIS (*Geographic Information Systems* ou Sistemas de Informações Geográficas) são aplicativos que tratam de informações com representação no espaço geográfico dispostas em camadas ("*layers*"), as quais são combinadas para a criação de mapas. Uma camada contém várias feições espaciais (entidades que representam pontos, linhas ou polígonos).

Este projeto descreve o desenvolvimento de um *software desktop GIS* que viabilize a utilização de *shapefiles* para a manipulação de dados georreferenciados.

A partir do que foi exposto, o principal objetivo deste trabalho é estudar o desenvolvimento de uma aplicação SIG com enfoque na disponibilização de dados, sendo desenvolvida através da linguagem de programação Java com o auxílio da biblioteca GeoTools.

É importante ressaltar a utilização de tecnologias livres, programas com código fonte aberto, representa uma grande vantagem no desenvolvimento de uma aplicação SIG, pois possibilita a redução do valor final do aplicativo.

No mundo tecnológico atual, a aquisição de equipamentos, tais como computadores, *PDAs*, *notebooks*, e GPS estão ao alcance de qualquer ser humano que busca conhecimentos científicos para seu crescimento profissional. Sendo assim, com um pequeno investimento, o pesquisador desta área poderá se beneficiar com a facilidade e comodidade que o geoprocessamento oferece ao seu usuário.

1.1 -Motivação

A tarefa de buscar soluções e visualizar de forma ampla situações geográficas exige múltiplas ações na manipulação de dados para buscar soluções que funcionem com praticidade no mundo moderno.

A partir desses fatos, criamos um aplicativo que será capaz de manipular dados geográficos, essa aplicação é um bom exemplo na utilização as tecnologias de informática que estão relacionadas a geoprocessamento, para melhor atingir os objetivos da necessidade de melhor entendimento dos benefícios da cartografia digital.

A integração do geoprocessamento motiva o desenvolvimento do atual projeto, o qual promove a união de tecnologias modernas que atinjam com facilidade visualização e

manipulação dos dados nele carregados, visando a criação de um Sistema de Informações Geográficas.

Baseando-se em estudos a respeito do geoprocessamento, percebe-se que a sociedade está, cada vez mais, integrada a estudos relevantes em relação à pesquisa e utilidade do mesmo. Haja vista, que até o Governo Federal vem apoiado significativamente programas para empresas públicas e privadas, para que acompanhe o desenvolvimento tecnológico e geotecnológico.

O que promove o estímulo e desejo de desenvolver o projeto é o incessante empenho de melhorar a formação profissional com o prazer e estar desenvolvendo programas visto que, esse é um dos principais motivos que levaram a escolha do curso "Sistemas de Informação".

1.2 - Objetivos e escopo do trabalho

Desenvolver uma aplicação onde é possível manipular dados georreferenciados armazenados em um *shapefile* ou em um banco de dados, representando as informações neles contidas.

1.3 -Organização do Trabalho

Este trabalho está organizado em 8 capítulos. No capítulo 2 são apresentados os conceitos introdutórios de geoprocessamento, detalhamos a cartografia, coordenadas geográficas e o *datum*. No capítulo 3 nos aprofundamos mais no geoprocessamento, desde um breve histórico até algumas noções de SIG. No capítulo 4 detalhamos o que são *shapefiles* e como estão organizados. No capítulo 5 falamos sobre o PostGIS, desde a sua instalação, passando pela criação de *scripts* SQL a partir de *shapefiles*, até a manipulação de bancos para armazenar dados geográficos. No capítulo 6 detalhamos o GeoTools, sua criação e os padrões usados em sua implementação. No capitulo 7 colocamos em prática os capítulos anteriores e exemplificamos implementações usando o GeoTools. Por fim, no capítulo 8 são apresentadas as conclusões do projeto e temas para trabalhos futuros.

2 - Conceitos Introdutórios de Geoprocessamento

2.1 - Cartografia

A superfície terrestre ao contrário do que muitos pensam não é redonda, e sim uma superfície de curvas irregulares, contudo sua forma se aproxima de um elipsóide, então, por prática aproxima-se a terra de um elipsóide de revolução. (FUCK JR., 2009)

De acordo com Fuck Jr. (2009), conceitua-se elipsóide de revolução como uma superfície utilizada como referência para cálculos de posicionamento, verificação de distância, direções, em geral, elementos de geométricos de mensuração.

Sendo a superfície terrestre completamente composta por curvas irregulares, seria impossível criar uma reprodução plana desta superfície sem desfigurá-la ou alterá-la. Desse modo, os mapas existentes são representações aproximadas da superfície terrestre.

2.1.1 - Elipsóide

A superfície terrestre, por ser completamente irregular (FUCK JR., 2009), apresenta valores geodésicos diferentes, para o uso de elipsóides, marcados em muitos pontos da Terra, por isso, cada país ou região adotou um elipsóide que se encaixe melhor em suas dimensões, ou seja, que se ajuste melhor ao geóide, que tenha uma melhor aproximação. Geóide, um modelo físico aproximado do formato real da terra. De acordo com a fig. 1, a linha preta está representando o elipsóide, e a vermelha, o geóide. Assim é possível observar as irregularidades do planeta terra, mesmo utilizando um elipsóide que mais se aproxime de seu formato real.

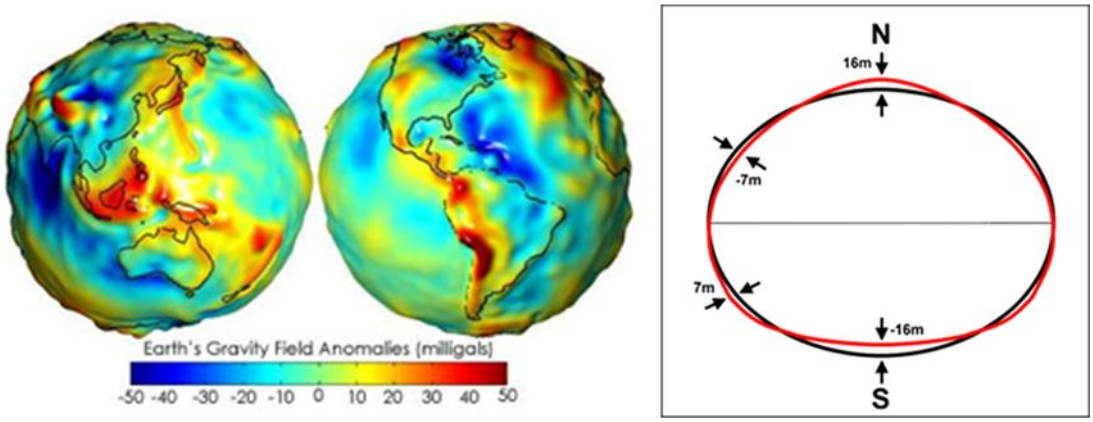

Figura 1: Representação real da superfície terrestre (FUCK JR., 2009)

No caso do Brasil, foi adotado o elipsóide de *Hayford,* pois suas dimensões ficaram mais apropriadas para a América do Sul. No entanto, de acordo com a Associação Internacional de Geodésia, é utilizado com mais frequência o elipsóide da União Astronômica Internacional.

2.2 -Coordenadas geográficas

Cada ponto localizado no globo terrestre está na intercessão de um meridiano e um paralelo, ou seja, uma latitude e uma longitude, que segue por base duas linhas, o equador e o meridiano principal, como podem ser exemplificadas na fig. 2 (CEUB/ICPD, 2009).

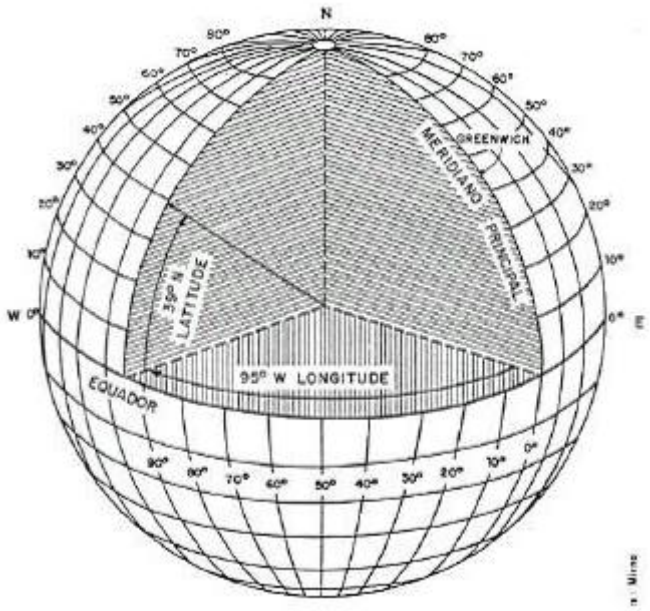

Figura 2 - Exemplo de coordenadas geográficas (CEUB/ICPD, 2009)

A latitude é a distância em graus, minutos e segundos, a partir da linha do Equador, que percorre a terra horizontalmente (CEUB/ICPD, 2009). Este valor varia de 0 a 90° tanto para o sul quanto para o norte. Este ângulo é formado pelas linhas que partem do Equador até o paralelo onde se localiza o ponto em questão.

A longitude é a distância também em graus, minutos e segundos, mas à partir do Meridiano de Greenwich, este valor varia de 0 a 180° tanto para oeste quanto para leste. Este ângulo é formado pelas linhas que partem do meridiano principal até o meridiano onde está o ponto a ser localizado.

2.3 -Datum

Datum, do latim dado, é um modelo da representação da superfície da Terra na forma plana (FORTES, 1993), ou seja, encarrega-se de representar um corpo curvo em três dimensões, num plano em duas dimensões, porém, mantendo perpendiculares seus meridianos e paralelos.

Naturalmente não é possível representar um elipsóide em um plano, para garantir uma representação gráfica proporcional, existem diferentes datum dependendo da localização e do tamanho do objeto a ser representado. Cita-se como exemplo, o ponto de projeção da representação do Japão, não está no centro da terra, mas em um ponto sobre o próprio Japão, isto se deve, principalmente devido ao seu tamanho. A sua curvatura no globo é bem pequena, assim quando transportado para um plano, sua variação de tamanho é relativamente pequena, o mesmo não ocorre com países maiores, como Rússia, Estados Unidos ou até mesmo o Brasil.

Os mapas mais antigos do Brasil adotavam o datum planimétrico Córrego Alegre (FORTES, 1993). A partir da década de 60, passou a ser utilizado o datum planimétrico SAD - 69. Atualmente, com o surgimento do GPS, tem sido comum o emprego do World Geodetic System (WGS), o mais usado é o WGS 84 (criado em 1984, mas revisado em 2004). A origem das coordenadas do WGS 84 está localizada no centro da terra, dando uma projeção igualitária para todo o mundo. Outro datum importante é o SIRGRAS (Sistema de Referência Geocêntrico para a América do Sul), iniciado em outubro de 1993, no Paraguai. Atualmente, este é o datum oficial do Brasil, porém, o SAD 69 poderá ser utilizado até 2014.

Cada instância espacial tem um SRID (*Spatial Reference Identifiers* ou Identificadores de referência espacial) único. O SRID faz referência a um local específico no globo terrestre (SPATIAL REFERENCE, 2009).

3 -Geoprocessamento

3.1 -Histórico

Segundo Gomes (2005), em Londres no ano de 1854, surgiu uma grave epidemia de cólera na capital inglesa em uma época que não se conhecia sua fonte de transmissão, mais de 500 mortes já havia sido registrada. O doutor John Snow teve uma idéia repentina em marcar em um mapa da cidade à localização das pessoas contaminadas pela doença e dos poços de água localizados na cidade.

A partir da interpretação dos dados demarcados no mapa o doutor Snow constatou que as maiorias dos casos estavam concentradas nas proximidades do poço da Broad Street. Ao interditar esse poço houve uma grande redução dos novos casos da doença. Com o passar do tempo comprovou-se que a cólera é transmitida através da ingestão de água contaminada.

O mapa do doutor Snow ficou para história como um exemplo de que é possível fazer uma análise com cruzamento de dados, mostrando bem o poder explicativo da análise espacial, como exemplificado na Figura 3.

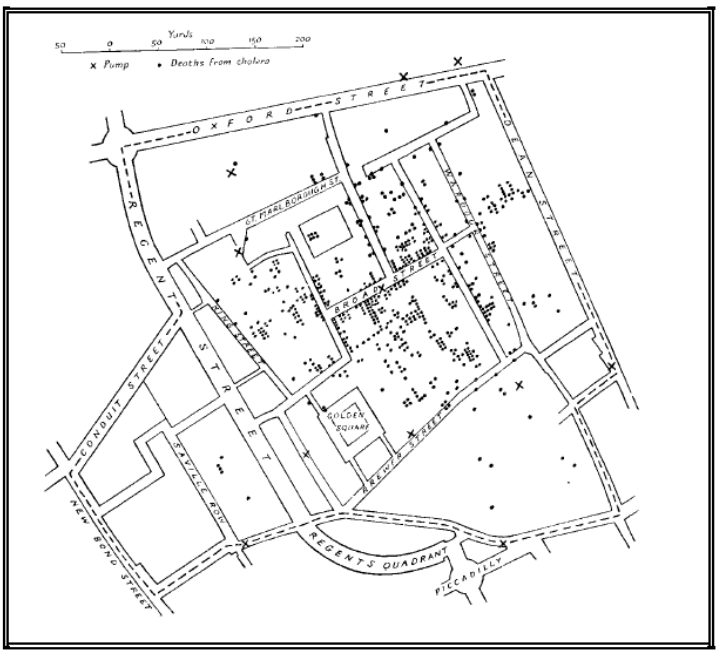

Figura 3 - Mapa de Londres. Pontos- Casos de cólera. Cruzes- Poços de água. (Adaptado de TUFTE, 1983).

3.2 -Definição

Os mapas estão sendo utilizados há muitos anos, com objetivo de registrar informações espaciais para atividades de interesse humano, assim como, a apresentação e comunicação de informações geográficas (GOMES, 2005). A representação do espaço tem evoluído paralelamente com a evolução tecnológica humana, a cartografia foi marcada por um grande avanço, evoluindo-se para resolver assuntos de interesses militares, levantamento de recursos naturais e controle de meio ambiente. O processamento de informações geográficas vem assumindo um papel estratégico de planejamento, administração e pesquisa de municípios, cidades e regiões.

Mapas apresentados em folha, contendo diversas informações, era uma forma que gerava muitas limitações para manipular informações geográficas. Unindo avanços recentes de diversas áreas tecnológicas – fotogrametria, banco de dados, sensoriamento remoto, computação gráfica, CAD (*Computer Aided Design*) – unidas a disciplinas que lidam com questões espaciais, como, Geodésica, Urbanismo, Geometria e Cartografia, criou-se uma área de conhecimento multidisciplinar conhecida como Geoprocessamento, nome que se tornou usual no Brasil.

O Geoprocessamento utiliza um conjunto de técnicas computacionais para o processamento de dados geográficos, que inclui a sua coleta, armazenamento e tratamento das informações georreferencias. Estas informações estão sempre relacionadas a uma posição geográfica, ou seja, a sua localização no globo terrestre, que pode ser apenas um ponto, uma localização única, uma seqüência de pontos, formando uma estrada, por exemplo, ou pontos formando um polígono, delimitando uma área.

Visando a questão tecnológica, o geoprocessamento utiliza intensivamente tecnologia computacional. Pode se considerar como um grande apoio no crescimento do Geoprocessamento o desenvolvimento em grande escala dos componentes tecnológicos – *hardware* e *software* – unidos a avanços em computação gráfica, banco de dados, o aumento da capacidade computacional adicionado a redução de custos, o que ocasionou uma participação significativa no avanço da área de geoprocessamento, que se baseia fortemente em tecnologia.

3.3 -SIG

Partindo da década de 80, época onde o Brasil começou trabalhos de pesquisa, conquistou um avanço significativo em estudos no que diz respeito a construção de SIGs (DIAS, 2005). Há muitas características que pode marcar a utilização desses sistemas como: facilidade de acesso, rapidez e praticidade para manipular as informações.

SIG (Sistema de informações Geográficas) são ferramentas de hardware e software com habilidades de coleta, armazenamento e processamento de informações geográficas. Através deste sistema as informações geográficas podem ser organizadas, sintetizadas, mescladas e apresentadas de forma em que o usuário final compreenda de forma clara, o que está explicitado na pesquisa desejada e que a análise desses dados possa ser feita por qualquer pessoa, e não apenas por profissionais da área.

Os SIGs além de cuidarem da organização das informações, o acesso as informações podem ser feito direto pela representação do objeto geograficamente representado. Deve-se analisar a área a ser aplicada primeiramente antes de definir a estruturação e o uso de tecnologias. Entretanto não é tão simples desenvolver um SIG, já que existe uma grande variedade de ferramentas que é disponibilizada para sua construção e também, se faz necessário o conhecimento nas áreas principais, assim como, cartografia e ciência da computação, que são áreas um tanto complexas, porém com dedicação, leitura e estudo do assunto desejado consegue-se atingir, formular e organizar projetos que favoreçam a atividade na área tecnológica desejada.

Para a construção de modelos de SIGs apropriados, ainda requer conhecimento e técnicas de modelagem. Contudo, pode se notar um grau de complexidade e partindo daí a necessidade de estudo profundo de modelos de metodologia de desenvolvimento. Diante de tantas particularidades impostas pelos diversos tipos de SIGs, os conceitos envolvidos deverão ser analisados de acordo com a área de aplicação determinada. Não devendo ocasionar desistência e sim, maiores objetivos na realização final.

4 -Shapefiles

De acordo com as especificações do *shapefile* da ESRI (1998), um *shapefile* é um formato de armazenamento digital para dados georreferenciados, voltado para utilização em softwares de sistemas de informações geográficas e atributos associados. Foi desenvolvido pela empresa ESRI, especializada em tecnologia GIS, sendo introduzido no mercado na década de 90 com o ArcView GIS. Hoje, o arquivo *shapefile* está mais popular, e já é possível ler e escrever este arquivo através de uma variedade de programas.

Shapefiles espacialmente representam geometrias: pontos, linhas, polígonos abertos e polígonos fechados. Um ponto, por exemplo, poderia representar poços de água, um polígono aberto poderia representar rios e lagos, respectivamente.

Estas representações geométricas são de uso limitado, sem atributos que especifiquem o que elas representam. Portanto, uma tabela de registros fará o armazenamento de suas propriedades, atributos para cada formato primitivo no *shapefile*. Os *shapes* (formas geométricas), unidos a esses atributos geram uma grande quantidade de representações em dados geográficos. Representações que possibilitam cálculos de precisão.

Na sua forma mais simples um *shapefile* é composto de três arquivos individuais obrigatórios para o armazenamento de dados que compreende um *shapefile*, são as extensões .shp, .shx e .dbf, que serão mencionados adiante. Existem mais oito arquivos opcionais, principalmente voltados para um melhor desempenho.

Os arquivos obrigatórios são:

- **Shp**: Os principais dados estão armazenados nesse arquivo, ou seja, os dados de referências geográficas.
- **Shx**: É um arquivo de índice utilizado para percorrer os pontos do *shapefile*.
- **Dbf**: Onde são registrados os atributos para cada forma, sendo armazenados no formato dBase, ou o opcional xBase que é variação do formato original dBase.

Opcionais:

- **Sbn**: Este é um arquivo binário de índice espacial de formato não documentado, sendo somente utilizado pelo software da ESRI. Uma vez que

o arquivo shp contém todas as informações que necessita para a análise de dados, o arquivo sbn não é necessário.

- **Prj**: Utiliza um arquivo texto para descrever as projeções usadas. Esse texto é fundamental para entender as informações contidas no shp.
- **Fbn**: Índice espacial, unicamente de leitura de recursos para *shapefile*.
- **Ain**: Tabela de atributos de um tema ou índice de atributo para um campo ativo na tabela.
- **Ixs**: Índice de geocodificação para leitura e escrita de *shapefile*.
- **Mxs**: Índice de geocodificação para leitura e escrita de *shapefile* no formato Odb.
- **Atx**: Índice de atributo para o arquivo dbf na forma de *shapefile*.
- **Cpg**: Especifica páginas de código dbf, identificando a codificação de caracteres a se utilizado. Os arquivos obrigatórios (.shp,.shx,.dbf), seguem uma sequência, isto é, o primeiro arquivo shp, corresponde ao primeiro registro no shx e no dbf e assim por diante.

5 -PostGIS

O PostGIS (REFRACTIONS, 2009) foi desenvolvido pela empresa canadense Refractions Research Inc., especializada em armazenamento de dados geoespaciais e em desenvolvimento de software personalizados. Com a experiência de mais de 10 anos no mercado, cria tecnologias com código livre e proprietário, beneficiando, assim, a todos os seus clientes.

O PostGIS adiciona suporte para processamento e armazenamento de dados geográficos ao banco de dados objeto-relacional Postgresql, possibilitando o uso de objetos GIS em aplicações que manipulam dados georreferenciados.

5.1 -Instalação

Por ser um complemento do PostgreSQL, o PostGIS necessita deste banco de dados previamente instalado e configurado para funcionar.

O PostgreSQL pode ser baixado em seu site oficial em `http://www.postgresql.org/download`. Após a sua instalação o PostGIS (`http://postgis.refractions.net/download/`), pode ser adicionado ao sistema, sua instalação, no Windows, é bem simples e intuitiva, porém deve-se prestar a atenção a dois detalhes importante para a conclusão da instalação.

O primeiro é o caminho que é requerido na instalação do PostGIS, este endereço deve ser o caminho da instalação do PostgreSQL, como por exemplo: `C:/Arquivos de Programas/PostgreSQL/8.3/`.

Outro detalhe que vale a pena ressaltar é o usuário e senha requeridos pela instalação, este deve ser o super usuário e não o de usuário normal, deste modo a instalação criará o banco de dados modelo do PostGIS, com as funções necessárias para manipular os dados georeferenciais.

5.2 -Geração de scripts SQL para carga de dados presentes nos Shapefiles

Um shapefile é composto de três arquivos diferentes e obrigatórios: .shp, .dbf e .shx (ESRI, 1998), porém a conversão destes arquivos se torna bem simples com a função shp2pgsql do PostGIS.

Primeiramente todos os três arquivos do *shapefile* devem estar em um mesmo diretório para que a função os encontre a partir do arquivo principal (.shp).

Dentro da pasta *bin*, do diretório de instalação do PostgreSQL, existe um arquivo *shp2pgsql.exe*, este é o qual iremos executar para realizar a conversão, para isso devemos rodar a seguinte linha no *prompt* do Windows:

```
C:\PostgresPlus\8.3\bin\shp2pgsql -s [SRID] [caminho do
shapefile] [nome da tabela a ser criada] > [nome do arquivo SQL]
```

Onde:

- `C:\PostgresPlus\8.3\bin\shp2pgsql`: Caminho para a instalação do PostgreSQL no computador, podendo variar o local e a versão da instalação;
- [SRID] : Define o campo SRID. Se não for especificado o padrão é -1. Neste exemplo usamos o 29100 (Datum SAD69 / Brazil Polyconic).
- [caminho do shapefile]: Referência do arquivo .shp do shapefile. Ex.: `C:\shapefiles\br_regioes\BR_regioes.shp`
- [nome da tabela a ser criada]: Nome da tabela que será criada posteriormente no banco de dados.
- [nome do arquivo SQL]: Local e nome do arquivo SQL que será criado ao final desta execução.

Exemplo:

```
C:\PostgresPlus\8.3\bin\shp2pgsql -s 29100 C:\shapefiles\
br_regioes\BR_regioes.shp br_regioes  > C:\shapefiles\arquivo.sql
```

5.3 -Criando um novo banco para armazenar dados geográficos

Para criar um novo banco de dados do PostgreSQL, clique com o botão direito em Bancos de Dados, e em seguida em “Novo banco de dados...”.

Na janela que se abre informe o nome do banco a ser criado e no campo Modelo, selecione a opção *template_postgis*, para que este banco importe as funções do PostGIS, como ilustra a fig. 4. Vale lembrar que o banco *template_ postgis* não pode estar sendo usado, senão ocorrerá um erro e o novo banco não poderá ser criado.

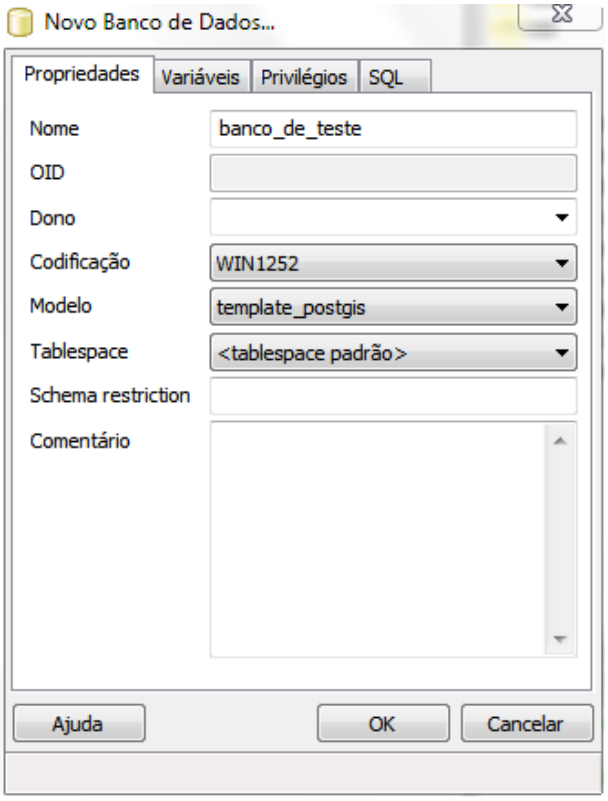

Figura 4 - Criando tabelas e inserindo os dados

A criação das tabelas e a inserção dos dados são bem simples, pois o PostGIS já criou o arquivo SQL contendo todas estas informações.

Basta abrir o arquivo, no PostgreSQL, criado com o comando *shp2pgsql* anteriormente, e rodá-lo para que a tabela seja criada e os dados inseridos.

5.4 -Função AddGeometryColumn

A função *AddGeometryColumn*, é uma função do PostGIS (importada para o banco atual do modelo *template_ postgis*) esta função adiciona um campo a tabela recém criada, chamada *the_geom*, este campo guardará as coordenadas geográficas. É passado como parâmetro nesta função, entre outros, valores como o SRID, informado na execução do comando `shp2pgsql`, e a forma geométrica dos dados inseridos, no nosso exemplo é o MULTIPOLYGON. Esta informação é gerada automaticamente de acordo com o *shapefile* utilizado.

6 -GeoTools

6.1 -Como surgiu

Segundo Hall (2008), GeoTools é uma biblioteca open source codificada em Java. Essa biblioteca fornece métodos para manipulação de dados geoespaciais, que são utilizados na implementação de Sistemas de Informação Geográfica (SIG).

A Sun lançou a primeira versão pública de Java em 1995 (HALL,2008), no mesmo ano em que o desenvolvedor do GeoTools, James Macgill, iniciou seu mestrado em SIG na Universidade de Leeds, no Reino Unido. Em 1996, tornou-se evidente que a libertação do Java, estava vindo para mudar a face da navegação na Web. Contudo, a plataforma também estava sendo utilizado por Macgill para visualizar resultados na detecção de cluster espacial. Com um crescente número de projetos no Centro de recém-formados pela Center for Computational Geography (CCG) da Universidade de Leeds, tornou-se necessário o mapeamento, o espectador Macgill, que simplesmente o havia construído visando o sucesso na conclusão de seu doutoramento, a pesquisa foi ampliada e o projeto GeoTools nasceu.

Uma das áreas de interesse inicial na investigação do CCG foi a detecção de cluster espacial. Este é o processo onde um programa é executado para determinar se a contaminação de uma forma específica de câncer, por exemplo, se é agrupado no espaço ou não. No início foram utilizados métodos como a máquina de análise geográfica (GAM). Eram relativamente fáceis de usar, os métodos mais complexos com base em flocagem boids estavam sendo investigadas. Isso levou Macgill a desenvolver um sistema de mapeamento básico em Java que pode ler e exibir shapefiles representados geograficamente, e ao mesmo tempo ser capaz de mover representações geométricas sobre a superfície do mapa com rapidez e facilidade.

Desde o início do projeto GeoTools, os desenvolvedores consideraram a fonte do código, algo que deve ser desenvolvido e compartilhado abertamente. O GeoTools nasceu em um ambiente universitário, e como as universidades da Grã-Bretanha na década de 1990 focaram-se estreitamente nas comercializações de pesquisas, o licenciamento de software real teve inúmeras implicações legais e administrativas. Assim, foi decidido dar ao GeoTools uma comunidade de desenvolvimento livre, permitindo-lhe atingir seu potencial sem ser envolvido em burocracia.

O interesse principal foi, e continua a sendo, produzir resultados consideráveis. Isto era extremamente difícil de conseguir no campo emergente da ciência computacional, na geografia geral e computacionais, em particular durante a década de 1990. GeoTools foi escrito em parte para evitar este problema, permitindo que todos baseiem suas experiências no mesmo código.

6.2 -Código aberto

Os sub-tópicos a seguir descrevem três tipos específicos de códigos aberto que os desenvolvedores buscaram apoiar no desenvolvimento do GeoTools, ou seja, padrões abertos.

6.2.1 -Open Standards

Open Standards é aquele que é livre para todos os desenvolvedores, podendo ser lido e implementado.

No desenvolvimento de informações geográficas são estabelecidos padrões, os mais utilizados são os desenvolvidos pela Open Geospatial Consortium (OGC).Os desenvolvedores do GeoTools decidiram incorporar o máximo de padrões OGC possíveis. Isso permite que programas que utilizem GeoTools, possam interoperar com outros programas que também implementarem essas normas. Os primeiros padrões que foram desenvolvidos pela OGC que o GeoTools fez parceria foram os relacionados ao mapeamento web, com o Web Map Service (WMS). O projeto GeoTools, a alguns anos atrás apoiou o GeoAPI e o projeto OGC, que estão desenvolvendo um conjunto de interfaces Java para a implementação de aplicações de interfaces de programação (API), para a criação de uma melhor interoperabilidade entre o Java e os diferentes projetos de base geográfica.

6.2.2 -Open Source

Open Source refere-se a códigos de programação que são disponibilizados para outros programadores aperfeiçoarem seu conteúdo, permitindo a identificação de possíveis erros ou até mesmo que o código seja reescrito de uma maneira mais simples e eficiente, gerando benefício a outros desenvolvedores e usuários. Umas grandes partes das licenças Open Source permitem a contribuição de desenvolvedores com suas alterações para que outros também possam usufruir com suas melhorias. No caso do GeoTools, os

contribuidores responsáveis por mudança permanente no código oficial, são convidados a ter participação nas próximas versões.

6.2.3 -Open Science

A ciência sempre se apoiou na capacidade computacional para a tomada de decisões. Com isso, fica cada vez mais complicado o repasse de experiências realizadas. Cita-se como exemplo, uma mesma experiência realizada por dois grupos trabalhando com programas computacionais distintos. Torna-se difícil verificar diferenças encontradas se ambos os grupos trabalham com programas de código fonte fechado. Essas diferenças ocorrem por coisas tão simples como um *bug* em um algoritmo, ou como uma conversão de dados de um formato para outro, para a utilização em outro programa, com a utilização de um código aberto, há recursos para a identificação desses problemas. Utilizando bibliotecas Open Source, torna-se fácil o uso da mesma base de código e a verificação dos algoritmos aplicados.

Também foi essa a intenção dos desenvolvedores do GeoTools ao disponibilizarem seu código fonte, permitindo uma constante melhoria em seu conteúdo.

6.3 -OGC

Para tornar o GeoTools padronizado com as outras tecnologias e aparelhos de geoprocessamento, os desenvolvedores tiveram que aderir a um padrão internacional de manipulação de dados georreferenciados, optaram pelo OGC.

O Open Geospatial Consortium (OGC), ou Consórcio Geoespacial aberto, é um consórcio internacional composto por 384 companhias, agências governamentais e universidades que participam em um processo de consenso para desenvolver padrões de interface disponíveis publicamente (OGC, 2009). Suporte a soluções sem fio e serviços baseados em localização e integração de TI. Cria normas a fim de capacitar desenvolvedores de tecnologia para tornar informações espaciais complexas acessíveis e úteis a todos os tipos de aplicações.

OpenGIS é uma marca associada com as normas e os documentos produzidos pelo OGC. Padrões OpenGIS são desenvolvidos em um processo de consenso único suportado pelos governos e membros acadêmicos para permitir a interoperabilidade de tecnologias de

geoprocessamento. Também é possível encontrar o OpenGIS associada a produtos que implementam ou cumprem essas normas, como é o caso do GeoTools.

7 - Exemplo de implementação com GeoTools

A aplicação desenvolvida baseia-se em exibir mapas carregados tanto de um *shapefile* quanto de um banco de dados PostGIS. Foi utilizada a biblioteca do GeoTools para fazer a manipulação dos dados geográficos.

7.1 - Dependências

Para que a aplicação funcione, se faz necessário possuir algumas bibliotecas e aplicativos instalados e configurados, descreveremos abaixo quais são e onde encontrá-los.

7.1.1 - GeoTools

O GeoTools (GEOTOOLS, 2009), como já explicando no capítulo 6, é uma biblioteca em Java para manipulação de dados geográficos, neste projeto foi utilizada a versão 2.6 do GeoTools que pode ser encontrada em `http://www.geotools.org/`.

7.1.2 - JAI

O Java Advanced Imaging foi adicionado à plataforma Java para dar aos desenvolvedores uma ferramenta de alto desempenho no processamento de imagens. A API fornece um rico conjunto de recursos para manipulação de imagens (JAI, 2009).

No aplicativo o JAI é utilizado em conjunto com a biblioteca do GeoTools para manipulação dos estilos geográficos. Ao adicionarmos uma camada abre-se a opção de editar o seu estilo, ou seja, alterar cor da linha e do fundo, opacidade, legendas entre outros, fig. 5, (GEOTOOLS, 2009).

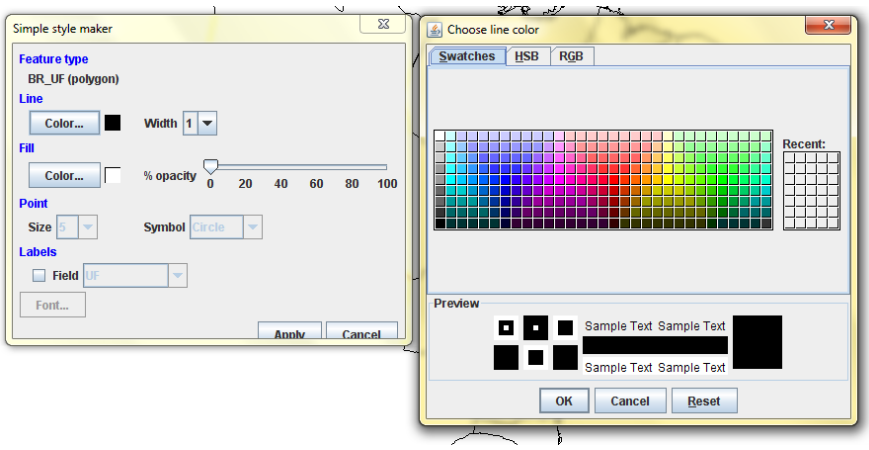

Figura 5: Manipulando estilos geográficos com o GeoTools usando o JAI

A biblioteca do JAI pode ser baixada em seu repositório oficial, em: `http://java.sun.com/products/java-media/jai/downloads/download-1_1_2_01.html`

7.1.3 - PostGIS

A obtenção, instalação e configuração do PostGIS já foram explicadas no capítulo 5.

7.2 -MapContex

Interface responsável por armazenar informações sobre a exibição do mapa. O objeto instanciado por esta classe é baseado nas especificações do OGC.

Para criar um novo objeto é utilizado o construtor `DefaultMapContext`, implementação padrão da interface `MapContext`.

```
MapContext map = new DefaultMapContext();
```

Para adicionar uma camada ao *MapContext* é utilizado o método *addLayer*, sendo passado dois valores como referência:

- Um objeto *FeatureSource*, onde são guardados os dados geográficos propriamente ditos, seja de um *shapefile*, seja de um banco de dados.
- E, opcionalmente, um objeto *Style* (será explicado mais à frente), caso não exista um *Style* definido para esta camada deve-se passar um valor nulo neste atributo.

O exemplo abaixo mostra como adicionar uma cama com ou sem um estilo:

```
map.addLayer(FeatureSource featureSource, Style style)

map.addLayer(FeatureSource featureSource, null)
```

7.3 -FeatureSource

Nesta interface, pode ser referenciado um *shapefile*, uma tabela do banco de dados, entre outros. Esta é uma das grandes características do GeoTools, o de poder trabalhar com diferentes fontes de dados de forma igual, pois uma vez carregado um *FeatureSource*, objetos distintos são manipulados de forma idêntica.

7.3.1 - Importando de um Shapefile

Como já foi explicado anteriormente, um *shapefile* é composto por três arquivos distintos, estes devem estar dentro de um mesmo diretório para que seja reconhecido como um *shapefile* válido, assim podemos instanciá-los usando o GeoTools:

Primeiramente criamos um objeto *File* apontando para o *shapefile* principal (.shp), como mostra o exemplo abaixo:

```
String arquivo = "C:\shapefiles\br_regioes\BR_regioes.shp";

File url= new File(arquivo);
```

Com o objeto *File* devidamente criado, podemos instanciar um *FeatureSource*, exemplificado no exemplo abaixo:

```
FeatureSource<SimpleFeatureType,SimpleFeature> featureSource;
```

Criamos também um *FileDataStore*, esta interface herda as características de um *DataStore*, que representa o conteúdo um arquivo.

```
FileDataStore store;
```

Para passar um valor para a variável *store*, declarada acima, usamos a classe *FileDataStoreFinder*, que encontra todas os *DataStores* válidos, e passamos a mensagem *getDataStore()*, passando como parâmetro a variável *url* já instanciada anteriormente, o método *getDataStore()*, verifica cada *DataStore* válido criado pelo *FileDataStoreFinder*, e retorna o primeiro a suportar o recurso passado por parâmetro.

```
store=FileDataStoreFinder.getDataStore(url);
```

E por último, a variável *featureSource* recebe um *FeatureSource* em si, que é o retorno da mensagem *getFeatureSource()* passado para a variável *store.*

```
featureSource = store.getFeatureSource();
```

Agora basta adicionar uma camada ao mapa com o método *addLayer()*:

```
map.addLayer(featureSource, null);
```

Abaixo o código completo, já com os tratamentos das exceções:

```
FeatureSource<SimpleFeatureType,SimpleFeature> featureSource=null;
String url="C:/shapefiles/Hidrografia/municipios.shp";
File sourceFile=new File(url);
try {
      FileDataStore store=
      FileDataStoreFinder.getDataStore(sourceFile);
      featureSource = store.getFeatureSource();
}catch (IOException e) {
      e.printStackTrace();
}
```

7.3.2 - Importando do Banco de dados

A criação de um *FeatureSource* a partir do banco da dados é bem parecida com do *shapefile*, já que o GeoTools trabalha da mesma forma, independente da origem.

Primeiramente criamos um *HashMap* para armazenar as configurações da conexão:

```
Map params = new HashMap();
```

Então incluímos os parâmetros de configuração para conectar ao banco:

```
params.put("dbtype", <tipo>);
params.put("host", <host>);
params.put("port", <porta>);
params.put("database", <banco>);
params.put("user", <usuário>);
params.put("passwd", <senha>);
```

Onde:

- **<tipo>:** Tipo do banco a ser conectado, em nosso exemplo será "*PostGIS"*.
- **<host>:** Geralmente *"localhost",* se o banco estiver na mesma máquina da aplicação.
- **<porta>:** Porta usada pelo banco, geralmente *"`5432`"*, deve-se passar como inteiro ex.: `new Integer(5432)`.

- **<banco>:** Nome do banco de dados a ser conectado.
- **<usuário>:** Nome do usuário deste banco.
- **<senha>:** Senha deste usuário.

Assim como no exemplo com *shapefile*, criamos um `FeatureSource`, depois um *DataStore*, que recebe um objeto *DataStore* válido do método *getDataStore()*, que é passado como parâmetro a variável *params*, com os dados da conexão:

```
FeatureSource <SimpleFeatureType,SimpleFeature> featureSource;

DataStore store = DataStoreFinder.getDataStore(params);
```

Por fim, a variável *featureSource*, recebe um *FeatureSource*, que é o retorno da mensagem *getFeatureSource()* passado para a variável *store*, este método deve passar como parâmetro o nome da tabela a ser recuperada.

```
featureSource = store.getFeatureSource(String tabela);
```

Agora basta adicionar uma camada ao mapa com o método *addLayer()*:

```
map.addLayer(featureSource, null);
```

Abaixo um exemplo de código completo, já com os tratamentos das exceções:

```
Map params = new HashMap();

params.put("dbtype", "PostGIS");

params.put("host","localhost" );

params.put("port", new Integer(5432));

params.put("database", "mapas");

params.put("user", "root");

params.put("passwd", "12345678");

FeatureSource <SimpleFeatureType,SimpleFeature>featureSource=null;

try {

     DataStore store = DataStoreFinder.getDataStore(params);

     featureSource = store.getFeatureSource("mundo");

} catch (IOException e1) {

     e1.printStackTrace();

}

map.addLayer(featureSource, null);
```

7.4 -CQL

O CQL (*Common Query Language*) é uma linguagem que, no *GeoTools*, filtra os dados geográficos para alterar os estilos, por exemplo. É definido por uma especificação do catálogo OGC, desta forma, cria consultas tão fáceis e legíveis quanto a clausula *where* do SQL.

A partir da consulta CQL são criados filtros para selecionar partes específicas de um mapa, abaixo alguns exemplos da utilização do CQL:

- Comparando valores

```
Filter result = CQL.toFilter( "ATTR1 < (1 + ((2 / 3) * 4))" );

Filter result = CQL.toFilter( "ATTR1 < (1 + ((2 / 3) * 4))" );

Filter result = CQL.toFilter( "ATTR1 < 10 OR ATTR3 > 10" );
```

- Usando Texto

```
Filter result = CQL.toFilter( "ATTR1 LIKE 'abc%'" );

Filter result = CQL.toFilter( "ATTR1 NOT LIKE 'abc%'" );
```

- Valores nulos

```
Filter result = CQL.toFilter( "ATTR1 IS NULL" );

Filter result = CQL.toFilter( "ATTR1 IS NOT NULL" );
```

- Comparando dadas

```
Filter result = CQL.toFilter("ATTR1 BEFORE 2006-11-30T01:30:00Z");

 Filter result = CQL.toFilter("ATTR1 AFTER 2006-11-30T01:30:00Z");
```

- Se existe ou não

```
Filter result = CQL.toFilter( "ATTR1 EXISTS" );

Filter result = CQL.toFilter( "ATTR1 DOES-NOT-EXIST" );
```

7.5 -Style

Os *Styles* indicam como um dado geográfico deve ser mostrado, definem cores para as linhas, fundos e textos do mapa. Pode definir, por exemplo, a cor de fundo de uma camada inteira, ou, através de consultas do CQL, definir partes a serem pintadas de cores diferentes, por exemplo, colocar cada região do Brasil de uma cor, como mostra a fig. 6.

Figura 6: Mapa das Regiões do Brasil usando o CQL

Para criarmos um *style*, primeiramente criamos um *StyleBuilder*, uma classe para construir de forma simples, estilos e métodos, veremos mais a frente o uso desta classe. Depois criamos um *Style* recebendo o retorno da mensagem *createStyle()* do *StyleBuilder*.

```
StyleBuilder sb = new StyleBuilder();

Style style = sb.createStyle();
```

A partir disto podemos criar vários tipos de estilos, dependendo do formato da camada, os tipos abordados neste projeto são: Polígono, linha e ponto. Para identificá-los criamos um método capaz de reconhecer o tipo dado um *FeatureSource*:

```
public int geraTipo(FeatureSource featureSource){

String s =
featureSource.getSchema().getAttributeDescriptors().toString();

int inicio=s.indexOf("<")+1;

int fim=s.indexOf(":");
```

```
String s2=s.substring(inicio,fim);

if (s2.equals("Point")){

     //códigos para tratar um ponto

}else if (s2.equals("Line")){

     //códigos para tratar uma linha

}else if (s2.equals("MultiPolygon")){

     //códigos para tratar um polígono

}

return tipo;

}
```

Uma vez identificado o tipo, podemos trabalhar com a interface *Symbolizer*, ele descreve como um recurso deve aparecer em um mapa. O *symbolizer* descreve não apenas a forma que deve aparecer, mas também propriedades gráficas, tais como cor e opacidade.

Um *symbolizer* pode ser obtido por diferentes sub-interfaces, neste projeto descreveremos três: *PointSymbolizer*, *LineSymbolizer* e *PolygonSymbolizer*, suas implementações são descritas abaixo:

1. **Ponto (*PointSymbolizer*)**

Primeiramente criamos um *Mark* usando o método do *StyleBuilder* `createMark`:

```
Mark mk = sb. createMark(String nome, Color cor_de_fundo,
                Color cor_da_borda, double largura);
```

Criamos então um *Graphic*, baseado no *Mark*, criado anteriormente:

```
Graphic gr = sb.createGraphic(null, mk, null, double opacidade,
                double tamanho, double rotação);
```

E então um *PointSymbolizer*, baseado no *Graphic*:

```
PointSymbolizer symbolizer = sb.createPointSymbolizer(gr);
```

2. **Linha (*LineSymbolizer*)**

Basta criar um *LineSymbolizer:*

```
LineSymbolizer symbolizer =sb.createLineSymbolizer(Color cor,
                        double largura);
```

3. **Polígono (*PolygonSymbolizer*)**

Criamos um *PolygonSymbolizer*:

```
PolygonSymbolizer symbolizer = sb.createPolygonSymbolizer(Color
  cor_de_fundo, Color cor_da_borda, double largura_da_borda);
```

Criado o *symbolizer*, independente do tipo, criamos um objeto da classe *Rule*, a classe *Rule* é usada para incluir uma condição usada para o processamento dos *symbolizers*, que neste caso é o *PointSymbolizer*, criado anteriormente:

```
Rule rule = sb.createRule(new Symbolizer[]{symbolizer});
```

Criamos um *FeatureTypeStyle* a partir da regra criada anteriormente, e adicionamos este ao *Style*, também já criado anteriormente.

```
FeatureTypeStyle fts = sb.createFeatureTypeStyle(null, rule);

style.addFeatureTypeStyle(fts);
```

Para inserirmos uma condição CQL à regra criada, criamos um objeto da classe *Filter* e o adicionamos na regra já criada.

```
Filter filtro = CQL.toFilter(String filtro);

rp.setFilter(filtro);
```

Pode-se também criar um texto para ser exibido em cima do componente geográfico, para isso utilizamos o *TextSymbolizer*:

```
TextSymbolizer ts = sb.createTextSymbolizer(Color cor, Font fonte,
                    String nome_do_atributo);
```

Este deve ser adicionado junto com o *Symbolizer* atual, independente do tipo:

```
rp = sb.createRule(new Symbolizer[]{symbolizer,ts});
```

7.6 - JMapFrame

O *JMapFrame* cria uma janela *Swing* contendo um painel de exibição de mapa (*MapContext*) e opcionalmente uma barra de ferramentas, barra de status e uma tabela de camadas do mapa, como mostra a fig. 7.

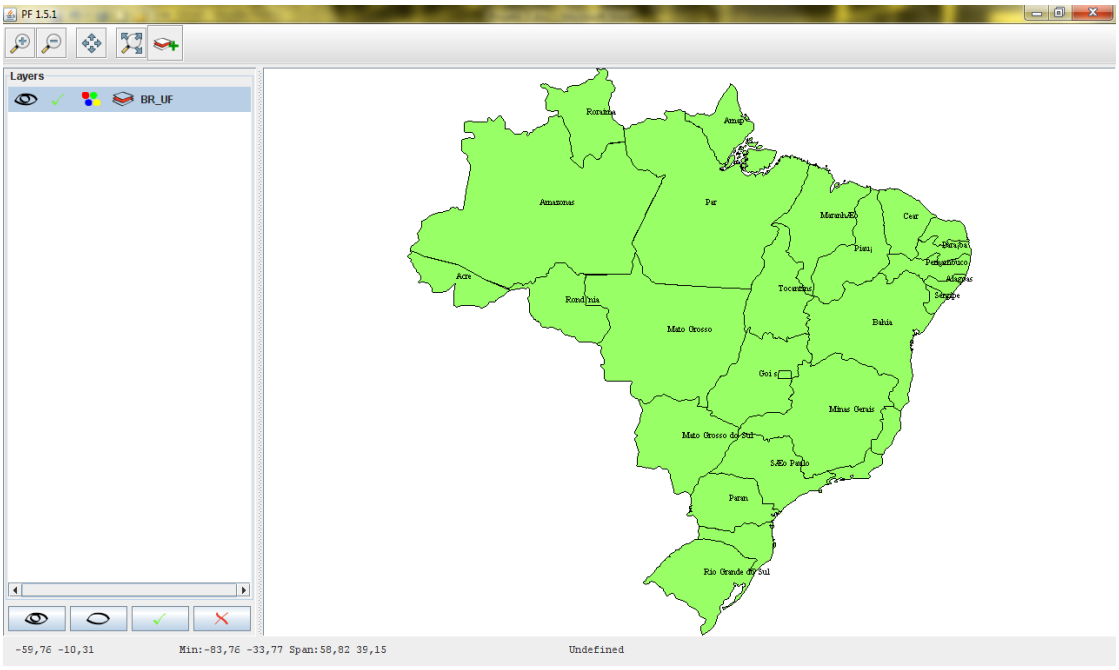

Figura 7: JMapFrame em ação

Instanciando um novo *JMapFrame:*

```
JMapFrame mapFrame = new JMapFrame(map);
```

Para inserir um título nesta janela basta usar o comando abaixo:

```
mapFrame.setTitle(String);
```

O *enableLayerTable* define uma tabela que mostrar a lista de camadas do *MapContext*, define a ordem, visibilidade, status, estilo, nome e permite remover cada camada do mapa. Deve ser passado como parâmetro um valor booleano, para mostrar ou ocultar a tabela.

```
mapFrame.enableLayerTable(true);
```

O *JMapFrame* também conta com uma barra de ferramentas, que deve ser inicializada do o seguinte modo:

```
mapFrame.enableToolBar(true);
```

Este método chama a barra de ferramentas padrão, porém ela pode ser personalizada com o *enableTool:*

```
mapFrame.enableTool(JMapFrame.Tool... tool);
```

Podendo ser passado os seguintes atributos:

- **INFO –** Informações de um recurso.
- **NONE –** Cria uma ferramenta vazia.
- **PAN –** Ferramenta para mover o mapa com o *mouse.*
- **ZOOM –** Habilita o *zoom in* e o *zoom out* com o *mouse.*
- **RESET –** Centraliza o mapa, o posicionamento e o zoom são alterados para o padrão.

Exemplo de uso do *enableTool*:

```
mapFrame.enableTool(JMapFrame.Tool.PAN,JMapFrame.Tool.ZOOM,
JMapFrame.Tool.RESET);
```

É possível também adicionar novos botões a barra de ferramentas do *JMapFrame*. Para isso é necessário criar um objeto do tipo *JToolBar*, e adicionar objetos `JButton` a ele, como mostra o exemplo abaixo:

```
JToolBar toolbar = mapFrame.getToolBar();
JButton bnt = new JButton();
toolbar.add(bnt);
```

Outro método muito útil do *JMapFrame* é a barra de status, que exibe a posição do cursor e os limites do mapa.

```
mapFrame.enableStatusBar(true);
```

Dois outros métodos adicionais, porém recomendados, podem ser incluídos no código:

Maximizar a janela:

```
mapFrame.setExtendedState(mapFrame.MAXIMIZED_BOTH);
```

Tornar a janela visível:

```
mapFrame.setVisible(true);
```

Abaixo o exemplo completo de uma classe que instancia uma nova janela:

```
public class JanelaPrincipal {

      private JMapFrame mapFrame;

      public JanelaPrincipal(){

            MapContext map = new DefaultMapContext();

            mapFrame = new JMapFrame(map);

            mapFrame.setTitle("Olá mundo!");

           mapFrame.enableLayerTable( true );

            mapFrame.enableTool(JMapFrame.Tool.PAN,
JMapFrame.Tool.ZOOM, JMapFrame.Tool.RESET);

           mapFrame.enableStatusBar(true);

            mapFrame.setExtendedState(mapFrame.MAXIMIZED_BOTH);

            mapFrame.setVisible(true);

}}
```

8 -Conclusão

Desde o início deste projeto, nossa vontade era abordar um tema atual para que pudéssemos aplicar de forma prática os conhecimentos adquiridos neste curso, então a partir de pesquisas para a escolha do tema, nos deparamos como o geoprocessamento, tecnologia que vem trazendo comodidade ao nosso dia-a-dia, optamos, então, pelo desenvolvimento de um aplicativo em Java, já que a intenção inicial e a única certeza, mesmo quando o tema do projeto ainda não estava definido, era a de trabalhar no desenvolvimento de um projeto que envolvesse esta linguagem de programação. Então, surgira a oportunidade de envolver o geoprocessamento no projeto que manteria o objetivo inicial de trabalhar na área de desenvolvimento e ao mesmo tempo abordar este assunto , que tem grande destaque neste projeto, já que a mesclagem dos dois assuntos originariam num SIG.

Aplicam-se os conceitos introdutórios de geoprocessamento, para os assuntos que envolvem a utilização do mesmo, como cartografia, posicionamento global, *datuns* entre outros, o que aborda informações geográficas para facilidade na manipulação de dados geográficos. Também aborda o GeoTools, complemento a linguagem de programação Java, além de *shapefiles* e PostGIS, tecnologias que armazenam dados georreferenciados.

Observados todos os assuntos e tecnologias apresentadas, conclui-se que o aplicativo desenvolvido atinge os objetivos para o qual foi projetado e abre margens para implantação de projetos mais sofisticados na área de Tecnologia da Informação, visando maior aplicabilidade na representação e manipulação de dados geográficos.

8.1 -Trabalhos Futuros

A partir do projeto desenvolvido, que não passa de uma aplicação básica de SIG, propomos para trabalhos futuros o suporte a uma variedades de necessidades relacionadas à área de geoprocessamento, tal como a opção de cadastramento de informações, que levaria o aplicativo a possuir inúmeras funcionalidades, tais como: manter dados de uma determinada localidade; cruzamento dos mesmos trazendo a identificação de prováveis problemas e propondo soluções mais rápidas com clareza na representação das informações. Também propomos a adição de uma ferramenta que seja capaz de criar novas camadas manualmente, a partir da adição de elementos geográficos, tais como: pontos,

linhas, áreas, textos e imagens. Assim como uma camada que se conecte em serviços de mapas on-line, como o Google Maps.

Na perspectiva de melhoria na funcionalidade do aplicativo vale ressaltar que um dos recursos sugestionáveis seria torná-lo um serviço disponível na Web que daria mais facilidade no transporte de informações.

Referências Bibliográficas

1. BARNES, David J. & KÖLLING, Michael. *Programação Orientada a Objetos com Java*. Prentice Hall, 2004.
2. CASTANHO, Roberto Barboza & ROSA, Roberto. O Geoprocessamento como instrumento de análise territorial do Espaço agropecuário da Microrregião Geográfica de Carazinho – RS, Brasil. Uberlândia: UFU, 2007.
3. CEUB/ICPD, Cartografia e GPS. Disponível em <http://www.etecjbento.com.br/downloads/cristina/cartografia_e_gps.pdf>. Acesso em 3 Setembro de 2009.
4. DA SILVA, Jorge Xavier & ZAIDAN, Ricardo Tavares. *Geoprocessamento & análise ambiental*. Bertrand Brasil, 2004.
5. DEITEL, H. M. *Java como programar 6° edição.* Prentice Hall, 2007.
6. DIAS, Hugo, *O SIG Municipal ao serviço do Cidadão/Munícipe*. Disponível em < http://geo-competitivo.tagus.ist.utl.pt/Anexos/SIG%20Municipal.pdf >. Acesso em 12 Maio de 2009.
7. ESRI. *ESRI Shapefile Technical Description.* Disponível em < http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf >. Acesso em 26 de Outubro de 2009.
8. FILHO, Lauro Luiz Francisco. Uso do geoprocessamento como apoio na gestão do município: Petrópolis, um estudo de caso. Rio de Janeiro: UFRJ: 1999.
9. FORTES, Luiz Paulo Souto, *SIRGAS: O Sistema de Referência Para o Novo Milênio*. Disponível em < ftp://geoftp.ibge.gov.br/documentos/geodesia/pmrg/ Apresentacao_em_eventos/2000/ SEMINARIO_2000_SIRGAS_D.pdf >. Acesso em 3 Setembro de 2009.
10. FRANÇA, Rovane Marcos de. Sistema Geodésico de Referência, Projeções Cartográficas e GPS. Disponível em < http://www.topografia.ufsc.br/Geodesia.pdf>. Acesso em 2 Setembro de 2009.
11. GEOTOOLS. *GeoTools The Open Source Java GIS Toolkit GeoTools v2.6 documentation*; Disponível em: < http://www.geotools.org/> Acesso em 17 Nov 2009.

12. GOMES, Marisa Prado & AGUIAR, Marcelo Cabral de. Noções Básicas sobre Geoprocessamento. Disponível em < http://www.lapig.iesa.ufg.br/ novosite/apresentacoes/treinamentos05/Nocoes_basicas_Geoprocessamento.pdf>. Acesso em 3 Maio de 2009.

13. HALL, G. Brent & LEAHY, Michael G. *Open Source Approaches in Spatial Data Handling.* Springer, 2008.

14. JAI. *Java Advanced Imaging (JAI) API.* Disponível em: < *http://java.sun.com/javase/technologies/desktop/media/jai/*> Acesso em 17 de Novembro de 2009.

15. OGC. *Welcome to the OGC Website | OGC®*. Disponível em: < http://www.opengeospatial.org/> Acesso em 16 de Novembro de 2009.

16. PEREIRA, Gilberto Corso & SILVA, Bárbara-Christine Nentwing. *Geoprocessamento e Urbanismo.* Disponível em < http://www.ageteo.org.br/download/livros/2001/ 05_Pereira.pdf >. Acesso em 3 Maio de 2007.

17. POSTGIS. *PostGIS : Home*; Disponível em: < http://postgis.refractions.net/> Acesso em 8 Nov 2008.

18. SPATIAL REFERENCE. *Home -- Spatial Reference*. Disponível em: <http://spatialreference.org/> Acesso em 3 de Outubro de 2009.

19. WILDAUER, Egon Walter. Sistema computacional para projeção de cenários de temperaturas favoráveis ao plantio de pinus taeda no estado do Paraná. Curitiba: UFPR, 2007.