ALBERT MASSAYUKI KUNIYOSHI
BRUNO UMEDA GRISI
MARCOS NOGUEIRA

SISTEMAS DE INFORMAÇÃO PARA TOMADA DE DECISÃO EM AGRICULTURA DE PRECISÃO

SÃO PAULO
2009

ALBERT MASSAYUKI KUNIYOSHI
BRUNO UMEDA GRISI
MARCOS NOGUEIRA

SISTEMAS DE INFORMAÇÃO PARA TOMADA DE DECISÃO EM AGRICULTURA DE PRECISÃO

Trabalho de conclusão de curso apresentado para obtenção do título de Engenheiro da Computação na Escola Politécnica da Universidade de São Paulo.

Área de Concentração:
Engenharia da Computação.

Orientador:
Prof. Dr. Antonio Mauro Saraiva
Co-Orientadora:
Prof[a]. Dr[a]. Fabiana Soares Santana

SÃO PAULO
2009

RESUMO

Este trabalho destinou-se à pesquisa de serviços geoespaciais padronizados pela *Open Geospatial Consortium* com o intuito de utilizá-los no desenvolvimento de aplicações para Agricultura de Precisão e Biodiversidade. Foram estudados os serviços WMS (*Web Map Service*), WFS (*Web Feature Service*) e WCS (*Web Coverage Service*). Desenvolveram-se aplicações principalmente focando na utilização do WMS devido à sua característica de trabalhar com imagens. Por fim, foram utilizados algoritmos específicos de Agricultura de Precisão e Biodiversidade, o *Filtering* e o GARP, respectivamente, para criar uma solução que integrou estes algoritmos com as capacidades geoespaciais providas pelos serviços e suas operações.

ok

ABSTRACT

This work researched the geospatial web services standardized by the *Open Geospatial Consortium,* so as to use them in the development of applications for Precision Agriculture and Biodiversity. The geospatial web services studied were the WMS (*Web Map Service*), the WFS (*Web Feature Service*) and the WCS (*Web Coverage Service*). Applications were developed focusing on the usage of WMS due to its main characteristic, which is to work with images. At last, specific algorithms used in Precision Agriculture and Biodiversity, respectively *filtering* and GARP, were applied to create an application which combines these algorithms with the geospatial capabilities provided by the web services and their operations.

DEDICATÓRIA

Dedico este trabalho de conclusão de graduação aos meus pais, Luiz Massahiro Kuniyoshi e Maria Michiko Kuniyoshi, e à minha irmã, Aline Lie Kuniyoshi, por me apoiarem na concretização deste trabalho.

Albert Massayuki Kuniyoshi

Dedico este trabalho à minha mãe, Lumi Maristela Umeda Grisi, pela educação que me foi ensinada e pela qual hoje colho frutos bons. Dedico ao meu pai, Rubens Grisi Júnior, pelo apoio e tranquilidade que sempre a mim foram transmitidos, proporcionando um equilíbrio muito importante para a realização dos meus objetivos. Dedico à minha irmã, Patricia Umeda Grisi, pela amizade e pelo cuidado que temos um com outro.

Bruno Umeda Grisi

Dedico este trabalho primeiramente a minha mãe, Maria José dos Reis Nogueira, devido a todo o seu tempo, dedicação e, principalmente, paciência direcionada a mim ao longo de todos esses anos.

Aos meus irmãos Edward Nogueira Júnior e Elaine Cristina Nogueira devido às lições que me ensinaram durante toda minha infância.

E por fim, a meu pai, Edwardi Nogueira, por todo o suporte e rigidez durante o tempo em que estive com ele. Descanse em paz.

Marcos Nogueira

AGRADECIMENTOS

Ao professor Antonio Mauro Saraiva, pelo papel fundamental em nos auxiliar no desenvolvimento deste trabalho.

À professora Fabiana Soares Santana, pela orientação, atenção e apoio indispensáveis para a conclusão deste trabalho.

À Escola Politécnica da Universidade de São Paulo, pela excelência na formação em Engenharia, provendo-nos oportunidade de aprendizado e crescimento.

Ao Departamento de Engenharia de Computação e Sistemas Digitais, por prover a seus alunos não apenas com conhecimento técnico, mas também com a capacitação esperada de um Engenheiro de Computação.

A nossos colegas, pela amizade e cooperação.

Ao Winston França pela atenção dada ao nosso trabalho e pelas dúvidas solucionadas.

Por fim, a todos que direta ou indiretamente contribuíram para este trabalho.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	*Agricultura de Precisão*
CCE	*Centro de Computação Eletrônica*
CGI	*Common Gateway Interface*
ESALQ	*Escola Superior de Agricultura Luiz de Queiroz*
FINEP	*Financiadora de Estudos e Projetos*
GARP	*Genetic Algorithm for Rule-set Production*
GML	*Geography Markup Language*
GPS	*Global Positioning System*
KML	*Keyhole Markup Language*
LAA	*Laboratório de Automação Agrícola*
OGC	*Open Geospatial Consortium*
P&D	*Pesquisa e Desenvolvimento*
POLI	*Escola Politécnica*
POM	*Project Object Model*
SAD	*Sistema de Apoio à Decisão*
SBIAgro	*Associação Brasileira de Agroinformática*
SIICUSP	*Simpósio Internacional de Iniciação Científica da USP*
SIG	*Sistema de Informação Geográfica*
SLD	*Styled Layer Descriptor*
SOA	*Service-Oriented Architecture* ou *Arquitetura Orientada a Serviços*
SRD	*Sistema de Referência Geoespacial*
USP	*Universidade de São Paulo*
WCS	*Web Coverage Service*
WFS	*Web Feature Service*
WKB	*Well Known Binary*
WKT	*Well Known Text*
WMS	*Web Map Service*
XML	*Extensible Markup Language*
BDD	*Biodiversity* Data *Digitizer*

SUMÁRIO

1 INTRODUÇÃO

O Agronegócio brasileiro (Saraiva, 2003), em particular a Agricultura de Precisão, AP, demanda cada vez mais um significativo suporte da Tecnologia da Informação, TI, em várias das suas atividades. A coleta, o armazenamento, o processamento e a análise das numerosas variáveis e da grande quantidade de dados envolvidos em seus processos requerem sistemas de informações. A tomada de decisão bem fundamentada demanda a consideração de um conjunto cada vez mais abrangente de informações. Para que essa decisão ocorra no momento oportuno o acesso aos dados deve ser feito de maneira rápida, muitas vezes em tempo real, ainda que as fontes estejam distantes e disponíveis em sistemas diferentes.

O dinamismo e a complexidade do agronegócio fazem com que seja necessário integrar diferentes fontes de dados, módulos de software e sistemas de informações completos para entender e gerenciar a complexidade dos processos e a velocidade com que a AP evolui.

Todavia, a característica principal da maioria dos sistemas de informação desenvolvidos para o agronegócio (como ocorre também para outras áreas) é o seu caráter fechado e proprietário. Isso dificulta ou mesmo impede a integração com outros sistemas, sejam eles fonte de dados necessários para a tomada de decisão, sejam eles destino dos dados gerados como resultado do processo de análise e decisão (Saraiva, 2003).

Por mais completos que possam ser os sistemas existentes e embora existam sistemas que atendam parcialmente às necessidades dos usuários, eles sempre poderão se beneficiar da integração com diversas fontes de dados e de processamento que estão distribuídos e disponíveis na Internet, como: dados econômicos, técnicos, legais; modelos de simulação para a avaliação de cenários alternativos de manejo; clusters de computadores para execução com alto desempenho, para citar alguns exemplos.

Se o desenvolvimento de software for realizado com base em paradigmas que favoreçam a interoperabilidade, a integração e o reuso de código, essas dificuldades

serão minimizadas e se poderá utilizar de modo mais efetivo o potencial da TI para o agronegócio.

A tecnologia de *Web Services*, uma possível implementação para o conceito de serviço em Arquiteturas Orientadas a Serviços (SOA), permite o desenvolvimento e a integração de sistemas de informações como uma alternativa para resolver os problemas de interoperabilidade, de integração e de reuso de software. Ela tem sido fortemente adotada e apoiada pelas principais empresas de computação no mundo, razão pela qual, embora seja relativamente recente no mercado, teve forte avanço (Saraiva, 2003).

Foi desenvolvida no Laboratório de Automação Agrícola, LAA, do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, uma infraestrutura baseada em SOA. Esta infraestrutura possui uma arquitetura de referência para sistemas de informações para agricultura de precisão, uma linguagem padrão para troca de dados entre serviços agrícolas, um barramento para conexão de serviços agrícolas, um provedor de serviços geoespaciais e um portal para serviços agrícolas. O seu propósito é fornecer elementos básicos para apoiar e facilitar o desenvolvimento de sistemas de informação para agricultura de precisão mais abrangentes e flexíveis, capazes de acompanhar a rápida evolução da AP. (Murakami, 2006)

No projeto de Pesquisa e Desenvolvimento, P&D, a ser desenvolvido com apoio da FINEP, FINanciadora de Estudos e Projetos do Ministério da Ciência e Tecnologia, esta infraestrutura será detalhada, refinada e colocada em operação para acesso público através da Internet, disponibilizando serviços de interesse para a agricultura de precisão.

Este projeto de formatura faz parte do projeto FINEP e nele foram desenvolvidos serviços fundamentais para esta infraestrutura, que são os serviços de tratamento de dados geoespaciais, uma aplicação para permitir o uso orquestrado destes serviços com outros serviços para tomada de decisão e alguns algoritmos para modelar a produtividade.

Segundo Soares (2003), Sistemas de Informação Geográfica, SIG, são ferramentas projetadas para coletar, manipular e apresentar grandes volumes de dados espaciais. As principais funções que um SIG completo deve ter são: captura dos dados (gráficos ou atributos na forma de importação de dados, digitalização, scanner e levantamentos de campo, entre outros), gerência dos atributos (edição,

gerência da base de dados), manipulação espacial (edição), análise dos dados (consultas condicionadas, sobreposições, modelagens) e saída dos dados (mapas, relatórios e imagens).

Quanto ao aspecto estrutural, um SIG possui os seguintes componentes: interface com o usuário, entrada e integração dos dados, funções de consulta e análise espacial, visualização e plotagem e armazenamento e visualização de dados (organizados sob a forma de um banco de dados geográficos) (Soares, 2003).

Os três tipos básicos de elementos utilizados nos mapas para modelar todos os objetos do mundo real são:

- Pontos: define localizações discretas de elementos geográficos demasiadamente pequenos para serem descritos como linhas ou áreas.
- Linhas: são definidas como um conjunto ordenado de pontos interligados por segmentos de reta ou por linhas e são utilizadas na representação de objetos sem largura suficiente para serem consideradas áreas.
- Áreas: são definidas como um conjunto ordenado de pontos interligados, em que o primeiro ponto e o último coincidem, utilizados quase sempre na representação de zonas que possuem uniformemente uma dada propriedade; ou seja, figura fechada, cujo os limites encerram uma área homogênea.

Esses três tipos de elementos se relacionam no mapa, constituindo-se nas camadas de dados dos mapas (*layers*) que compõem o espaço geográfico em estudo.

1.1 Objetivos

Este trabalho visa o desenvolvimento de um sistema SIG para apoio à decisão para AP, como parte de um projeto de P&D para auxiliar os usuários na quantificação, no entendimento e no gerenciamento da variabilidade espaço-temporal dos dados obtidos em campo. Assim, o sistema a ser desenvolvido deverá permitir a manipulação de mapas de diversas variáveis agrícolas, de acordo com o padrão definido pelo Open Geospatial Consortium (OGC)

[http://www.opengeospatial.org/], e realizar processamentos típicos da AP, tais como conversão de formatos, limpeza de dados e combinação de mapas.

Por meio da utilização de uma arquitetura aberta e orientada a serviços, baseada na tecnologia de *Web Services*, o sistema pode contribuir para o desenvolvimento de serviços que atendam melhor aos fatores de escalabilidade, simplicidade, modificabilidade, extensabilidade, configurabilidade, customizabilidade, reusabilidade, portabilidade e confiabilidade para AP.(Murakami, 2006)

1.2 Motivação

O sistema de apoio à decisão a ser desenvolvido neste trabalho é parte de um projeto de P&D para AP que está sendo desenvolvido com apoio da FINEP. O projeto consiste da especificação, integração e testes de serviços de um sistema de apoio a decisão para AP via Internet, que permita aos usuários o acesso e processamento de dados para auxílio na quantificação, entendimento e gerenciamento da variabilidade espaço-temporal em campo. O sistema será baseado em arquitetura orientada a serviços (SOA – *Service-Oriented Architeture*) facilitando a integração com outros sistemas (como os de informação de dados de mercado, climáticos, etc.) e também a integração de novos algoritmos de processamento, em particular os desenvolvidos em pesquisas nacionais ainda não disponíveis em pacotes de software no mercado. Este sistema conterá os algoritmos, processos, metodologias e dados obtidos/desenvolvidos pelos demais grupos integrantes desta proposta, formando a base para a tomada de decisão. Além da POLI-USP, a Escola Superior de Agricultura "Luiz de Queiroz" da Universidade de São Paulo, ESALQ-USP, também é uma das instituições executoras deste projeto, enquanto a Jacto S/A [http://www.jacto.com.br/], a maior empresa nacional de maquinários agrícolas instalada no país, é a empresa co-financiadora. Assim, existem outros requisitos funcionais a serem desenvolvidos para atender as especificações do projeto de P&D que não fazem parte do escopo deste trabalho.

Porém, no contexto deste projeto, um dos maiores desafios é a implementação das principais funcionalidades de um SIG na forma de serviços, para fazer o tratamento de dados georreferenciados dentro do contexto da arquitetura

SOA proposta. Uma vez que grande parte das aplicações de AP tratam mapas em algum nível, o uso de serviços para tratar dados georreferenciados é essencial para o sucesso do projeto.

Neste trabalho, portanto, é proposto um sistema SIG simplificado. Este sistema é escalável e interoperável, e já possui aplicação prática em um dos principais sistemas da Jacto S/A.

1.3 Justificativa

A Agricultura de Precisão (AP) atua em uma área que, no Brasil, tem sido um dos pilares da economia nacional: o agronegócio. Graças ao agronegócio, o país tem tido um superávit na balança de comércio exterior, como se pode observar pela Figura 1 (Saraiva, 2003).

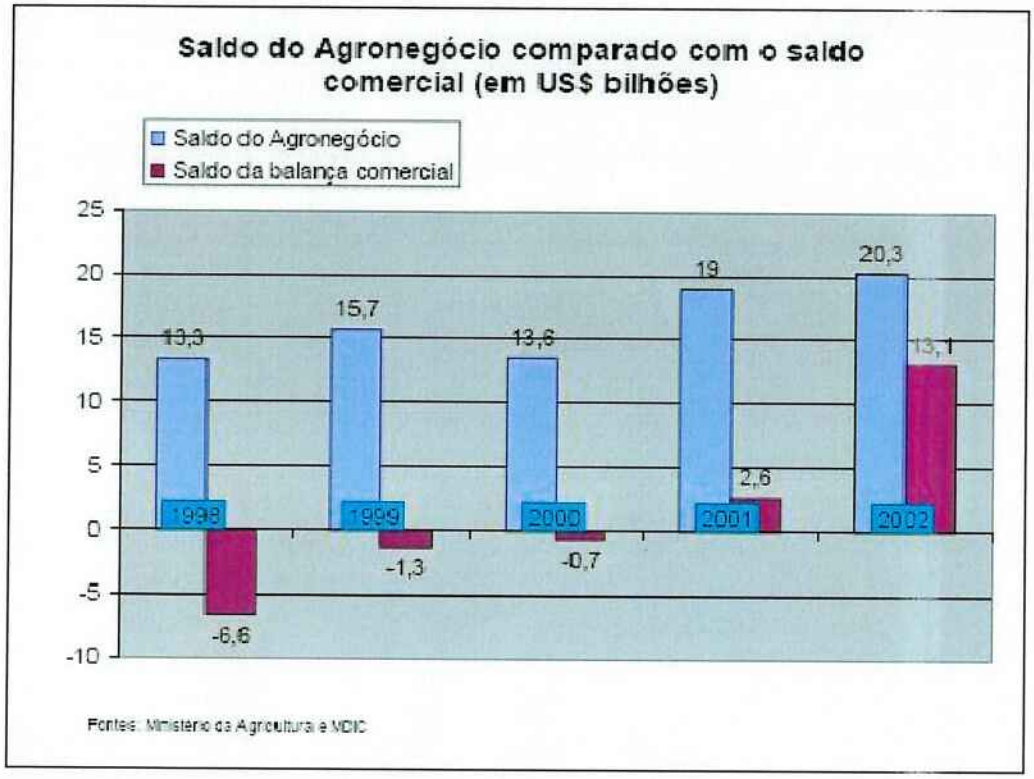

Figura 1.1 – Saldo da balança comercial e saldo do agronegócio no Brasil. (Saraiva, 2003)

A competitividade do agronegócio se deve aos investimentos tecnológicos de vários de seus setores, o que implica em aumento de produtividade e de qualidade dos produtos gerados. Entretanto, os desafios para a agricultura neste século são maiores e seguem uma preocupação com questões relacionadas a sustentabilidade da produção e à conservação do ambiente, tais como: o correto uso da água para evitar a ameaça de escassez e a poluição dos mananciais; a manutenção da fertilidade do solo; o controle das pragas e doenças que afetam as plantas e os rebanhos, entre outros. (Saraiva, 2003)

Tais questões precisam ser enfrentadas em diferentes níveis e exigem informações e conhecimento para a tomada de decisão. Existe um número muito grande de variáveis associadas, advindas de uma maior compreensão humana do mundo físico. Como conseqüência, é alto o grau de dificuldade das interações e enorme o volume de dados que podem ser utilizados atualmente. Portanto, é imprescindível o uso de ferramentas que auxiliem no processo de transformar os dados em informações úteis para a tomada de decisão. Isto requer a realização de múltiplas operações e de processamento sobre os dados, o que demanda áreas distintas de conhecimento, como geoestatística e o processamento de imagens, além dos conhecimentos agronômicos básicos, o que dificulta a realização de todo o processamento por um único usuário, ainda que especialista. Por este motivo, os sistemas de apoio à decisão em AP são tão importantes. (Referência, Ano)

1.4 Estrutura do Trabalho

Esta monografia está dividida em oito partes distintas:

Introdução: aborda a importância do uso da Tecnologia de Informação no paradigma Agricultura de Precisão e descreve os objetivos, as justificativas e a motivação deste trabalho;

Material e Métodos: descreve os principais conceitos, técnicas, ferramentas e tecnologias relacionados ao projeto a fim de que sejam transmitidas as informações necessárias para um bom entendimento do trabalho;

Metodologia: descreve as etapas definidas ao desenvolvimento deste trabalho, envolvendo desde a especificação até a validação do projeto;

Especificação: possui informações a respeito da descrição e arquitetura do projeto, seus requisitos funcionais e não-funcionais, assim como uma descrição sucinta dos principais serviços geoespaciais do interesse deste trabalho;

Implementação: são detalhados os algoritmos desenvolvidos que utilizam o sistema servidor de mapas;

Resultados e Discussão: descreve os resultados finais obtidos neste trabalho e os avalia de modo crítico, evidenciando pontos a serem melhorados;

Conclusão e Trabalhos Futuros: são retomados os itens mais importantes do processo de desenvolvimento e são apresentadas possibilidades para estender este trabalho.

2 MATERIAL E MÉTODOS

A OGC (*Open Geospatial Consortium*) (Percivall, 2003) é uma organização internacional, sem fins lucrativos, com 385 companhias, agências governamentais e universidades, cujo objetivo é o desenvolvimento de padrões para serviços geoespaciais. A idéia é habilitar o uso de soluções geoespaciais interoperáveis através da internet. O objetivo de se criar padrões é tornar acessíveis tanto as informações espaciais quanto os serviços disponíveis para diversos tipos de aplicação.

Dentro dos serviços padronizados por essa organização, existem três principais, a saber: WMS, WFS e WCS. Estes são os mais utilizados atualmente para o tratamento de dados geoespaciais, sendo o WMS o mais usado entre eles.

2.1 OGC Web services

O *Web Map Service*, WMS, permite que se obtenha, através de uma comunicação HTTP, imagens geoespaciais de mapas provenientes de diferentes fontes de dados espaciais. Este serviço é capaz de retornar essas imagens em um formato de imagem comum, como JPG ou PNG, e de manipulá-las de modo a criar sobreposições entre elas em estrutura de camadas.

O *Web Feature Service*, WFS possibilita a requisição de características geográficas. Pode-se pensar em uma *feature* geográfica como o "código-fonte" por detrás de um mapa. O WFS retorna este código em um arquivo de extensão em *Geographic Markup Language*, GML (referência para GML), com a informação espacial do mapa requisitado. GML é uma linguagem que tem por intuito descrever características geográficas. É um subconjunto da linguagem XML. O WFS é mais voltado para o tratamento de dados vetoriais.

O *Web Coverage Service*, WCS retorna imagens matriciais, como arquivos do tipo *raster*. Os arquivos são devolvidos em um formato manipulável, diferentemente do WMS, que os retorna apenas como imagem.

Para uma maior compreensão do que significam estes servidores, é necessário primeiro compreender melhor quais suas principais operações:

- WMS – *Web Map Service*

  Trata-se de um serviço espacial estabelecido pela OGC (*Open Geospatial Consortium*) capaz de retornar informações georreferenciadas no formato de imagem (JPEG, PNG, etc.) em um *browser*, e também permite obter informações sobre pontos específicos. As possíveis operações com o WMS são:

1. GetMap – Retorna as imagens. Parâmetros: version, service, request, format, bbox, srs, width, height, layers, TIME, styles, sld, transparent, bgcolor, exceptions. Exemplo de requisição: http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.1.1&service=WMS&request=GetMap&srs=EPSG:4326&bbox=-180,-90,180,90&format=image/png&layers=land_shallow_topo_2048,rivers&styles=,&transparent=true&width=500&height=300

2. GetFeatureInfo: Retorna informações sobre um ponto específico. Parâmetros: Todos os do GetMap, só que alterando *request* para GetFeatureInfo, x, y, query_layers, info_format. Exemplo de requisição: http://127.0.0.1:8081/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.1.1&service=WMS&request=GetFeatureInfo&srs=EPSG:4326&bbox=-180,-90,180,90&format=image/png&layers=land_shallow_topo_2048,rivers&styles=,&transparent=true&width=500&height=300&query_layers=rivers&info_format=text/html&x=141&y=91&radius=10

3. GetLegendGraphic: Esta operação retorna uma imagem da legenda. Parâmetros: version, service, request, format, layer, sld. Exemplo de requisição:
   http://127.0.0.1:8081/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.1.1&service=WMS&request=GetLegendGraphic&layer=rivers&format=image/png

4. GetStyles: Retorna o SLD (*Styled Layer Descriptor*) para uma *layer*. Parâmetros: version, service, request, layers. Exemplo de requisição: http://127.0.0.1:8081/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/sld/demo.map&version=1.1.1&service=WMS&request=GetStyles&layers=rivers

5. GetCapabilities: Retorna um XML com metadados do serviço e das *layers*. Parâmetros: version, service, request. Exemplo de requisição: http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.1.1&service=WMS&request=GetCapabilities

Usualmente os mais utilizados são 1,2 e 5.

- WFS – *Web Feature Service*

Trata-se de um serviço geoespacial estabelecido pela OGC que retorna um arquivo GML, contendo informações sobre dados georreferenciados vetoriais. As possíveis operações para se realizar com o WFS são:

1. GetCapabilities - Retorna um XML com metadados do serviço e das *layers*. Parâmetros: version, service, request. Exemplo de requisição: http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.0.0&service=WFS&request=GetCapabilities

2. DescribeFeatureType – Fornece a estrutura de um *feature* (campos, etc.). Parâmetros: service, version, request, typename. Exemplo de requisição:

   http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.0.0&service=WFS&request=DescribeFeatureType&typename=rivers

3. GetFeature – Retorna o GML do *feature* em si. Parâmetros: version, service, request, typename, filter, bbox. Exemplo de requisição:
   http://127.0.0.1:8081/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.0.0&service=WFS&request=GetFeature&typename=rivers&filter=%3CFilter%3E%3CPropertyIsEqualTo%3E%3CPropertyName%3ENAME%3C/PropertyName%3E%3CLiteral%3EGreat%20Bear%3C/Literal%3E%3C/PropertyIsEqualTo%3E%3C/Filter%3E&outputFormat=GML2

- WCS – *Web Coverage Service*

  Trata-se de um serviço especial estabelecido pela OGC e que realiza as mesmas ações que o WFS, só que para dados matriciais em vez de vetoriais.

  As possíveis operações que podem ser realizadas com o WCS são:

1. GetCapabilities - Retorna um XML com metadados do serviço e das *layers*. Parâmetros: *version, service, request*. Exemplo de requisição:
   http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.0.0&service=WCS&request=GetCapabilities

2. DescribeCoverage: Fornece a estrutura de um *coverage* (*bands*, *resolution*, etc). Parâmetros: version, service, request, coverage. Exemplo de requisição:
   http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.0.0&service=WCS&request=DescribeCoverage&coverage=toronto

3. GetCoverage: Retorna o arquivo matricial em si. Parâmetros: version, service, request, coverage, crs, bbox. Exemplo de requisição: http://127.0.0.1:8081/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/config.map&version=1.0.0&service=WCS&request=GetCoverage&coverage=toronto&crs=EPSG:26917&resx=500&resy=500&format=GEOTIFF_RGB&BBOX=456850,4732100,706850,4957100

   Vale ressaltar que todos os exemplos mostrados foram criados se utilizando o *MapServer*.

O objetivo principal da implementação destes serviços é possibilitar sua utilização em quaisquer aplicações. No escopo deste projeto, tais serviços foram aplicados nas áreas de Biodiversidade e Agricultura de Precisão, o que não restringe a interoperabilidade que os serviços OGC implementados oferecem. Portanto, os serviços desenvolvidos podem ser aplicados a outras áreas do conhecimento utilizando o mesmo padrão de acesso.

Neste cenário, foram desenvolvidos sistemas servidores que encapsularam os serviços geoespaciais OGC em um ambiente próprio, com a finalidade de disseminar o desenvolvimento de SIG's de modo geral. Para atender os requisitos de uma arquitetura baseada em SOA, foram criados também sistemas clientes para facilitar a integração desses serviços nas aplicações de interesse.

2.2 Sistemas Servidores para os Serviços Geoespaciais OGC

2.2.1 MapServer

O *MapServer* [http://mapserver.org/] é também um *software* livre e *open source*, mas que foi desenvolvido utilizando-se a linguagem C. Também tem o objetivo de disponiblizar dados geoespaciais de maneira interoperável através de

padrões da OGC e disponibiliza maneiras de trabalhar com os serviços WMS, WFS e WCS.

Segundo o estudo comparativo entre ferramentas SIG publicado no VII Congresso Brasileiro de Agroinformática, o SBIAgro, o *MapServer* tem como vantagem ser um projeto mais maduro e, portanto, apresenta maior confiabilidade que o *GeoServer*. Ademais, as aplicações SIG de *MapServer* estão focadas no desenvolvimento de mapas interativos ou em operações de processamento de geometrias, específicas de uma aplicação cliente. O *MapServer* possui o melhor desempenho na velocidade de processamento de uma grande quantidade de dados georreferenciados em um SIG. (Grisi et al., 2009).

Em vista do exposto, o *MapServer* foi a ferramenta SIG escolhida para o desenvolvimento deste trabalho.

Na figura abaixo, é apresentada a estrutura de uma aplicação típica em *MapServer*. Nesta figura podemos ver como entrada os *Web Services* espaciais e também os *Data Stores*, banco de dados espaciais. Percebe-se que o fluxo de dados passa por uma configuração em *Mapfile*, e então pode ir diretamente para a aplicação CGI ou passar por uma aplicação desenvolvida em *MapScript*, para então se entregar a informação através de um servidor Apache ou IIS. Temos como possíveis saídas arquivos nos formatos SWF, SVG ou PDF, além de o simples retorno de uma imagem ou de uma imagem aliada a uma página HTML. Por fim, temos também como possibilidade de saídas o retorno dos serviços geospaciais WMS, WFS e WCS.

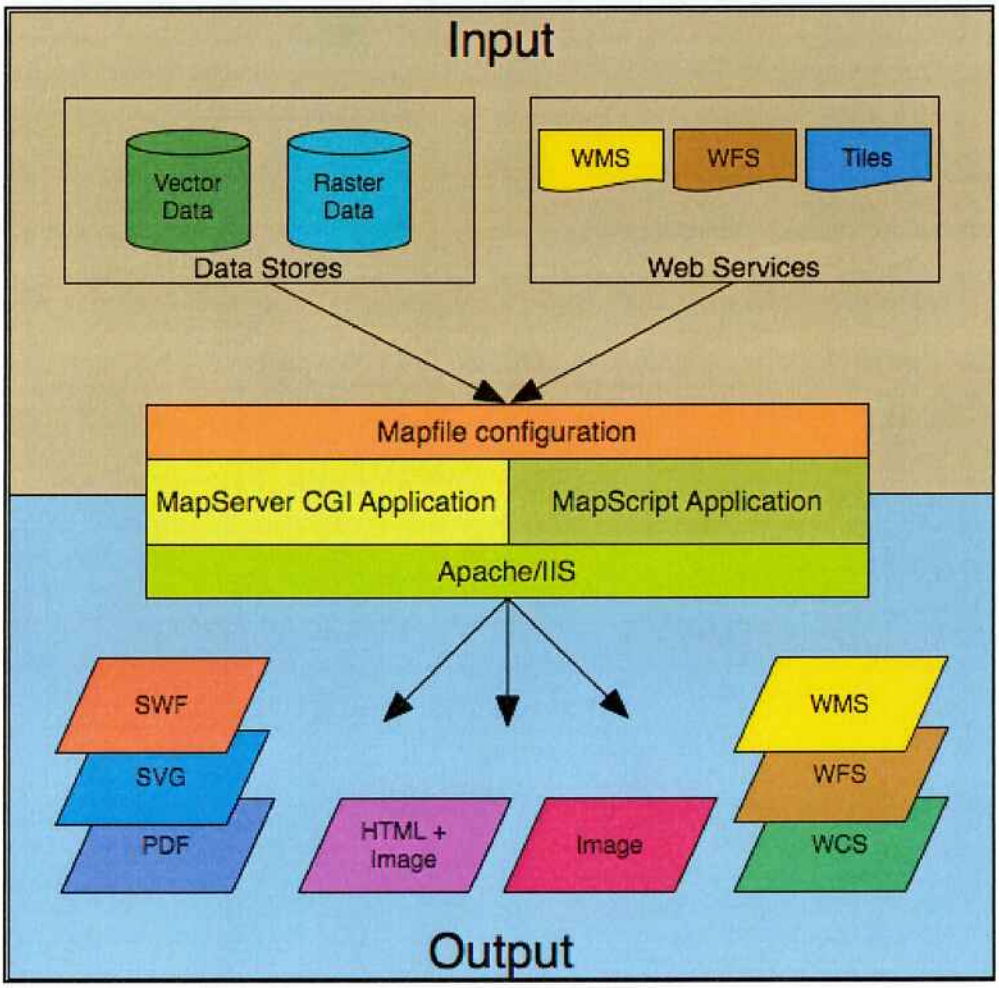

Figura 2.1 – Estrutura de uma Aplicação *MapServer* -

[http://mapserver.org/introduction.html] - Acessado: 21/11/2009

2.3 Sistemas Clientes para os Serviços Geoespaciais OGC

2.3.1 OpenLayers

O *OpenLayers* é também um *software* livre e aberto que atua no lado cliente de um sistema de mapas, consumindo requisições WMS e WFS. O *OpenLayers* foi desenvolvido em *Javascript* e AJAX, o que permite maior usabilidade e torna a experiência para o usuário mais amigável, eliminando o *overhead* das requisições síncronas entre cliente e servidor.

O *OpenLayers* permite uma integração simples com o servidor de mapas *MapServer*, que foi escolhido para o desenvolvimento deste trabalho. Por exemplo, esta parceria entre *OpenLayers* e *MapServer* permite associar mapas de uma aplicação simplificada da produção apiária no sudeste do estado de São Paulo com o mapa físico terrestre provido pelo *Google Maps*, cujo resultado foi publicado no XVII Simpósio Internacional de Iniciação Científica da USP, o SIICUSP, e está apresentado na Figura 2.2.

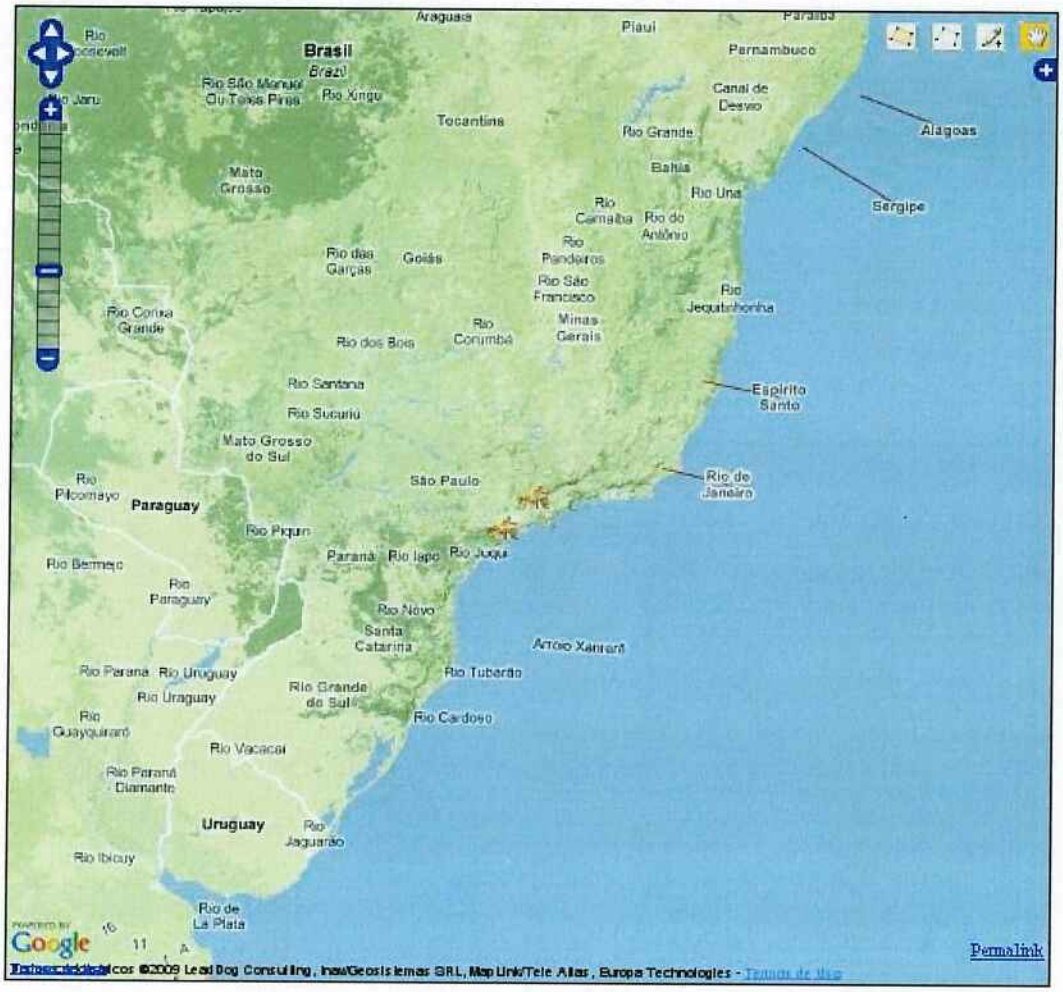

Figura 2.2 – Exemplo de Apresentação de Mapas usando o *OpenLayers* (Nogueira et al., 2009)

3 METODOLOGIA

Neste capítulo, é descrita a metodologia aplicada no desenvolvimento do projeto e o seu acompanhamento. As estratégias e abordagens definidas durante a fase de concepção do sistema estão elencadas nesta metodologia.

3.1 Metodologia de Software

A metodologia utilizada para o desenvolvimento do sistema consistiu nas seguintes etapas:

- *Especificação do Projeto*: compreende os itens a seguir:
  - *Escopo do Projeto*: Definem-se as fronteiras do sistema, bem como todos os processos envolvidos e necessidades atendidas;
  - *Especificação dos Requisitos do Sistema*: define os requisitos funcionais e não-funcionais do sistema de forma a atender o objetivo do projeto;
  - *Modelo Estático*: identifica e descreve os casos de uso e as classes do sistema;
  - *Definição da Arquitetura de Software*: descreve a Arquitetura de Software utilizada, considerando os aspectos de integração de tecnologias e os requisitos do sistema;
  - *Definição do Plano de Testes*: corresponde à elaboração dos testes para validar o sistema de acordo com a especificação de requisitos.
- *Estudo e Exploração dos Conceitos Teóricos*: estudo das técnicas ainda não conhecidas pela equipe do projeto.
- *Escolha das Ferramentas*: seleção das ferramentas de auxílio ao desenvolvimento do sistema, com base nos requisitos do sistema e na disponibilidade de uso.
- *Provas de Conceito (PoC's)*: realização de provas de conceito para análise da viabilidade de utilização e integração de algumas tecnologias ainda inexploradas na área de aplicação do sistema.

- *Desenvolvimento e Implementação*: apresenta as técnicas empregadas no desenvolvimento do sistema.
- *Testes*: execução dos testes definidos no Plano de Testes para identificar eventuais erros (bugs) e correção destes bugs a fim de validar o sistema.
- *Resultados*: apresenta os resultados gerados.

Os documentos técnicos e a monografia foram elaborados e refinados ao longo do período de desenvolvimento do Projeto de Formatura.

O controle da implementação do sistema foi realizado pelo SVN (*Subversion*), sistema de controle de versão conhecido por todos da equipe.

O gerenciamento de atividades foi realizado por meio de um cronograma compartilhado entre os membros da equipe do projeto, desenvolvido com o auxílio da ferramenta *Microsoft Project 2007*. Este cronograma está apresentado pelo Diagrama de Gantt na seção *Apêndice A* deste documento.

4 ESPECIFICAÇÃO

Nesta seção, é feita uma apresentação geral do projeto, abordando seus aspectos técnicos e funcionais, sua arquitetura de componentes e sua documentação *UML*, contendo as informações relevantes para o desenvolvimento do sistema como um todo.

4.1 Descrição do Projeto

Em linhas gerais, o projeto consiste em desenvolver funcionalidades de um Sistema de Informações Geográficas para o processamento e a apresentação de mapas. O foco principal do projeto é implementar os serviços padronizados e especificados pelo OGC, que permitem a interoperabilidade do sistema proposto para quaisquer aplicações. Este sistema foi nomeado *easyGIS*.

A apresentação de mapas possibilita a sobreposição de camadas de imagens geográficas, tais como as fotos da Terra captadas por satélite pelo *Google Maps* [http://maps.google.com.br/] e *Yahoo! Maps* [http://maps.yahoo.com/]. Assim, as aplicações que utilizarem o *easyGIS* poderão receber apoio destes provedores de mapas terrestres, facilitando a localização da imagem apresentada em qualquer ponto da Terra. Portanto, cria-se uma ferramenta capaz de auxiliar a tomada de decisão (Sistema de Apoio à Decisão, SAD, simplificado).

4.2 Arquitetura do Projeto

A arquitetura do projeto faz parte de uma arquitetura maior para um projeto de P&D patrocinado pela FINEP mencionado anteriormente, com o intuito de criar um portal para agricultura de precisão que permita aplicações de modelagem e de simulação, utilizando serviços especificados pelo OGC.

É justamente a porção que compreende os serviços OGC e os serviços de agricultura para modelagem e simulação que corresponde à arquitetura delimitada neste trabalho. O modelo da arquitetura representado pela Figura 4.1 foi concebido com base na arquitetura desenvolvida no projeto *OpenModeler* que está referenciada na Tese de Doutorado de Santana (2009).

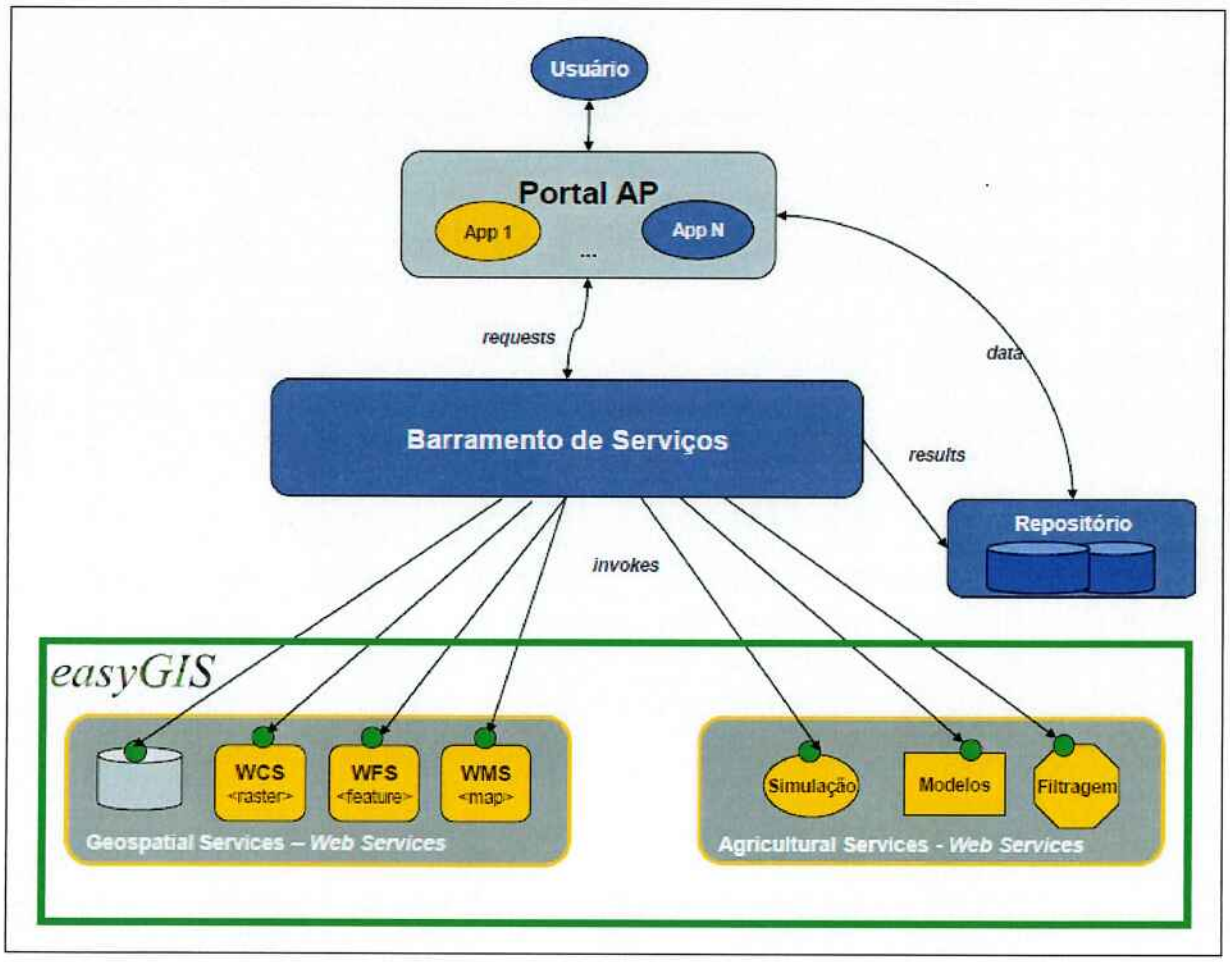

Figura 4.1 – Arquitetura FINEP do Projeto de P&D (Santana, 2009)

Esta arquitetura estabelece os principais serviços geoespaciais (*WMS*, *WFS* e *WCS*) para a manipulação de mapas, função principal do *easyGIS*. Nota-se a persistência das informações processadas por estes serviços em um Banco de Dados Espacial, no caso, o *Postgis* (uma extensão do banco de dados *PostgreSQL* que fornece capacidades geoespaciais).

O servidor de mapas escolhido, conforme justificado anteriormente, foi o *MapServer* e a aplicação cliente utilizada, o *OpenLayers*. Tais softwares foram escolhidos por se mostrarem mais adequados ao *easyGIS* (Grisi et al., 2009).

Convém explicitar que não haveria impedimentos para se optar por outras ferramentas e tecnologias. Como exemplos de outros softwares, o *Spatial MySQL* atenderia as necessidades consumidas pelo *Postgis*. No caso do *MapServer*, poderia ser substituído pelo *GeoServer*. Por fim, o *OpenLayers* poderia ceder lugar para o *KaMap*.

O *Postgis* foi escolhido, pois se trata da solução *free* e *open source* mais madura e líder do mercado, assim como ocorre para o *MapServer* e o *OpenLayers*.

4.3 Requisitos Funcionais

Segue a descrição dos requisitos funcionais do trabalho.

4.3.1 Uso de Serviços de SIG

A habilidade para lidar com mapas, dados georreferenciados e coordenadas espaciais são fortes exigências na agricultura de precisão, em virtude da natureza do problema estar basicamente no entendimento da relação entre o desenvolvimento de um talhão, que é uma área cultivável, com seus respectivos dados ambientais e geoespaciais (Bongiovanni, 2004).

Em vista dos conceitos aplicados em SOA no projeto, a integração dos Sistemas de Informação Geográfica deve ser fundamentada em serviços no tratamento de mapas e dados georeferenciados. A principal organização que elabora padrões nas áreas geoespaciais, o Open Geospatial Consortium definiu um conjunto de serviços que provêem as principais funcionalidades dos SIG's, tais como os OGC *Web Services*. Destes serviços, os que serão desenvolvidos neste projeto são o WMS (*Web Map Service*), o WFS (*Web Feature Service*) e o WCS (*Web Coverage Service*).

A adoção dessas normas SIG definidas pelo OGC é imprescindível para instaurar e manter a interoperabilidade dos serviços desenvolvidos, exigência primordial para a validação do sistema.

4.3.2 Serviços para Tratamento de Mapas

A funcionalidade principal dos serviços para tratamento de mapas é descrita pelo documento de especificação de implementação do WMS da *OpenGIS* [http://www.opengeospatial.org/standards/wfs]. Este documento define exatamente três operações (*GetMap*, *GetCapabilities* e *GetFeatureInfo*) que permitem a visualização de dados georeferenciados no formato de mapa, a partir de parâmetros dimensionais e geográficos bem definidos no acordo estabelecido entre cliente e servidor. A informação necessária para realizar serviços WMS são os dados georeferenciados provenientes de diversas fontes remotas e heterogêneas, aplicando o paradigma de dados distribuídos.

As três operações que constituem o WMS são responsáveis pelas seguintes funções:

- *GetCapabilities* obtém os metadados de serviço de um servidor;
- *GetMap* gera um mapa de acordo com parâmetros dimensionais e geográficos previamente definidos;
- *GetFeatureInfo* obtém informações sobre as características (*features*) particulares aplicados num mapa.

Com estas operações, o WMS manipula informações geográficas, tipicamente na forma de dados georreferenciados, produzindo mapas com características bem definidas.

4.3.3 Serviços para Tratamento de Características

O *OpenGIS Web Feature Service* permite que o cliente requisite e atualize dados geoespaciais codificados em *Geography Markup Language* (GML), potencialmente originados de diferentes plataformas.

A especificação do WFS define as interfaces para as operações na manipulação de dados com características geográficas. As habilidades das operações de manipulação são:

- Enviar ou receber dados geoespaciais;
- Criar dados geoespaciais;
- Remover dados geoespaciais;
- Atualizar dados geoespaciais.

4.3.4 Serviços de Cobertura

O *OpenGIS Web Coverage Service* (WCS) define uma interface padrão que permite ao cliente requisitar e obter informações geográficas do servidor sob forma de coberturas, sendo estes compostos por valores ou propriedades associados a localizações geográficas espaçadas de forma regular, podendo também conter informação temporal regular ou irregularmente espaçado.

As coberturas geralmente são obtidas de três tipos de fontes:

- Imagem de satélite;
- Matriz de elevação digital;
- Mapa de temperatura.

Por meio deste serviço, o cliente pode realizar três operações no servidor:

- *GetCapabilities*: retorna um arquivo XML com os metadados do serviço e das coberturas oferecidos;
- *DescribeCoverage*: retorna um arquivo XML com detalhes da cobertura especificada;

- *GetCoverage*: retorna a cobertura especificada no formado de arquivo solicitado.

4.3.5 Vantagem da Adoção de Padrões

Os principais aspectos são:

- Os pacotes de software disponíveis normalmente têm a transmitir arquivos grandes com mapas e outras imagens para o usuário final, que serão substituídas por arquivos XML, reduzindo exigências da rede ou melhorar o desempenho da rede, já que transmiti-los normalmente requisita muito menos do que transmitir imagens;
- A interoperabilidade da solução aumentará por aplicação de normas;
- Arquitetura proposta poderá aumentar as oportunidades para futuras parcerias nestes domínios de investigação;
- Dispensar pacotes externos para tratamento de dados georeferenciados.

Portanto, a proposta aqui apresentada é abrangente e pode ser estendida a outros problemas similares.

4.3.6 Prover Integração de Serviços com Portais de Agricultura de Precisão

Ao oferecer serviços de SIG, a solução é capaz de ignorar o uso de diferentes pacotes de *software* que, muitas vezes, exigem peritos, com a finalidade de avaliar os seus modelos e auxiliar a tomada de decisão nas atividades correlatas (Santana, 2009).

4.3.7 Avaliação da Solução easyGIS

Uma vez integrados os serviços OGC por meio de interfaces com o usuário, criou-se a necessidade de avaliar a solução proposta, o *easyGIS*, relacionando-a com algoritmos de modelagem, sobretudo das áreas de Agricultura de Precisão e de Biodiversidade.

O uso de inteligência artificial e outras técnicas algorítmicas avançadas sobre o conjunto de dados georreferenciados é realizado por algoritmos de modelagem da produtividade de uma propriedade rural que produzem informações de fundamental importância à tomada de decisão por parte do agricultor, por exemplo.

Em conformidade com os serviços do SIG, o *easyGIS* deve permitir a sobreposição de camadas para uma representação mais inteligente do mapa e para a comparação dos resultados obtidos antes e depois do processamento do algoritmo. A arquitetura da solução completa na qual os algoritmos estão integrados é apresentada no capítulo 5 deste documento.

Os dois algoritmos selecionados para avaliar a solução *easyGIS* são:

- *Algoritmo 1*: Mapa de produtividade processada por uma máquina agrícola e aplicação do algoritmo *Filtering* para eliminação de dados inconsistentes;
- *Algoritmo 2*: Mapa de ocorrências de espécies *Furcata boliviana* e aplicação do algoritmo genético GARP.

A descrição mais detalhada destes algoritmos encontra-se no capítulo 5 referente ao desenvolvimento das aplicações web para cada área de estudo.

4.4 Requisitos Não-Funcionais

Segue a descrição dos requisitos não-funcionais do trabalho.

4.4.1 Infraestrutura

A infraestrutura necessária para o funcionamento do projeto deve possuir no mínimo:

- Hardware para o servidor: HP ProLiant DL380 G5 Storage Server;
- Software: sistema operacional Windows Server 2003; Java Web e EE compatíveis com especificações J2EE, PHP, MS4W, OpenLayers, PostgreSQL, PostGIS.

4.4.2 Desenvolvimento Colaborativo

O desenvolvimento do software deve ser aberto para que seja possível o desenvolvimento por uma comunidade *Open Source*, sendo hospedada em um repositório aberto no *SourceForge*, de modo que o projeto possa ser colaborativamente documentado em um sistema de *Wiki*.

4.4.3 Usabilidade

O sistema deve possuir uma boa usabilidade para os usuários de modo que facilite a iteração homem-máquina. Como forma de verificação desse requisito, deve ser realizada uma experimentação por alguns usuários e deles obter notas e opiniões quanto à facilidade no uso deste software.

4.4.4 Desempenho

O sistema deve suportar um número pequeno de usuários (em média, são 100 acessos a cada instante de tempo), pois é previsto que este software não será utilizado em grande escala. Para realizar a verificação desse requisito, no final do projeto, deverá ser realizado um teste de estresse padrão em demanda sobre o servidor que carrega a aplicação.

4.4.5 Segurança de acesso

Como os dados dos usuários podem ser sigilosos, pois, em alguns casos, tratam-se de informações confidenciais de importância comercial e estratégica, deve-se utilizar de recursos de criptografia na transferência dos dados e evitar o uso malicioso na autenticação de usuários.

4.4.6 Disponibilidade

O sistema deve possuir alta disponibilidade, de modo que não haja a indisponibilidade do sistema durante o horário comercial padrão. Este requisito é de grande importância, visto que se trata de um sistema que servirá de apoio nas decisões diárias do agricultor. Como a solução final do projeto suportado pela FINEP será hospedado no CCE-USP e também em um provedor externo sob responsabilidade da empresa Jacto S/A, espera-se que o sistema tenha disponibilidade 24/7.

5 IMPLEMENTAÇÃO

Neste capítulo, são detalhados os algoritmos desenvolvidos, a aplicação das tecnologias e a utilização das ferramentas apresentadas no escopo do projeto. Seguindo a metodologia discutida, as duas partes do projeto foram sendo desenvolvidas paralelamente e, através da realização de provas de conceitos com os protótipos citados, foi possível chegar, de forma evolutiva, ao produto final da aplicação.

5.1 Manipulações de Mapas

A principal atividade deste módulo foi realizar uma análise de cada serviço OGC e as possibilidades do mercado para ferramentas de servidor e de cliente. Assim, um servidor era escolhido e avaliado de acordo com seu desempenho para atender às especificações de cada serviço OGC.

Por exemplo, para o WFS foram testadas as funções *GetCapabilities*, *GetFeatureType* e *GetFeature*, bem como as possíveis aplicações de cada uma delas e seu relacionamento com a aplicação cliente. Neste ponto, também foram definidas aplicações mais concretas, como as relativas ao OpenModeller e ao BDD.

O openModeller (doi:10.1016/j.ecoinf.2007.12.003) é um software que faz o processamento de informações de espécies biológicas e de dados abióticos (e.g.: temperatura, umidade), aplicando algoritmos a estes dados para obter modelos sobre a localização das espécies estudadas. Estas informações são devolvidas em formato geoespacial, gerando imagens que mostram a probabilidade de ocorrência das espécies em uma determinada região de estudo. O BDD é uma solução de digitalização, análise e publicação de dados de biodiversidade. Ela permite ao usuário cadastrar dados de espécimes e interação entre elas para posterior visualização.

Para esta atividade, foi necessário montar cada servidor relativo a cada serviço e também verificar as possíveis aplicações de cada um, além de explorar as

possibilidades que os clientes atuais *free* e *open source*, principalmente o *OpenLayers*, podem oferecer. Também foi assumido o papel de ajudar a instalar a opção de mapas no BDD e o de listar novas possibilidades de saída de dados para o openModeller.

Para esta atividade, foi necessário estudar e definir as aplicações dos serviços geoespaciais. Foram realizados diversos testes experimentais e provas de conceito, até que o foco foi definido nas aplicações do openModeller e do BDD. Neste ponto, surgiu o conceito de Banco de Dados Espacial (*Spatial Database*). Um Banco de Dados Espacial, em linhas gerais, é um banco de dados comum que adiciona a funcionalidade de manipular informações espaciais vetoriais, isto é, as primitivas espaciais: Pontos, Linhas e Polígonos (Soares, 2003).

A partir do Banco de Dados Espacial, ocorre uma comunicação com o servidor de mapas, que fornece os mapas através de serviços para a aplicação cliente. Este processo de transações configura a arquitetura apresentada pela figura 5.1.

Figura 5.1 – Arquitetura de Apresentação de Mapas

Nesta arquitetura, o *Banco de Dados Espaciais* é o *Postgis*, uma extensão ao banco de dados *PostgreSQL* que fornece capacidades espaciais. O servidor de mapas é o *MapServer* e a aplicação cliente é *OpenLayers*.

Atualmente, tal arquitetura se encontra implementada no projeto do BDD.

5.2 Aplicações de Mapas Tridimensionais

Após a implementação do WMS, foi possível estender sua funcionalidade de apresentação de mapas bidirecionais nos formatos IMG, JPG, PNG e TIFF para

aplicações que lidam com mapas em três dimensões (Chapman; Muñoz; Koch, 2005). A fim de atender este objetivo, foi escolhida uma das aplicações para mapas em 3D mais conhecidas: *Google Earth* [http://earth.google.com/].

No início do projeto, quando o *GeoServer* era o servidor dos serviços OGC utilizado, as aplicações para o *Google Earth* dos mapas requisitos já estavam implementadas pelo próprio *GeoServer*. Neste caso, o *GeoServer* gerava automaticamente o arquivo KML/KMZ pronto para ser lido e executado pelo *Google Earth*. Basicamente, este arquivo obtinha a localização exata do mapa requisitado no planeta e aplicava o mapa em sobreposição à superfície terrestre tridimensional. A informação da localização do mapa no planeta estava embutida nos dados geoespaciais que o descrevem.

O fato desta aplicação ser um recurso já pronto do *GeoServer* limitava quaisquer modificações para usos mais específicos. Obviamente, era possível modificar o *GeoServer* para atender as necessidades específicas das aplicações em 3D, dado que o *GeoServer* é um servidor de mapas *open source*. No entanto, foram encontradas muitas dificuldades para uma implementação customizada das aplicações 3D usando o *GeoServer*.

O *MapServer* não possui um recurso para aplicações de mapas em três dimensões. Porém, uma análise da estrutura dos arquivos KML/KMZ reconhecidos pelo *Google Earth* mostrou que é possível fazer uma requisição HTTP, permitindo integrar o serviço WMS ao *Google Earth*.

A estratégia para o desenvolvimento da integração entre o WMS e o *Google Earth* foi definida em módulos da seguinte maneira:

1) Implementar um leitor das informações geoespaciais de um mapa;
2) Implementar um gerador de arquivos KML/KMZ;
3) Definir as requisições WMS permitidas usando o *MapServer*;
4) Integrar os três itens anteriores em um único sistema para aplicação de um mapa requisitado via WMS no *Google Earth*.

O primeiro módulo consistiu em obter as informações geográficas utilizadas pela requisição WMS para obter a imagem do mapa e pelo *Google Earth* para localizar a posição do mapa na superfície terrestre.

Primeiramente, faz-se uma requisição *GetCapabilities* do mapa de interesse por meio do WMS. Este mapa pode estar em qualquer formato que seja reconhecido

pelo WMS. Esta requisição retorna um arquivo XML com as principais informações do mapa requisitado, em uma estrutura padronizada pelo serviço WMS.

A requisição HTTP utilizada para o *GetCapabilities* por meio do WMS é apresentada pela Tabela 5.1.

```
MAPSERVER_DOMAIN/cgi-bin/mapserv.exe?
    map=MAPFILE_PATH\arquivo_mapfile.map&
    SERVICE=servico_geoespacial&
    VERSION=versao_do_tipo_de_requisicao&
    REQUEST=tipo_de_requisicao
```

Tabela 5.1 – Requisição HTTP do *GetCapabilities* implementado pelo WMS

Na operação *GetCapabilities*, os parâmetros utilizados na requisição HTTP são:

- *MAPSERVER_DOMAIN* – domínio no qual foi instalado o *MapServer*;
- *MAPFILE_PATH* – caminho no qual se encontra o arquivo de configuração *MapFile*;
- *arquivo_mapfile* – arquivo de configuração do *MapServer* para acesso a informações geográficas;
- *serviço_geoespacial* – WMS;
- *versao_do_tipo_de_requisicao* – 1.1.1;
- *tipo_de_requisicao* – GetCapabilities.

A estrutura do arquivo de resposta da requisição *GetCapabilities* do WMS com as informações geográficas relevantes está simplificada e descrita pela Tabela 5.2.

```
<WMT_MS_Capabilities>
    <Layer>
        <Name>nome_da_camada_principal</Name>
        <Title>titulo_da_camada_principal</Title>
        <SRS>sistema_de_referencia_geoespacial</SRS>
        <LatLonBoundingBox minx="valor" miny="valor"
                           maxx="valor" maxy="valor" />
```

```
        <BoundingBox SRS="sistema_de_referencia_geoespacial"
                     minx="valor" miny="valor" maxx="valor"
                     maxy="valor" />
        <Layer queryable="valor" opaque="valor" cascaded="valor">
            <Name>nome_da_camada_01</Name>
            <Title>titulo_da_camada_01</Title>
            <SRS>sistema_de_referencia_geoespacial_01</SRS>
            <LatLonBoundingBox minx="valor" miny="valor"
                               maxx="valor" maxy="valor" />
        </Layer>
        <Layer> ... </Layer> ... <Layer> ... </Layer>
      </Layer>
    </Capability>
</WMT_MS_Capabilities>
```

Tabela 5.2 – Estrutura Simplificada da Requisição *GetCapabilities* do WMS

Esta estrutura simplificada descreve as camadas que compõem o mapa requisitado e está padronizado pelo WMS para qualquer tipo de mapa reconhecido pelo serviço. Cada mapa possui uma camada principal e uma ou mais camadas internas. A descrição do mapa compreende os seguintes itens:

- *WMT_MS_Capabilities* – consiste nas informações do mapa obtido pela requisição *GetCapabilities* do WMS;
- *Layer* – camada principal do mapa, contendo:
    - *Name* – nome da camada principal;
    - *Title* – título da camada principal/
    - *SRS* – Sistema de Referência Geoespacial (*Spatial Reference System*) utilizado pela camada principal;
    - *LatLonBoundingBox*

Descrição: delimitadores geográficos que identificam a posição da camada na superfície terrestre usando valores do SRS da camada.

- **Propriedades**:
    - *minx* – longitude mínima da camada principal;
    - *miny* – latitude mínima da camada principal;
    - *maxx* – longitude máxima da camada principal;
    - *maxy* – latitude máxima da camada principal.

- *BoundingBox*
  - **Descrição**: delimitadores geográficos que identificam a posição da camada na superfície terrestre usando o SRS da *BoundingBox*.
  - **Propriedades**:
    - *SRS* – Sistema de Referência Geoespacial da *BoundingBox*;
    - *minx* – longitude mínima da camada principal;
    - *miny* – latitude mínima da camada principal;
    - *maxx* – longitude máxima da camada principal;
    - *maxy* – latitude máxima da camada principal.
- *Layer*:
  - **Descrição**: camadas internas que compõem a camada principal.
  - **Propriedades**:
    - *queryable* – indica se o servidor da camada permite requisições *GetFeatureInfo* por meio do WMS
      - **Valores Permitidos:** 0, false, 1, true
    - *cascaded* – indica a quantidade de vezes que a camada foi retransmitida por servidores de mapas
      - **Valores Permitidos:** 0, inteiro positivo
    - *opaque* – indica se a camada possui uma área de dados significativa
      - **Valores Permitidos:** 0, false, 1, true
  - **Tags**:
    - *Name* – nome da camada interna;
    - *Title* – título da camada interna;
    - *SRS* – Sistema de Referência Geoespacial utilizada pela camada interna;
    - *LatLonBoundingBox* – delimitadores geográficos descritos no SRS da camada interna.

O primeiro módulo deve implementar um leitor das informações geoespaciais de um mapa. Este mapa deve ter sido obtido como arquivo de

resposta da requisição *GetCapabilities* do mapa de interesse por meio do WMS, usando o *MapServer*. Assim, foi preciso desenvolver um leitor deste arquivo XML que retorne os dados relevantes identificados anteriormente.

O programa desenvolvido para esta tarefa denomina-se *GetCapabilitiesReader* e está implementado em Java. O arquivo *GetCapabilities* deve estar localizado em uma pasta padrão do projeto, utilizada exclusivamente para este tipo de arquivo. Identificado pelo nome, este arquivo é lido e percorrido até o nó que contém as informações do mapa (*WMT_MS_Capabilities*). A partir deste nó, o programa identifica a camada principal (*Layer*) que é sempre o nó-filho direto do *WMT_MS_Capabilities*. Em seguida, são armazenadas as informações da camada principal, incluindo as características dos nós-filhos referentes às camadas internas. Todas as informações armazenadas pelo *GetCapabilitiesReader* estão identificadas pela Tabela 5.2.

O segundo módulo consiste em um gerador de arquivos KML/KMZ que são extensões reconhecidas pelo *Google Earth*. A extensão KML deriva de *Keyhole Markup Language* e é usado para exibir dados geoespaciais em um navegador da Terra. O KMZ é a versão compactada do KML e deriva de KML-Zipped. A estrutura desses formatos de arquivo é proveniente do XML, o que facilita a compreensão e a análise das informações neles contidas. (Conroy et al., 2008)

O gerador de arquivos KML/KMZ denomina-se *KMLWriter* e também está implementado em Java, para facilitar a futura integração dos programas desenvolvidos para aplicações de mapas em 3D. Para facilitar a implementação do *KMLWriter*, foi definida uma estrutura padrão do formato KML para atender a necessidade de integração entre o WMS e as aplicações 3D. Na Tabela 5.3 é possível identificar os principais elementos desta estrutura.

```
<Folder>
  <GroundOverlay>
    <name>nome_da_sobreposicao_de_imagem</name>
    <Icon>
      <href><![CDATA[requisicao_GetMap_WMS]]></href>
    </Icon>
    <viewBoundScale>valor_padrao</viewBoundScale>
```

```
            <LatLonBox>
              <north>maxy</north>
              <south>miny</south>
              <east>maxx</east>
              <west>minx</west>
            </LatLonBox>
          </GroundOverlay>
          <LookAt id="identificador_de_visualização">
            <longitude>longitude_de_visualizacao</longitude>
            <latitude>latitude_de_visualizacao</latitude>
            <altitude>altitude_padrao</altitude>
            <range>alcance_de_visualizacao</range>
            <tilt>valor_padrao</tilt>
            <heading>valor_padrao</heading>
            <altitudeMode>valor_padrao</altitudeMode>
          </LookAt>
        </Folder>
```

Tabela 5.3 – Estrutura dos Arquivos gerados pelo *KMLWriter*

Note que, na estrutura do arquivo gerado pelo *KMLWriter*, existe uma requisição *GetMap* pelo serviço WMS no interior do nó *GroundOverlay*, justamente para a sobreposição de uma imagem de um mapa na superfície terrestre. Logo, é necessário definir um padrão para este tipo de requisição, a fim de evitar que o usuário tenha que se preocupar com questões de configuração, normas ou outros detalhes muito específicos da aplicação. O terceiro módulo trata exatamente desta questão, ressaltando o fato de que será utilizado o servidor de mapas *MapServer*.

A requisição HTTP utilizada para o *GetMap* por meio do WMS é apresentada pela Tabela 5.4.

```
MAPSERVER_DOMAIN/cgi-bin/mapserv.exe?
    map=MAPFILE_PATH\arquivo_mapfile.map&
    SERVICE=servico_geoespacial&
    VERSION=versao_do_tipo_de_requisicao&
    REQUEST=tipo_de_requisicao&
```

```
LAYERS=nome_da_camada&
SRS=sistema_de_referencia_geoespacial&
BBOX=minx,miny,maxx,maxy&
WIDTH=largura_da_imagem&
HEIGHT=altura_da_imagem&
FORMAT=formato_do_arquivo_de_saida
```

Tabela 5.4 – Requisição HTTP do *GetMap* implementado pelo WMS

Na operação *GetMap*, os parâmetros utilizados na requisição HTTP recebem, por padrão, os seguintes valores:

- *MAPSERVER_DOMAIN* – domínio no qual foi instalado o *MapServer*;
- *MAPFILE_PATH* – caminho no qual se encontra o arquivo de configuração *MapFile*;
- *arquivo_mapfile* – arquivo de configuração do *MapServer* para acesso a informações geográficas;
- *serviço_geoespacial* – WMS;
- *versao_do_tipo_de_requisicao* – 1.3.0;
- *tipo_de_requisicao* – GetMap.

Por fim, o último módulo integra os três módulos anteriores em um único sistema. Neste sistema, as entradas são um arquivo de resposta da requisição *GetCapabilities* e um arquivo de configuração *MapFile*, referentes ao mapa de interesse. A saída deste sistema é um arquivo KML, reconhecido por qualquer navegador do planeta Terra, como o *Google Earth*.

5.3 Algoritmos de Modelagem

Paralelamente à implementação do sistema de apresentação de mapas, outra arquitetura foi projetada para atender a necessidade do processamento de dados geoespaciais por meio de algoritmos de modelagem.

Esta arquitetura está esquematizada pela Figura 5.2 e evidencia a utilização de um Banco de Dados Geoespaciais para a persistência dos dados processados pelos algoritmos.

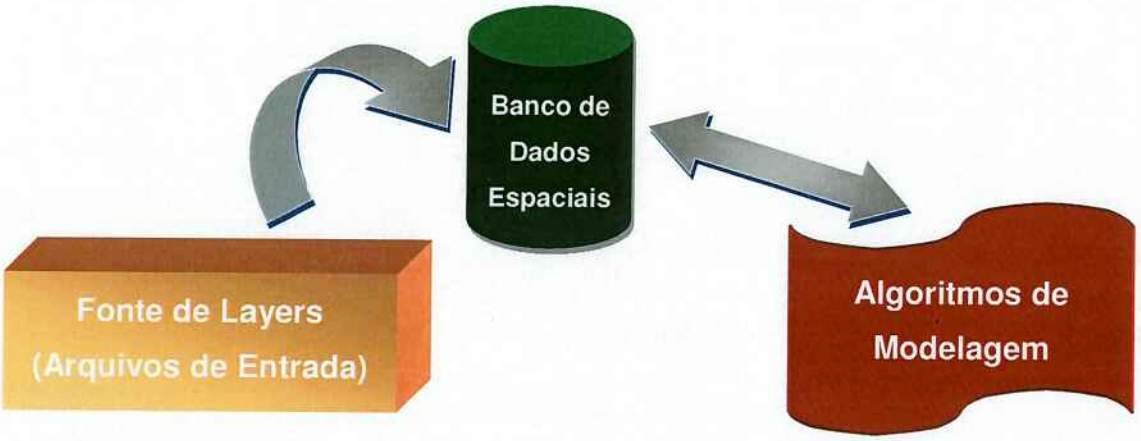

Figura 5.2 – Arquitetura de Processamento de Dados Geoespaciais por Algoritmos de Modelagem

Neste projeto, foram escolhidos dois algoritmos de modelagem em áreas distintas, justamente para avaliar a interoperabilidade garantida pela solução:

- *Filtering*;
- GARP (*Genetic Algorithm for Rule-set Production*).

5.3.1 Filtering

Este algoritmo foi desenvolvido com o propósito de realizar a filtragem de dados de monitores de produtividade. A metodologia de filtragem consiste na identificação, caracterização e remoção de erros de mapas de rendimento gerados por monitores de produtividade, instalados em máquinas colhedeiras equipadas com receptores de *Global Positioning System*, GPS, e sensores de produtividade (Murakami, 2006). Esta metodologia foi desenvolvida por Menegatti (2002) (Molin; Menegatti, 2002) na Escola Superior de Agricultura Luiz de Queiroz da Universidade de São Paulo.

No processo de filtragem de dados, o material inicial é composto por dados brutos que provêm diretamente do monitor de produtividade, quando os arquivos são

gravados em texto, ou do programa específico de cada monitor, quando é gerado em código, e então convertido em formato texto (Molin; Menegatti, 2002).

Os arquivos com dados brutos podem conter erros grosseiros de posicionamento, de produtividade nula ou ausente, de interpretação de largura de plataforma, de umidade nula ou valor ausente de umidade, de distância nula entre pontos, de intervalo de enchimento além de valores discrepantes de produtividade (Molin; Menegatti, 2002). A Figura 5.3 mostra o fluxograma que representa as etapas de aplicação do processo de filtragem (Molin; Menegatti, 2002).

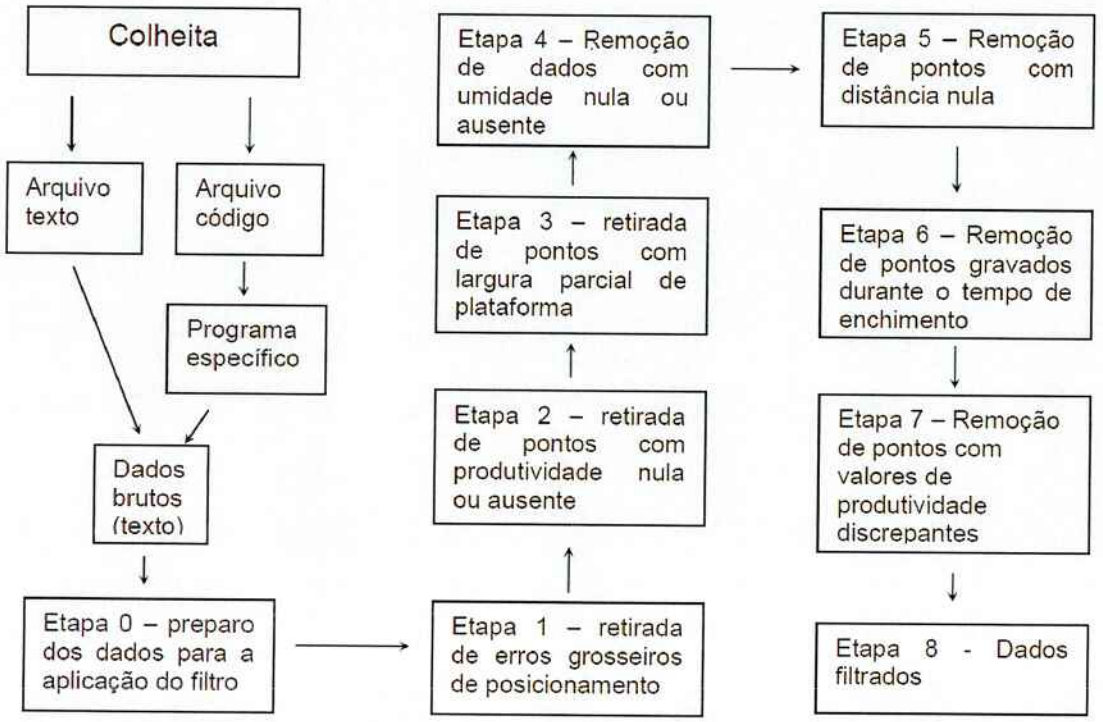

Figura 5.3 – Fluxograma de Etapas de Aplicação do Processo de Filtragem dos Dados Brutos (Molin; Menegatti, 2002)

Originalmente, as etapas de aplicação do processo de filtragem eram executadas com auxílio de apenas uma planilha de cálculos. Assim, foi desenvolvido por Murakami (2006) um protótipo que integrava o algoritmo *Filtering* à uma aplicação *web*, disponibilizando-o na forma de serviço reutilizável independente de tecnologia.

O *Filtering* envolve diversas funcionalidades de Agricultura de Precisão no que diz respeito à qualidade dos dados manipulados, à interpretação e à análise de correlação de mapas para tomada de decisão (Murakami, 2006).

5.3.2 GARP

O projeto openModeller [http://openmodeller.sourceforge.net/] é um ambiente computacional multiplataforma que gera modelos de distribuição potencial de espécies, o que pode auxiliar, por exemplo, na indicação de áreas prioritárias para conservação da biodiversidade, na avaliação do potencial de invasão de espécies exóticas, no estudo de impactos de mudanças climáticas na biodiversidade, entre outras [http://www.openmodeller.cria.org.br/]. Dentre vários algoritmos suportados pelo openModeller, este trabalho utilizou o *Genetic Algorithm for Rule-set Production* (GARP) como gerador do modelo a ser representado no sistema de apresentação de mapas.

O GARP é um algoritmo genético para processamento de regras que foi adaptado para realizar modelagem de nicho ecológico. Este algoritmo utiliza conceitos de nicho ecológico para definir uma população e, a partir daí, processar as regras que definem o nicho da população ou da espécie em estudo. Como o algoritmo tenta encontrar o nicho de uma espécie entre diversos *layers* ambientais, as regras a que o GARP se refere estão relacionadas a estes *layers*. Uma regra, por exemplo, seria: a temperatura ser inferior a 40° e superior a 10° Celsius (Santana, 2009).

Desse modo, o algoritmo utiliza informações sobre condições ambientais e disposição geográfica conhecida das espécies para gerar modelos que podem ser projetados em mapas de distribuição de espécies para estudos de biodiversidade, conforme apresentado pela Figura 5.4 [http://openmodeller.sourceforge.net/].

Por fim, o modelo gerado pelo openModeller é persistido no banco de dados espaciais e o sistema de apresentação de mapas exibe os resultados obtidos a partir dessas informações.

Figura 5.4 – Geração do modelo de biodiversidade pelo openModeller [http://openmodeller.sourceforge.net/]

5.4 Aplicações Web para Sistemas Geoespaciais

5.4.1 GeoServer: Visualização automática de arquivo GeoTiff

Uma das primeiras tarefas que tivemos no trabalho foi a visualização automática de um arquivo do tipo *GeoTiff*. O objetivo é um usuário leigo na área de sistemas de informação geográficos poder ter acesso ao arquivo *GeoTiff* produzido pelo *OpenModeller* sem a necessidade de trabalhar com um software SIG.

Em um primeiro instante, chegamos a constatação de que o *Geoserver* era capaz de exibir tal tipo de arquivo, porém para isso era necessário seguir alguns passos em sua interface do usuário, como segue:

- Página inicial do *Geoserver* (vide Figura 5.5)

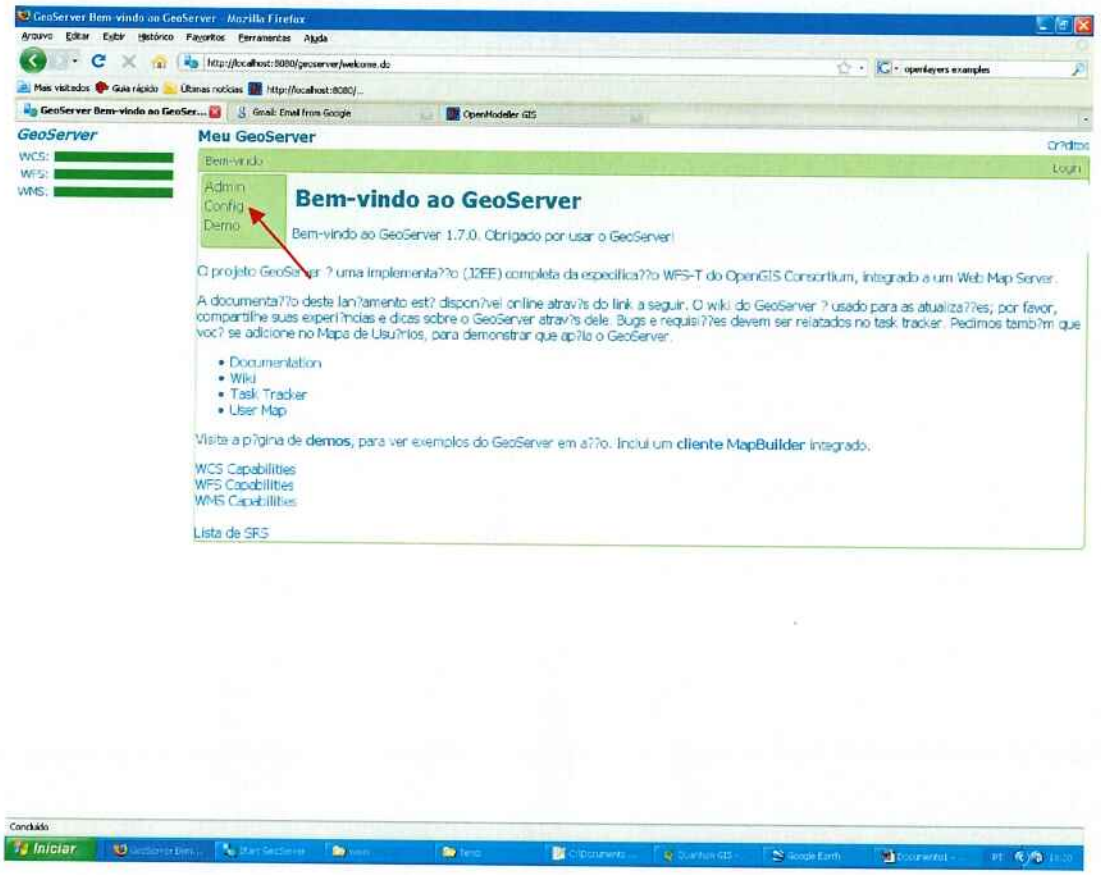

Figura 5.5 – Home do *GeoServer*

- Clicar em *Config*.
  - Entrar com usuário e senha (vide Figura 5.6)
  - Default: *admin // geoserver*

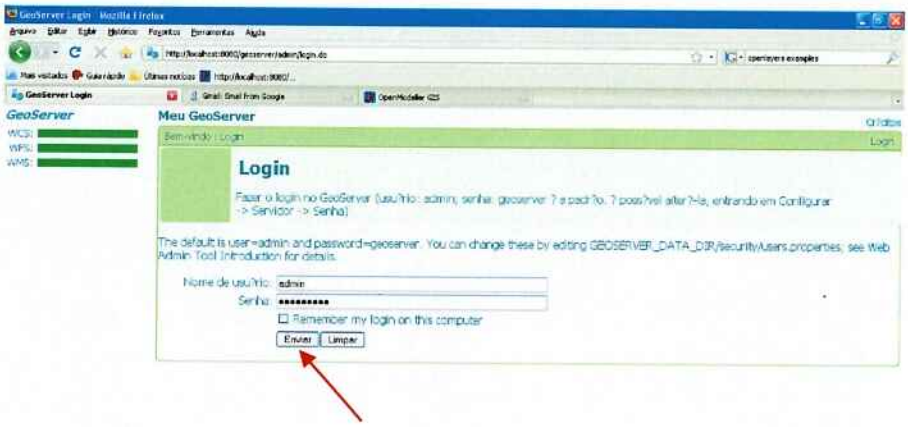

Figura 5.6 – Autenticação do Usuário no *GeoServer*

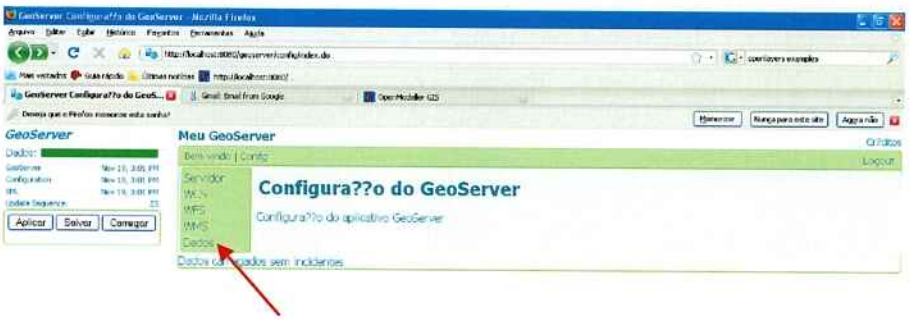

Figura 5.7 – Home do Módulo de Configuração do *GeoServer*

- Clicar em *Dados* (vide Figura 5.7)
- Clicar em *CoverageStores* (vide Figura 5.8)

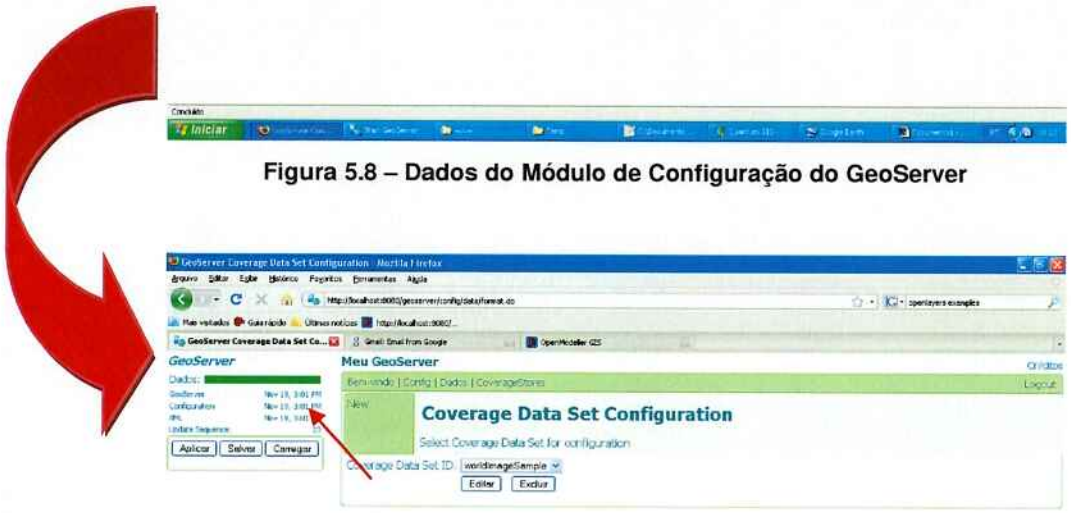

Figura 5.8 – Dados do Módulo de Configuração do GeoServer

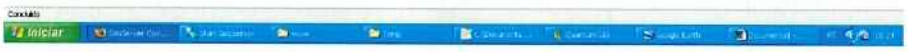

Figura 5.9 – Adicionar Dados no Módulo de Configuração do *GeoServer*

- Clicar em *New* (vide Figura 5.9)
- Selecionar tipo de arquivo *GeoTIFF* e definir o nome da *layer* (por exemplo, *Layer* "*raster*")
- Clicar em Novo (vide Figura 5.10)

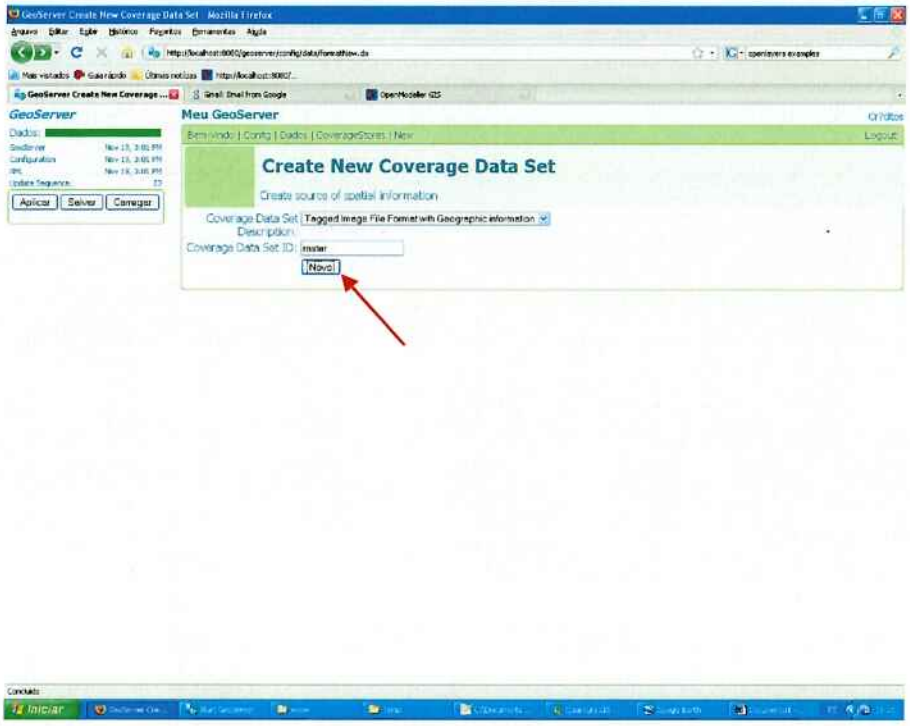

Figura 5.10 – Cadastro de Nova Camada de Dados

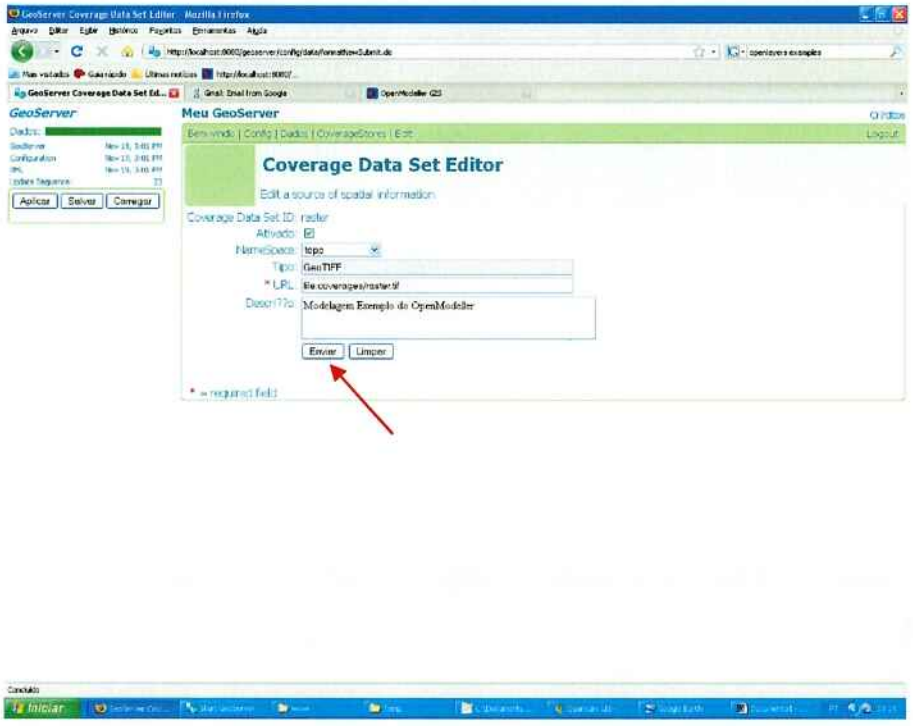

Figura 5.11 – Envio da Camada de Dados ao Servidor

- Informar a localização (*URL*) do arquivo *GeoTIFF*
- Clicar em *Enviar* (vide Figura 5.11)
- Página de edição de metadados. Manter todos os valores-padrão.
- Clicar em *Enviar* (vide Figura 5.12)

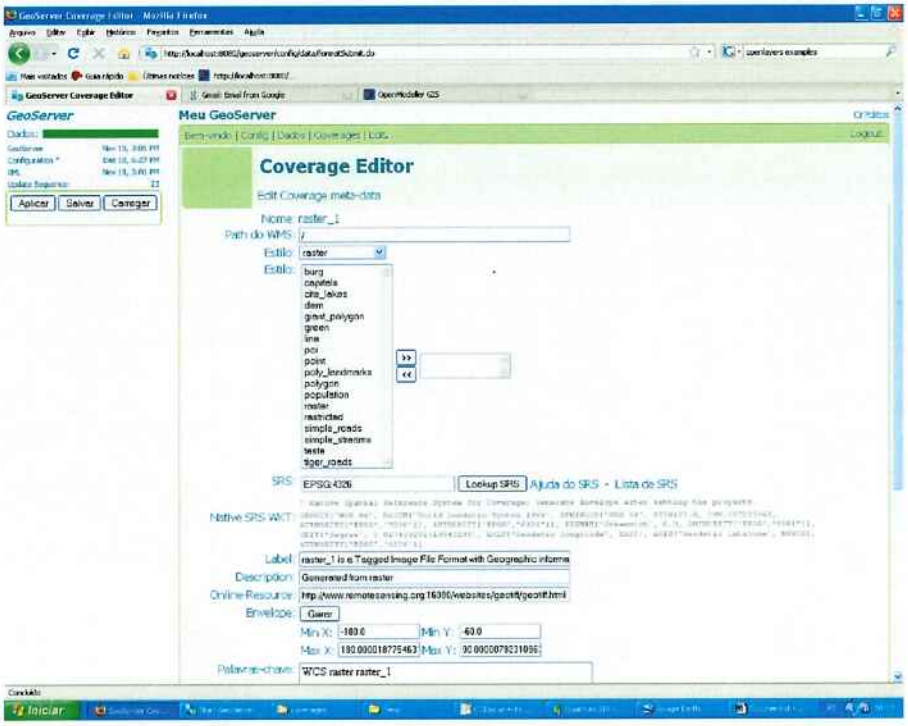

Figura 5.12 – Edição dos Metadados da Camada Cadastrada

Figura 5.13 – Aplicar e Salvar as Configurações

- Clicar em *Aplicar* e *Salvar* (vide Figura 5.13)

Ao final deste procedimento, a *Layer* "*raster*" já fica disponível no serviço WMS do *GeoServer*, conforme apresentado na Figura 5.14, por meio da seguinte requisição HTTP GET:

http://localhost:8080/geoserver/wms?bbox=-180.0,-60.0,180.0000187754631,90.00000782310963&styles=&Format=application/openlayers&request=GetMap&version=1.1.1&layers=topp:raster&width=800&height=313&srs=EPSG:4326

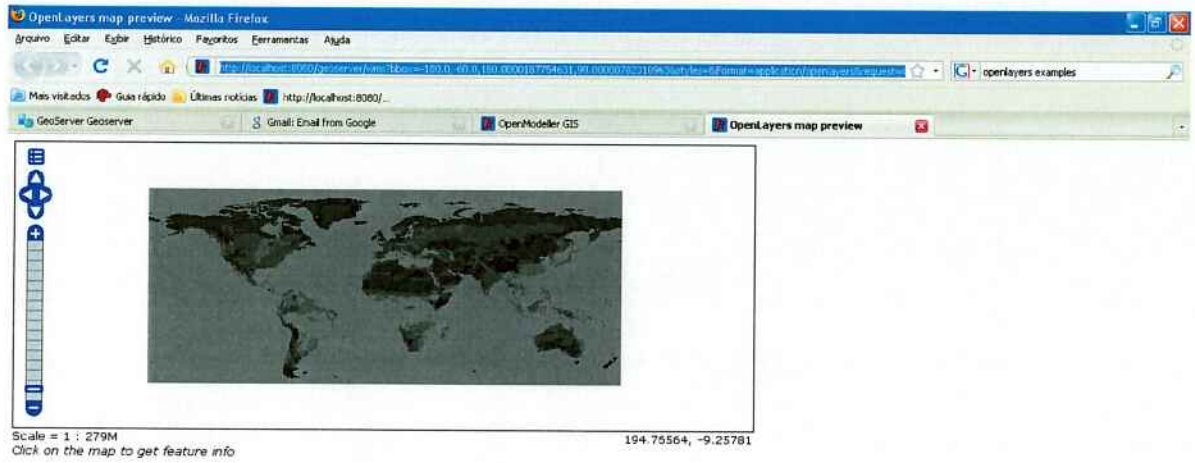

Figura 5.14 – Apresentação da Camada via WMS

Embora seja possível exibir o arquivo *GeoTIFF*, este procedimento é extremamente trabalhoso, mesmo com o auxílio de uma interface de Administração amigável como a do *Geoserver*. Portanto, foi necessário estudar uma solução para permitir a automatização de todo esse processo.

Uma vez que o *Geoserver* é uma solução *free* e *opensource*, a primeira e mais óbvia tentativa de automatizar o processo foi justamente obter o código fonte do *GeoServer* e realizar as modificações necessárias. O *GeoServer* é desenvolvido na linguagem Java e utiliza a ferramenta *Maven,* que é um software de gerência de projeto, para dar o *build* na solução.

No primeiro instante, foram seguidos os passos especificados no Manual do Desenvolvedor do *GeoServer* para obter acesso ao código fonte:

- Obter o código fonte do *GeoServer* diretamente de seu repositório conforme a Figura 5.15:

```
svn co https://svn.codehaus.org/geoserver/branches/1.7.x
geoserver_1.7.x
```

Figura 5.15 – Repositório do *GeoServer*

- Realizar o *build* do código fonte usando o *Maven* (Figura 5.16):

```
cd geoserver_1.7.x/src

mvn clean install
```

Figura 5.16 – Comandos do *Maven* para build do *GeoServer*

- Em uma situação de sucesso, o resultado seria (Figura 5.17):

```
[INFO]

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Reactor Summary:

[INFO] ------------------------------------------------------------------------

[INFO] GeoServer ..................................... SUCCESS [10.271s]

[INFO] GeoServer Maven Plugins ....................... SUCCESS [0.865s]

[INFO] Configuration Deployment PlugIn ..................... SUCCESS [3.820s]
```

```
[INFO] GeoServer Maven Archetypes ..... SUCCESS [0.054s]
[INFO] GeoServer WFS Output Format Archetype ..... SUCCESS [0.390s]
[INFO] Core Platform Module ..... SUCCESS [5.270s]
[INFO] Data Module ..... SUCCESS [4.521s]
[INFO] Open Web Service Module..... SUCCESS [2.730s]
[INFO] Main Module ..... SUCCESS [10.077s]
[INFO] Web Coverage Service Module ..... SUCCESS [3.785s]
[INFO] Web Coverage Service 1.1.1 Module ..... SUCCESS [5.254s]
[INFO] Validation Module ..... SUCCESS [1.131s]
[INFO] Web Feature Service Module ..... SUCCESS [6.695s]
[INFO] Web Feature Service Module ..... SUCCESS [1.197s]
[INFO] Web Map Service Module ..... SUCCESS [8.519s]
[INFO] Geoserver REST Support Code ..... SUCCESS [3.366s]
[INFO] GeoWebCache (GWC) Module ..... SUCCESS [0.255s]
[INFO] Web Application Module ..... SUCCESS [27.386s]
[INFO] Community Space ..... SUCCESS [0.312s]
[INFO] GeoServer Extensions ..... SUCCESS [0.071s]
[INFO] ------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
```

Figura 5.17 – Build de Sucesso do *GeoServer*

Porém, raramente esse *build* funcionava. De fato, ele nunca chegou a funcionar por completo. Uma segunda tentativa de manipulação do código-fonte do *GeoServer* foi utilizando a IDE NetBeans com suporte ao *Maven*.

Os passos eram os seguintes:

- Instalar o plug-in do *Maven* para o *NetBeans* (Figura 5.18):

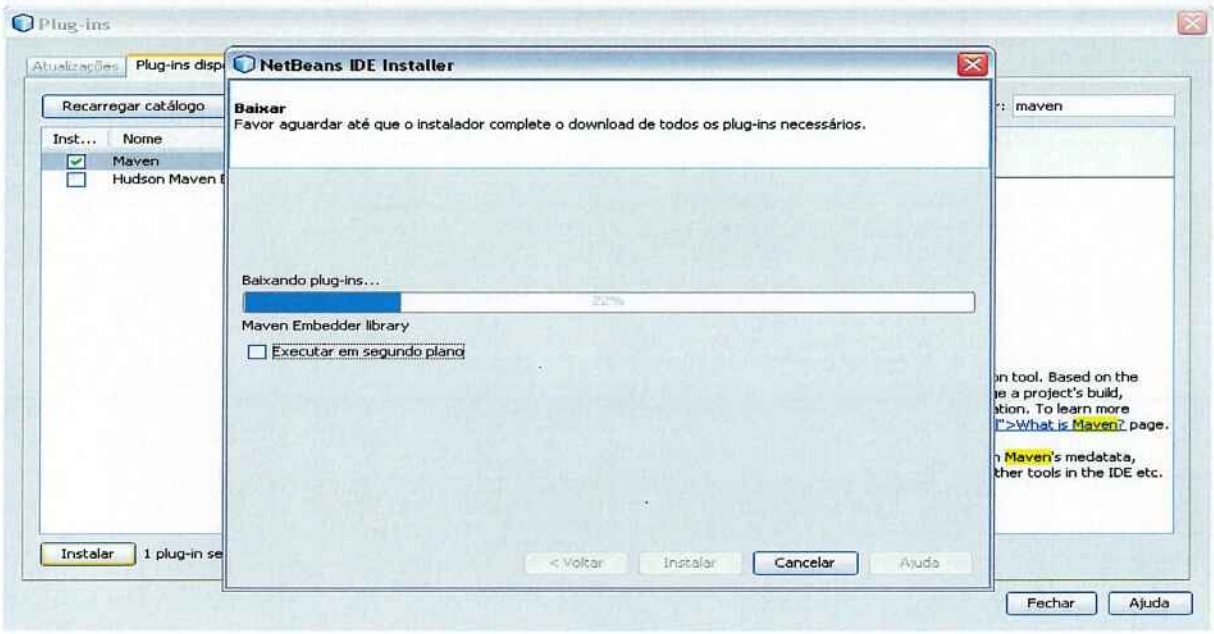

Figura 5.18 – Instalação de plugin do Maven para o Netbeans

- Criar um projeto *Maven* novo com POM (Project Object Model – basicamente um arquivo XML com uma descrição do projeto) já existente (Figura 5.19):

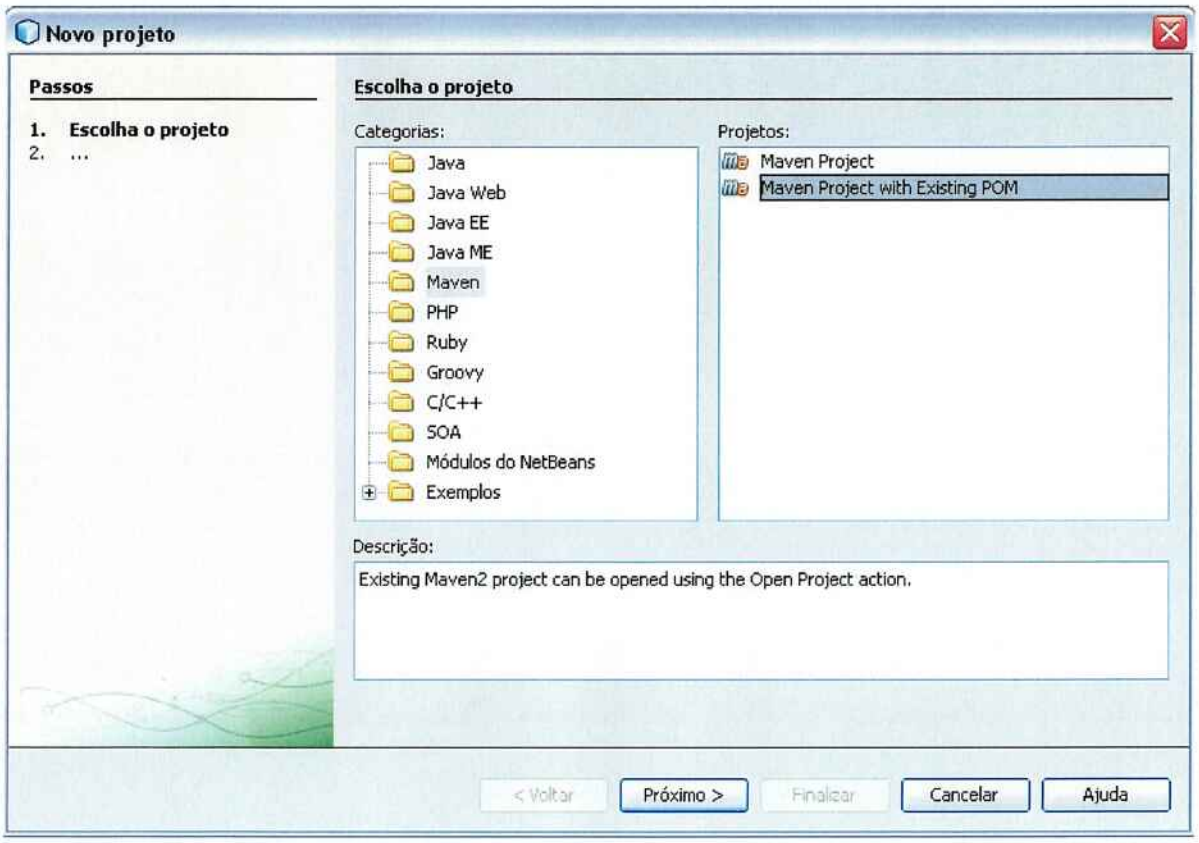

Figura 5.19 – Criação de projeto *Maven* no NetBeans

Apesar de com esta segunda maneira ser possível o acesso ao código do *GeoServer*, o problema do *build* ainda continuava. O caminho foi tentar compreender o código do *GeoServer* e então criar a aplicação necessária. Porém, alguns problemas foram encontrados para realizar esta tarefa:

- A complexidade do código era alta, demandando tempo e esforços muito grandes mesmo para aquilo que seria a última conseqüência da tarefa;
- O fato de o projeto ser muito complexo e o próprio grupo obter testemunhos de usuários do *GeoServer* avançados que não conseguiam nem mesmo realizar coisas simples com o código-fonte do *GeoServer*;
- O fato de não funcionar o build no mesmo inviabilizava qualquer possibilidade de testes e estudos em cima da solução.

Sendo assim, a possibilidade de manipulação do código do *GeoServer* para criar uma aplicação se mostrou uma alternativa inviável. Portanto, uma mudança de abordagem para o problema foi adotada. Após novas pesquisas, foi descoberto o aplicativo *Restful*.

5.4.2 Restful

O *Restful* é uma solução do *Geoserver*, que, nas palavras deles:

Trata-se de um trabalho ainda em progresso. A principal motivação para esta interface é a necessidade dos usuários poderem facilmente adicionar novas camadas e para desenvolvedores e usuários mais avançados poderem fazer o mesmo **programaticamente**.

[http://geoserver.org/display/GEOSDOC/RESTful+Configuration+API. Acessado em: 20/11/2009]

Essa interface permitiria que fosse possível fazer manipulações no *GeoServer* sem a necessidade de interagir com a interface do usuário.

O *Restful* funciona basicamente através de uma API, com a qual é possível fazer requisições HTTP ao *GeoServer* para realizar as tarefas. Foram então realizados testes iniciais utilizando o CURL, o qual é um software que permite realizar

requisições HTTP através de uma interface. Os passos para realizar a tarefa de visualização automática foram os seguintes:

- Inserir o *Coverage Store* através do comando (Figura 5.20):

```
curl -u admin:geoserver -v -XPOST -H "Content-type: text/xml" -d "<coverageStor
e><name>cs_teste</name></coverageStore>" http://localhost:8080/geoserver/rest/workspac
es/topp/coveragestores
```

Figura 5.20 – Comando HTTP para criação de *Coverage Store* no GeoServer

Este comando funciona, porém existe uma limitação que é a que todos os *Coverage Stores* eram criados dentro do *Workspace Topp*, que é o *Workspace* padrão do Geoserver. Portanto, não era possível criar um *Workspace* e então criar um *Coverage Store* dentro dele. Neste ponto foi adotado simplesmente que os *Coverage Stores* seriam criados dentro do Topp.

- Inserção do arquivo *Geotiff* (no caso, o arquivo sample.tiff) através do comando (Figura 5.21):

```
curl -u admin:geoserver -v -XPUT -H "Content-type: image/tiff" --data-binary @C
:\tiffs\sample.tiff http://localhost:8080/geoserver/rest/workspaces/topp/coverag
estores/cs_teste/file.geotiff
```

Figura 5.21 – Comando HTTP para inserção de *Geotiff* por requisição HTTP

Esta era a parte crucial para a automatização do processo. Porém, mesmo executando este comando, no máximo que se conseguia era o seguinte erro, conforme apresentado pela Figura 5.22:

```
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'admin'
> PUT /geoserver/rest/workspaces/
topp/coveragestores/cs_teste/file.geotiff HTTP

1.1
> Authorization: Basic YWRtaW46Z2Vvc2VydmVy
> User-Agent: curl/7.19.4 (i386-pc-win32) libcurl/7.19.4 OpenSSL/0.9.8j zlib/1.
.3
> Host: localhost:8080
> Accept: */*
> Content-type: image/tiff
> Content-Length: 692230
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
< HTTP/1.1 500 Internal Server Error

< Transfer-Encoding: chunked
< Server: Jetty(6.1.8)
<
Error auto-configuring coverage:java.lang.NullPointerException* Connection #0 to
```

Figura 5.22 – Erro obtido ao se subir o arquivo Geotiff com o comando correto

Mais tarde, foi descoberto que este é um *bug* específico do Geoserver [http://jira.codehaus.org/browse/GEOS-2926 - Acessado em 20/11/2009], que ainda não havia sido resolvido.

Esse foi o ponto no qual o *Geoserver*, apesar de suas características favoráveis e de o *Restful* dar uma resposta rápida para os serviços espaciais, foi abandonado devido aos diversos *bugs* da ferramenta e das dificuldades encontradas para a realização da tarefa.

A partir desta constatação, foi necessário recomeçar o trabalho, iniciando a busca por outros servidores geoespaciais. As opções recomendadas deveriam implementar os padrões OGS e a comunidade internacional de pesquisa na área indicou duas possibilidades: 1) O *Mapserver*, uma implementação antiga e mais madura, porém mais difícil de se utilizar devido às suas características de manipulação, já que ela possui uma linguagem própria chamada *Mapfile*, e as aplicações devem ser criadas a partir dela, além de ser necessário aprofundar os conhecimentos em conceitos geoespaciais; e 2) O *Deegree*, uma solução interessante, porém com a documentação disponível apenas em alemão. A decisão

foi então partir para o estudo do *Mapserver*, visto a maturidade da solução e a documentação acessível em inglês.

5.4.3 - Mapserver: Visualização automática de arquivo GeoTiff

Partiu-se então para o estudo e testes com o *MapServer*. O *MapServer* é um servidor geoespacial com uma manipulação muito diferente do *GeoServer*. O *MapServer* não possui uma interface para o usuário, mas uma linguagem de programação própria, constituída basicamente de operações de atribuição e algumas condições, sendo que com ela é possível criar aplicações tanto do lado servidor quanto do lado cliente. Basicamente, utilizar bem o *MapServer* significa programar bem em sua linguagem, chamada *MapFile*.

Para exibir o arquivo em formato *Geotiff* para o usuário, é necessário criar, em primeiro lugar, um servidor WMS para a imagem (o que o *GeoServer* faz sozinho), para então consumir deste servidor a imagem através de um cliente - no caso, o cliente escolhido foi o *OpenLayers*. Basicamente, a criação de um servidor WMS no *MapServer* é feita através do seguinte código (Figura 5.23):

```
MAP
NAME    "WMS"
SHAPEPATH "C:\tiffs\"
SIZE 480 480
IMAGETYPE JPEG
EXTENT -79.00 23.00 -77.00 25.00

WEB
    IMAGEPATH "/ms4w/tmp/ms_tmp/"
    IMAGEURL "/ms_tmp/"
    METADATA
     "wms_title"        "WMS_Server"  ##required
     "wms_onlineresource" "http://127.0.0.1:8081/cgi-bin/mapserv.exe?"   ##required
     "wms_feature_info_mime_type"  "text/html"
    END
  END
PROJECTION
    "init=epsg:4326"
END
LAYER
 NAME "sample"
```

```
        DATA "sample.tiff"
          TYPE RASTER
          STATUS DEFAULT
          OFFSITE 0 0 0
        METADATA
        "wms_title" "sample"
        "wms_srs""epsg:4326"
        END # End of Metadata
        DUMP TRUE
        HEADER "C:\ms4w\apps\ms_ogc_workshop\templates\query_header.html"
        #FOOTER "C:\ms4w\apps\ms_ogc_workshop\templates\query_footer.html"
        TEMPLATE
"C:\ms4w\apps\ms_ogc_workshop\templates\land_shallow_topo_2048_query_body.html"
        END # End of Layer
        END # Map File
```

Figura 5.23 – Código de servidor WMS básico criado com o MapServer

Realizando a seguinte requisição (Figura 5.24):

http://127.0.0.1:8081/cgi-bin/mapserv.exe?map=c:\\maps\\sample_WMS.map&mode=map

Figura 5.24 – Requisição WMS para o servidor criado em questão

Desta forma, como desejado inicialmente, foi possível obter a imagem como resultado (Figura 5.25):

Figura 5.25 – Resultado de Requisição WMS

Para se entender o código criado na linguagem *MapFile* é imprescindível a compreensão da sua estrutura. O que a linguagem faz é basicamente relacionar o "onde" para os dados geoespaciais e o "como" eles deverão ser exibidos. Um conceito importante para entendê-la é o conceito de *Layer*.

Layer é justamente uma camada que será entregue pelo servidor. Ela combina dados com estilo. Estilo é justamente uma definição de como os dados serão mostrados. Um arquivo com extensão map consiste basicamente de um grande objeto MAP e, dentro dele, vários objetos LAYER e mais alguns objetos de configuração, como o WEB. Dentro de cada LAYER temos as diretivas CLASS e STYLE, que permitem definir o estilo. Dentro de cada objeto temos muitos possíveis atributos para qualificar, como: NAME, TYPE, DATA, STATUS, entre outros.

Concluída a tarefa de construir o servidor WMS capaz de retornar um arquivo *Geotiff* na forma de uma imagem, através de uma requisição WMS, o passo seguinte é construir a aplicação cliente, baseada no *OpenLayers*. Naturalmente, existiram algumas complicações. Como se pode reparar ao se observar o código em *MapFile*, existem algumas informações sobre a imagem que devem ser obtidas previamente

antes da sua publicação. Por exemplo, é necessário conhecer as coordenadas das bordas da imagem.

Como se tratava de um arquivo matricial no formato *Geotiff*, foi utilizado um programa chamado GDALINFO, disponível na biblioteca *Geospatial Data Abstraction Library*, GDAL [http://www.gdal.org/]. Este programa é capaz de obter as informações necessárias, bastando ter o arquivo à disposição. Segue exemplo de execução do GDALINFO (Figura 5.26):

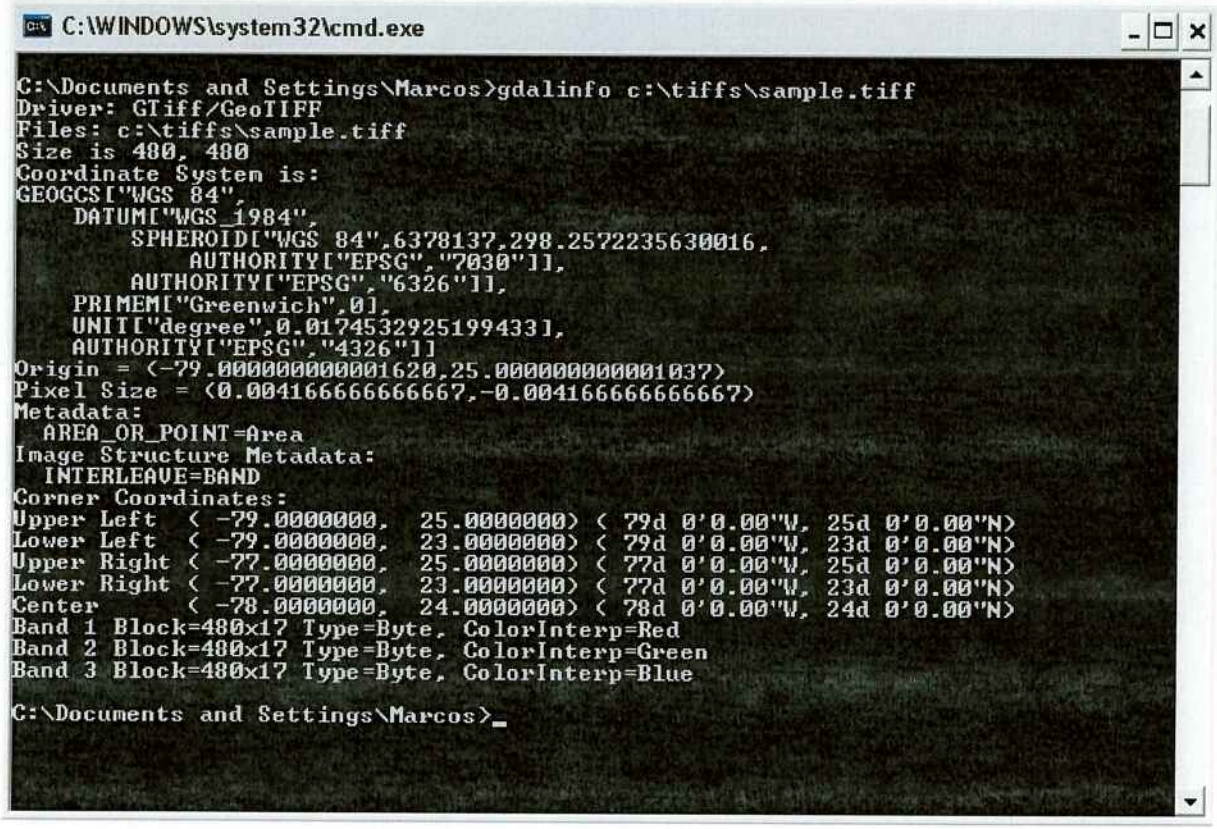

Figura 5.26 – Resultado de execução de GdalInfo sobre arquivo geotiff

Outras informações podem ser obtidas, incluindo o tamanho exato da imagem ("Size is 480 480") e quais as coordenadas dos cantos. Com estas informações é possível, finalmente, criar o servidor WMS com as informações corretas.

Para cada imagem diferente era necessário criar um servidor WMS novo, dinamicamente. Também seria possível atualizar um servidor existente, porém criar um novo é uma solução melhor porque permite otimizar a organização dos arquivos e dos servidores, além de simplificar a programação. Isto foi feito através da implementação de uma aplicação *web* em PHP, que é capaz de realizar uma

execução de GDALINFO, obter seu resultado, extrair as informações necessárias e configurar o cliente *OpenLayers* utilizado, em *JavaAcript*.

A imagem de um possível resultado final está na Figura .5.27

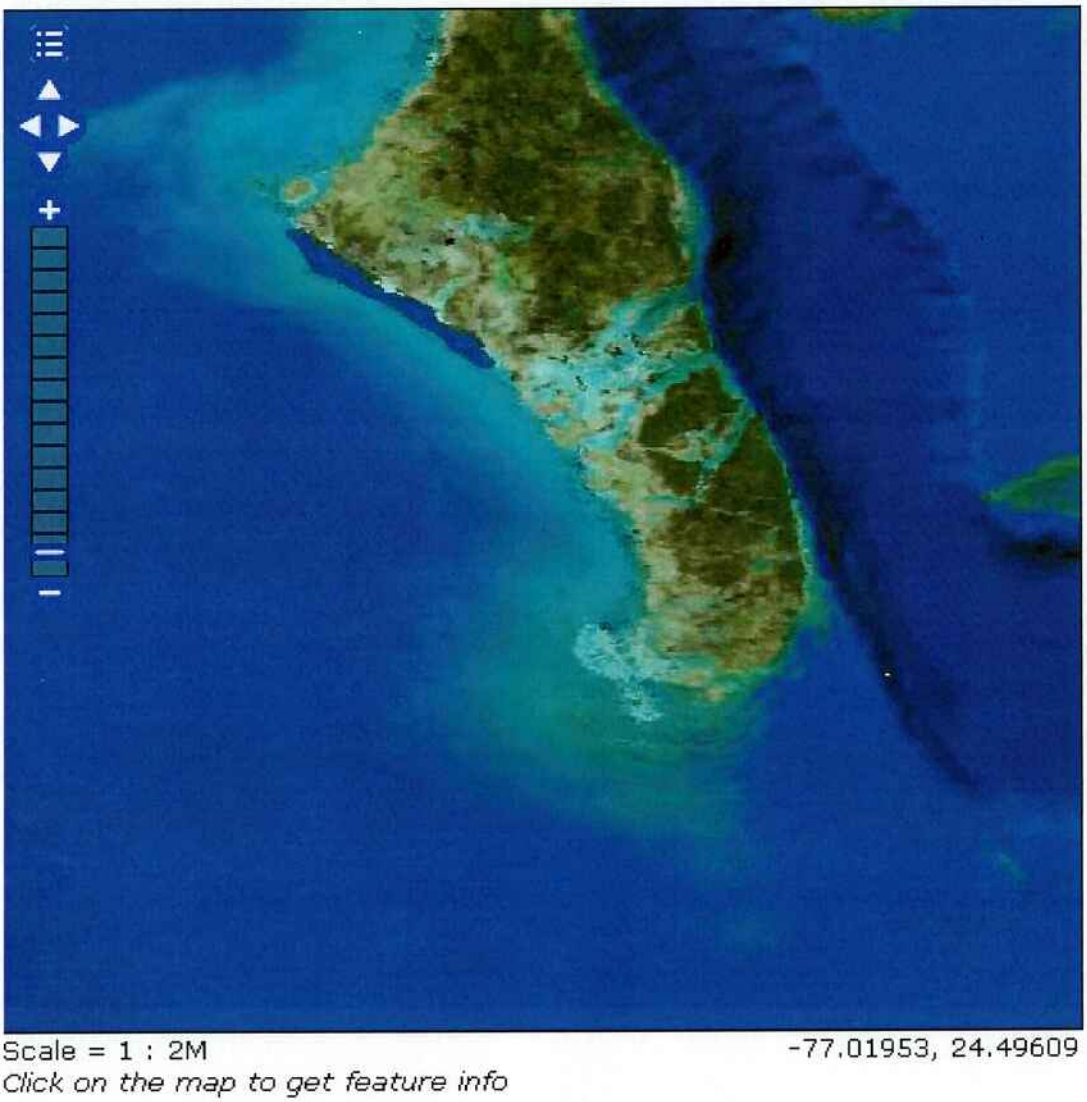

Figura 5.27 – Exibição de *Geotiff* através do *MapServer* e *OpenLayers*

Quanto a solução cliente, a recomendação é não utilizar a versão 2.8 do *OpenLayers* . Esta versão é mais recente e pode apresentar alguma incompatibilidade com a solução descrita, então seria necessário refazer todos os testes da aplicação. De qualquer forma, esta versão foi usada em alguns testes nos passos seguintes e não foi detectada nenhuma incompatibilidade.

5.4.4 - Aplicação Base

Em primeiro lugar, é preciso entender qual serviço seria o principal para uma aplicação web envolvendo mapas. Dentre os serviços explicitados, o mais usado é justamente o WMS pelo seu caráter de exibir a informação de uma maneira visual. Sendo assim, para uma aplicação cuja intenção é expor dados espaciais como resultados de algoritmos e similares, é natural que ele venha a ser o principal serviço utilizado.

Orientando-nos pelo BDD, a principal necessidade que existia era integrar mapas à sua aplicação, o que, em outras palavras, se traduz na exibição das informações de espécimes que eles possuem, além de resultados de análises sobre esses dados. Tais dados sobre espécies são obtidos através de GPS, o que disponibiliza, basicamente, coordenada de latitude e longitude sobre a localização destas espécies. O caminho entre a obtenção destas informações em forma textual e a sua transformação em algo que possa ser visualizado foi justamente o objetivo da aplicação desenvolvida, explicitado a seguir.

Um sistema de projeção é uma representação, ou modelo, da superfície terrestre. Ele especifica uma maneira de representar a Terra, através de uma aproximação matemática do planeta (elipsóides, também conhecidas como esferóides) aliado a um tipo específico de representação de coordenadas (Chapman et al., 2005). Para ilustrar isso, vamos analisar o sistema de projeção EPSG:4326, o qual é o mais comum e também o usado em GPS's.

5.4.4.1 – EPSG:4326

EPSG:4326 é uma sigla que significa *European Petroleum Survey Group* (uma entidade ligada a Europa e que trabalhava com a ciência da geodésia e também com extração de óleo, e que montou um banco de dados de projeções que acabou por se tornar comumente usado) e o código 4326 se refere ao WGS84 (WGS significa *World Geodetic System*) em coordenadas de latitude e longitude. O WGS84 adota *Greenwich* como seu meridiano central e adota como origem de suas

coordenadas o centro de massa da Terra. Apesar de todos esses detalhes, para termos de processamento computacional, o importante é que sempre o fornecedor dos dados deixe claro em qual sistema de coordenadas estão as informações que ele está fornecendo, pois caso contrário o consumidor destas informações poderá obter resultados incorretos.

Uma vez definido um sistema de projeção, é necessário compreender como o banco de dados é capaz de armazenar informações espaciais. Comumente, estamos acostumados a compreender informações de latitude e longitude no formato (20,45), onde 20 seria a latitude e 45 a longitude. Os bancos de dados espaciais representam a informação ao contrário, ou seja, eles não a representam a informação no formato (latitude, longitude) e sim como (longitude, latitude). Este ponto é uma fonte muito comum de erros e, portanto, é necessário cuidado no tratamento destas informações. Além disso, quanto uma informação geométrica, ou um objeto geométrico, é salvo no banco, ele é salvo no formato WKB (*Well Known Binary*), em oposição ao WKT (*Well Known Text*), que não é nada mais do que o texto comum. Por exemplo: Seja o ponto (-47, -24) (lembrando que -47 é informação de longitude e -24 é informação de latitude) no formato WKT. Ao ser guardado no banco ele seria representado como:

"0101000020E6100000000000000008047C000000000000038C0"

Ou seja, no formato WKB. Para ser compreensível a leitura de tal informação no banco, é necessário utilizar a função St_AsText, que é capaz de traduzir WKB para WKT. Exemplo de consulta com St_AsText (Figura 5.28):

```
SELECT ST_AsText(thepoint_geospatialelements) from geospatialelements;
```

Figura 5.28 – Consulta SFSQL

Nesta consulta, *thepoint_geospatialelements* trata da coluna da tabela *geospatialelements*, que tem capacidade para guardar informações geométricas no formato WKB. A função St_AsText faz parte da SFSQL.

SFSQL é uma sigla para *Simple Features for SQL*. Basicamente, é um conjunto de funções que adiciona capacidades espaciais à linguagem SQL. Por exemplo, traçando um paralelo entre a SQL e a SFSQL, pode-se afirmar que, enquanto a SQL é capaz de responder a perguntas como "Quais foram as vendas totais naquele

mês?", "Quais os códigos dos produtos que obtiveram mais rendimento no ano? ", com a SFSQL é possível se responder perguntas como "Quais cidades seriam atingidas por uma inundação na costa?" ou "Quantas ocorrências de espécimes temos num raio de 50 km?". Em outras palavras, a SFSQL é capaz de responder questões relativas ao espaço. Exemplos de aplicação serão vistos adiante.

As informações disponíveis eram, justamente, informações georreferenciadas sobre localizações de espécies, como se pode perceber na Figura 5.29:

	idgeospatiale [PK] serial	decimallongitude double precision	decimallatitude double precision
1	2	-47	-24
2	3	-47	-24
3	4	-47	-24
4	5	-47	-24
5	6	-46	-23
6	7	-46	-23
7	8	-46	-23
8	9	-46	-23
9	10	-46	-23
10	11	-46	-23
11	12	-46	-23

Figura 5.29 – Trecho dos dados disponíveis

Transforma-se então essa informação em dados geométricos através de um script SFSQL (Figura 5.30):

```
SELECT AddGeometryColumn('public', 'geospatialelements',
'thepoint_geospatialelements', 4326, 'POINT', 2 );  (1)

UPDATE geospatialelements
SET thepoint_geospatialelements = PointFromText('POINT(' ||
decimallongitude || ' ' || decimallatitude || ')',4326);  (2)

CREATE INDEX idx_zctas_thepoint_geospatialelements ON geospatialelements
USING GIST (thepoint_geospatialelements); (3)

VACUUM ANALYZE geospatialelements; (4)
```

Figura 5.30 – Script SFSQL para criação de coluna geoespacial

A consulta (1) tem por intuito criar uma coluna geométrica chamada *thepoint_geospatialelements* na tabela *geospatialelements*. É esta coluna que

guardará as informações espaciais geométricas sobre as espécies. A consulta (2) é quem realiza a transformação dos dados textuais (as informações de longitude e latitude) em informações geométricas, no caso, os pontos, e no sistema de projeção EPSG:4326, posto que tais informações foram obtidas com GPS. A consulta (3) realiza a criação de índices no banco, visando a melhoria do desempenho das consultas. A consulta (4), apesar de estar junto com o *script*, não pode ser realizada junto com o mesmo. Ela deve ser realizada à parte, devido a uma imposição do Postgre, que exige que ela seja feita isoladamente. A consulta verifica espaços que estejam sendo usados inutilmente e elimina os mesmos liberando, assim, mais espaço na memória.

Temos como resultado então a seguinte coluna (Figura 5.31):

thepoint_geospatialelements geometry
0101000020E6100000000000000008047C000000000000038C0
0101000020E6100000000000000008047C000000000000038C0
0101000020E6100000000000000008047C000000000000038C0
0101000020E6100000000000000008047C000000000000038C0
0101000020E6100000000000000000047C000000000000037C0
0101000020E6100000000000000000047C000000000000037C0
0101000020E6100000000000000000047C000000000000037C0
0101000020E6100000000000000000047C000000000000037C0
0101000020E6100000000000000000047C000000000000037C0
0101000020E6100000000000000000047C000000000000037C0

Figura 5.31 – Amostra de tabela geométrica

Como se pode ver, os objetos geométricos foram criados a partir dos pontos fornecidos, para serem exibidos na solução final. Pode-se efetuar um teste utilizando um software de SIG como o *Quantum GIS*, por exemplo. O resultado é a imagem produzida a partir dos pontos apresentados na Figura 5.32.

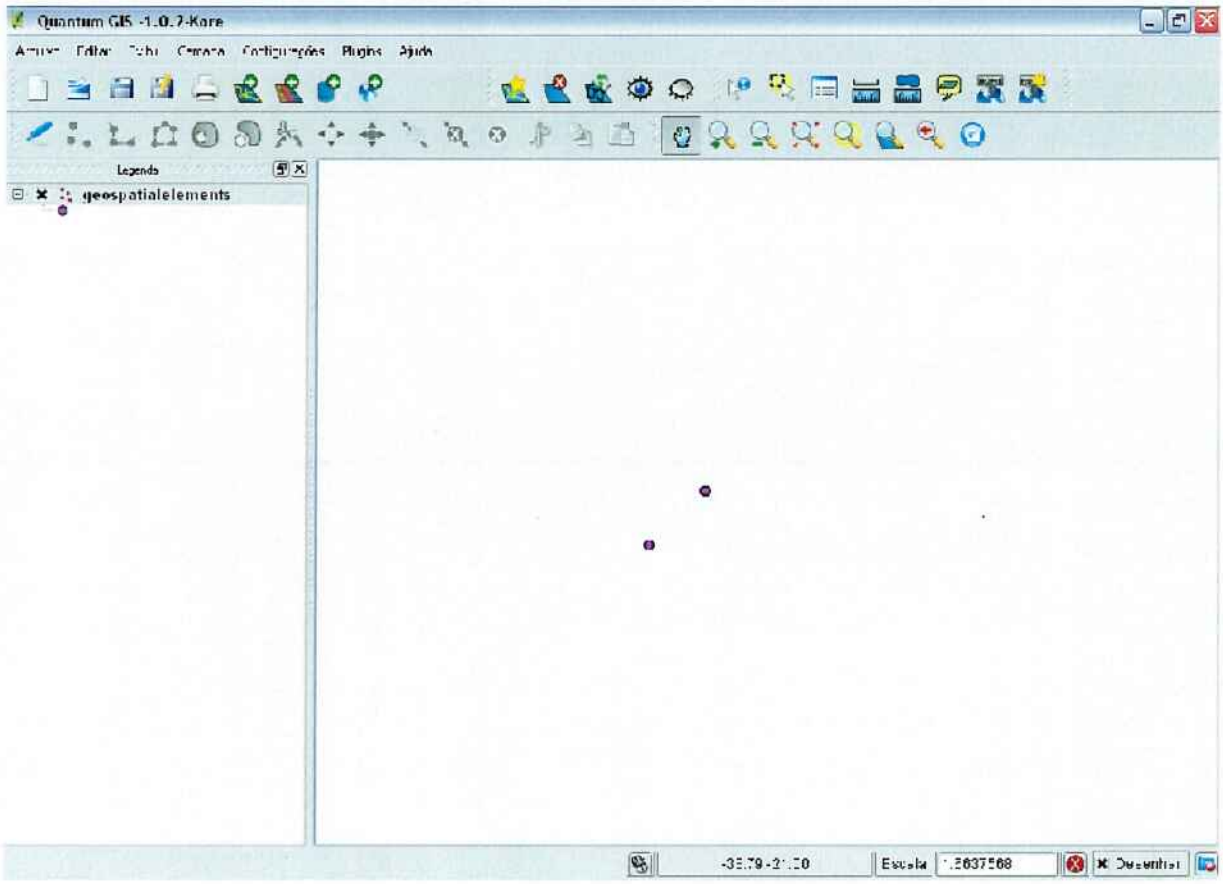

Figura 5.32 – Pontos do BD vistos sob um software GIS

Com as informações corretas já disponíveis no banco, elas devem ser servidas de maneira interoperável através de um *Web Service*, no caso o WMS. Criou-se então, um WMS capaz de obter as informações do banco (Figura 5.33):

```
MAP
NAME      "WMS"
SIZE 600 400
SHAPEPATH "C:\tiffs\"
EXTENT -180 -90 180 90 # World
SYMBOLSET "C:\maps\symbols.txt"

WEB
    IMAGEPATH "/ms4w/tmp/ms_tmp/"
    IMAGEURL "/ms_tmp/"
    METADATA
     "wms_title"         "WMS_Server"  ##required
     "wms_onlineresource" "http://127.0.0.1:8081/cgi-bin/mapserv.exe?"   ##required
     "wms_feature_info_mime_type"  "text/html"
    END
  END
PROJECTION
```

```
"init:epsg4326"
END
LAYER
        NAME "geospatialelements"
        STATUS Default
        TYPE Point
        CONNECTIONTYPE postgis
        CONNECTION "dbname=postgis user=postgres host=localhost password=tcc09"
DATA "thepoint_geospatialelements FROM (select idgeospatialelements, thepoint_geospatialelements from
geospatialelements) as subquery using UNIQUE idgeospatialelements"
        CLASS
    NAME "Bee"
    STYLE
      SYMBOL "Bee"
        SIZE 35
            END
          END

         METADATA
        # sets:
          # /WMT_MS_Capabilities/Capability/Layer/*/Title
          # /WFS_Capabilities/FeatureTypeList/FeatureType[*]/Title
          "ows_title"    "zcta"
        # sets:
          # /WMT_MS_Capabilities/Capability/Layer/*/Abstract
          # /WFS_Capabilities/FeatureTypeList/FeatureType[*]/Abstract
          "ows_abstract"    "These are wfs tests... zcta"
        # sets:
          # /WMT_MS_Capabilities/Capability/Layer/*/KeywordList/Keyword[]
          # /WFS_Capabilities/FeatureTypeList/FeatureType[*]/Keywords
          "ows_keywordlist"    "Wfs,zcta"
        # sets:
          # /WMT_MS_Capabilities/Capability/Layer/*/MetadataURL/@type
          # /WFS_Capabilities/FeatureTypeList/FeatureType[*]/MetadataURL/@type
          "ows_metadataurl_type" "FGDC"
        # sets:
          # /WMT_MS_Capabilities/Capability/Layer/*/MetadataURL/OnlineResource/@xlink:href
          # /WFS_Capabilities/FeatureTypeList/FeatureType[*]/MetadataURL
          "ows_metadataurl_href" "http://127.0.0.1:8081/ms_ogc_workshop/index.html"
```

```
        # sets:
          # /WFS_Capabilities/FeatureTypeList/FeatureType[*]/MetadataURL/@format
          "wfs_metadataurl_format" "TXT"
          # specify which fields to include when returning queries
          "gml_include_items" "all"
         END#end of metadata
        # data is queryable
         DUMP TRUE
         HEADER "C:\ms4w\apps\ms_ogc_workshop\templates\query_header.html"
         #FOOTER "C:\ms4w\apps\ms_ogc_workshop\templates\query_footer.html"
         TEMPLATE
"C:\ms4w\apps\ms_ogc_workshop\templates\land_shallow_topo_2048_query_body.html"

         # fuzziness for querying
         TOLERANCE 5
        END#end of layer
        END # Map File
```

Figura 5.33 – Implementação de WMS mais completa

Testando a requisição WMS (Figura 5.34):

```
http://127.0.0.1:8081/cgi-bin/mapserv.exe?map=c:\\maps\\wms_postgis.map&mode=map
```

Figura 5.34 – Requisição WMS para dados em banco

Obtemos no *browser* a imagem apresentada na Figura 5.35:

Figura 5.35 – Resultado da requisição no *browser*

Com isso, as informações do no banco de dados estão corretas e existe um servidor WMS funcional para elas. Parte-se então para a construção da parte cliente, que é responsável por consumir os dados do *Web Service* espacial.

Neste momento do projeto ocorre a combinação de camadas.
São utilizadas camadas de servidores bem conhecidos como *Google* ou *Yahoo!* (no caso, será usado o *Google*), e por cima das camadas *Google*, será sobreposta a camada com informação das espécies. Quanto às camadas *Google*, é importante esclarecer que elas não estão em um sistema de projeção comum, como o EPSG:4326, e sim em um sistema de projeção próprio, cujo código é 900913. Ou seja, as informações obtidas por GPS em EPSG:4326 precisam ser tratadas. Como o *MapServer* não vem preparado para lidar com esse sistema de projeção, é necessário realizar as seguintes modificações:

1. Acessar a pasta proj/nad;
2. Abrir o arquivo epsg, que é uma coletânea de sistemas de projeção;
3. Adicionar o *Google Projection* (Figura 5.36):

```
## Google Projection
<900913> +proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m
+nadgrids=@null +no_defs <>
##
```

Figura 5.36 – Projeção Google: 900913

4. Salvar.

Uma vez que o *MapServer* conhece o *Google Projection*, pode-se então criar a aplicação utilizando o framework do *OpenLayers* normalmente.

Existem, porém, ainda, alguns pontos a serem considerados. A solução faz uso tanto de consulta SFSQL, quanto das operações *GetMap* e *GetFeatureInfo* do serviço WMS. Através do GetMap a *layer* de espécimes é apresentada em cima das *layers* provenientes do Google. Através do *GetFeatureInfo* informações pertinentes ao ponto selecionado, ou que estejam armazenadas no banco, são retornadas através de um *framed cloud*. Um *framed cloud* nada mais é do que a janela em formato de nuvem que se abre ao se clicar em um ponto, dentro da qual o conteúdo de retorno da operação *GetFeatureInfo* é exibido. O retorno da consulta SFSQL ocorre no *iframe* (um *iframe* é uma *tag* em HTML que cria uma janela de browser dentro de outra janela de browser) inferior ao mapa, e é executado a cada ponto da *layer* das espécies quando ocorrer um evento de clique.

Com relação à apresentação do conteúdo da operação *GetFeatureInfo*, o código em *MapFile* contém as *tags* HEADER, FOOTER e TEMPLATE. Elas indicam qual a estrutura HTML que se deseja utilizar para apresentar o resultado no *framed cloud*.

Segue exemplo de possível código HTML (Figura 5.37):

```
<table class="centered" border="1">
    <tr>
        <th>Bounding box</th>
        <td>[minx],[miny],[maxx],[maxy]</td>
    </tr>
    <tr>
        <th>Image Coordinates</th>
        <td>[img.x], [img.y]</td>
    </tr>
    <tr>
        <th>Map Coordinates</th>
        <td>[mapx], [mapy]</td>
    </tr>
    <tr>
        <th>Number of results</th>
        <td>[nr]</td>
    </tr>
    <tr>
        <th>Scale</th>
        <td>[scale]</td>
    </tr>
    <tr>
        <th>Cellsize</th>
        <td>[cellsize]</td>
    </tr>
</table>
```

Figura 5.37 – Código HTML que trata requisição *GetFeatureInfo*

O conteúdo entre colchetes deve ser substituído pela informação proveniente do *GetFeatureInfo*, quando da apresentação no *framed cloud*. Por exemplo, se temos a coluna *cellsize* no banco, e a chamamos na consulta ao banco presente no servidor WMS, e então a referenciamos no html como "<td>[cellsize]</td>". Isto signifca que, para cada requisição de *GetFeatureInfo* que retorne uma informação sobre cellsize, é precisamente neste ponto do HTML que ele deverá ser colocado.

Para que o *GetFeatureInfo* funcione, é preciso configurar um arquivo de proxy, aqui chamado de *proxy.cgi*.

Segue *Proxy.cgi* na Figura 5.38:

```
#!c:/python25/python.exe -u


"""This is a blind proxy that we use to get around browser
restrictions that prevent the Javascript from loading pages not on the
same server as the Javascript.  This has several problems: it's less
efficient, it might break some sites, and it's a security risk because
people can use this proxy to browse the web and possibly do bad stuff
with it.  It only loads pages via http and https, but it can load any
content type. It supports GET and POST requests."""

import urllib2
import cgi
import sys, os

# Designed to prevent Open Proxy type stuff.

allowedHosts = ['www.openlayers.org', 'openlayers.org',
                'labs.metacarta.com', 'world.freemap.in',
                'prototype.openmnnd.org', 'geo.openplans.org',
                'sigma.openplans.org', 'demo.opengeo.org',
                'www.openstreetmap.org', 'sample.avencia.com',
                                        '127.0.0.1:8081']


method = os.environ["REQUEST_METHOD"]

if method == "POST":
    qs = os.environ["QUERY_STRING"]
    d = cgi.parse_qs(qs)
    if d.has_key("url"):
        url = d["url"][0]
    else:
        url = "http://www.openlayers.org"
else:
    fs = cgi.FieldStorage()
    url = fs.getvalue('url', "http://www.openlayers.org")
```

```
try:
    host = url.split("/")[2]
    if allowedHosts and not host in allowedHosts:
        print "Status: 502 Bad Gateway"
        print "Content-Type: text/plain"
        print
        print "This proxy does not allow you to access that location (%s)." % (host,)
        print
        print os.environ

    elif url.startswith("http://") or url.startswith("https://"):

        if method == "POST":
            length = int(os.environ["CONTENT_LENGTH"])
            headers = {"Content-Type": os.environ["CONTENT_TYPE"]}
            body = sys.stdin.read(length)
            r = urllib2.Request(url, body, headers)
            y = urllib2.urlopen(r)
        else:
            y = urllib2.urlopen(url)

        # print content type header
        i = y.info()
        if i.has_key("Content-Type"):
            print "Content-Type: %s" % (i["Content-Type"])
        else:
            print "Content-Type: text/plain"
        print

        print y.read()

        y.close()
    else:
        print "Content-Type: text/plain"
        print
        print "Illegal request."

except Exception, E:
    print "Status: 500 Unexpected Error"
    print "Content-Type: text/plain"
    print
    print "Some unexpected error occurred. Error text was:", E
```

Figura 5.38 – Código *Python* de *Proxy.cgi*
[http://trac.openlayers.org/browser/trunk/openlayers/examples/proxy.cgi] – Acessado: 12/08/2009

Trata-se de um script na linguagem *Python* o qual lida com a questão de que devido a restrições de segurança da linguagem *Javascript* não é possível retornar informações de domínios remotos via XmlHTTPRequest. Torna-se então necessário o desenvolvimento de um *Proxy* que lide com este padrão, o que é justamente o script acima. As alterações necessárias para o script funcionar em um ambiente *Windows* e com o *MapServer* configurado na porta 8081 se restringem a incluir mais um endereço aos *allowedHosts* "127.0.0.1:8081", além de mudar o caminho do executável *Python* para um ambiente *Windows*.

Também é necessário abordar a questão dos *symbols* utilizados. Foi utilizado o arquivo symbols.txt, o qual é, na verdade, uma coleção de possíveis símbolos a serem utilizados pelas camadas, conforme a Figura 5.39:

```
SYMBOL
  NAME "Bee"
  TYPE PIXMAP
  IMAGE "bee.png"
END

SYMBOL
  NAME "lblue"
  TYPE PIXMAP
  IMAGE "lblue-marker.png"
END

SYMBOL
  NAME "blue"
  TYPE PIXMAP
  IMAGE "blue-marker.png"
END
```

Figura 5.39 – Arquivo de símbolos

Uma vez com uma coleção de *symbols* criada e devidamente referenciada em SYMBOLSET, é uma questão de se escolher na *tag* CLASS e em SYMBOL qual símbolo se deseja para aquela *layer* específica.

5.4.3 Aplicações Web de Agricultura de Precisão

É possível integrar todas as aplicações anteriores em uma única solução, reutilizando as arquiteturas anteriores de modo a atender um espectro ainda maior de aplicações. A arquitetura completa desta solução que integra o algoritmo *Filtering*, descrito anteriormente na seção referente aos algoritmos de modelagem, para a área de AP está representada pela Figura 5.40.

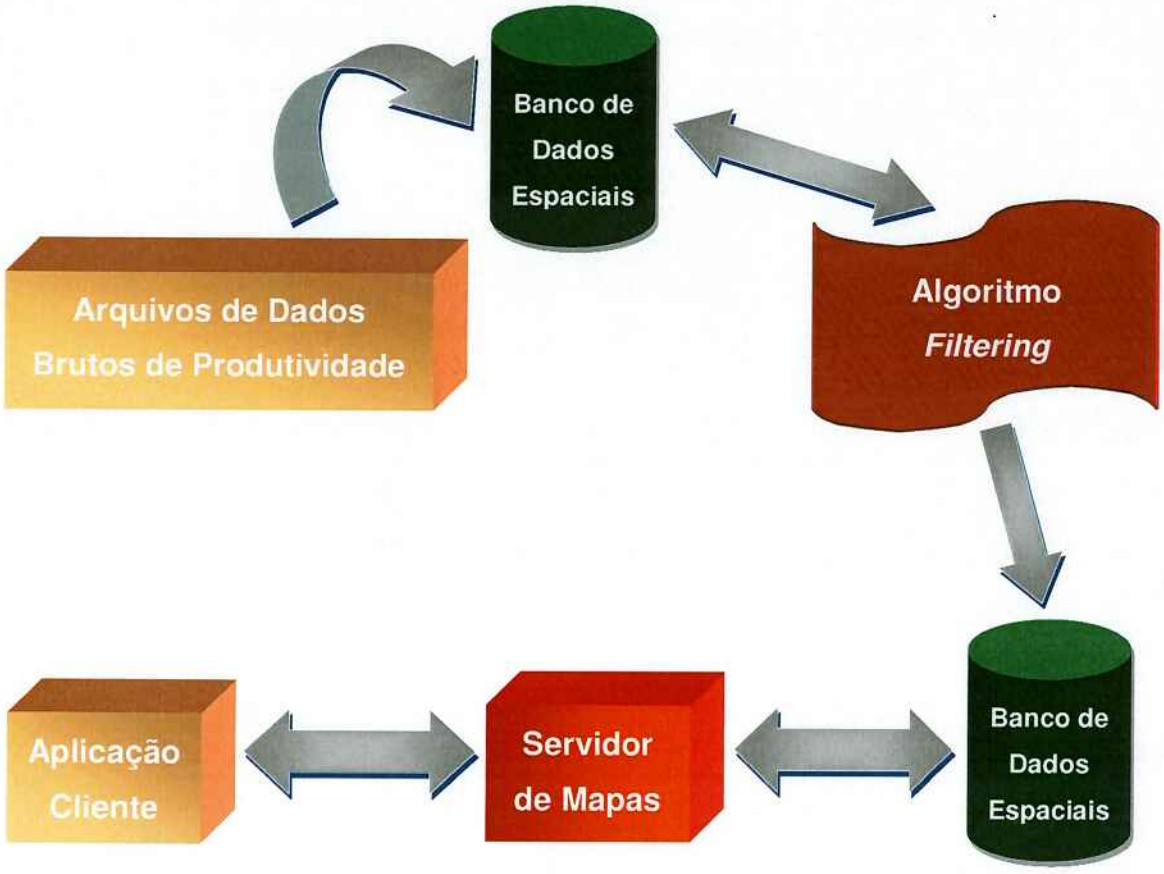

Figura 5.40 – Arquitetura Integrada de Aplicações Geoespaciais para Agricultura de Precisão

Primeiramente, os arquivos de dados brutos de produtividade são fornecidos pelo usuário e carregados pela aplicação web para o Banco de Dados Espaciais. Em seguida, estes dados brutos são processados pelo algoritmo *Filtering*. As informações resultantes do algoritmo são armazenadas no Banco de Dados Espaciais para o controle de todo o processo computacional envolvido. Por fim, o Servidor de Mapas é responsável por apresentar as informações armazenadas no Banco de Dados na forma de mapas de fácil interpretação ao usuário.

Portanto, no final de todo o processo, o usuário é capaz de visualizar os mapas de produtividade referentes à propriedade de interesse tanto antes como depois da aplicação do algoritmo.

6 RESULTADOS E DISCUSSÃO

Os resultados finais obtidos pela solução *easyGIS* foram:

- Aplicação Base;
- Apresentação de mapas bidimensionais em navegadores Terra;
- Aplicação Web de Biodiversidade;
- Aplicação Web de Agricultura de Precisão.

6.1 Aplicação Base

A imagem do resultado final da aplicação está na Figura 6.1. O resultado aqui demonstrado é de uma primeira aplicação utilizando o serviço geoespacial WMS, com capacidades para a operação *GetFeatureInfo* além de consultas SFSQL.

Figura 6.1 – Resultado Final

Na figura 6.2 temos uma demonstração da operação *GetFeatureInfo*. Nela é possível se ver o retorno de informações específicas sobre o ponto georreferenciado em questão.

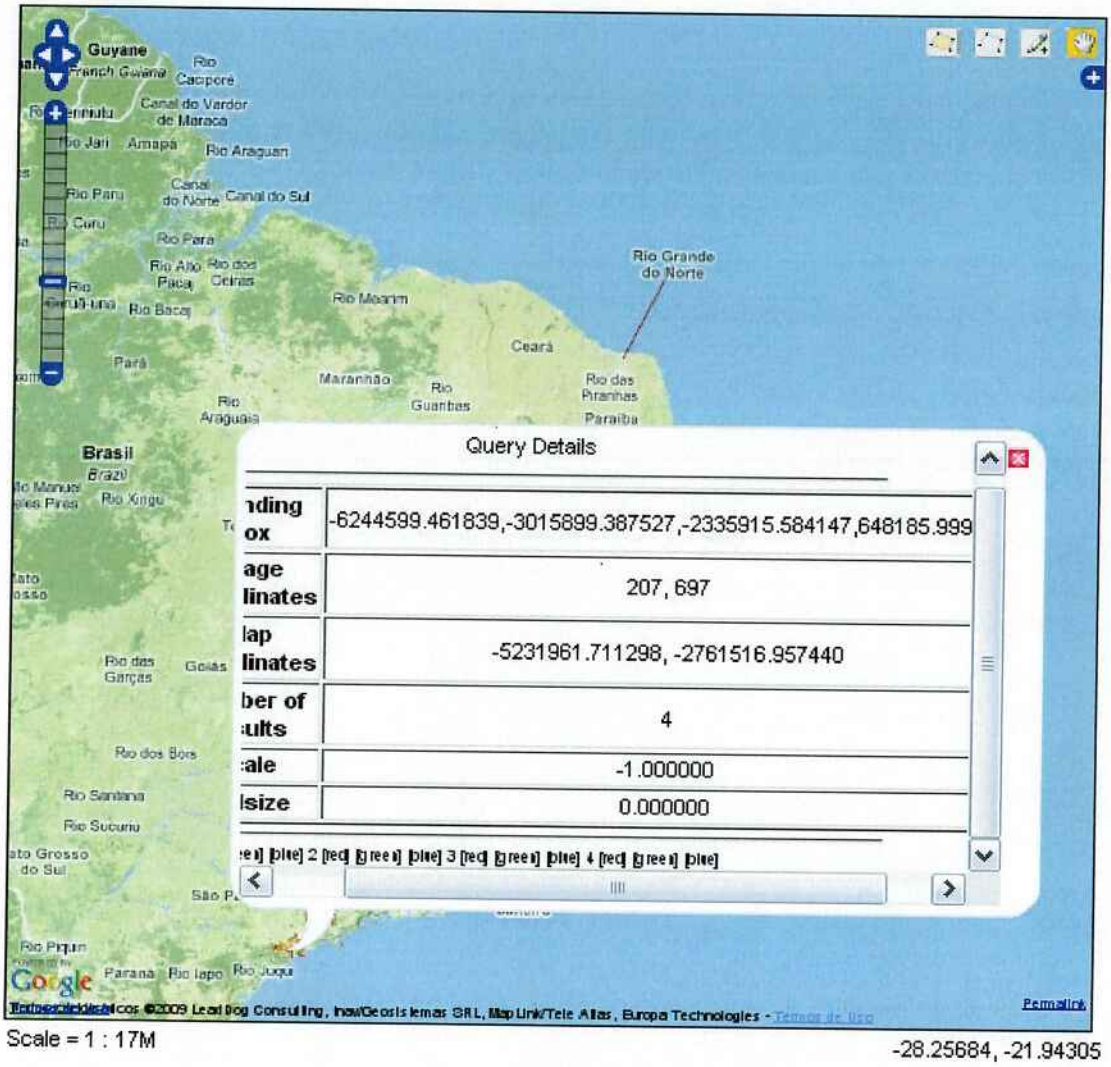

Figura 6.2 – Operação *GetFeatureInfo*

Na figura 6.3 está a demonstração de um resultado de uma consulta SFSQL realizada sobre o ponto (-47, -24). No exemplo, ela retorna um grupo de informações a serem exibidas através do *iframe,* mas nada impossibilita a realização de outros tipos de consultas. Existe a liberdade de se realizar qualquer consulta dependendo da aplicação de destino.

geometrytype	ST_Point
area	0
idgeospatialelements	2
decimallongitude	-47
decimallatitude	-24
coordinateuncertaintyinmeters	
georeferenceremarks	
geodeticdatum	
pointradiusspatialfit	
verbatimcoordinates	
verbatimlatitude	
verbatimlongitude	
verbatimcoordinatesystem	
georeferenceprotocol	
georeferencesources	

Figura 6.3 – Resultado de uma consulta relativa ao ponto (-47,-24)

Finalmente, a figura 6.4 ilustra a possibilidade de troca de camadas base e também da possibilidade de criação de dados vetoriais pelo usuário utilizando os ícones do lado superior direito.

Figura 6.4 - Escolha de layers e desenho de features

Quanto ao *script Phyton.cgi* utilizado para funcionar como um *proxy* que lida com domínios remotos via XmlHTTPRequest, a sua implantação foi realizada com sucesso de modo que as restrições de segurança da linguagem *Javascript* foram resolvidas.

6.2 Apresentação de Mapas Bidimensionais no *Google Earth*

A Figura 6.5 apresenta uma *Layer* de um mapa bidimensional do arquipélago das Bahamas, em destaque, utilizando o navegador Terra *Google Earth* [http://earth.google.com].

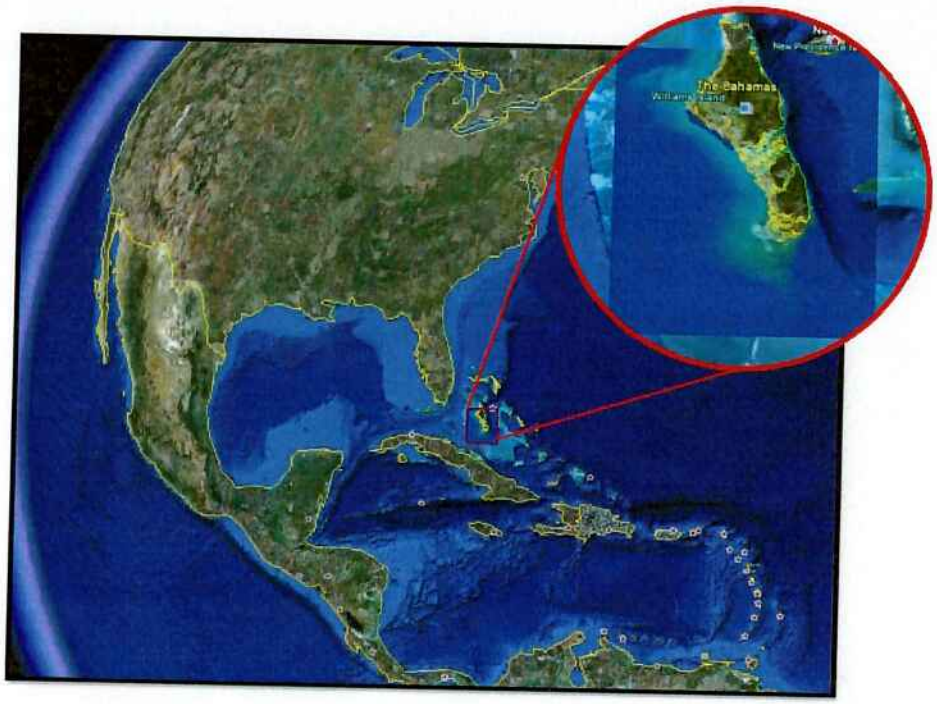

Figura 6.5 – Apresentação de Mapas Bidimensionais no *Google Earth*

Esta aplicação restringe-se apenas a realizar sobreposição de *Layers* no *Google Earth*, utilizando o serviço OGC WMS para captura do mapa e de seus atributos geográficos, descrito em detalhes no capítulo 5 referente às aplicações de mapas tridimensionais. No entanto, o código está aberto e disponível para interessados e colaboradores que desejam agregar outras funcionalidades a esta aplicação.

Como melhorias a esta aplicação, existem diversos atributos do próprio *GoogleEarth* que podem ser explorados, tais como o *Placemark* que identifica posições no mapa usando ícones de *flags*.

Uma observação importante quanto a este tipo de aplicação é o uso bastante limitado dos recursos disponíveis pelos serviços OGC, dificultando a escalabilidade do sistema como interatividade com o usuário no que diz respeito a Sistemas de Informação Geográfica.

6.3 Aplicação Web para Biodiversidade

A Figura 6.6 mostra as *Layers* do mapa de ocorrências da espécie *furcata boliviana* após a aplicação do algoritmo GARP.

Note que o usuário pode selecionar *Layers* de mapas terrestres online disponíveis pelos servidores *Google Maps* [http://maps.google.com/] e *Yahoo!Maps* [http://maps.yahoo.com/]. Todas as informações que estão apresentadas nos mapas são oriundas do Banco de Dados Espacial, o que possibilita a integração de outras funcionalidades, tais como obtenção de dados relacionados a cada ponto desenhado no mapa. Esta funcionalidade, em particular, foi implementada e está descrita em detalhes no capítulo 5 referente às aplicações web para sistemas geoespaciais.

Esta aplicação é extremamente escalável segundo as funcionalidades permitidas pelo *MapServer* e pelo *Openlayers* que apresentam uma enorme variedade de tratamento de dados geoespaciais.

Figura 6.6 – Aplicação do Algoritmo GARP para Ocorrências de Espécies *furcata boliviana*

É possível aplicar zoom no mapa apresentado com o auxílio da barra superior à esquerda, bem como deslocar o mapa para quaisquer direções bidimensionais com o auxílio do mouse ou mesmo das setas acima da barra lateral de zoom. No canto inferior esquerdo, o *Yahoo! Maps* indica a escala na qual o mapa está representado. Caso seja interesse do usuário, é permitido eliminar quaisquer *Layers* desenhadas no mapa com exceção da *Layer Base* que corresponde à camada mais interna do mapa.

6.4 Aplicação Web para Agricultura de Precisão

Na Figura 6.7, é possível identificar uma aplicação mais completa na qual são apresentados dois mapas para análise comparativa do tomador de decisão. Assim, as informações dos dados brutos de produtividade são mostradas em um mapa e as informações após a aplicação do algoritmo *Filtering*, no mapa ao lado.

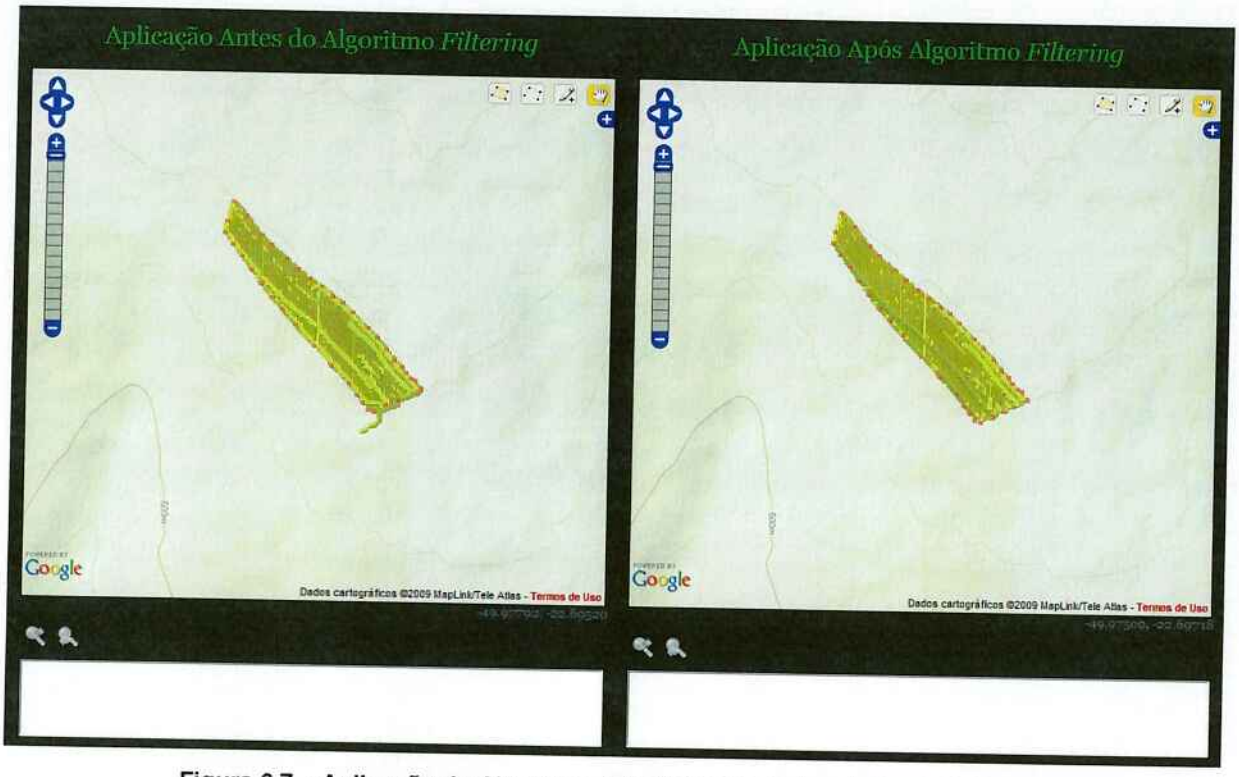

Figura 6.7 – Aplicação do Algoritmo *Filtering* para Dados de Produtividade

Esta aplicação de mapas possui as mesmas funcionalidades apresentadas na aplicação web de Biodiversidade e algumas outras. Uma delas que também foi desenvolvida e documentada no capítulo 5 é o armazenamento do histórico de ações do usuário, representados pelos ícones na forma de lupa logo abaixo dos mapas. Eles funcionam analogamente aos botões Voltar e Avançar dos navegadores web (ou *browsers*). A única diferença é que apenas a aplicação do mapa em questão sofre alteração, visto que são realizadas requisições assíncronas. Os blocos de texto abaixo desses botões permitem exibir informações capturadas no Banco de Dados quando o usuário clica em um dos pontos do mapa.

Assim como, a aplicação de Biodiversidade, as restrições da aplicação são relacionadas ao *MapServer* e ao *Openlayers*. Uma questão importante quanto à limitação desses tipos de aplicação com processamento de dados geoespaciais é o desempenho. Existem aplicações que processam uma quantidade de dados muito grande, tornando a aplicação bastante lenta. Em alguns casos, inclusive, é necessário solicitar o processamento com horas de antecedência para então serem visualizados os resultados.

Neste trabalho, foi realizado um estudo para otimizar o desempenho do processamento de dados geoespaciais. No entanto, não foi possível integrar a ferramenta estudada com a solução *easyGIS*.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma área relativamente recente do mundo tecnológico: o geoprocessamento. *Google Maps*, *Google Earth*, *Yahoo! Maps*, *Microsoft Virtual Earth*, entre outros, tem ganhado atenção recentemente, grande parte pela liderança das soluções do *Google*, trazendo uma forte convergência de pessoas e tecnologias para a área.

Os resultados deste trabalho nesta área do conhecimento foram a implementação de serviços de informações geográficas aliado ao processamento de algoritmos para tratamento de modelos em agricultura de precisão e biodiversidade, baseado em *Web Services* espaciais padronizados pela OGC. As tecnologias utilizadas foram *free* e *open source*, o que significa basicamente que é uma solução cujo custo é muito próximo do zero em termos materiais, mas que ainda assim é capaz de alcançar os resultados almejados. Como este trabalho é parte de um projeto maior, seus resultados serão, de fato, utilizados em aplicações para o agronegócio, na forma de modelos em AP, e para o estudo de questões relacionadas ao aquecimento global, resultado das aplicações para a biodiversidade.

A área de geoprocessamento, por ser relativamente nova, não possui material de referência em abundância para ser estudada. Assim, durante a condução do projeto foi necessária uma pesquisa muito forte atrás de fontes que pudessem ser úteis e confiáveis, elucidando o caráter de P&D do projeto. Além disso, fruto deste trabalho foi também a produção acadêmica, com a publicação de dois artigos pelo grupo na área, um para a SBIAgro e outro para o SIICUSP, ambos aprovados e já expostos. Outro artigo, com a solução final proposta, será escrito e submetido nos próximos meses, como parte dos resultados do projeto financiado com recursos do FINEP. Também vale ressaltar a interdisciplinaridade do trabalho, pois ele envolveu conceitos de uma ciência chamada geodésia, a qual tem por intuito o estudo da medida e da representação da Terra, ciência esta não comumente estudada em um curso de engenharia de computação, além de estudos em agronegócio e biodiversidade.

Quanto às evoluções desta solução e trabalhos futuros, existem principalmente duas frentes que podem ser adotadas. Uma delas é o estudo de soluções de cacheamento para a solução, com o intuito de tornar a renderizaçao do

mapa mais rápida e eficiente no caso de mapas com muitas camadas ou com muitas feições espaciais. Essa solução implementa um processo de renderização por meio de *caching* que armazena partes do mapa que já foram requisitados pelo cliente em um servidor independente, evitando assim o reprocessamento de alguns dados pelo servidor de mapas. Existem soluções *free* e *open source* disponíveis, como o *Tilecache* e o *GeoWebCache*. Tais soluções prometem uma melhora de desempenho no processamento das imagens, porém ainda requerem um estudo maior para serem aplicadas de fato. A outra é estudar o *GeoTools*, uma biblioteca em *Java* que possui ferramentas para lidar com dados espaciais e que permite até mesmo a implantação de um GIS a partir dela. Esta possibilidade permite a criação de um framework com funcionalidades que venham a estender as possibilidades de processamento geoespacial do *MapServer* e/ou do *GeoServer*.

Além destes, pode-se também citar como um possível trabalho futuro o uso do GeoExt, um cliente baseado no *OpenLayers*, porém com uma interface mais avançada e que permitiria a implementação de novas funcionalidades para o usuário.

REFERÊNCIAS

[1] BONGIOVANNI, R.; LOWENBERG-DEBOER, J. Precision Agriculture and Sustainability. Precision Agriculture. p.29, 2004.

[2] Bostongis – a testbed for GIS and mapping solutions. Disponível em: http://www.bostongis.com/. Acessado em: 22/07/2009.

[3] BROVELLI, M.A.; MAGNI, D. An ARCHAEOLOGICAL WEB GIS APPLICATION BASED ON MAPSERVER AND POSTGIS. 2003. Disponível em: http://www.commission5.isprs.org/wg4/workshop_ancona/proceedings/19.pdf Acessado em: 21/08/2009.

[4] Centro de Referência em Informação Ambiental (CRIA) – openModeller. Disponível em: http://www.openmodeller.cria.org.br/. Acessado em: 05/12/2009.

[5] Fernando Quadro - Esporte & Tecnologia. Disponível em: www.fernandoquadro.com.br. Acessado em: 14/04/2009.

[6] Geo.NET. Disponível em: http://geoprocessamento.net/. Acessado em: 02/08/2009.

[7] GeoExt. JavaScript Toolkit for Rich Web Mapping Applications. Disponível em: http://www.geoext.org/. Acessado em: 22/10/2009.

[8] Geoserver official home page. Disponível em: http://geoserver.org/display/GEOS/Welcome. Acessado em: 10/04/2009.

[9] Getools. GeoTools is an open source Java library that provides tools for geospatial data. Disponível em: http://www.geotools.org/. Acessado em: 15/08/2009.

[10] CHAPMAN, A.D.; MUÑOZ, M.E.S.; KOCH, I. Environmental information: placing biodiversity phenomena in an ecological and environmental context. Biodiversity Informatics 2. 2005. p 24-41.

[11] GOMES, M. M.; ROCHA, A.; COELHO, A. F.; SOUZA A. A. Acesso Interoperável à Informação Geográfica para Disponibilização de Modelos Urbanos 3D em Dispositivos Móveis. 2006. p 126-137.

[12] HILTON, B. N. Emerging Spatial Information Systems and Applications. In: ZHAO, P.; YU, G.; LIPING, D. (eds.). Geospatial Web Services. George Mason University. p. 1-35, 2006.

[13] Gis self learning tool. Acessível em: http://www.geom.unimelb.edu.au/gisweb/menu.html. Acessado em: 04/10/2009.

[14] GRISI B.U.; SANTANA, F.S.; NOGUEIRA, M.; KUNIYOSHI A.M.; SARAIVA A.M. Estudo Comparativo entre Ferramentas de Sistemas de Informação Geográfica para Integração Via Web Services. VII Congresso Brasileiro de Agroinformática (SBIAgro), Viçosa, Minas Gerais. 2009.

[15] Jacto S/A. Disponível em: http://www.jacto.com.br/. Acessado em: 15/04/2009

[16] MS4W – Mapserver for Windows. Disponível em: http://www.maptools.org/ms4w/. Acessado em: 15/06/2009.

[17] Mapserver official home page. Disponível em: http://mapserver.org/. Acessado em: 27/04/2009.

[18] MARSDEN, Richard. Overview: WMS, WFS, WCS. Geoweb Guru. 2009. Disponível em: http://geowebguru.com/articles/95-overview-wms-wfs-wcs. Acessado em: 27/08/2009.

[19] MOLIN, J.P.; MENEGATTI, L.A.A. Methodology for identification, characterization and removal errors on yield maps. In: 2002 ASAE Annual

International Meeting/CIGR XVth World Congress, 2002. Chicago. Proceedings. Illinois. 2002.

[20] MURAKAMI, E. Uma infraestrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão. 2006. 192p. Tese (Doutorando) – Escola Politécnica, Universidade de São Paulo. São Paulo, 2006.

[21] MURAKAMI, E.; SARAIVA A.M.; JUNIOR L. C.M. R.; CUGNASCA, C. E.; HIRAKAWA, A. R.; CORREA, P. L. P. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. Computers and Electronics in Agriculture, , v.58, p.37 - 48, 2007.

[22] NOGUEIRA, M.; GRISI B.U.; KUNIYOSHI A.M.; SANTANA, F.S.; SARAIVA A.M. Análise de Ferramentas para Prover Serviços Geoespaciais. XVII Simpósio Internacional de Iniciação Científica da USP (SIICUSP). 2009.

[23] OBE. Regina O. ; HSU Leo S. Postgis in Action. Disponível em: http://www.manning.com/obe/PostGIS_MEAPCH01.pdf. Acessado em: 17/07/2009.

[24] OGC – Open Geospatial Consortium. Disponível em: http://www.opengeospatial.org/. Acessado em: 15/04/2009.

[25] OGP Surveying & Positioning Committee. Disponível em: http://www.epsg.org/. Acessado em: 02/06/2009.

[26] OpenGeo. Acessível em: http://workshops.opengeo.org/. Acessado em: 12/09/2009.

[27] Openlayers official home page. Disponível em: www.openlayers.org. Acessado em: 08/06/2009.

[28] OpenModeller. Disponível em: http://www.openmodeller.sourceforge.net/. Acessado em: 05/12/2009.

[29] PERCIVALL, G. OGC Reference Model OGC 03-040. Open Geospatial Consortium. 2003. Disponível em: http://portal.opengeospatial.org/files/?artifact_id=3836. Acessado em 23/08/2009.

[30] RAMSEY, P., Refractions Research Inc, 2003. PostGIS Manual. Acessível em: http://www.geoconnections.org/developersCorner/devCorner_devNetwork/meetings/2002.05.30/postgis.pdf. Acessado em: 15/07/2009.

[31] RAMSEY, P. The State of Open Source GIS. Refractions Research Inc. 49p, 2007.

[32] RATKE, C. Implementation of an OGC conform distributed GIS. Tese de Bacharelado. Georg-August-University Göttingen. Göttingen. 73p, 2005.

[33] SANTANA, F.S. Uma infraestrutura orientada a serviços para a modelagem de nicho ecológico. 2009. 141p. Tese de Doutorado – Escola Politécnica, Universidade de São Paulo. São Paulo, 2009.

[34] SARAIVA, A.M. Tecnologia da Informação na agricultura de precisão e biodiversidade: estudos e proposta de utilização de *Web services* para desenvolvimento e integração de sistemas. 2003. 185p. Tese (Livre-Docência) – Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

[35] SOARES, A.C. Diagnóstico e modelagem da rede de distribuição de derivados de petróleo no Brasil. 2003. 156p. Dissertação de Mestrado – Departamento de Engenharia Industrial da Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 2003.

[36] SHEEHAN, M. 2009. "Geospatial Solutions in Challenging Economic Times". Acessível em: http://www.directionsmag.com/printer.php?article_id=3219. Acessado em: 18/07/2009.

[37] SONG, X.; KONO, Y.; SHIBAYAMA, M. An Open Source GIS Solution for Discovering Cambodia through Maps and Facts. Proceedings of the FOSS/GRASS Users Conference, Bangkok, Thailand. p.280-290, 2004.

[38] TU, S.; ABDELGUERFI, M. Web Services for Geographic Information Systems. IEEE Internet Computing, vol.10, n. 5, p.13-15, 2006.

[39] Worboys, M.; Duckham M.. GIS: A Computing Perspective. CRC Press, 2nd Edition. p.1-24, 2004.

APÊNDICE A – DIAGRAMA DE GANTT

7.1 A.1 Macrodivisões

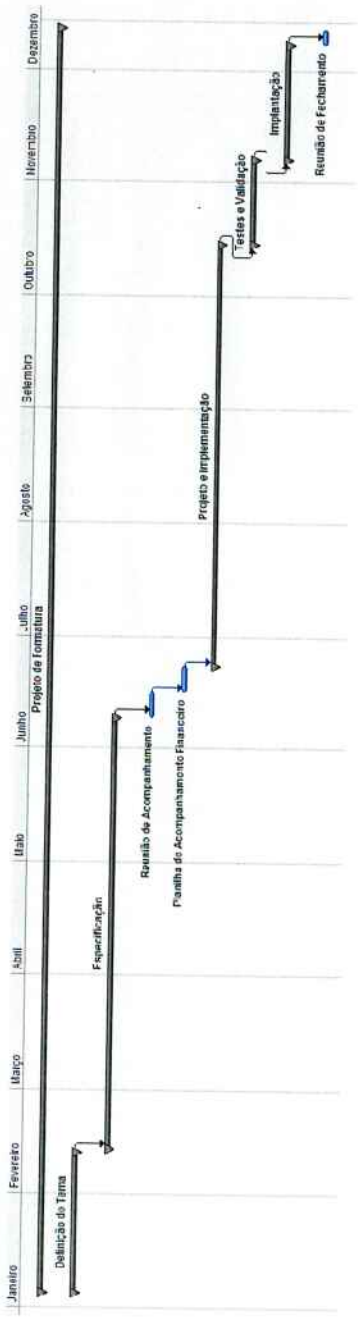

Figura A.1 – Macrodivisões do Diagrama de Gantt

7.2 A.2 Especificação

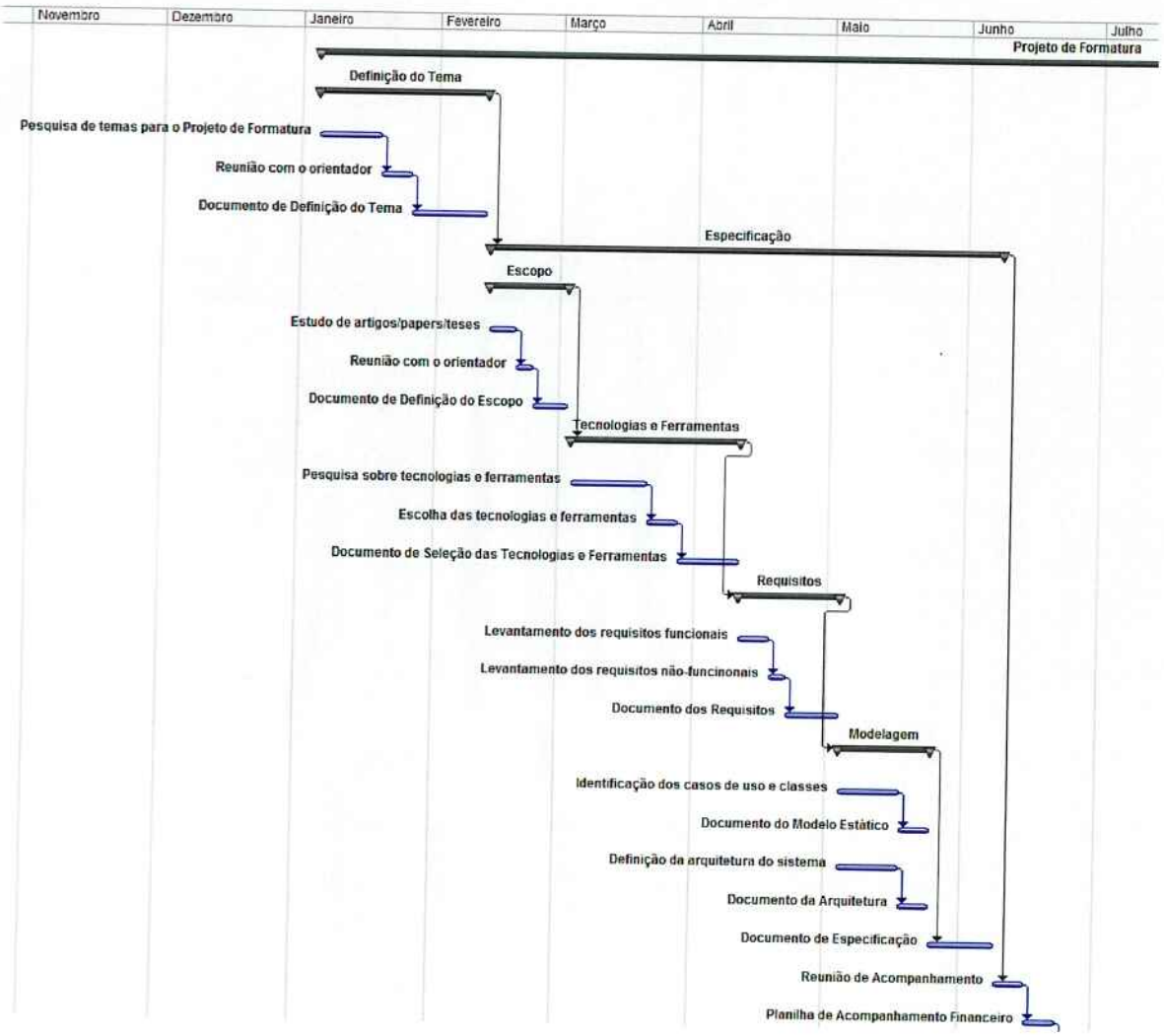

Figura A.2 – Módulo de Especificação do Diagrama de Gantt

7.3 A.3 Desenvolvimento

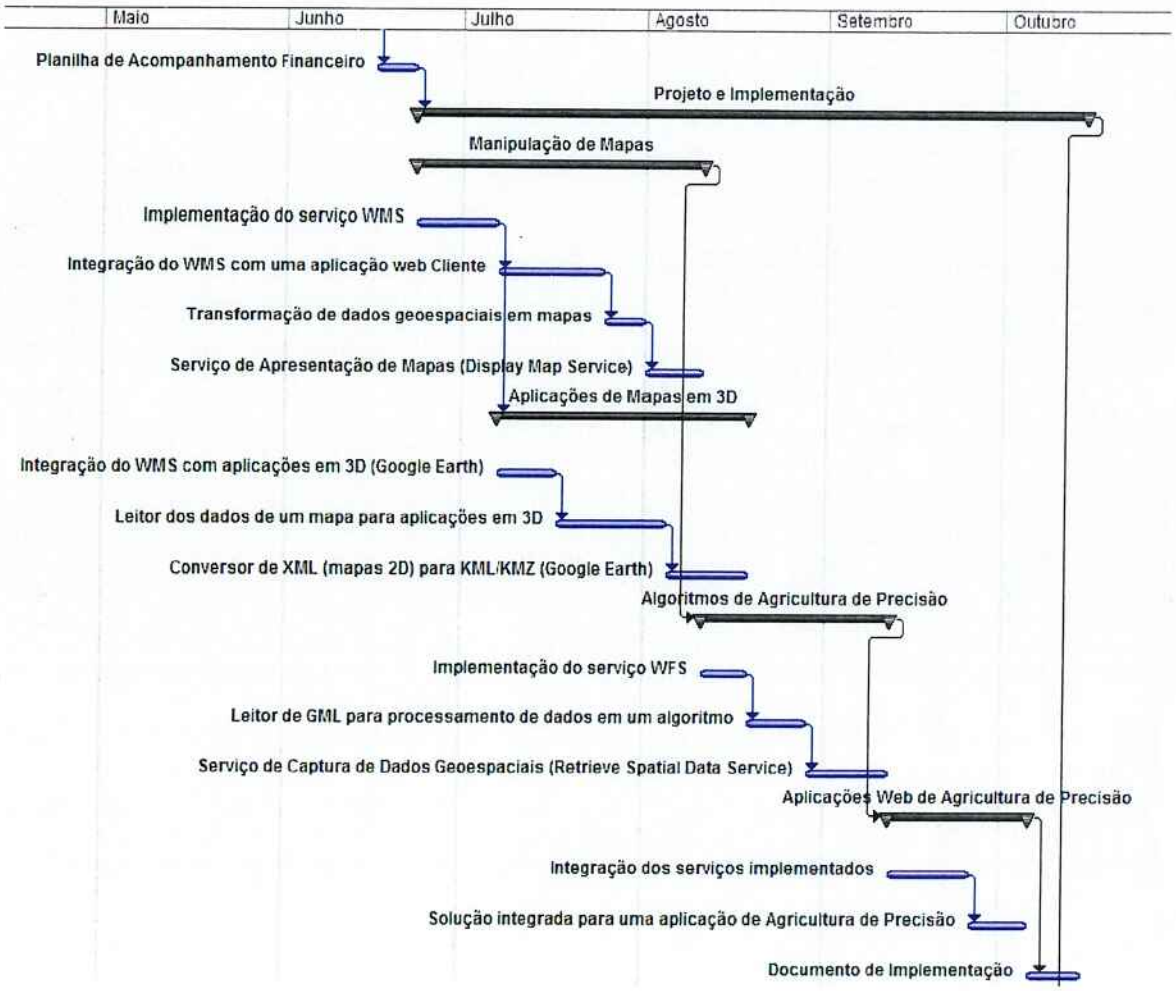

Figura A.3 – Módulo de Desenvolvimento do Diagrama de Gantt

7.4 A.3 Validação

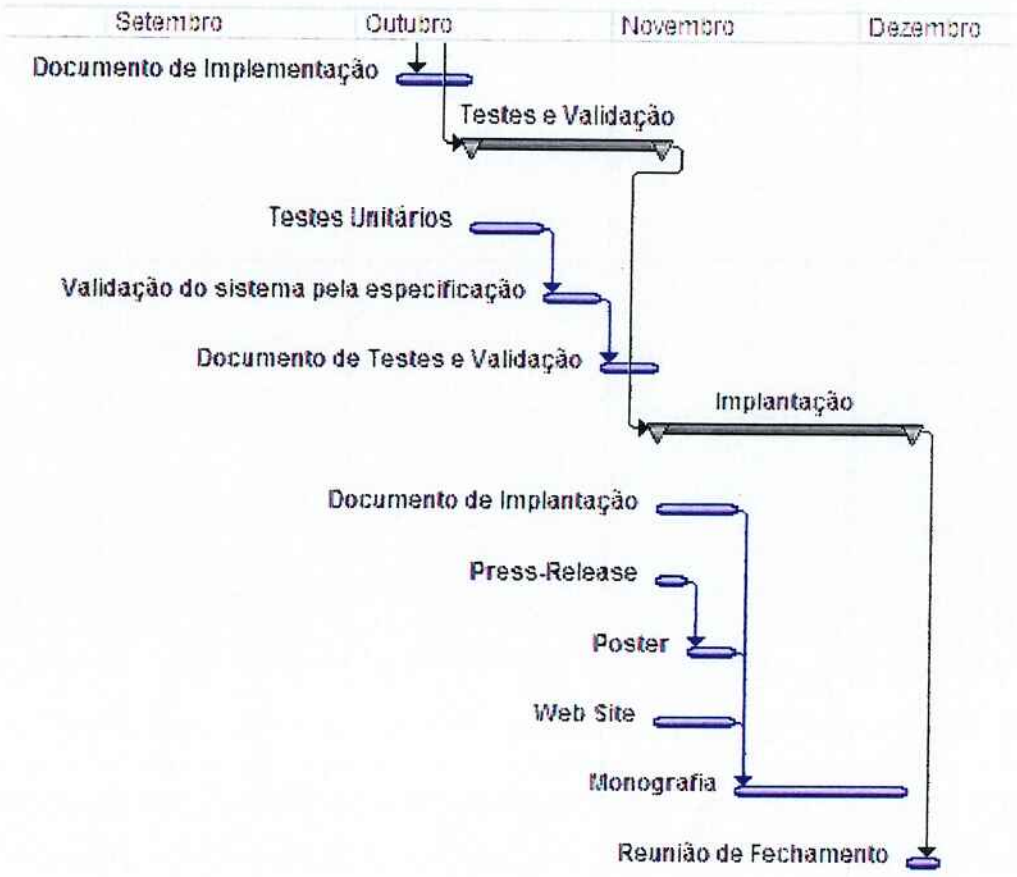

Figura A.4 – Módulo de Validação do Diagrama de Gantt