IGIS: um Framework para Sistemas de Informações Geográficas em N-Camadas usando um SGDB Objeto-Relacional

RODRIGO A. VILAR MIRANDA
CLÁUDIO S. BAPTISTA
RODRIGO R. ALMEIDA
BRUNO CATÃO
EDER PAZINATTO
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande,
Av. Aprigio Veloso, 882 Bodocongó 58107-970 Campina Grande – Pb – Brasil
{vilar, baptista, rodrigor, catao, eder}@dsc.ufcg.edu.br
http://www.lsi.dsc.ufcg.edu.br

Abstract. Many efforts to accomplish web based spatial information systems have been realized so far. However, some proposed solutions are too expensive and most of the times proprietary. Therefore, important issues in such web based systems such as interoperability and portability have not been contemplated. This paper describes a framework based on n-tier architecture, which aims to implement a web-based spatial information system. The main features of the framework is extensibility; the use of well-established standards such as OpenGIS and GML; lightweight and portable map rendering using SVG; and data persistence and spatial querying using an object relational database management system.

1 Introdução

Sistemas de Informações Geográficas (SIG) são hoje uma realidade em diversas empresas públicas e privadas, onde são usados em diferentes domínios. A aplicabilidade de SIG vai da produção de mapas a sistemas de apoio a decisão. SIGs têm sido usados em agricultura, meio-ambiente, controle de florestas, serviços de rede, dentre outros [1].

A larga aceitação da Web que, através dos browsers, tem se tornado um meio de comunicação com interface universal, vem pressionando a indústria de software de SIG para produzir soluções voltadas para Internet[2,3]. Entretanto, muitas das soluções de mercado ainda estão baseadas em arquiteturas duais e proprietárias, e, principalmente, têm sido disponibilizadas a um alto custo.

A importância de disponibilizar SIG na Web é notória dada a onipresença desta última. Além disso, a integração de SIG na Web propicia o uso de 'thin-clients', reduz os custos a longo prazo, aumenta a escalabilidade, e caso haja uso de padrões, pode-se ter interoperabilidade e transportabilidade destes sistemas.

Na arquitetura dual tem-se um SIG acoplado a um Sistema de Gestão de Banco de Dados (SGBD) de forma que os dados não -espaciais são gerenciados por este último, enquanto os dados espaciais são mantidos pelo primeiro. Esta arquitetura apresenta várias limitações, incluindo a necessidade de lidar com duas tecnologias (SIG + SGBD), o não aproveitamento de importantes serviços de SGBD para os dados espaciais tais como controle de concorrência, controle de integridade, tolerância à falhas, otimização de consultas, dentre outros [4].

Os recentes avanços dos SGBDs, sobretudo os orientado a objetos e objeto-relacionais, permitem que se estenda a arquitetura destes com o suporte a dados espaciais. Estes SGBDs constituem a chamada arquitetura integrada, em que as limitações mencionadas anteriormente na arquitetura dual não mais se aplicam. Entretanto, ainda existem alguns problemas na arquitetura integrada dentre as quais destacamos:

- A maioria das soluções existentes ainda é limitada a Application Programming Interfaces (API) proprietárias, sacrificando, portanto, a interoperabilidade e transportabilidade das aplicações espaciais;
- As interfaces gráficas disponibilizadas são muito limitadas, sem nenhum apoio a customização de interface para solução de uma aplicação específica. Isto dificulta o convencimento do usuário de SIG em

migrar para um SGBD com suporte espacial, pois lhe faltam mais atrativos do ponto de vista de usabilidade.

Ultimamente, vários padrões vêm sendo propostos para permitir interoperabilidade entre SIGs. Dentre estes, o que mais tem se destacado e tem sido aceito pelos fornecedores de SIG e SGBDs espaciais é o padrão do consórcio OpenGIS[5]. Outra proposta interessante, sobretudo para o transporte e representação de dados espaciais na Internet é a linguagem Geographic Markup Language (GML) [6]. GML é uma aplicação de XML que permite a representação de dados espaciais num formato compatível com o OpenGIS. No tocante à interface, o Scalable Vector Graphics (SVG) da W3C é um outro padrão que tem sido proposto para o desenho de dados vetoriais em aplicativos independente de plataforma. Com SVG podemos desenhar pontos, linhas e polígonos; colocar textos e apresentar dados raster. Além disso, o plugin SVG já traz algumas funções embutidas como zooming e panning, além de busca textual em textos contidos no mapa apresentado[7].

Neste trabalho, apresentamos uma arquitetura em 3 camadas para SIG na Web usando a arquitetura integrada, baseada no OpenGIS e fazendo uso de GML e SVG. Na camada de apresentação usamos SVG para o desenho dos mapas e imagens; na camada de aplicação, implementamos um framework que permite acessar dados espaciais e não-espaciais na camada de dados (usando OpenGIS SQL) e enviar estes dados para camada de aplicação; a camada de dados possui um tradutor OpenGIS SQL para o SQL proprietário do SGBD específico denominado EasyGeo. No protótipo desenvolvido, usamos o SGBD Objeto relacional Oracle 9i na camada de dados [8].

O restante deste artigo está estruturado como segue. A seção 2 relata trabalhos correlatos, a seção 3 apresenta a arquitetura do IGIS. A seção 4 discute aspectos de implementação. A seção 5 apresenta cenários de uso da ferramenta e, finalmente, a seção 6 apresenta as conclusões e discute futuros trabalhos a serem realizados.

2 Trabalhos Relacionados

A integração da Internet com SIG vem sendo discutida há bastante tempo. Várias soluções já foram propostas pela academia como também pela indústria. Destacamos aqui duas soluções da indústria: Alov [9] e Oracle MapViewer [8].

Alov é um SIG, oferecido de forma gratuita, que tem duas versões: 'standalone' e cliente-servidor. Na versão 'standalone', um applet é executado e o usuário pode configurar a aplicação via XML. Na versão cliente-servidor usa-se servlets que implementam a lógica da aplicação espacial que se comunica com um applet que implementa a camada de apresentação. Alov implementa o padrão OpenGIS e suporta dados vetoriais de SIGs renomados como Arcview [10] e Mapinfo [11], como também suporta raster (gif, jpg, e MrSID). Alov implementa zooming, panning e overlay, porém, tem pesquisa limitada apenas a dados não-espaciais. Desta forma, podemos consultar o nome de uma cidade no mapa, mas não podemos saber quais os hospitais estão mais próximos de um dado ponto, ou seja não há consultas espaciais.

O Oracle MapViewer possui 3 componentes: uma biblioteca de classes Java que permite a exibição de mapas; uma API XML que provê uma interface programática ao MapViewer; e uma ferramenta de definição de mapas que pode ser usada para gerenciar metadados num banco de dados espacial Oracle. O MapViewer permite a exibição de mapas armazenados no SGBD Oracle usando o Spatial Cartridge. Suporta apenas 2 dimensões e não trabalha com conectividade. Como reconhecido pelo fabricante, o MapViewer não pode ser considerado como um servidor de mapas.

Outras soluções de fabricantes renomados existem como o GeoMedia WebMap da Intergraph, o Autodesk MapGuide, o MapInfo MapXtreme e o ESRI ArcIMS. Porém, todas estas soluções são proprietárias e requerem um alto investimento.

O projeto IGIS difere das soluções apresentadas acima em vários aspectos, dentre os quais destacamos:

- Uso de padrões OpenGIS e GML que possibilita interoperabilidade de SIG;
- Uso da linguagem SVG que oferece alta performance (quando comparada a Applets) e não requer a instalação de Java Virtual Machine nos clientes, apenas é requerido um browser e um plugin SVG (que é obtido gratuitamente). Além de ser uma solução leve, podemos usá-la em outros dispositivos tais como PDAs e telefones celulares;
- Arquitetura em n-camadas que permite uma maior escalabilidade à solução e um melhor balanceamento de carga, uma vez que a camada de dados é responsável pelo gerenciamento destes, a camada de apresentação é responsável pela exibição dos mapas; e a camada de aplicação implementa a lógica do negócio.

3 Arquitetura do IGIS

O IGIS é um framework distribuído em três camadas: *apresentação*, com páginas JSP que geram HTML e SVG; *aplicação*, composta de classes em Java que implementam o modelo de dados ao nível de aplicação e controlam a lógica do sistema (nesta camada se encontra a maior extensibilidade do framework); *dados*, que consiste de um banco de dados objeto-relacional com extensões espaciais. A figura 1 apresenta a arquitetura do IGIS.

A escolha desta arquitetura foi baseada no padrão de projeto Model-View-Controller (MVC) [12], por prover maior estensibilidade e reusabilidade para o sistema, reduzindo o acoplamento dos dados com a lógica do sistema e com a interface gráfica.

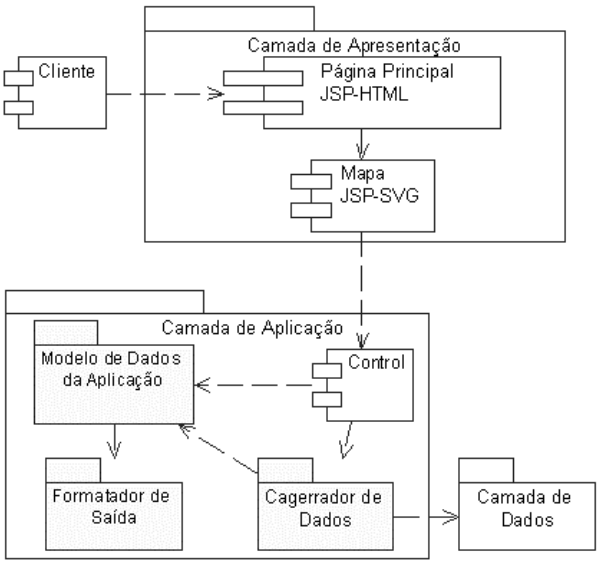

Figura 1- Arquitetura do IGIS

Nas próximas subseções, serão discutidas as três camadas do IGIS.

Camada de apresentação

A interface do IGIS consiste de páginas JSP divididas em duas categorias:

1. *interface funcional do sistema*, com a lista das camadas pertencentes ao mapa atual, barra de ferramentas e campos para interação com o usuário;
2. *mapa*, o desenho do mapa propriamente dito.

O arquivo JSP da interface funcional gera HTML com algumas funções JavaScript, que são úteis para a interação Usuário-Mapa. Para obter um mapa, esta página deve instanciar um objeto do tipo *IGISRequest*, informando onde e como obter os dados geográficos. O request (requisição) é submetido para um objeto da classe Map, que retorna os dados formatados em SVG.

A maior parte da formatação da saída dos dados foi transferida para a camada de aplicação, que possibilita a configuração da saída através de XML. Desta forma, o IGIS poderá ser personalizado para exportar os mapas em outros formatos como, por exemplo, GML, PNG, GIF, JPG, além de formatos proprietários (ArcInfo, MapView).

Nesta camada estão presentes as funcionalidades: visualização de dados raster e vector, sobreposição (overlay) de camadas, zoom, pan, exibição das coordenadas onde está o mouse e consultas. Na figura 2, pode-se ver uma tela típica do IGIS.

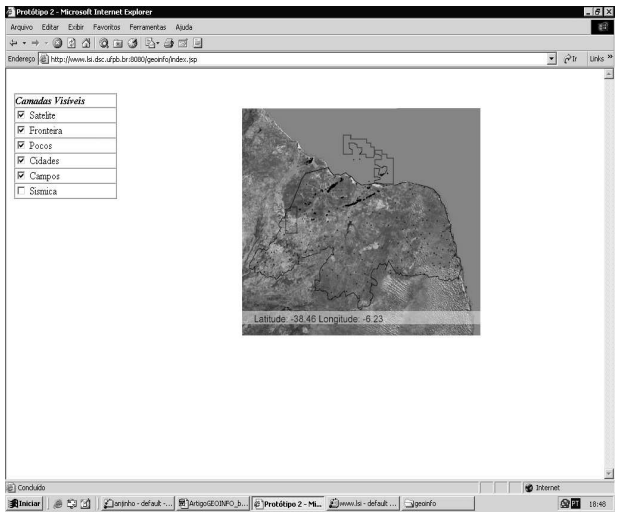

Figura 2- Interface do IGIS

Camada de aplicação

Esta camada é o núcleo do IGIS, pois recebe as requisições do cliente, obtém as informações desejadas na camada de dados, convertendo-as para o modelo OpenGIS, formata as informações para um tipo de saída específico e as retorna para a camada de apresentação.

Como pôde ser visto na Figura 1, a camada de aplicação é dividida em quatro módulos:

1. *Controle*, que define a seqüência de ações que esta camada deve fazer deste a requisição dos dados até sua devolução para a interface. Eis a seqüência (ver algoritmo a seguir em pseudo-código):

```
imprimeMapa(requisicao)
  //Repassa a requisição para o
  //Carregador de Dados
  Colecao c =
        carregador.carrega(requisicao);
  //Solicita a formatação de cada dado
  //Junta o resultado em uma String
  String resultado = "";
  Itera em c
    String geom =
        c.next().formata(requisicao);
    resultado += geom;
  retorna resultado;
```

As linhas em negrito representam os pontos onde o framework pode ser estendido, pois todos os carregadores possuem uma interface comum; da mesma forma todos os dados devem implementar um método informando com se formata a saída.

2. *Carregador de Dados*, uma interface comum para a obtenção de uma coleção de dados espaciais, a partir de uma consulta em SQL . Conseqüentemente, esta interface torna o IGIS independente do SGBD utilizado. Estão implementados os suportes para o SGBD Oracle 9i com o cartucho Spatial; porém novos suportes podem ser adicionados, apenas implementando a interface do Carregador de Dados para, por exemplo, Postgresql e IBM DB2.

3. *Modelo de Dados da Aplicação*, que foi gerado a partir da necessidade de se manipular os dados de uma forma independente do banco de dados. Portanto, foram implementados Java Beans segundo a definição do Open GIS Consortium. Todavia, houve uma extensão desta definição, pois foram encontradas lacunas para o uso de labels (textos descritivos), imagens raster, e para o agrupamento de dados em camadas (layers).

Por conseguinte, um novo modelo para dados geográficos foi criado, conforme apresentado na figura 3, onde a interface mais genérica é *Formatable*, que representa qualquer elemento que pode ser formatado para ser exibido na interface do usuário. A classe *AbstractFormatable* possui todos os atributos e métodos comuns a todos os "formatables". Desta forma elimina-se nas suas subclasses a necessidade de implementar os métodos da interface *Formatable*, pois uma geometria não precisa implementar nenhum método, além dos que implementaria normalmente, para ser acoplada no framework.

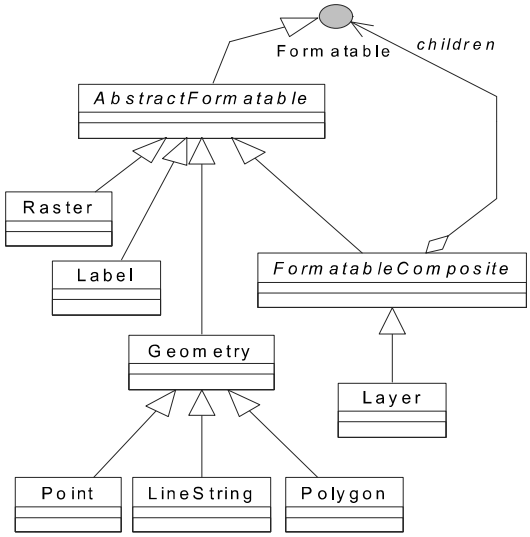

Figura 3- Modelo de Dados do IGIS

Herdando da classe *AbstractFormatable*, estão os objetos concretos, que serão exibidos no mapa: Imagens Raster, Labels e Geometrias; além do conceito de camadas (Layer), que é um conjunto comum ou temático de dados geográficos agrupados. Para projetar a implementação de camadas, foi usado o padrão de projeto *Composite* [13], inclusive com um Composite Abstrato: o *FormatableComposite*.

Pode-se exemplificar a flexibilidade deste modelo de dados com um fato ocorrido durante o desenvolvimento do IGIS: a necessidade de implementar um componente Java bean para imagens raster que não tinha sido notado no início do projeto. Todavia, a inserção deste novo tipo de dado custou menos de uma hora de programação e todo o sistema se adaptou as imagens raster. Não foi preciso nenhum esforço adicional para que as camadas também tratassem com imagens, por causa do padrão de projeto *Composite,* que desacoplou as camadas dos outros tipos de dados.

4. *Formatador de Saída*, pacote que surgiu como conseqüência do uso do padrão de projeto *Strategy* [13], pois o formato da saída foi modelado como estratégia. Assim sendo, um objeto do modelo de dados (*formatable*), possui um formatador próprio (*formatter*). Para alterar o formato de saída (para GML, por exemplo), não será necessário alterar o modelo dados, deve-se apenas mudar de formatador.

Cada formatador de saída recebe uma referência para o objeto que o possui, além dos detalhes da requisição como o 'minimum bounding rectangle' do mapa completo.

Extrai as informações necessárias e monta uma String com os dados formatados. No caso de SVG, por exemplo, cria uma 'tag' <g> para camada, <image> para imagens raster, <path> para linhas e polígonos, dentre outros.

A classe *AbstractFormatable* implementa todos os métodos necessários para a associação de um *formatable* com seu *formatter*.

Todavia, devido à não-padronização da estrutura das tags SVG, foi necessário criar uma classe para cada tag SVG diferente, com diferentes parâmetros de configuração. Para ocultar do resto do sistema esta complexidade, foi definida uma interface para fábricas de formatadores (*FormatterFactory*), baseada no padrão de projeto *Abstract Factory*.

O resto do sistema só precisa invocar os métodos para criar formatadores dos tipos presentes no modelo de dados: label, point, line, polygon e raster; a fábrica procura, na requisição do mapa, a localização de um arquivo XML que contém os dados para formatação de saída, e retorna um formatador pronto para cada tipo de dado.

Atualmente, existe uma implementação da fábrica de formatadores para SVG, porém uma fábrica para GML será implementada com a continuidade do projeto.

Camada de dados

IGIS fez uso do Oracle 9i para persistência dos objetos espaciais. O Oracle 9i tem um cartucho espacial "Oracle Spatial", que provê o armazenamento, recuperação e alteração de dados espaciais. Este cartucho implementa a especificação "SQL with Geometry Types" do OpenGIS, através do esquema chamado MDSYS. O esquema MDSYS é composto dos seguintes componentes: um esquema que prescreve o armazenamento, sintaxe e semântica para a manipulação de tipos geométricos; um mecanismo de indexação espacial; e um conjunto de operadores e funções espaciais.

Apesar de oferecer um conjunto bastante útil de ferramentas para manipulação de dados espaciais, o esquema Oracle Spatial não simplifica o trabalho do programador. Pois, o esquema implementa todos os objetos espaciais suportados em apenas um tipo, o *SDO_GEOMETRY*. Este tipo é usado para representar todos os tipos previstos no modelo OpenGIS. Todavia, ao criar um SDO_GEOMETRY, devemos passar por parâmetro o tipo deste objeto geométrico, o que elimina a característica objeto-relacional no Spatial. Veja abaixo um exemplo de criação de um objeto do tipo linha, o qual informamos através do código "2002".

```
INSERT INTO linhas_sismica VALUES(
  ... -- Outros campos desta tabela
  , -- Campo geométrico
  MDSYS.SDO_GEOMETRY(
    -- Metadados da Geometria
    -- 2002 = Linha
    2002,NULL,NULL,
    -- Mais metadados da geometria ....
    MDSYS.SDO_ELEM_INFO_ARRAY(1,2,2),
    MDSYS.SDO_ORDINATE_ARRAY(-35.94,
-4.56, -36.74, -4.19, -37.31, -3.54)
  )
);
```

Para simplificar a implementação do modelo espacial e tornar sua persistência no SGBD mais elegante, criamos um esquema Objeto Relacional, o qual chamamos "**EasyGeo**". Na figura 4, apresentamos o modelo implementado pelo EasyGeo.

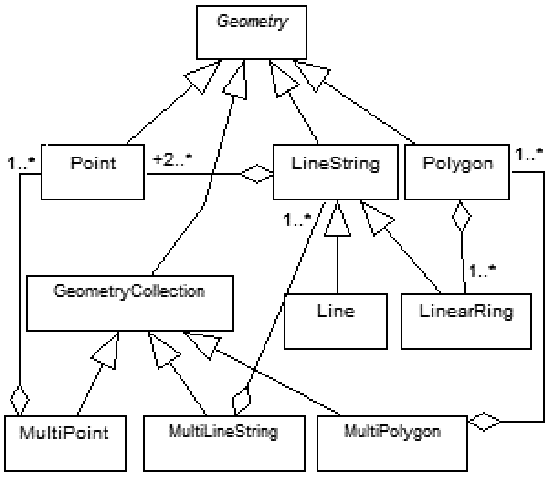

Figura 4- Esquema do EasyGeo

O EasyGeo oferece, ao programador, um conjunto de objetos geométricos que seguem o modelo proposto pelo OpenGIS, todavia aproveitando as características do modelo Objeto-Relacional. O EasyGeo é uma camada de objetos geométricos implementados usando o esquema "Oracle Spatial"; tirando do programador a necessidade de conhecer as especialidades da implementação provida pela Oracle. Com o EasyGeo, torna-se bem mais simples e legível a manipulação de dados geométricos, como pode-se ver a seguir:

```
INSERT INTO linhas_sismica VALUES(
  ... -- Outros campos desta tabela
  , -- Campo geométrico do tipo linha
  LineString_obityp.init(
    -- Possui um conjunto de pontos
    Point_varray(
      -35.94, -4.56, -36.74, -4.19,
      -37.31, -3.54
    )
  )
```

No EasyGeo, a semântica dos tipos geométricos não é definida através de códigos numéricos, e sim pelos próprios object types.

4 Aspectos de implementação

O IGIS foi desenvolvido sob a plataforma Java 2 Enterprise Edtion, da Sun Microsystems, portanto ele pode ser empacotado, distribuído e instalado (*deployed*) através de um Web Archive (.war).

Experimentalmente, o IGIS foi instalado no servidor JBoss. O JBoss é responsável por preparar um contexto onde as páginas JSP e HTML estarão acessíveis por HTTP; disponibilizar as classes da camada de aplicação para a camada de apresentação e preparar a conexão ao banco de dados.

A camada de dados está usando o Oracle 9i com o Spatial Cartridge para os dados espaciais.

5 Uma aplicação exemplo

Para exemplificar o uso do IGIS, foi criada uma aplicação exemplo sobre dados da indústria de petróleo no estado do Rio Grande do Norte. As fontes onde foram obtidas as informações foram: o IBGE, a ANP e a Emprapa.

A aplicação consiste de um mapa com as camadas:

- Fronteira: a fronteira política do estado do Rio Grande do Norte;
- Poços: poços de petróleo do tipo produção;
- Sísmica: linhas de sísmica;
- Campos: campos que a ANP lançará na quarta rodada de licitações;
- Cidades: cidades do Rio Grande do Norte e
- Satélite: foto de satélite da região.

A figura 5 ilustra o mapa com todas as camadas exibidas.

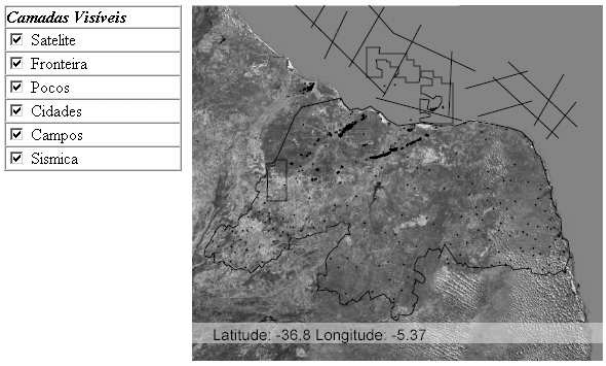

FIGURA 5- Mapa com todas as camadas

À esquerda do mapa, está a lista de camadas. Pode-se ocultar qualquer das camadas (ver figura 6).

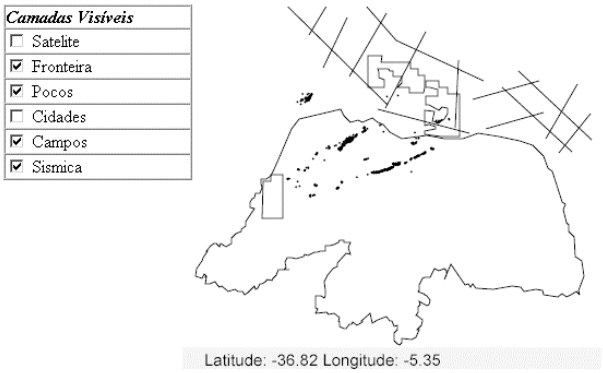

Figura 6- Mapa com as camadas satélite e cidades ocultas

O IGIS oferece a possibilidade de executar consultas especiais, portanto vejamos um exemplo: "Quais cidades estão dentro dos campos que a ANP disponibilizará na rodada 4?". O resultado desta consulta está apresentado na figura 7 (com zoom na região da resposta). Ao executar esta consulta, o IGIS solicita ao banco de dados, através de OpenGIS SQL, que seja aplicada a operação CONTAINS entre o polígono que representa o campo, e os pontos, que são as cidades.

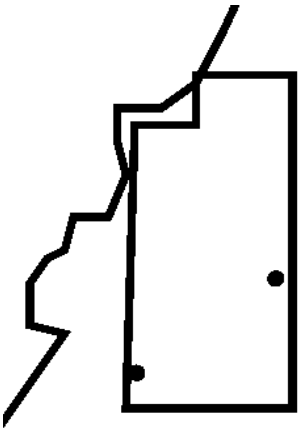

Figura 7- Resultado de uma consulta espacial

O plugin de SVG provê alguns serviços para o usuário como 'zooming' (figura 8) e 'panning' (figura 9).

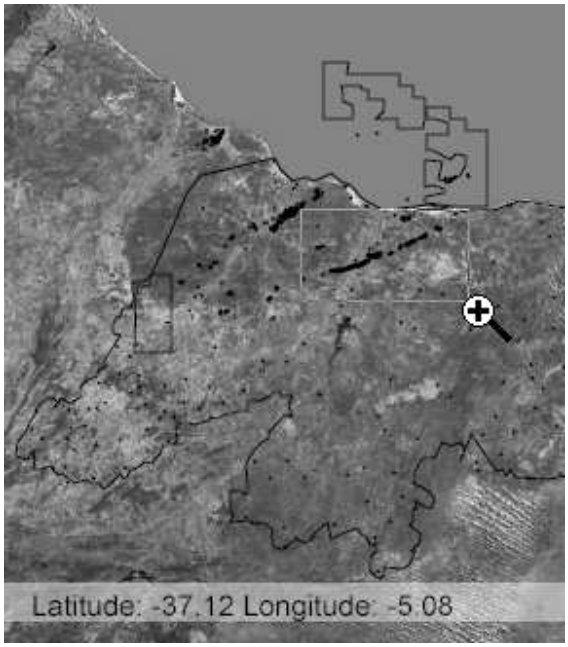

Figura 8- Operação de Zoom in

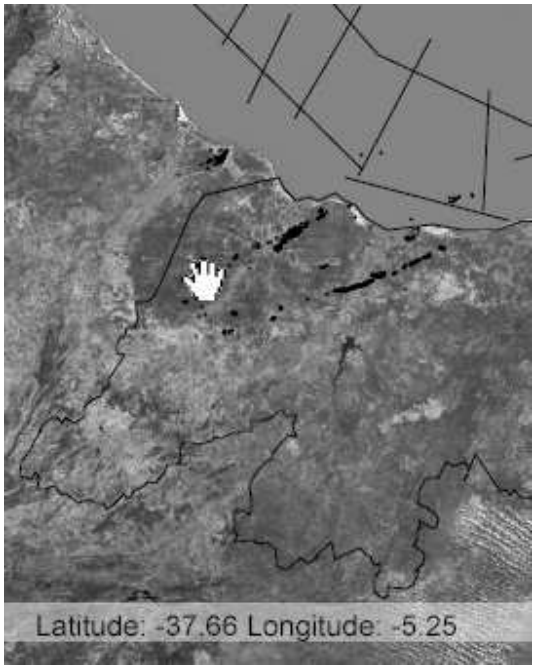

Figura 9- Ferramenta Pan

Para visualização de dados não espaciais o IGIS provê um mecanismo de consulta às tabelas relacionadas às camadas dos mapas. Por exemplo na figura 10, o usuário usa a interface de consulta não espacial para definir da tabela Cidades quais campos deseja consultar. A interface é inspirada na linguagem QBE (Query By Example) [14].

Consultar Tabela "Cidades"

Selecione as colunas da tabela

Nome	Exibir?	Restrição
Cidade_Nome	☑	
População	☐	
Prefeito	☐	
Estado	☑	Rio Grande do Norte
Altitude	☑	
Ponto	☐	

Consultar Apagar

Figura 10- Especificando a consulta à tabela de cidades

Na figura 10, o usuário quer saber quais os nomes das cidades do estado do Rio Grande do Norte com respectivas altitudes. O resultado desta consulta está apresentado na figura 11.

Tabela "Cidades"

Cidade_Nome	Estado	Altitude
Acari	Rio Grande do Norte	270
Açu	Rio Grande do Norte	27
Afonso Bezerra	Rio Grande do Norte	62
Água Nova	Rio Grande do Norte	270
Alexandria	Rio Grande do Norte	319
Almino Afonso	Rio Grande do Norte	236
Alto do Rodrigues	Rio Grande do Norte	13
Angicos	Rio Grande do Norte	110
Antônio Martins	Rio Grande do Norte	312
Apodi	Rio Grande do Norte	67
Areia Branca	Rio Grande do Norte	3
Arês	Rio Grande do Norte	52
Augusto Severo	Rio Grande do Norte	96
Baía Formosa	Rio Grande do Norte	4
Baraúna	Rio Grande do Norte	94
Barcelona	Rio Grande do Norte	124
Bento Fernandes	Rio Grande do Norte	111
Bodó	Rio Grande do Norte	560

Figura 11- Resultado da consulta não-espacial

6 Conclusão e trabalhos futuros

A demanda por sistemas de informações geográficas baseados na Web tem aumentado a cada dia. Várias soluções têm sido propostas, entretanto, ainda existe muito campo para investigação de melhores formas de prover serviços espaciais via Web.

Neste artigo, apresentamos o framework multi-camadas IGIS que usa padrões estabelecidos na indústria e usa uma arquitetura fortemente acoplada a um SGBD. Foi desenvolvido um protótipo para validar as idéias propostas na arquitetura.

Como trabalhos futuros pretendemos:

- fazer refatoramento da camada de apresentação, para suportar outros formatos de saída (GML, GIF, PNG, JPG);
- fazer refatoramento da interface formatter factory, para receber uma chave representando o tipo do formatador.
- Melhorar as interfaces de consulta espaciais e não-espaciais, através de uma avaliação de usabilidade.
- Incorporar um gazetteer (geo-localizador de nomes de lugares), que está sendo desenvolvido em outro projeto;
- implementar a camada de interface a consultas espaciais e interface para dispositivos móveis como PDA's e telefones celulares.

Agradecimentos

Os dois primeiros autores gostariam de agradecer a ANP/PRH25 pelo suporte financeiro a este projeto.

References

[1] P. A. Longley, M. F. Goodchild, D. J. Maguire, D. W. Rhind, *Geographic Information Systems and Science,* John Willey & Sons, 2001.

[2] C. S. Baptista, Z. Kemp "Spatial Information Systems and the Internet", *Innovations in GIS 6*, Taylor & Francis, 1999.

[3] F. Fonseca, C. Davis, "Using the Internet to Access Geographic Information: An Open GIS Prototype", *Interoperating Geographic Information Systems,* Kluwer Academic Publishers, 1999.

[4] P. Rigaux, M. Scholl, A. Voisard, *Spatial Databases with Applications to GIS*, Morgan Kaufmann, 2002.

[5] L. McKee, K. Buehler (eds) *The Open GIS Guide*. http://www.opengis.org/guide.

[6] S. Cox, A. Cuthbert, R. Lake, R. Martell (eds), *Geography Markup Language (GML) 2.0*, OpenGIS Implementation Specification, Fevereiro, 2001. http://opengis.net/gml/01-029/GML2.html

[7] J. Ferraiolo (ed), *Scalable Vector Graphics (SVG) 1.0 Specification*, W3C Recommendation, Setembro, 2001. http://www.w3.org/TR/SVG/

[8] Oracle Technology Network, *Oracle Documentation*, http://technet.oracle.com/docs/content.html

[9] Alov, ALOV Map, Free Java GIS, http://www.alov.org

[10] Arcview, ESRI – GIS & Mapping Software, http://www.esri.com

[11] Mapinfo, Mapinfo, http://www.mapinfo.com

[12] S. Allamaraju (ed), *Professional Java Sever Programming J2EE, 1.3*, Wrox Press, 2001.

[13] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns – elements of reusable object-oriented software*, Addison-Wesley, 1997.

[14] M. Zloof. "Query-By-Example: a Database Language", *IBM Systems Journal* 16 (1977), 324--343.