UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL**

ALEX CATANELLI MELLO

CRIAÇÃO DE MÓDULO DE RELATÓRIOS INTEGRADO A FERRAMENTA BRAIN OLAP PARA ANÁLISES MULTIDIMENSIONAIS

Bandeirantes

2011

ALEX CATANELLI MELLO

CRIAÇÃO DE MÓDULO DE RELATÓRIOS INTEGRADO A FERRAMENTA BRAIN OLAP PARA ANÁLISES MULTIDIMENSIONAIS

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná Campus Luiz Meneguel, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Msc. André Luis Andrade Menolli.

´

Bandeirantes

2011

ALEX CATANELLI MELLO

CRIAÇÃO DE MÓDULO DE RELATÓRIOS INTEGRADO A FERRAMENTA BRAIN OLAP PARA ANÁLISES MULTIDIMENSIONAIS

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná Campus Luiz Meneguel, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Msc. André Luis Andrade Menolli.

COMISSÃO EXAMINADORA

_____

Prof. Msc. André Luís Andrade Menolli.
UENP – Campus Luiz Meneghel

_____

Profª. Msc. Daniela de Feritas G. Trindade.
UENP – Campus Luiz Meneghel

_____

Prof. Msc. Ricardo Gonçalves Coelho
UENP – Campus Luiz Meneghel

Bandeirantes, __ de_____ de 2011

RESUMO

Pequenas empresas não dispõem de ferramentas adequadas ao elevado nível de informatização existente no mercado, muitas delas não possuem sistemas ou qualquer outro tipo de ferramenta que tornam as informações úteis ao setor executivo do negócio. Para que isso aconteça, a criação de um Data Warehouse é necessária para que essas informações detalhadas possam ser convertidas em padrões e aspectos que contribuam de alguma forma ou detectem erros e problemas. Ferramentas OLAP de código aberto são complexas e de grande escassez, a geração de relatórios é peça chave junto a analise para que a ferramenta forneça o ambiente proposto pelo formato, e é a criação de uma aplicação que supra tal necessidade a principal proposta desse trabalho.

Palavras-chave: OLAP, BI, Relatórios, Mondrian, Análise Multidimensional

ABSTRACT

Small businesses do not have the right tools for the high level of computerization in the market, many of them do not have systems or any other kinds of tools that make information useful to the executive branch of business. For this to happen, creating a data warehouse is required for this detailed information can be converted into patterns and aspects that contribute in some way or detect errors and problems. Open Source OLAP tools are complex and of great scarcity, reporting is key to the analysis with the tool provides the environment for the proposed format, and is creating an application that meets this need the main purpose of this study.

.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API – Application Programming Interface

BI – Business Intelligence

CSV - Comma Separated Values

DW – Data Warehouse

DM – Data Mart

ETL – Extract Transform Load

HQL - Hibernate Query Language

HTML - Hypertext Markup Language

MDX – Multi Dimensional Expression

MVC - Model-view-controller

MXML – eXtensible Markup Language of Adobe Macromedia

OLAP – Online Analytical Processing

OLTP – Online Transaction Processing

PDF - Portable Document Format

RIA - Rich Internet *Applications*

RTF - Rich Text Format

RUP - Rational Unified Process

SQL - Structured Query Language

UML – Unified Modeling Language

XHTML - eXtensible Hypertext Markup Language

XLS - Excel Spreadsheet

XML - eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O conceito de Business Intelligence (BI) surgiu na década de 80 e traz um novo formato de se analisar as grandes quantidades de informação, estas são armazenadas normalmente em um Data Warehouse (DW) /Data Mart (DM). Estes repositórios armazenam os dados em uma modelagem voltada para leitura de dados no formato de analises, que quando executadas por uma equipe executiva competente proporcionam mudanças significativas e lucros para o negócio ao qual se relacionam. Ferramentas são desenvolvidas para ajudar a fixar o conceito de BI no escopo do negócio. Existem softwares que gerenciam essa informação e a armazena de forma a melhorar o entendimento e usabilidade da grande quantidade de informação oriunda dessas analises.

Segundo Christine e Ferraro (2006) BI é a utilização de variadas fontes de informação para se definir estratégias de competitividade nos negócios da empresa, com a definição de regras e técnicas para a formatação adequada de grandes volumes de dados, visando transformá-lo em depósitos estruturados de informações.

O objetivo maior das técnicas de BI, neste contexto, está exatamente na definição de regras e técnicas para a formatação adequada destes volumes de dados, com a finalidade de transformá-los em depósitos estruturados de informações, independentemente da sua origem (REZENDE, 2005). De acordo com Magalhães (2008), Sistema de Informação Gerencial (SIG) é representado pelo conjunto de subsistemas, visualizados de forma integrada e capaz de gerar informações necessárias ao processo decisório. Esse tipo de sistema emprega a utilização de relatórios gerenciais que são documentos que consolidam as informações de forma estruturada e fornecem informações precisas e pontuais, capazes de auxiliar na melhor tomada de decisão, conforme objetivos e estratégias da organização.

É muito comum ficar em dúvida quanto a qual ferramenta usar para gerar os relatórios de sistemas ligados a análise de informação ou BI. Sendo assim, diversas ferramentas *open source* estão disponíveis para melhor atender a cada tipo de sistema, se mostrando uma maneira bem ágil para exibir as informações ao usuário, que estão sempre em busca de relatórios concisos para exibir e armazenar o conteúdo e conhecimento extraído das analises.

1.1 OBJETIVOS

1.1.1 OBJETIVOS GERAIS

Desenvolver um módulo de relatórios para ser agregado à ferramenta de BI Brain, esse módulo deverá gerar relatórios diretos sobre cada análise realizada, converter os resultados em formatos para impressão e documentação e organizar a informação de forma a auxiliar a geração de conhecimento sobre as análises realizadas.

1.1.2 OBJETIVOS ESPECIFICOS

- Gerar relatórios a partir de análises Multidimensionais;

-Integrar os *Frameworks iReport, Mondrian e Flex* em uma aplicação que forneça relatórios a partir de dados obtidos na analise obtida pela ferramenta Brain;

- Criar Interface que suporte a alteração do formato/layout do relatório, sendo possível a remoção e adição de medidas ligadas as dimensões previamente delimitadas pela análise;

- Gerar arquivos externos com extensão *.pdf para impressão e documentação.

1.2 JUSTIFICATIVA

Atualmente muitas empresas possuem sistemas para controle e gerenciamento de suas informações, sendo esses dados de clientes, dados financeiros, dados de fornecedores, dentre outros. Esses diversos tipos de dados são atribuídos a setores do negócio e muitas vezes observados de forma dispersa pela direção, isso faz com que o conhecimento intrínseco nessas massas de informação dificilmente chegue a diretoria. A criação da modelagem multidimensional de dados juntamente com os d*ata warehouses* fizeram com que esse conhecimento fosse obtido com maior facilidade e comodidade pela direção. A aplicação desse modelo é uma iniciativa que deve ser estudada com atenção pela diretoria, uma vez que gera um alto custo devido a aquisição de componentes mais sofisticados e uma mudança direta na forma de como o negócio é gerido.

As aplicações de código aberto voltadas o conceito de análise de dados e BI são raras e de alta complexidade, esses frameworks e APIs já existentes possuem documentação e são, apesar de complexos, são altamente funcionais e oferecem opções que se assemelham a das ferramentas de custo elevado.

A aplicação Brain é uma aplicação *open source* para analises em bancos de dados que utilizam a modelagem multidimensional, esta aplicação possui diversas funcionalidades e dentre elas, se mostrou necessário a alteração do módulo de relatórios. Faz-se necessário uma aplicação que se integre ao ambiente em questão provido pela ferramenta Brain, no formato de um módulo que tem como objetivo a estruturação das informações obtidas nas análises na forma de relatórios.

1.3 ORGANIZAÇÃO DO TRABALHO

Na seção 2 é feita a fundamentação teórica do trabalho e a apresentação dos principais conceitos e componentes utilizados seguidos da proposta de arquitetura. Na seção 3 discute a proposta de arquitetura utilizada na aplicação. Na seção 4 é apresentado o desenvolvimento, contendo os métodos de desenvolvimento, a documentação, estrutura e a discutição do funcionamento geral da aplicação.

Na seção 5 são apresentados os resultados obtidos. Na seção 6 é apresentada a conclusão seguida da proposta de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A criação de uma aplicação envolvendo a geração de relatórios em uma modelagem diferente da modelagem convencional dos bancos de dados transacionais requer um estudo detalhado do problema e dos principais conceitos envolvidos para que se possa decidir por qual linguagem, framework e componentes utilizar. Esta seção apresenta os principais conceitos da modelagem de banco de dados multidimensional e os frameworks envolvidos na criação da aplicação.

2.1 BUSINNES INTELLIGENCE (BI)

A Businnes Intelligence ou BI é um termo do *Gartner Group,* que em português atende pelo significado de inteligência empresarial, reflete o conceito surgido na década de 80 que descreve as habilidades das empresas para analisar dados e explorar informações desenvolvendo assim percepções e entendimentos a seu respeito, o que lhes permite incrementar e tornar mais pautada em informações e fatos a tomada de decisão.

Segundo Barbieri (2001), BI “representa a habilidade de se estruturar, acessar e explorar informações, normalmente guardadas em DW/DM com o objetivo de desenvolver percepções, entendimentos, conhecimentos, os quais podem produzir um melhor processo de tomada de decisão”.

A Figura 1 exibe um diagrama de Venn mostrando a Business Intelligence inserida em uma intersecção entre o negócio, o setor gerencial e a tecnologia da informação, deixando claro quais setores são envolvidos na implantação desse conceito.

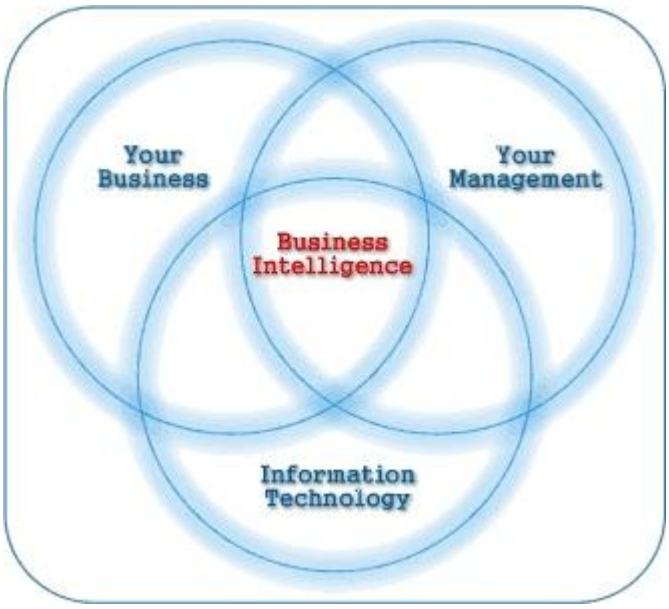

Figura 1 – Business Intelligence, demostrado no diagrama de Venn

Fonte: http://www.ronaldodavi.com/wp-content/uploads/2010/12/business-intelligence-graph.gif

Segundo TURBAN; EFRAIM (2009) é por meio da análise de dados históricos e atuais que gestores conseguem informações para servir de base para a tomada de decisão. O processo de BI baseia-se na transformação de dados em informações, após isto transformando as informações em decisões.

2.2 PROCESSO DE EXTRAÇÃO, TRANSFORMAÇÃO E CARGA DE DADOS (ETL)

Através de um processo conhecido como *Extraction, Transform and Load* (ETL) que traduzido ao pé da letra significa extração, transformação e carga, os dados dos sistemas de origens são extraídos, “limpos”, consistidos, unificados e carregados para um banco de dados, que é chamado de DW. Os dados carregados neste banco de dados utilizam-se de uma modelagem de dados específica para grandes volumes e alto desempenho, diferentemente das bases de dados transacionais controladas pelos sistemas voltados a esse tipo de modelagem.

ETL é um conjunto de processos e abrange tudo entre os sistemas operacionais de origem e a área de apresentação de dados (KIMBALL, 2002). É o processo de

preparação dos dados para que estes possam ser utilizados com integridade e formato adequados às consultas multidimensionais

As três etapas do processo de ETL de acordo com KIMBALL (2002)

1. Extração: Etapa do processo de obtenção de dados no ambiente de data warehouse. O processo de extração envolve a literatura e a compreensão de dados de origem e copia dos dados para manipulação nas etapas posteriores;
2. Transformação: Ocorre a filtragem de dados (correções de erros de digitação, solução de conflito de domínio, tratamento de elementos ausentes ou a divisão em formatos padrão), combinação de dados de varias origens, cancelamento de dados duplicados e atribuição de chaves de warehouse.
3. Carga: A fase de carga carrega os dados no Data Warehouse (DW). Dependendo das necessidades da organização, este processo varia amplamente. Alguns data warehouses podem substituir as informações existentes semanalmente, com dados cumulativos e atualizados, ao passo que outro DW (ou até mesmo outras partes do mesmo DW) podem adicionar dados a cada hora.

2.3 DATA WAREHOUSE (DW)

Segundo INMON (1999) Data Warehouse é um banco de dados contendo dados extraídos do ambiente empresarial, que foram extraídos e analisados e preparados para o processamento de consulta e não para o processamento de transações. Um DW como ilustrado na Figura 2, pode ter mais de uma fonte de dados, o que pode gerar problemas com relação ao formato da informação, uma vez que os demais repositórios podem estar em documentos de texto ou planilhas eletrônicas.

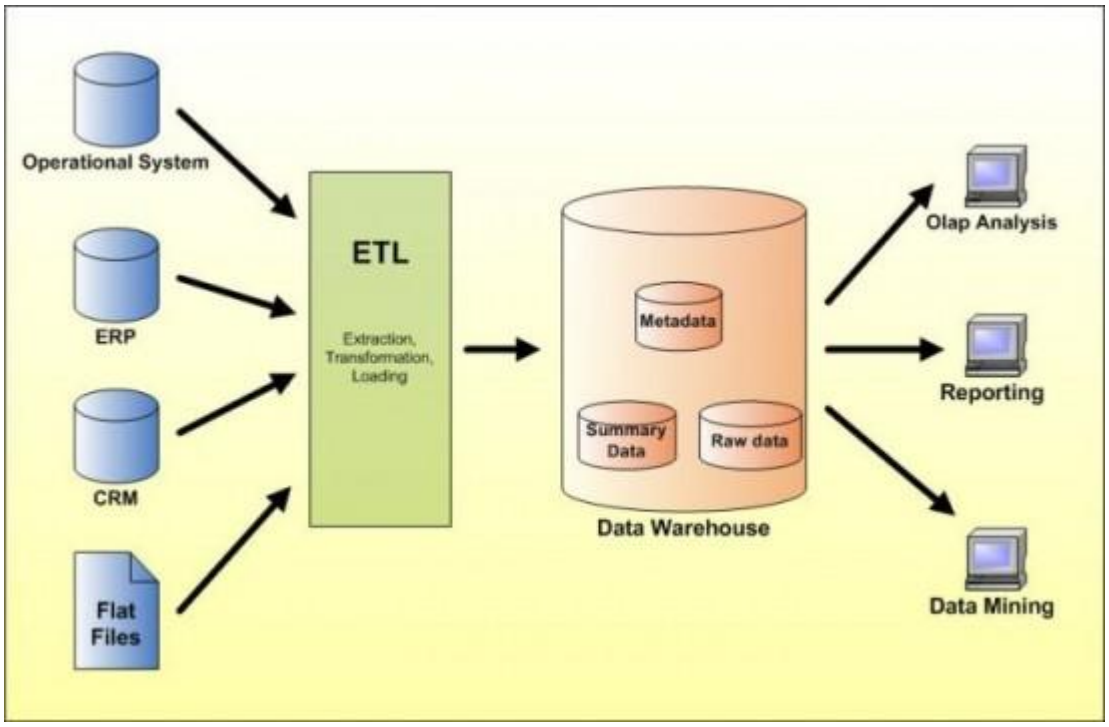

Figura 2 – Exemplo de Arquitetura Data Warehouse

Fonte: http://www.infoescola.com/informatica/data-warehouse

Os DW possuem uma arquitetura de banco de dados voltada ao aspecto gerencial abrangendo áreas como: suporte à tomada de decisão, planejamento estratégico, análise do comportamento de clientes e análise do desempenho de vendas. Funciona como um provedor de Informações, pois concentram todas as informações estratégicas e históricas, extraídas das bases de dados transacionais.

2.4 ONLINE ANALYTICAL PROCESSING (OLAP)

Segundo PENTAHO (2010) OLAP significa analisar grandes quantidades de dados em tempo real. Ao contrário do Online Transaction Processing (OLTP), onde as operações típicas de carga e modificação de um volume pequeno de registros, as operações são geralmente utilizadas para leitura de dados. O termo “online” implica que, apesar de enormes quantidades de dados estão envolvidos – geralemnte muitos

milhões de registros ocupando vários *gigabytes* – o sistema deve responder a consultas rápidas o suficiente para permitir uma exploração interativa dos dados.

Os passos para sua criação podem ser observados na Figura 3.

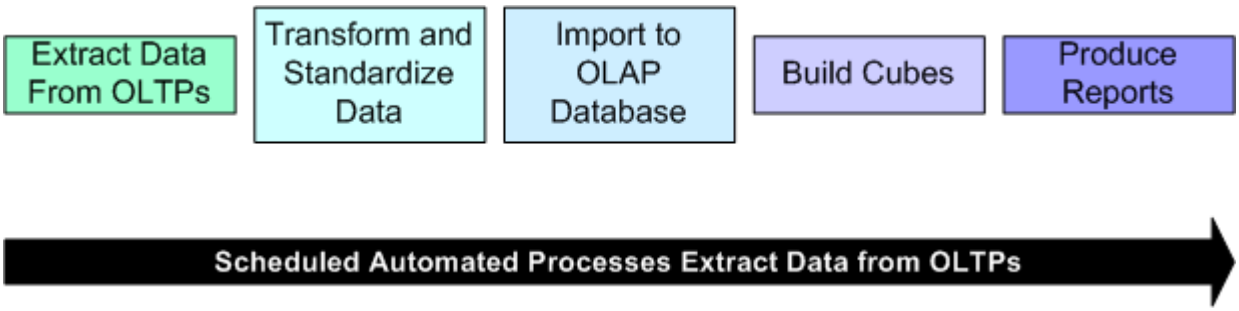

Figura 3 - Passos na criação de um sistema OLAP

Fonte: http://www.dwreview.com/OLAP/Introduction_OLAP.html

OLAP emprega uma técnica chamada Análise Multidimensional. Considerando que um banco de dados relacional armazena todos os dados na forma de linhas e colunas, um conjunto de dados multidimensional consiste de eixos e células (PENTAHO, 2010)

2.4.1 ANÁLISE MULTIDIMENSIONAL

A análise Multidimensional é uma forma representativa do cruzamento de informações. Dimensão é uma unidade de análise, representa um Eixo Principal no Estudo dos Dados.

Uma Dimensão pode possuir níveis hierárquicos. (Exemplo: A dimensão Região divide-se em Estados, que por sua vez divide-se em cidades ou em áreas de atuação, etc.).

Visão é o Cruzamento entre uma ou mais dimensões. (Exemplo: Estudo de um Produto ao longo do Tempo para uma determinada Região). Para melhor compreensão, tal análise é sempre associada a um Cubo, onde as arestas representam as Dimensões e cada célula representa um Indicador resultado de uma determinada visão como ilustrado na Figura 4.

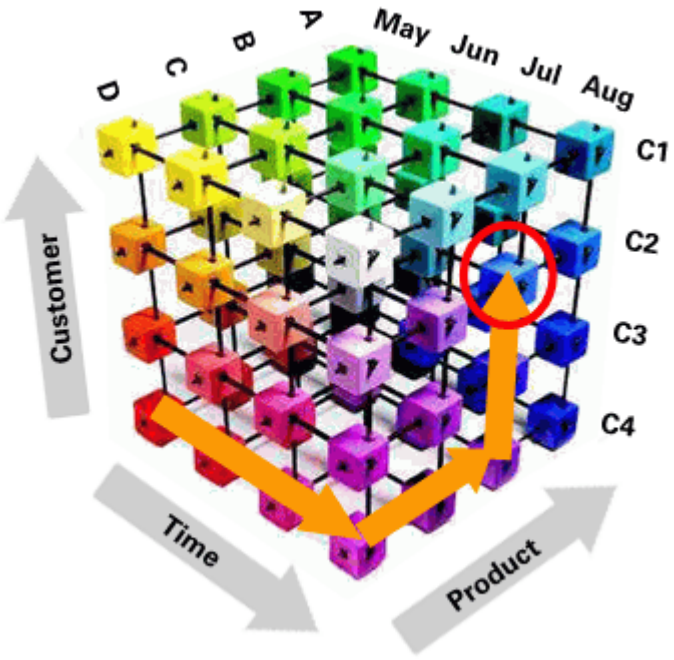

Figura 4 – Visão Multidimensional

Fonte: http://www.ealliancebusinessintelligence.com/images/articles/olap_obiee11g.gif

Uma tabela de fatos é segundo KIMBALL (2002) a principal tabela dimensional em que as medições numéricas de desempenho da empresa estão armazenadas. Utiliza-se da palavra fato para representar uma medição do negócio. A tabela de dimensão se relaciona com tabelas de fatos diretamente, possui colunas e atributos, estes que por sua vez descrevem linhas na tabela.

Funcionam como uma fonte primaria de restrições de consultas, agrupamentos e rótulos de relatórios KIMBALL (2002). Exemplos de ambas às tabelas podem ser observados na Figura 5:

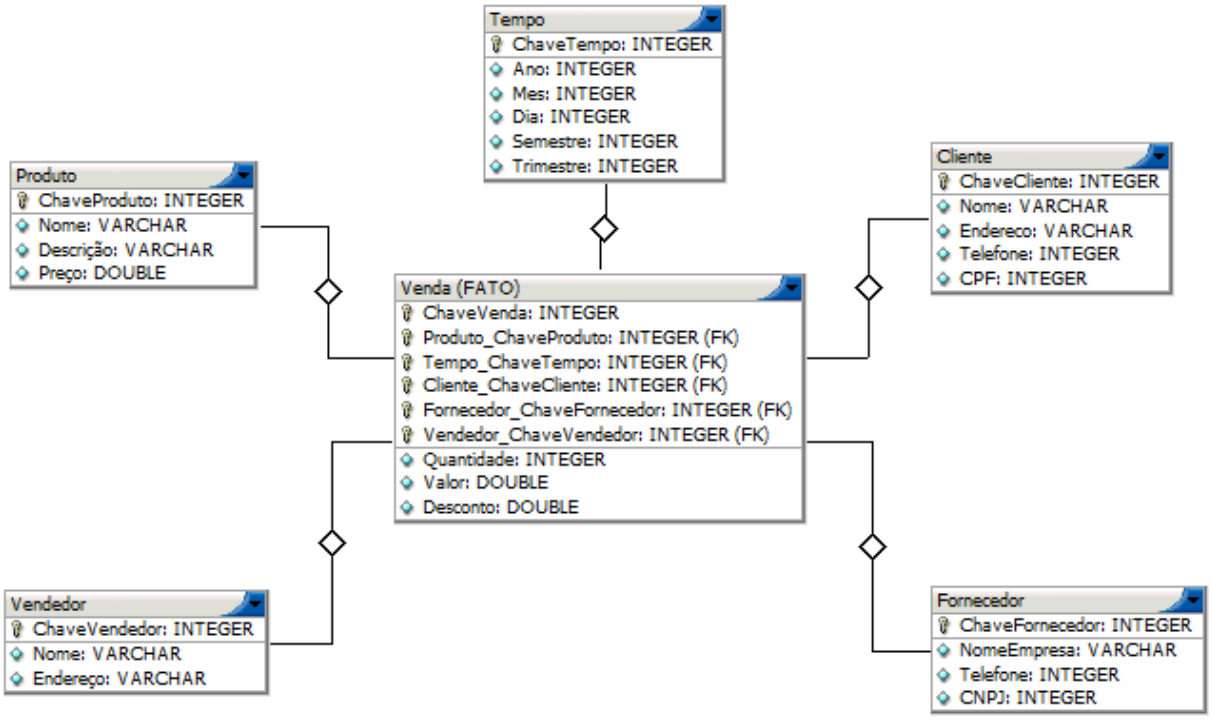

Figura 5 – Exemplo de modelagem multidimensional

Normalmente, os dados em uma organização são distribuídos em múltiplas fontes de dados e são incompatíveis entre si. Ex: Ponto-de-dados de vendas e vendas feitas via *call center* ou na Web são armazenadas em local e formatos diferentes. Seria um processo demorado para um executivo para obter relatórios OLAP, tais como: - Quais são os produtos mais comprados pelos clientes com idades entre 15-30? Parte do processo de implementação envolve a extração de dados OLAP a partir dos repositórios de dados diversos e tornando-os compatíveis.

Nem sempre é necessário criar um DW para análise OLAP. Os dados armazenados pelos sistemas operacionais tais como ponto-de-venda, estão em tipos de bancos de dados chamados OLTPs.Os processo de transação on-line nas bases de dados não tem nenhuma diferença do ponto de vista estrutural a partir de quaisquer outros bancos de dados. A principal diferença é a maneira pela qual os dados são armazenados.

2.5 SERVIDOR MONDRIAN

O sistema Mondrian OLAP é composto de quatro camadas sendo elas: a camada de apresentação, a camada dimensional, a camada de estrela e a camada de armazenamento, sua arquitetura ilustrada na Figura 6, (PENTAHO, 2010).

.

- A camada de apresentação determina o que o usuário final vê no monitor dele ou dela, e como ele ou ela pode interagir para fazer novas perguntas. Há muitas maneiras de apresentar conjuntos de dados multidimensionais, incluindo tabelas dinâmicas ,gráficos de linhas, circular e barras. Estes resultados podem ser apresentados nos formatos de formulários Swing, páginas web*.JSP ou transmitida a uma aplicação remota via XML. O que todas estas formas de apresentação têm em comum é a "gramática" multidimensional de dimensões, medidas e células em que a camada de apresentação realiza a análise desejada pelo usuário, obtendo a partir dessas analises a resposta enviada pelo servidor OLAP.

- A segunda camada é a camada dimensional, ela analisa, valida e executa consultas MDX. As consultas são avaliadas em várias fases. Os eixos são calculados em primeiro lugar, então os valores das células dentro dos eixos. Para a eficiência, a camada de células-dimensional envia os pedidos para a camada de agregação em lotes. Um transformador de consulta permite que o aplicativo para manipular as consultas existentes, ao invés de construir uma instrução MDX a partir do zero para cada solicitação. E metadados descrevem o modelo dimensional, e como ele é mapeado para o modelo relacional.

- A terceira camada é a camada estrela, ela é responsável por manter um *cache* agregado. Uma agregação é um conjunto de valores de medida ("células") na memória, qualificado por um conjunto de valores da coluna da dimensão. A camada dimensional envia pedidos de conjuntos de células,

se as células solicitadas não estão no *cache*, ou derivam de uma agregação no *cache*, o gerenciador de agregação envia uma solicitação para a camada de armazenamento.

- A camada de armazenamento é um RDBMS (*Relational Database Management System*) e é responsável por fornecer dados às células agregadas.

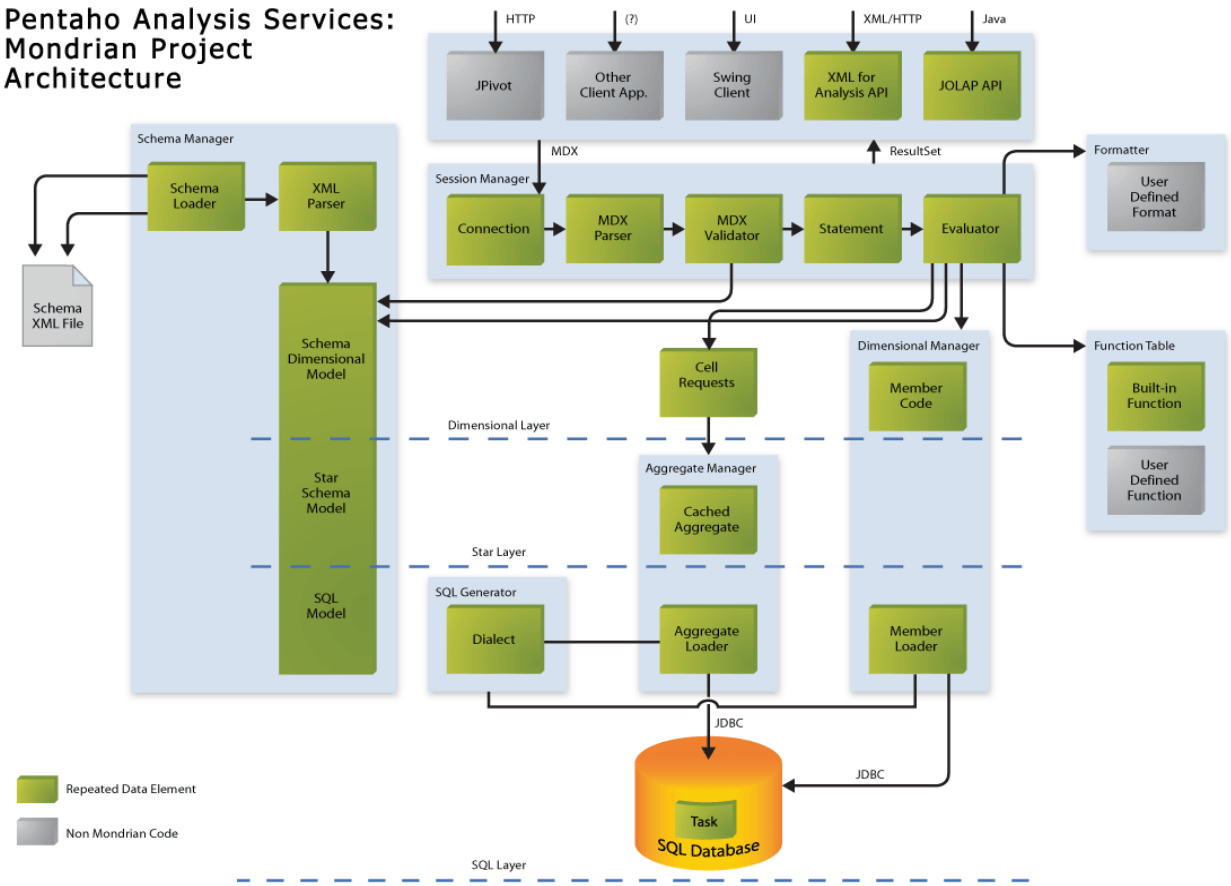

Figura 6 – Arquitetura Servidor OLAP Mondrian

Fonte: http://mondrian.pentaho.com/documentation/

Segundo PENTAHO (2011) esses componentes podem existir na mesma máquina, ou podem ser distribuídos em diversos usuários sendo que as camadas dois e três precisam necessariamente ser executadas no mesmo dispositivo, enquanto a

terceira camada pode se localizar em outra maquina dedicada exclusivamente ao armazenamento de informações.

Os dados então devem ser apresentados ao usuário de forma que os dados obtidos na analise sejam facilmente compreendidos e assim podem ser utilizados na construção de novas estratégias para o negócio.Um exemplo de apresentação de dados á partir de uma análise multidimensional pode ser observado na Figura 7.

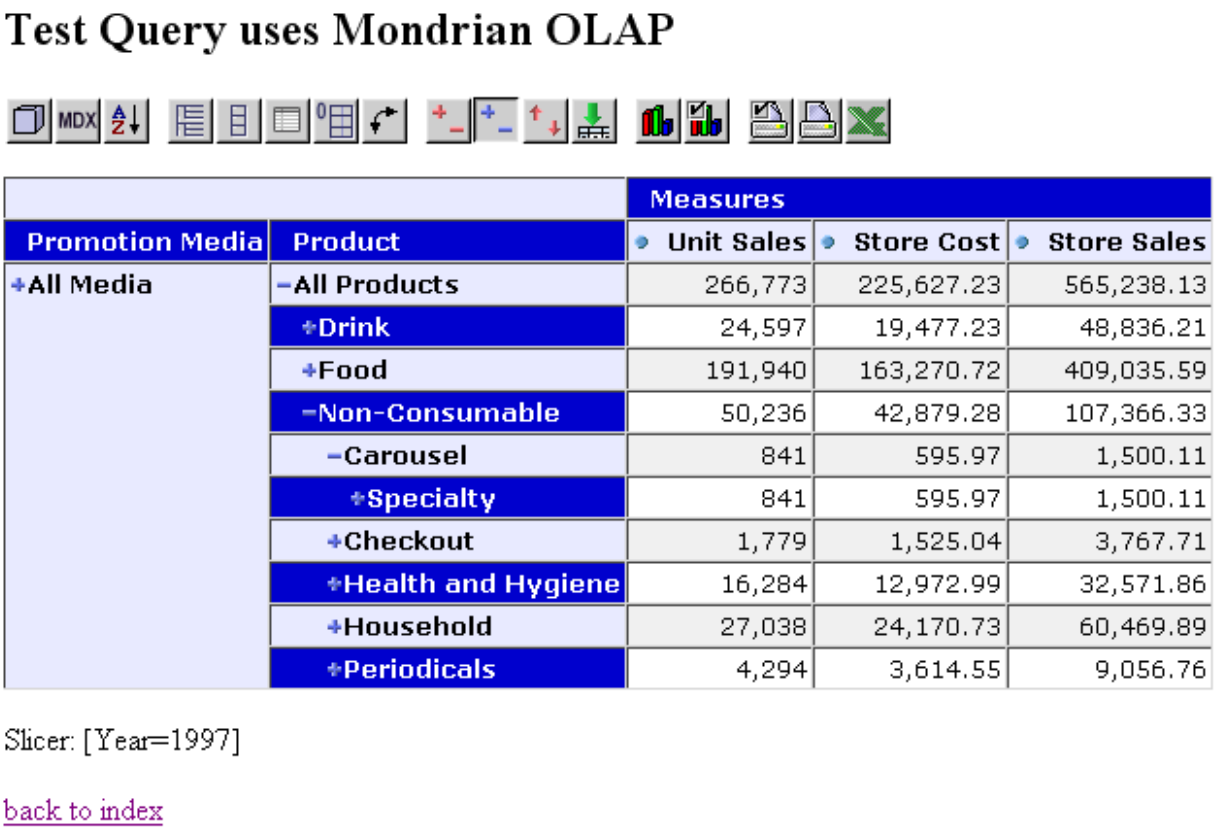

Figura 7 – Tabela JPivot para consultas multidimensionais

A Figura representa uma tabela JPivot que de acordo com a documentação apresentada em SOURCEFORGE (2011) é uma biblioteca de *tags* JSP personalizadas que torna uma tabela OLAP e permitir que os usuários executem análises OLAP típicas

como *slice and dice, drill-down e roll-up*. Ele usa o Mondrian como servidor OLAP e também suporta o acesso a bancos de dados XMLA.

2.6 MULTIDIMENSIONAL EXPRESSIONS (MDX)

De acordo com PENTAHO (2010) MDX é uma linguagem para consulta em bases de dados multidimensional, semelhante ao SQL é utilizado para consultar bancos de dados relacionais. Foi definido originalmente como parte do OLE DB para OLAP, especificação e uma linguagem similar, mdXML, faz parte do XML para a especificação.

MDX foi introduzido pela Microsoft com o Microsoft SQL *Server* OLAP *Services* em meados de 1998 como componente de linguagem do OLE DB que representa uma API para acesso universal a diversas bases de dados. Mais recentemente, o MDX apareceu como parte do XML for *Analysis* API.

A Microsoft propôs MDX como um padrão, e sua adoção entre os autores de aplicativos e outros fornecedores OLAP é cada vez maior relata (PENTAHO 2011).

```
SELECT {[Measures]. [Vendas Unit], [Measures]. [Store Sales]} on coluns,
  {[Produto]. Members} ON ROWS
FROM [vendas]
WHERE [Time]. [1997]. [Q2]
```

Figura 8 – Exemplo de consulta MDX

A Figura 8 representa uma expressão MDX contendo as medidas, dimensões e cubo, seguindo suas respectivas regas de sintaxe. É a peça chave de estudo do projeto e utilizada pela aplicação como fonte de informação.

2.7 FERRAMENTA DE DESIGN DE RELATÓRIOS (IREPORT)

Segundo JASPERFORGE (2011) IReport é uma ferramenta de designer para relatórios, que utiliza a biblioteca Java para relatórios a JasperReports. Com ele é possível fazer layouts sofisticados, contendo gráficos, imagens, sub-relatórios, tabelas de referência cruzada, etc. Os dados podem ser acessado via JDBC, TableModels, *JavaBeans*, XML, *Hibernate* e CSV, podendo também exportar para formatos em PDF, RTF, XML, XLS, CSV, HTML, XHTML, textos, DOCX ou *OpenOffice*.

É um projeto open-source exibido na Figura 9, foi desenvolvido pela equipe *JasperForge* que possui ferramentas para produção de relatórios, elaboração de relatórios operacionais etc. (JASPERFORGE, 2011).

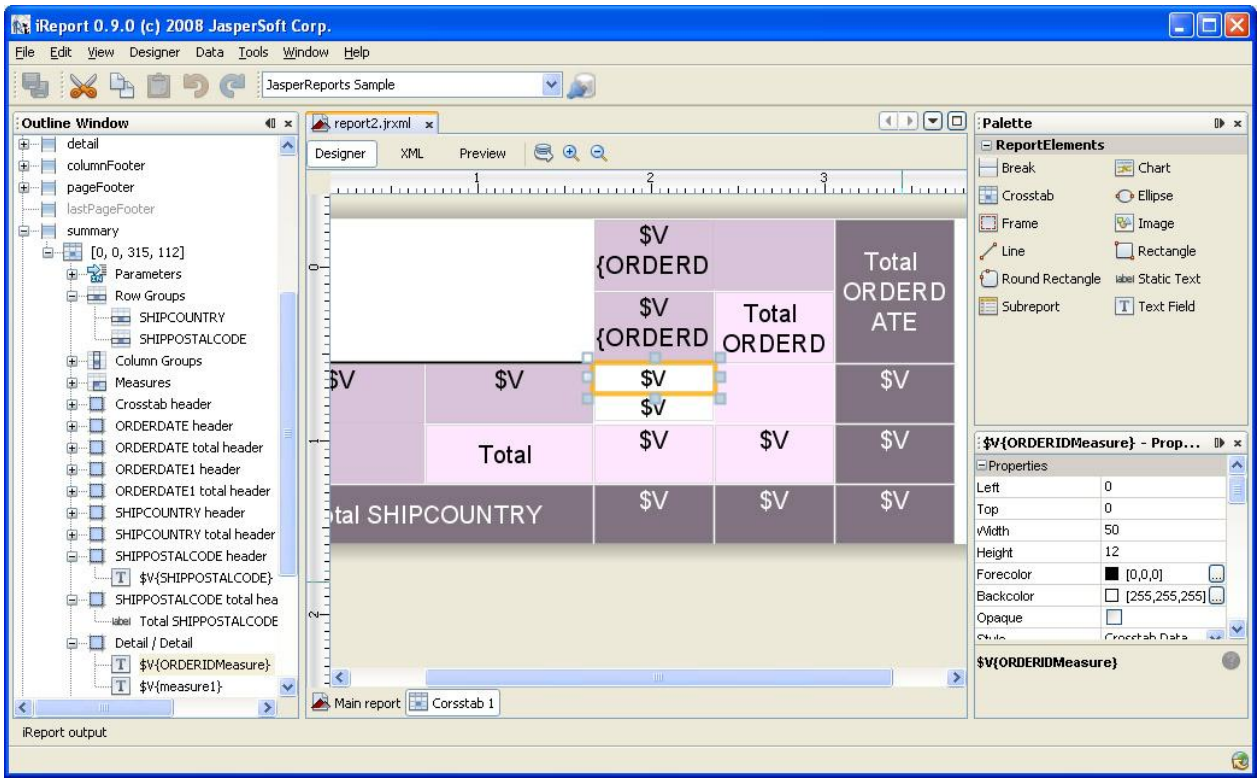

Figura 9 – Ferramenta IReport em execução

2.7.1 CICLO DE VIDA DO RELATÓRIO

O arquivo criado pelo IReport é um jrxml, arquivo XML que contém a definição do relatório, o relatório é desenhado completamente de uma forma visual através de uma estrutura em um arquivo XML. O jrxml deve ser compilado em um objeto binário chamado *.jasper, e é o que você precisa enviar para a aplicação para executar os relatórios, como mostrado na Figura 10 (JASPERFORGE, 2011).

Na execução do relatório será necessário, um arquivo de extensão *.jasper contendo a estrutura do relatório e os dados que serão distribuídos respeitando essa estrutura pré definida. De acordo com (GONÇALVES, 2011), o arquivo *.jasper e uma fonte de dados para o *JasperReports*. Há muitos tipos de fontes de dados como: arquivo XML, consulta SQL, arquivo CSV, uma (HQL), uma consulta com *JavaBeans*. O *JasperReports* é bem flexível proporcionando criar sua própria fonte de dados. Com o arquivo Jasper e a fonte dados o *JasperReports* são capazes de gerar o documento final em vários formatos. Essa sequência é ilustradapela Figura 10.

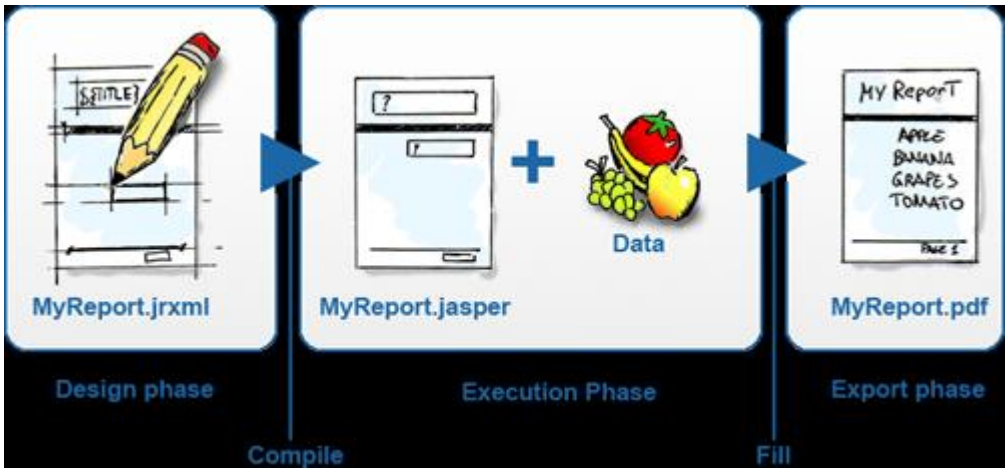

Figura 10 – Ciclo de vida do relatório

2.8 ADOBE FLEX

2.8.1 ESTRUTURA DO FLEX

O Flex é um *framework* gratuito e de código aberto para a criação de aplicativos expressivos móveis, da Web e para desktop. Permite que se crie aplicativos móveis e da Web que compartilham uma base de código comum, reduzindo o tempo e custo da criação de aplicativos e a manutenção de longo prazo. (ADOBE, 2011)

De acordo com ADOBE (2011) o framework Flex oferece um modelo moderno de linguagem e programação baseado em padrões que oferece suporte a modelos comuns de design., uma linguagem declarativa baseada em XML, é usada para descrever comportamentos e layout de interface de usuário, e ActionScript 3.0, uma linguagem de programação orientada por objetos, é usada para criar a lógica de cliente. O Flex inclui também uma biblioteca de componentes com mais de 100 componentes de interface de usuário comprovados e extensíveis para a criação de *Rich Internet Applications* (RIA) que em português significa aplicações avançadas para *internet*. Na Figura 11 é apresentada uma aplicação utilizando FLEX, pode ser observado seus aspectos de layout e alguns componentes.

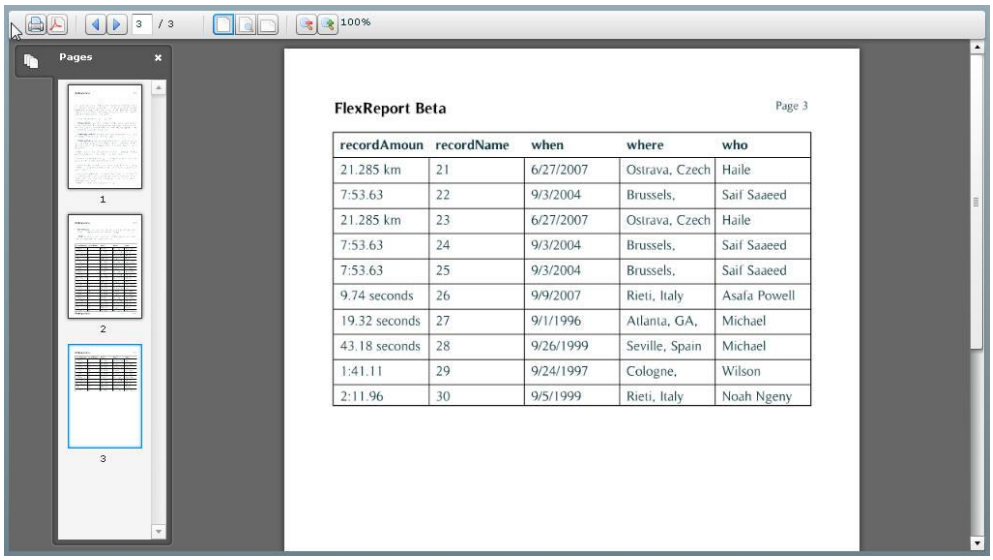

Figura 11 – Aplicação web para relatórios em Flex

O Flex pode ser usado para criar aplicativos que são executados no navegador através do software Adobe Flash. O Flash Player e o Adobe AIR são execuções de clientes de classe empresarial com gráficos vetoriais avançados, de alto desempenho capazes de lidar com os aplicativos mais exigentes, que processam grandes volumes de dados (ADOBE, 2011).

3 ARQUITETURA PROPOSTA

A proposta deste trabalho é a criação de um protótipo de aplicação para geração de relatórios a partir de analises multidimensionais, sua arquitetura se divide em interface gráfica, controle e modelo. É ilustrada em conjunto com os arquivos envolvidos na Figura 12 e segue os padrões de arquitetura *Model-View-Control* (MVC) que visam separar a lógica de negócios da lógica de apresentação. A interface gráfica fornece o ambiente de integração com o usuário e fornece as informações para o controle.

O controle é responsável pela análise desta informação, geração e utilização de arquivos externos que representam respectivamente o relatório gerado e arquivo de esquema de banco de dados utilizado.

O modelo tem como objetivo estruturar os objetos utilizados pela lógica de negócios contida no controle.

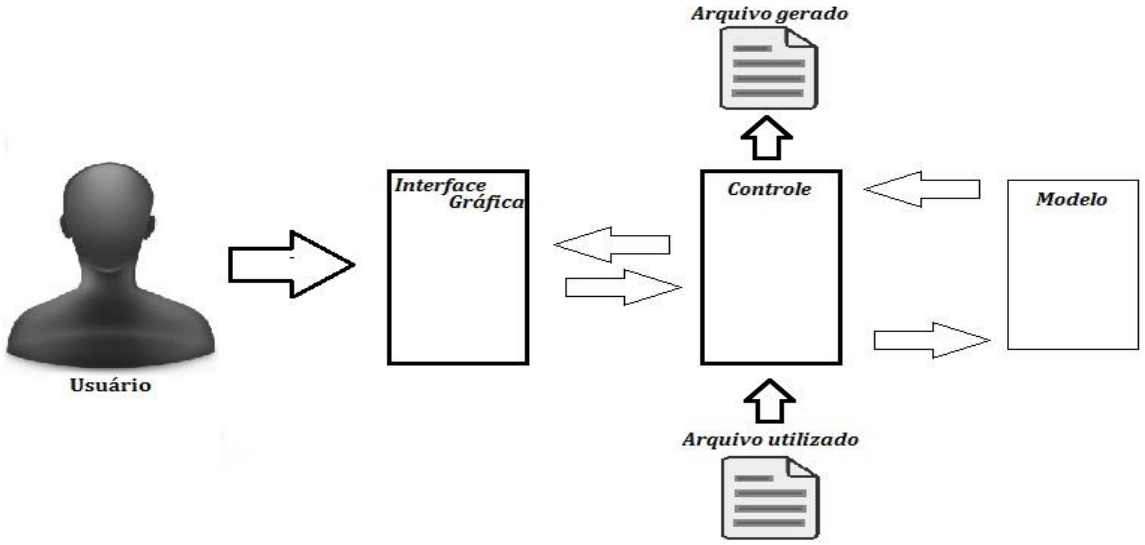

Figura 12 – Arquitetura proposta

A interface gráfica fornece o ambiente de integração com o usuário e fornece as informações para o controle.

O controle é responsável pela análise desta informação, geração e utilização de arquivos externos que representam respectivamente o relatório gerado e arquivo de esquema de banco de dados utilizado.

O modelo tem como objetivo estruturar os objetos utilizados pela lógica de negócios contida no controle.

A aplicação segue a sequência apresentada na Figura 13, que ilustra a proposta geral da aplicação.

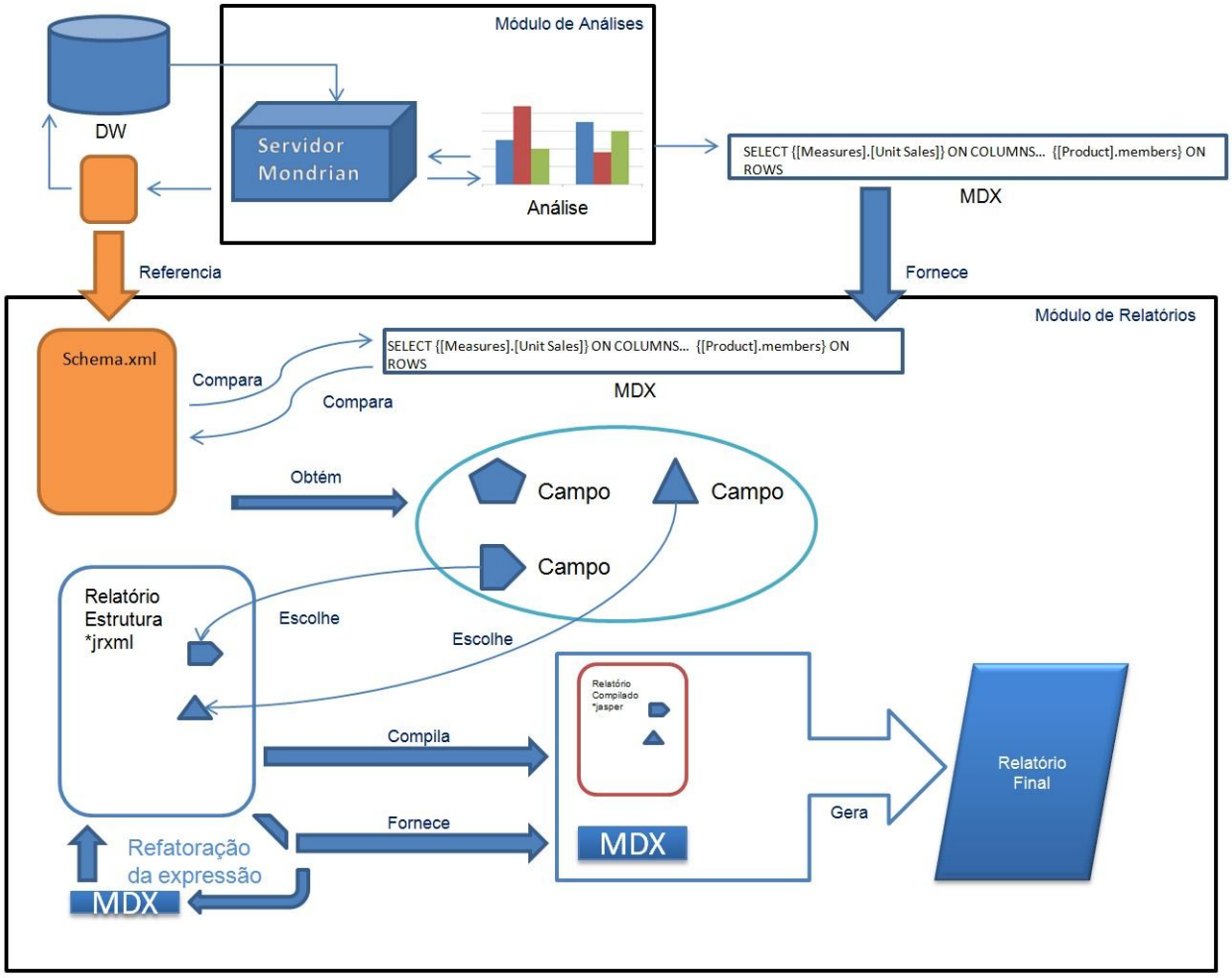

Figura 13 – Proposta geral da aplicação

O módulo de análises envia o MDX que ao ser recebido pelo módulo de relatórios é analisado, divido em expressões menores e comparado ao arquivo de esquema de banco de dados a fim de se obter os campos fornecidos como opção ao usuário.

Após o processo de escolha destes campos, a aplicação gera um arquivo de design de relatório contendo as *tags* referentes aos componentes do relatório, tais como titulo, ata, imagens, dentre outros, juntamente com os campos escolhidos, que por fim são compilados e exportados gerando um relatório final estruturado.

4 DESENVOLVIMENTO

4.1 MÉTODO DE DESENVOLVIMENTO

Este trabalho apresenta-se como pesquisa exploratória, sendo elaborada uma revisão bibliográfica e um levantamento dos componentes pertinente aos principais assuntos que envolvem a aplicação, com o objetivo de se obter conhecimento sobre os conceitos: BI, visão multidimensional, OLAP, *frameworks* para geração de relatórios, *frameworks* para analises multidimensionais e *frameworks* de interface gráfica.

Os métodos se dividem nas seguintes etapas:

- Revisão bibliográfica sobre os principais conceitos envolvidos;
- Definição de um formato para utilização da informação fornecida pela ferramenta de análises;
- Levantamento e análise de *frameworks* disponíveis e de código aberto.
- Integração dos frameworks obtidos: iReport e Mondrian para a criação da lógica de lógica de negócios, provendo ao usuário meio para a criação de relatórios a partir das analises realizada na ferramenta Brain-OLAP;
- Utilização do framework Adobe Flex para criação da interface gráfica;
- Exibir, Armazenar e gerar os relatórios nos formatos, *.pdf.

A especificação, documentação e diagramas apresentados á seguir da seguem os padrões e notações da *Unified Modeling Language* (UML) juntamente com a abordagem utilizada pelo *Rational Unified Process* (RUP) com o objetivo de proporcionar maior visualização lógica do desenvolvimento em geral.

4.2 MODELO DE VISÃO

A aplicação consiste em gerar relatórios á partir de análises OLAP feitas pela ferramenta *Brain*. Essa ferramenta utiliza de recursos como o servidor Mondrian, componente JPivot e interface em Flex integrado com a linguagem Java. A ferramenta fornece o MDX gerado pela análise e á partir da comparação do mesmo com o arquivo referente ao esquema da base de dados utilizada a aplicação define quais informação poderão ser utilizadas na criação do novo relatório.

4.2.1 DESCRIÇÃO DO PROBLEMA

A necessidade de se estruturar, documentar e armazenar o conteúdo provido de cada análise e uma interface intuitiva que auxilie o usuário na elaboração desses relatórios são os principais objetivos da aplicação. No Quadro 1 é exibida uma descrição geral do problema, os principais envolvidos e o impacto gerado pela situação problema e a solução proposta.

Quadro 1 – Visão do problema

O problema	Criar e armazenar relatórios provenientes das análises
Os afetados	Afeta o usuário final da aplicação BRAIN
O impacto	Não existe uma forma de se ter acesso ao conteúdo das análises já realizadas e isso afeta diretamente o escopo da aplicação base.
A solução	Criação de módulo externo, que ira trabalhar como um componente do sistema, esse módulo devera suprir as necessidades de criação e armazenamento desses relatórios.

4.2.1.1 DESCRIÇÃO GERAL DOS SISTEMAS ENVOLVIDOS

A ferramenta *Brain* propõe um Sistema de Gestão, apresentando uma ferramenta de BI que tem como objetivo auxiliar as empresas no processo de tomada de decisão. A ferramenta provê um ambiente *OLAP* que pode ser implantado e adaptado por

qualquer corporação. No Quadro 2 é feita uma breve descrição da ferramenta em conjunto com sua principal responsabilidade.

Quadro 2 – Descrição dos envolvidos

Nome	Descrição	Responsabilidade
Brain	Ferramenta que provê um ambiente de BI	Fornecer o mdx referente às analises realizadas.

4.2.1.2 DESCRIÇÃO DOS USUÁRIOS ENVOLVIDOS

O usuário envolvido é o mesmo da ferramenta *Brain*, a aplicação trabalha na forma de um módulo, utilizando a informação fornecida pelo módulo de análise na forma de um MDX. O usuário responsável por gerar os relatórios pode utilizar analises previamente salvas na ferramenta, excluindo assim a necessidade de se realizar uma nova análise sempre que um relatório for requerido. Abaixo o Quadro 3 representa esses usuários seguidos de um breve descrição e suas respectivas responsabilidades no sistema.

Quadro 3 – Descrição dos usuários

Nome	Descrição	Responsabilidades
Usuário principal. (logado)	Usuário principal da aplicação.	Realizar uma nova análise ou utilizar uma já existente para a geração de relatórios.
Usuário principal. (logado)	Usuário principal da aplicação.	Realizar a seleção dos campos escolhidos

4.3 LEVANTAMENTO DE REQUISITOS

Os requisitos foram obtidos através da analise e levantamento das principais necessidades da aplicação, através de um estudo das funcionalidades fornecidas e recursos utilizados pela ferramenta Brain. A partir desse estudo definiram-se os principais requisitos da aplicação juntamente com os atores envolvidos, ambos descritos nas seções seguintes.

4.3.1 REQUISITOS

Os requisitos obtidos foram:

1. Importar arquivo *schema*.xml;
2. Obter MDX referente á análise;
3. Definir campos que poderão ser exibidos no relatório;
4. Gerar relatório;
5. Visualizar relatório;
6. Exportar relatório para formato *.pdf.

4.3.2 ATORES

Os principais atores identificados foram:

1. Usuário (Brain).

Após a definição dos principais requisitos e atores envolvidos foi possível se definir o formato da informação utilizada e assim criar os diagramas de casos de uso apresentados no próximo item.

4.4 CASOS DE USO

O diagrama apresentado na Figura 14 representa o diagrama de casos de uso, referentes à modelagem da aplicação, são exibidas as principais funcionalidades e processos envolvidos seguidos de suas respectivas descrições.

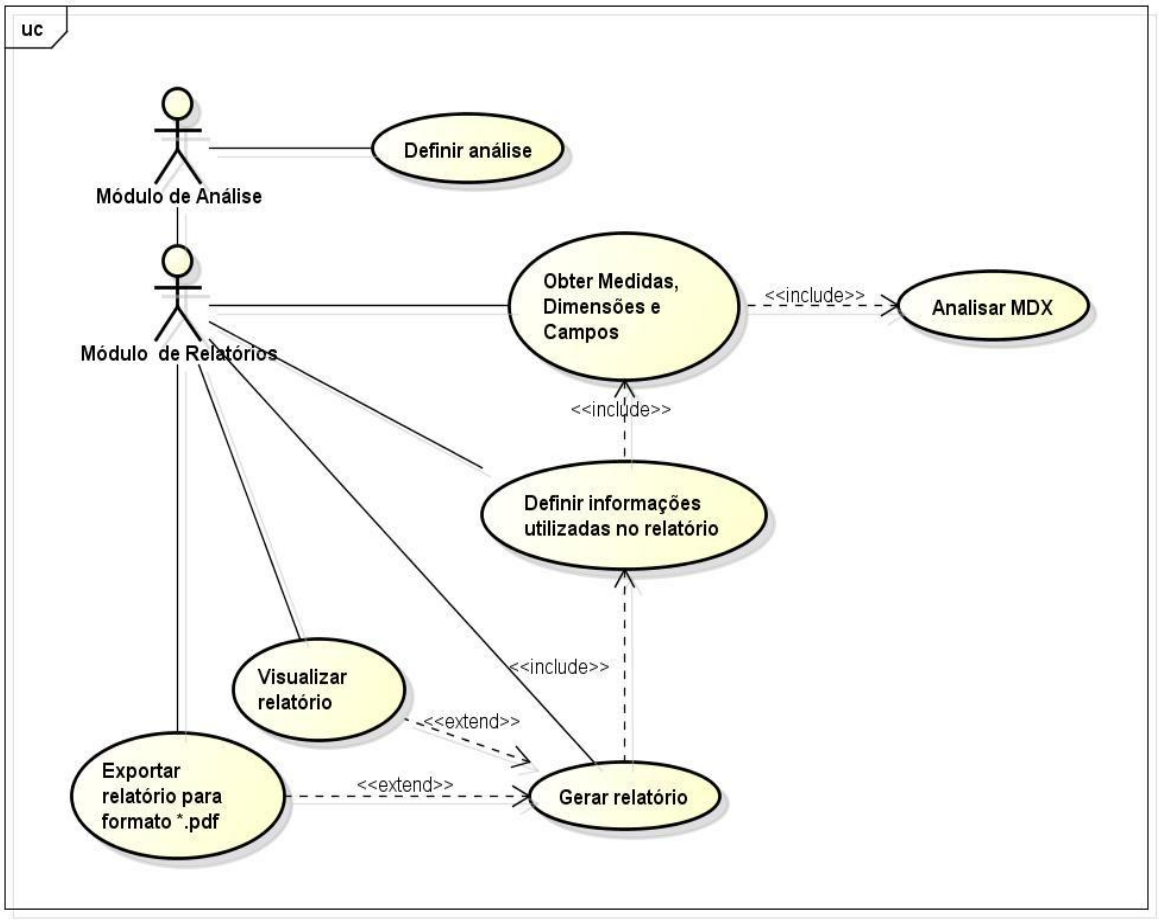

Figura 14 – Diagrama de casos de uso

A descrição do caso de uso apresentado no Quadro 4, exibe a etapa em que o usuário delimita o conjunto de informações que estarão disponíveis para exibição no relatório por meio da análise utilizada.

Quadro 4 – Caso de uso: Definir Análise

Caso de uso: Definir Análise
Ator: Usuário Brain
Pré-requisito:
Fluxo Principal: 1. Usuário realiza uma nova análise ou utiliza alguma previamente definida.

A descrição do caso de uso apresentado no Quadro 5, exibe a etapa de comparação da expressão proveniente da análise com o arquivo *schema*.xml que retorna para o usuário as medidas e dimensões envolvidas na análise com seus respectivos campos, referenciados no arquivo *schema*.xml pela *tag* <*Level*>*.*

Quadro 5 – Caso de uso: Analisar MDX

Caso de uso: Analisar MDX
Ator: Usuário Brain
Pré-requisito: Definir uma análise e seu respectivo MDX
Fluxo Principal: 1. A aplicação, á partir da comparação do mdx com o arquivo schema.xml, obtém as informações que podem ser utilizadas no relatório.
Fluxo Alternativo: 1. a: Referência de endereço físico do arquivo schema.xml inválida 2. a.1: O sistema exibe a mensagem informando o usuário da falha de referência 3. 1.2: Retorna ao passo 1.

O caso de uso apresentado no Quadro 6, representa o centro da lógica de negócio da aplicação que compara o MDX oriundo da análise com o arquivo *schema*.xml fornecendo ao usuário as opções de atributos que poderão ser inseridos no novo relatório.

Quadro 6 – Caso de uso: Obter medidas, dimensões e campos

Caso de uso: Obter medidas, dimensões e campos
Ator: Usuário Brain
Pré-requisito: Analisar MDX
Fluxo Principal: 1. A aplicação compara o mdx com o arquivo schema.xml e através das referencias contidas na expressão são definidas as medidas, dimensões e campos que poderão ser

O caso de uso apresentado no Quadro 7, indica a função na qual o usuário define os campos que irão preencher o novo relatório.

Quadro 7 – Caso de uso: Definir informações utilizadas no relatório

Caso de uso: Definir informações utilizadas no relatório
Ator: Usuário Brain
Pré-requisito: Obter as medidas, dimensões e campos contidos no MDX
Fluxo Principal: 1. Usuário escolhe quais campos serão inseridos no relatório.
Fluxo Alternativo: 1. a 1: Campos esperados pelo usuário não aparecem para escolha 2. a 2: Usuário deve realizar uma nova análise 3. a 3: Retorna ao caso de uso Definir Análise.

O caso de uso apresentado no Quadro 8, ilustra o método retorno do método *JasperManager.compileReport(),* onde o arquivo de estrutura, gerado pela aplicação na forma de um *.jrxml representado pelo objeto instanciado da classe *JasperDesign* do framework *iReport é compilado gerando um arquivo de extensão *.jasper*.

Quadro 8 – Caso de uso: Gerar relatório

Caso de uso: Gerar relatório
Ator: Usuário Brain
Pré-requisito: Estruturar e compilar arquivo de design
Fluxo Principal: 1. Arquivo de relatório, (*.jasper), é gerado á partir do retorno da função que recebe como parâmetro o arquivo de estrutura (*.jrxml);

Os casos de uso apresentados respectivamente no Quadro 9 e Quadro 10, possuem uma semelhança, ambos utilizam o objeto gerado pela classe *JasperPrint*, utilizado na visualização do relatório pelo método *jrviewer*.*show*() e na exportação para o arquivo de extensão *.pdf pelo método *JasperManager*.*printReportToPdfStream*();

Quadro 9 – Caso de uso: Exportar relatório para formato *.pdf

Caso de uso: Exportar relatório para formato *.pdf
Ator: Usuário Brain
Pré-requisito: Gerar relatório
Fluxo Principal: 1. Arquivo de extensão (*.pdf) é criado atravez do retorno da função que recebe com parâmetro

Quadro 10 – Caso de uso: Visualizar relatório

Caso de uso: Visualizar relatório
Ator: Usuário Brain
Pré-requisito: Gerar Relatório

4.5 DIAGRAMA DE COMPONENTES

O diagrama apresentado na Figura 15 representa o diagrama de componentes, que apresenta os principais envolvidos no projeto, a organização dos componentes sendo eles internos ou externos ao sistema.

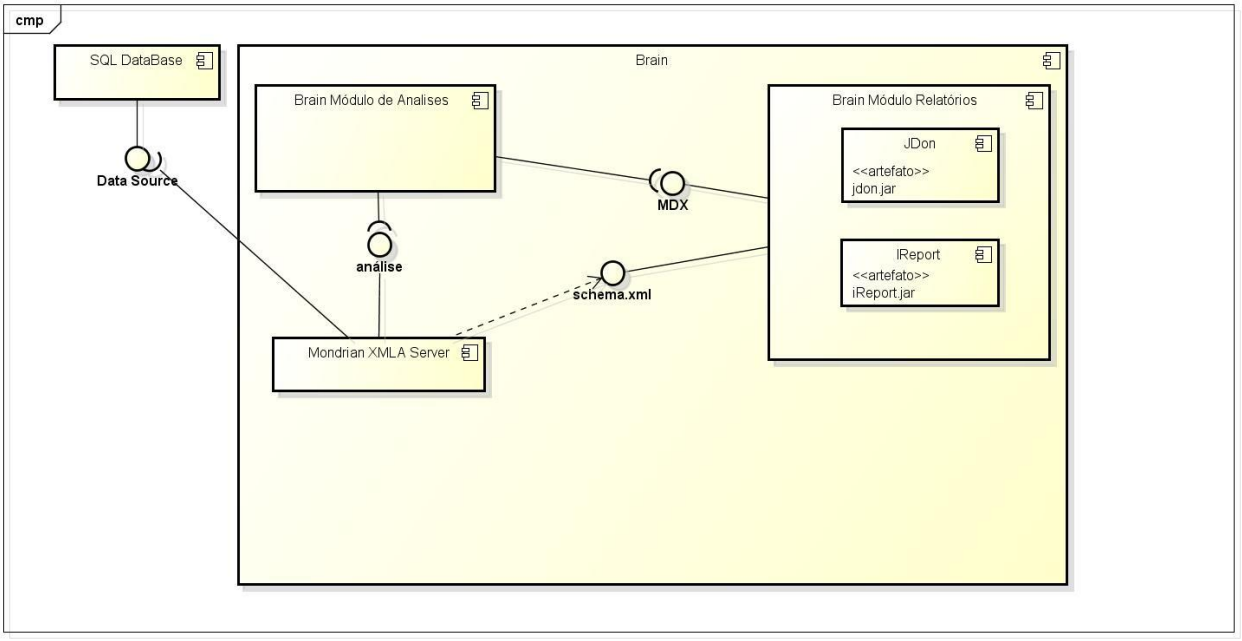

Figura 15 – Diagrama de componentes

O diagrama ilustra como os componentes trabalham dentro do processo executado na aplicação. O módulo de relatório recebe do módulo de analise a interface representada pela expressão MDX juntamente com o arquivo schema.xml provido pelo servidor xmla, nota-se que este se encontra em uma notação diferente por não se tratar de uma interface e sim de uma referência de arquivo externo. O módulo de relatórios se apresenta com seus *framewokrs* ilustrados na forma de bibliotecas com seus respectivos arquivos utilizados, representados pela notação <<*artefato*>>.

4.6 DIAGRAMA DE ATIVIDADES

Neste item é apresentado o diagrama de atividades, que representa um gráfico que ilustra o fluxo de controle de uma atividade para a outra, essas atividades são divididas em regiões, representadas a baixo no diagrama ilustrado na Figura 16 em conjunto com suas atividades.

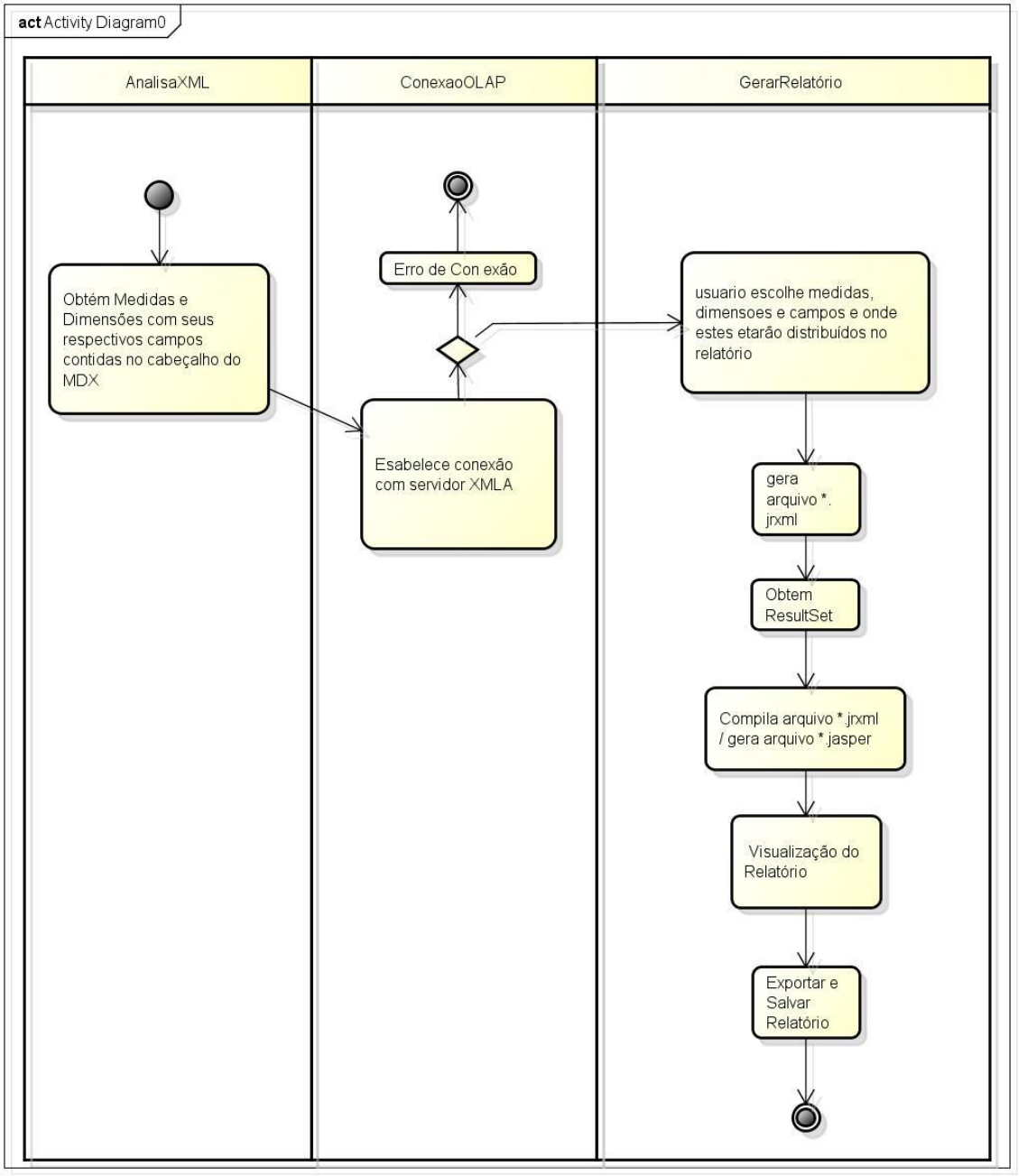

Figura 16 – Diagrama de atividades

As regiões se dividem em:

- AnalisaXML: Região que representa o inicio das atividades, a etapa em que a aplicação compara o MDX e obtém o conjunto de informação que estará disponível para o usuário estruturar o novo relatório e a passagem para a próxima região de atividades.
- Conexão OLAP: Região em que se estabelece uma conexão com o servidor xmla responsável pelo fornecimento dos dados que preenche o novo relatório, um erro nessa conexão compromete o funcionamento da aplicação e a sequência das atividades.
- GerarRelatório: Região principal da aplicação, a partir da definição das informações escolhidos pelo usuário, a aplicação constrói um novo arquivo *.jrxml,sendo possível a adição de componentes nesse novo arquivo na forma de *tags,* estruturadas seguindo o padrão estabelecido pelo *framework* iReport, tornando assim, possível a geração do arquivo *.jasper e posteriormente a visualização e armazenamento do relatório gerado.

4.7 DIAGRAMA DE PACOTES

O objetivo deste item é descrever os pacotes do sistema, mostrando as dependências entre estes, ou seja, pacotes podem depender de outros pacotes. Este diagrama é muito utilizado para ilustrar a arquitetura de um sistema e posteriormente exibir o agrupamento de suas classes.

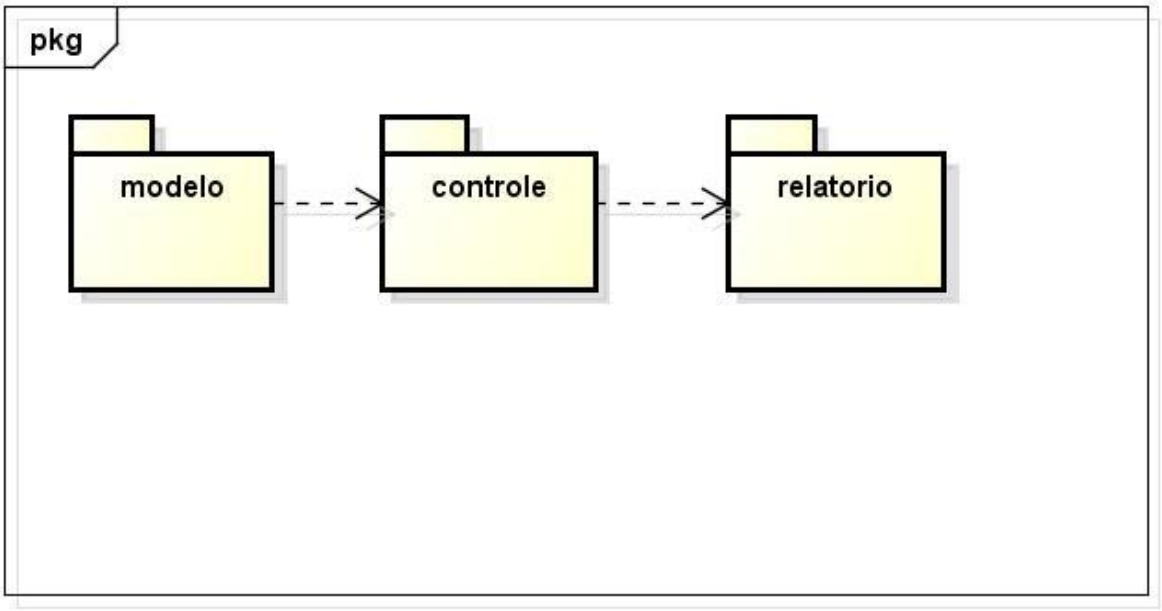

Figura 17 – Diagrama de pacotes

O diagrama ilustrado na Figura 17 representa os pacotes utilizados no processo de implementação da aplicação, são divididos em modelo, controle e relatório. O pacote modelo contém as classes responsáveis pela modelagem dos objetos utilizados pela lógica de negócio, contida no pacote controle. O pacote relatório possui classes de conexão e utilização do framework *IReport* especificamente.

4.8 DIAGRAMA DE CLASSES

O item se destina a especificar as classes contidas nos pacotes apresentados anteriormente, suas principais funções e relacionamentos.

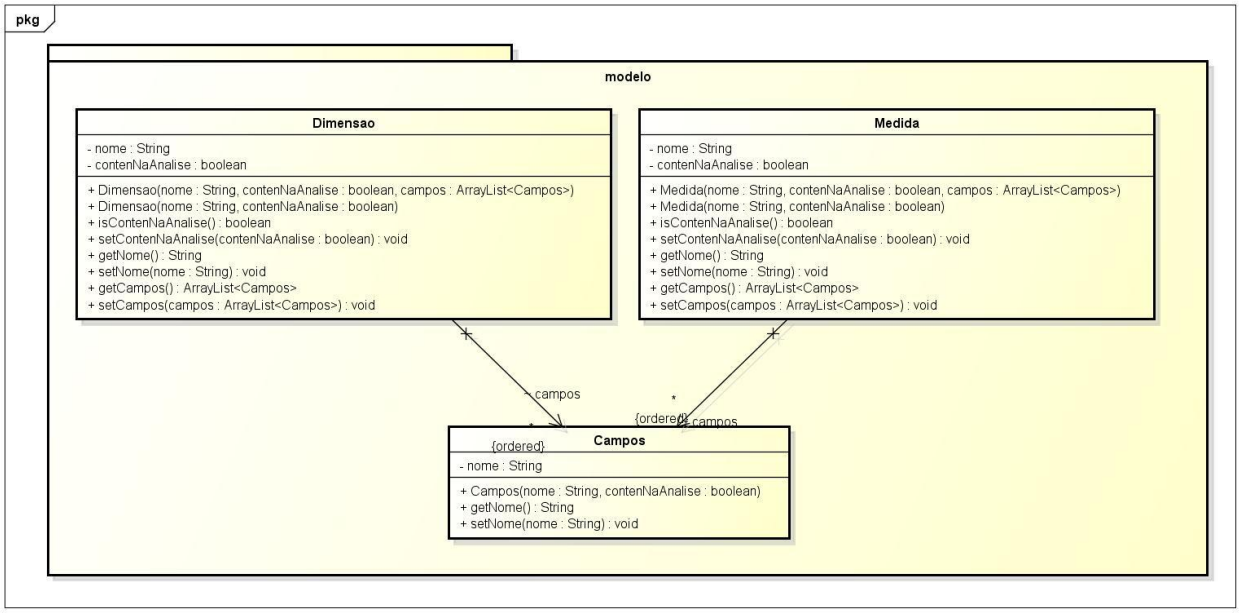

Figura 18 – Diagrama de classes: Pacote modelo

O pacote modelo possui as classes que compõem os objetos que serão utilizados na aplicação como resultado da comparação da expressão proveniente da análise com o arquivo de esquema da base de dados, ou seja, a partir de segmentos do mdx a aplicação detecta as dimensões e medidas utilizadas e as armazena em objetos geradas pela classes do diagrama ilustradas na Figura 18.

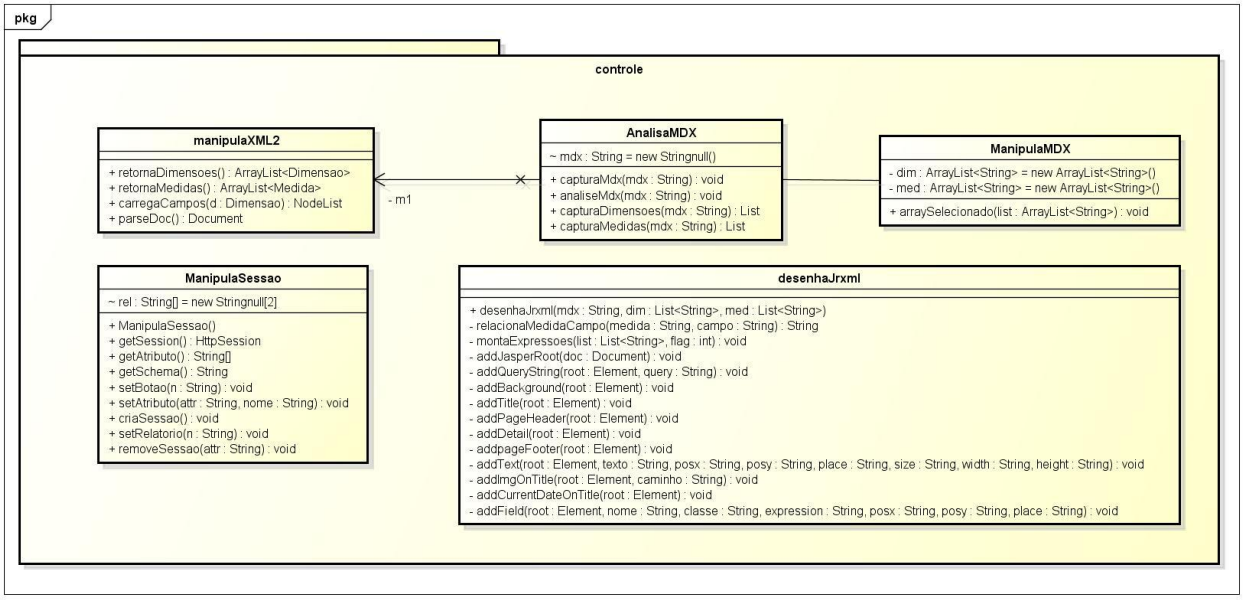

Figura 19 – Diagrama de Classes: Pacote controle

O pacote controle possui as classes ilustradas pelo diagrama apresentado na Figura 19, que compõem a lógica de negócios para obtenção das informações provenientes da analise e a modelagem da estrutura do relatório.

A classe manipulaXML2 é responsável leitura do arquivo esquema que envia as informações para a classe AnalisaMDX responsável pela comparação com a expressão MDX e criação dos objetos através da utilização das classes contidas no pacote modelo.

A classe ManipulaMDX é responsável pela criação dos arrays contendo os campos a serem oferecidos como opção para o usuário e reestruturação do MDX de acordo com a decisão do mesmo

A classe desenhaJrxml é responsável pela criação do arquivo de design do relatório, esse novo arquivo possui extensão *.jrxml. Os métodos criam as *tags* que preenchem o novo arquivo.

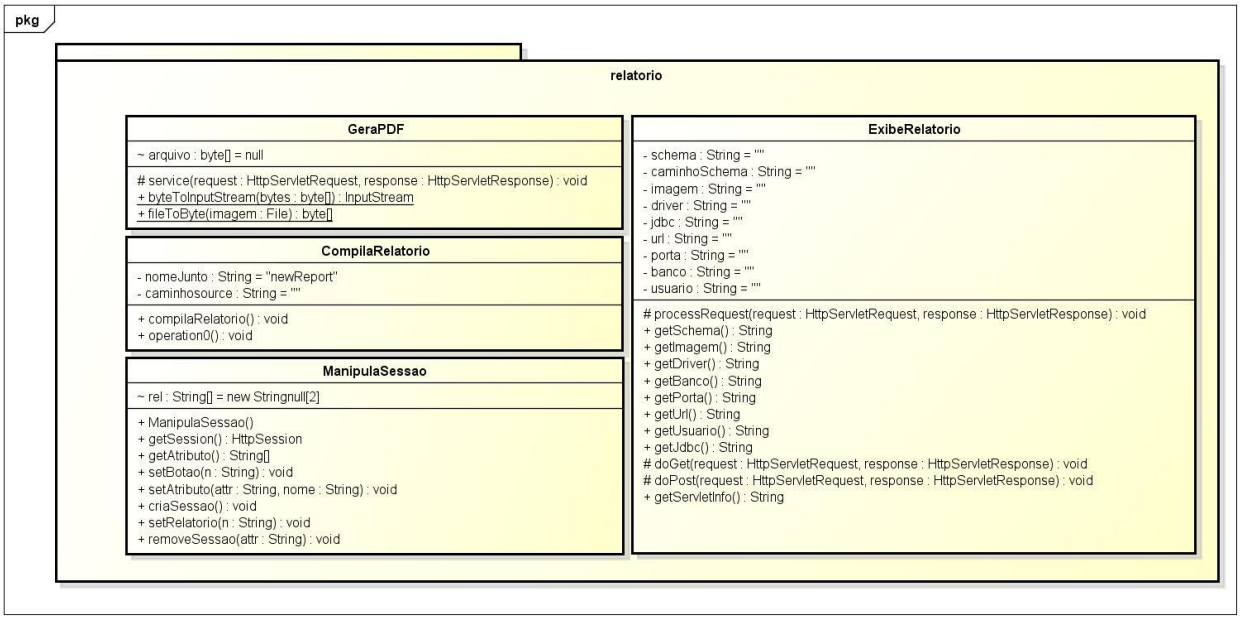

Figura 20 – Diagrama de Classes: Pacote relatório

O pacote relatório possui as classes que compõem as funções de geração do arquivo de relatório (*.jasper), visualização e exportação para o formato (*.pdf) do mesmo. O pacote tem função principal de agregar o framework iReport a aplicação e divide as funcionalidades nas classes apresentadas no diagrama da Figura 20.

5 RESULTADOS

A utilização de dos *frameworks* iReport, Mondrian e Flex geraram uma serie de situações adversas no processo de implementação da aplicação, o estudo destes *frameworks* aliado a definição dos conceitos chave foram fatores decisivos no sucesso do projeto.

A base de dados utilizada foi a FoodMart, base esta utilizada pelo servidor Mondrian como base de teste em formato multidimensional. Representa dados provinetes de uma loja de alimentos e é utilizada juntamente com um arquivo de esquema de banco de dados de extensão *.xml.

O problema inicial envolvendo a conexão com o servidor XMLA Mondrian com o framework iReport se mostrou uma duvida freqüente nos fóruns de tecnologia, com soluçoes limitadas que não resolviam totalmente o problema. Isso porque a base de dados deve ser criada no servidor da aplicação para que o iReport possa obter os dados que irão compor o novo relatório. A solução se deu com a troca do arquivo de esquema que possuía *tags* que o framework não identifica, o que tornava o arquivo inválido.

A aplicação então passa pelo seu primeiro processo, a comparação do MDX com o arquivo de esquema de banco de dados, identificando através das *tags* <*Measures*/> e <*Dimension*/> contidas no arquivo, com os segmentos da expressão condizentes para então se gerar objetos referenciando essa nova medida ou dimensão, se no caso de uma dimensão, a aplicação gera uma lista atribuída ao objeto, contendo seus atributos que serão possíveis campos do relatório.

A interface ilustrada na Figura 21 então cria componentes *checkbox* para proporcionar a escolha desses campos, para ser então possível a criação do arquivo de estrutura do relatório e a remontagem da expressão MDX de acordo com a escolha do usuário.

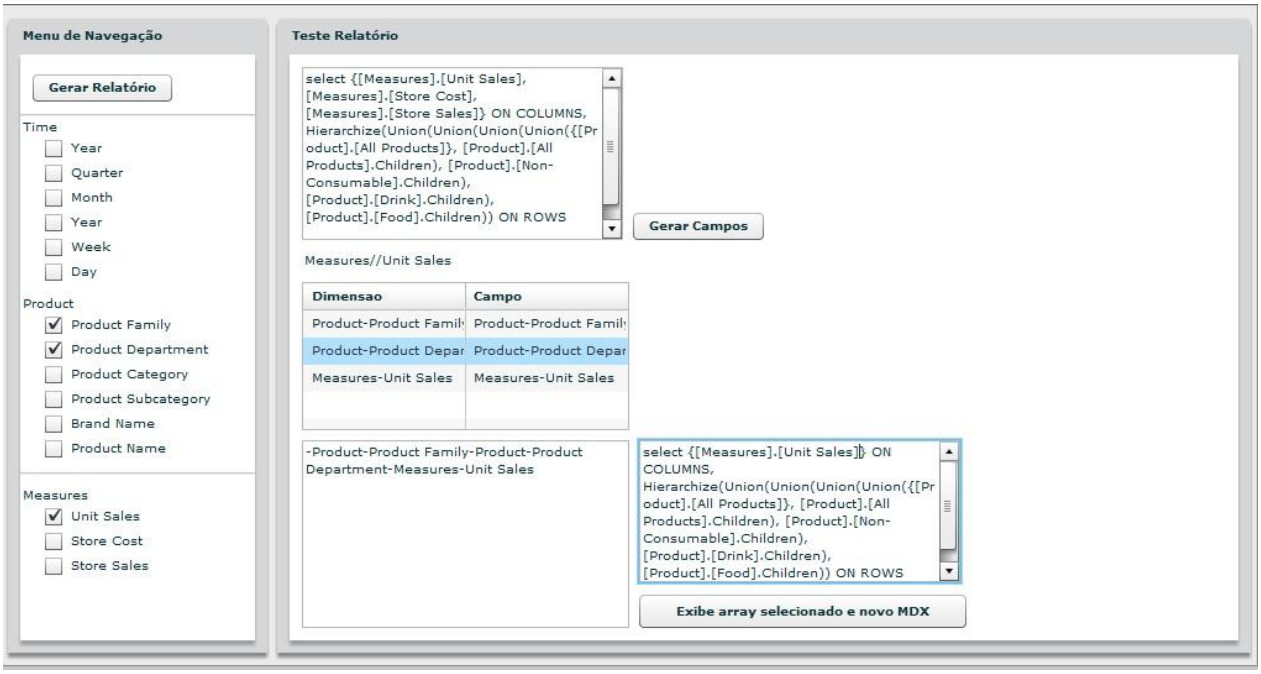

Figura 21 – Interface gráfica em Flex

O usuário gera os campos de acordo com a escolha do *checkbok*, a partir daí a aplicação gera as *tags* e cria o novo arquivo de estrutura de relatório para que então este possa ser visualizado, compilado, exibido e armazenado.

A aplicação não oferece ao usuário a opção de alocação dos campos, isso é executado pela lógica de negócios contida no pacote de controle especificamente na classe desenhaJrxml.class.

Os métodos para criação das *tags* que compõem o arquivo jrxml são o núcleo da aplicação, implementados de forma dinâmica proporcionam um padrão que favorece a usabilidade e o reuso do código em trabalhos futuros. O módulo se integra a outro sistema na forma de um componente, ou seja, fornece classes, métodos e objetos que auxiliam a elaboração de relatórios para sistemas em um ambiente multidimensional.

Foi então criado um relatório teste para verificação e validação dos métodos utilizados, os campos escolhidos foram *Product* F*amily* e *Product Departament* oriundos da Dimensão *Product* com a medida referente *Unit Sales,* e o resultado foi à geração do relatório de teste exibido na Figura 22*.*

Figura 22 – Relatório teste, pag. 1 de 5

6 CONCLUSÃO

A aplicação foi desenvolvida na forma de um módulo ou seja fornece meios para que outra aplicações utilizem seus métodos e objetos de forma a auxiliar a estruturação e geração de relatórios direcionados a modelagem multidimensional de banco de dados. Os relatórios são gerados totalmente baseados em expressões multidimensionais MDX, ou seja, as expressões criadas nas *tags* que compõem o arquivo de estrutura de relatório seguem o padrão imposto pelo *framework* iReport que segue as seguintes regras:

- Para campos relacionados a atributos contidos na seção *ROWS* do cabeçalho do MDX, as expressões seguem o padrão Rows[Dimension][Atributo];
- Para campos relacionado a atributos na seção COLUMNS do cabeçalho MDX, as expressões seguem o padrão Data([Measure][Medida],?);
- Para campos que relacionam uma medida a uma dimensão as expressões se mesclam formando o padrão Data(Rows[Dimension][Atributo] [Measure][Medida],?).

A integração dos *frameworks* ocorre primeiramente pela utilização de APIs como DOM e SAX para fornecer acesso programático ao conteúdo e estrutura do aqruivo *.xml referente ao esquema da base de dados multidimensional, tornando possivel a comparação com o MDX proveniente da análise. A comparação com esse MDX gera opções de dados que poderão ser inseridos no novo relatório e são estruturados pela classe desenhaJrxml.class, tambem responsavel pela adição dos componentes e desenho do arquivo de estrutura do relatório *.jrxml.

Esse arquivo então é compilado pelo *framework* iReport gerando um arquivo de relatório *.jasper, que em conjunto com o resultado obtido pela consulta no servidor Mondrian, utilizando-se da expressão MDX refatorada de acordo com as escolhas do usuario, gera um relatório final e propcia a exportação para formatos de armazenamento e impressão.

A interface criada serve de intermédio para a lógica de negócios, fornecendo as opções de campo e mostrando ao usuário suas escolhas e posteriormente o resultado obtido. Não supre totalmente a criação das *tags*, uma vez que as mesmas necessitam de atributos de localização como coordenadas de eixo x e y referente ao corpo do relatório. Para isso se faz necessária a criação de uma interface mais completa, com maior número de componentes de forma a suprir tal necessidade.

Os métodos contidos na aplicação, fornecidos pelo framework iReport compilam arquivos de estrutura validos, que em conjunto com o objeto *ResultSet*, obtido através da consulta realizada no servidor XMLA com a utilização da expressão multidimensional fornecida pela aplicação, geram arquivos de extensão *.pdf que tornam possíveis a visualização e armazenamento destes arquivos.

6.1 TRABALHOS FUTUROS

A criação de uma *Rich Internet Application* (RIA) com maior número de funcionalidades é uma interessante proposta para trabalhos futuros, uma vez que os métodos implementados já fornecem uma base para a criação e inserção dos componentes primordiais para a elaboração, geração e visualização de relatórios voltados para banco de dados multidimensionais. Ainda há componentes do iReport que não podem ser criados na aplicação, tais como:

- *List;*
- *Crosstab;*
- *Generic Element;*
- *Barcode.*

A utilização de um componente, responsável pela obtenção das coordenadas de posicionamento e tamanho dos itens inseridos no relatório, também se faz necessário, para isso deve ser feita uma pesquisa mais aprofundada dos componentes suportados pelo framework de criação de interface gráfica, neste caso o Adobe Flex.

Esses novos componentes serão responsáveis por forncer atributos de coordenadas e localização para as *tags* que compõem o arquivo de estrutura jrxml, possam ser distribuídos pelo relatório de acordo com a necessidade do usuário. Podem se basear nos métodos já implementados neste projeto, uma vez que seguem os mesmos padrões e são interpretados pelo mesmo framework.

REFERÊNCIAS

REZENDE, Solange Oliveira. Mineração de Dados. Mini-curso V ENIA. São Leopoldo: Unisinos, 2005.

BARBIERI, Carlos. Business Intelligence: modelagem e tecnologia. Rio de Janeiro: Axcel Books, 2001.

TURBAN; EFRAIM. Business Intelligence: um enfoque gerencial para a inteligência do negócio. Porto Alegre: Bookman, 2009.

SERRA, Laércio. A essência do Business Intelligence. São Paulo: Berkeley Brasil, 2002.

KIMBAL, Ralph. The Data Warehouse ToolKit:Guia Completo para modelagem dimensional. 2. ed. Rio de Janeiro: Campus Ltda., 2002.

GONÇALVES, Edson. Dominando Relatórios *JasperReports* com iReport. Rio de Janeiro: Editora Ciência Moderna Ltda., 2008.

INMON, W.H, WELCH, J. D., GLASSEY, K. L. Gerenciando Data Warehouse. São Paulo: Makron Books, 1999.

Pentaho. Mondrian. Disponível em: <http://mondrian.pentaho.org/documentation/>. Acesso em: 08 de julho 2011.

JasperForge. **IReport.** Disponível em: <http://jasperforge.org/projects/ireport./>. Acesso em: 03 de agosto 2011.

Adobe. **Flex.** Disponível em: <http://www.adobe.com/br/products/flex/ />. Acesso em: 07 de agosto 2011.

SourceForge**.** Disponível em: <http:// sourceforge.net / />. Acesso em: 15 de setembro 2011..

Apêndice A – Código fonte da interface gráfica da aplicação.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application creationComplete="{}"
xmlns:mx="http://www.adobe.com/2006/mxml" xmlns:maps="com.google.maps.*"
layout="absolute" backgroundGradientAlphas="[1.0, 1.0]"
backgroundGradientColors="[#C8C8C8, #C8C8C8]">



	<mx:RemoteObject id="AnalisaMDX" destination="AnalisaMDX"
showBusyCursor="true">
		<mx:method name="analiseMdx"/>
		<mx:method name="capturaMdx"/>
		<mx:method name="capturaDimensoes"
result="resultado_listagem_dimensoes(event)"/>
		<mx:method name="capturaMedidas"
result="resultado_listagem_medidas(event)"/>
	</mx:RemoteObject>

	<mx:RemoteObject id="ManipulaMDX" destination="ManipulaMDX"
showBusyCursor="true">
		<mx:method name="arraySelecionado"/>
		<mx:method name="remontaMDX" result="resultado_MDX(event)"/>
	</mx:RemoteObject>

	<mx:Canvas id="cvGeral" height="535" width="1020" y="0"
horizontalCenter="0" backgroundColor="#EBEBEB" backgroundAlpha="1.0"
borderStyle="solid" borderColor="#8D8D8D" borderSides="left, right, bottom"
borderThickness="1">
		<mx:Panel x="7" y="10" width="210" height="514"
borderColor="#B7BABC" layout="absolute" id="pnMenu" title="Menu de Navegação"
cornerRadius="4" backgroundAlpha="1.0">
			<mx:HRule x="0" y="48" width="190"/>
			<mx:Canvas x="0" y="0" width="190" height="48"
id="cvControle">
				<mx:Button x="10" y="16" label="Gerar Relatório"
id="btnGeraRelatorio" click="gerarRelatorio();"/>
			</mx:Canvas>


		</mx:Panel>

		<mx:Panel x="225" y="10" width="788" height="514"
layout="absolute" borderColor="#B7BABC" id="pnRelatorio" title="Teste
Relatório" backgroundAlpha="1.0">
			<mx:Button x="278" y="128" label="Gerar Campos"
id="btnMdx" click="{changeMdx();}"/>

			<mx:DataGrid x="10" y="184"  id="dg_dimensao"
dataProvider="{array}" width="265" height="120">
				<mx:columns>
					<mx:DataGridColumn headerText="Dimensao"
dataField="dim_nome" />
```

```
                                        <mx:DataGridColumn headerText="Campo"
dataField="dim_campo"/>
                                </mx:columns>
                        </mx:DataGrid>
                        <mx:Label x="10" y="158" text="Campos Adicionados"
id="lbTeste" visible="true"/>
                        <mx:Button x="283" y="437" label="Exibe array selecionado
e novo MDX"  click="{confereArray();exibeMDX();}" width="265" height="27"/>

                        <mx:TextArea x="10" y="312" id="inputArray" width="265"
height="152"/>
                        <mx:TextArea  x="10" y="10" width="260" height="140"
id="campoMdx" text="Digite o MDX"/>
                        <mx:TextArea x="283" y="312" id="novoMDXText" width="260"
height="115" text="novo MDX"/>
                </mx:Panel>
        </mx:Canvas>

        <mx:Script>
                <![CDATA[
                        import mx.messaging.channels.AMFChannel;
                        import mx.controls.VRule;
                        import mx.core.LayoutContainer;
                        import mx.controls.Label;
                        import mx.controls.CheckBox;
                        import mx.containers.*;
                        import mx.collections.ArrayCollection;
                        import mx.rpc.events.ResultEvent;
                        import mx.controls.Alert;

                        public function changeMdx():void{
                                query=campoMdx.text;
                                chamar_listagem_dimensoes();
                                chamar_listagem_medidas();

                                checaAssicrono();
                        }

                    [Bindable]
                private var dados_dimensoes:ArrayCollection;

                [Bindable]
                private var dados_medidas:ArrayCollection;
                [Bindable]
                private var query:String;
                [Bindable]
                private var novoMDX:String;
                [Bindable]
                private var dados_selecionados:ArrayCollection;


                        public function chamar_listagem_dimensoes():void
                {
                        AnalisaMDX.capturaDimensoes(query);
                }
                public function chamar_listagem_medidas():void
                {
```

```
                AnalisaMDX.capturaMedidas(query);
            }

                private var   resultadoChego:Boolean=false;
                private var   dimensaoChego:Boolean=false;
                private var   medidaChego:Boolean=false;

                public function
resultado_listagem_dimensoes(event:ResultEvent):void
            {
                dados_dimensoes = event.result as ArrayCollection;
                dimensaoChego=true;
                checaAssicrono();

            }

            public function
resultado_listagem_medidas(event:ResultEvent):void
            {
                dados_medidas = event.result as ArrayCollection;
                medidaChego=true;
                checaAssicrono();

            }
            public function resultado_MDX(event:ResultEvent):void
            {
                novoMDX = event.result as String;

            }
            public function exibeMDX():void
            {
                novoMDX = AnalisaMDX.remontaMDX(array,query);
                novoMDXText.text =novoMDX;

            }

            public function checaAssicrono():void{

                resultadoChego=medidaChego&&dimensaoChego;
                if(resultadoChego){
                    geraTela();
                }


            }
            public function geraTela():void{
                var layout:LayoutContainer = new LayoutContainer();

                var regua:HRule = new HRule();
                regua.width = 190;
                regua.x = 0;

                var canvas:Canvas = new Canvas();
                canvas.percentHeight = 100;
                canvas.percentWidth = 100;
                var lbDimensao:Label = new Label();
                var cbCampo: CheckBox = new CheckBox();
```

```
                    var verDimensao:String=
dados_dimensoes[0].dim_nome.toString();
                    var prim:int = 0;
                    var posY:int = 50;



                    for(var i:int = 0; i<dados_dimensoes.length; i++){
                           var lbDimensao:Label = new Label();

                        var cbCampo: CheckBox = new CheckBox();

                        if(dados_dimensoes[i].dim_nome.toString() ==
verDimensao){
                            if(prim==0){
                                   lbDimensao.text =
dados_dimensoes[i].dim_nome.toString();
                                   lbDimensao.y = posY;
                                   posY += 15;
                                          cbCampo.label =
dados_dimensoes[i].dim_campo.toString();
                                          cbCampo.id =
dados_dimensoes[i].dim_nome.toString();

       cbCampo.addEventListener(MouseEvent.CLICK, selecionado);
                                          cbCampo.y = posY;
                                          cbCampo.x = 20;
                                          posY+=20;
                                          canvas.addChild(lbDimensao);

                                          canvas.addChild(cbCampo);
                                   prim=1;
                                   verDimensao =
dados_dimensoes[i].dim_nome.toString();
                                   if(i+1 == dados_dimensoes.length){
                                          layout.addChild(canvas);
                                          posY = 0;
                                   }
                            }
                            else{
                                   cbCampo.label =
dados_dimensoes[i].dim_campo.toString();
                                   cbCampo.id =
dados_dimensoes[i].dim_nome.toString();
                                   cbCampo.addEventListener(MouseEvent.CLICK,
selecionado);
                                   cbCampo.y = posY;
                                   cbCampo.x = 20;
                                          posY+=20;
                                   canvas.addChild(cbCampo);
                                   verDimensao =
dados_dimensoes[i].dim_nome.toString();
                            }
                        }
                        else{
                            layout.addChild(canvas);
                            canvas = new Canvas();
```

```
                                posY = 0;
                                lbDimensao.text =
dados_dimensoes[i].dim_nome.toString();
                                lbDimensao.y = posY;
                                        posY+=15;
                                        cbCampo.label =
dados_dimensoes[i].dim_campo.toString();
                                        cbCampo.id =
dados_dimensoes[i].dim_nome.toString();
                                        cbCampo.addEventListener(MouseEvent.CLICK,
selecionado);
                                        cbCampo.y = posY;
                                        cbCampo.x = 20;
                                        posY+=20;
                                        canvas.addChild(lbDimensao);

                                        canvas.addChild(cbCampo);

                                verDimensao =
dados_dimensoes[i].dim_nome.toString();
                                if(i+1 == dados_dimensoes.length){

                                        layout.addChild(canvas);
                                }
                            }
                        }

                        posY += 10;
                        regua.y = posY;

                        posY += 10;
                        prim = 0;

                        for(var i:int = 0; i<dados_medidas.length; i++){
                                var lbDimensao:Label = new Label();

                            var cbCampo: CheckBox = new CheckBox();
                            if(prim ==0){
                                prim=1;
                                lbDimensao.text =
dados_medidas[i].med_nome.toString();
                                lbDimensao.y = posY;
                                posY += 15;
                                cbCampo.label =
dados_medidas[i].med_campo.toString();
                                cbCampo.id =
dados_medidas[i].med_nome.toString();
                                cbCampo.addEventListener(MouseEvent.CLICK,
selecionado);
                                cbCampo.y = posY;
                                        cbCampo.x = 20;
                                        posY+=20;
                                canvas.addChild(lbDimensao);

                                        canvas.addChild(cbCampo);
                            }
                            else{
```

```
                                cbCampo.label =
dados_medidas[i].med_campo.toString();
                                cbCampo.id =
dados_medidas[i].med_nome.toString();
                                cbCampo.addEventListener(MouseEvent.CLICK,
selecionado);
                                cbCampo.y = posY;
                                        cbCampo.x = 20;
                                        posY+=20;
                                //canvas.addChild(lbDimensao);

                                        canvas.addChild(cbCampo);
                            }

                        }
                        canvas.addChild(regua);
                        layout.addChild(canvas);

                        pnMenu.addChild(layout);
                        }


                [Bindable]
                var array:ArrayCollection = new ArrayCollection();

                public function selecionado(eventObj:MouseEvent){

                        var parc : CheckBox  = new CheckBox();
                        parc = eventObj.target.valueOf();
                        var txt:String = new String(parc.id+"-"+parc.label);
                        if(parc.selected == false){
                                for(var i:int=0;i<array.length;i++){
                                        if(array[i] == txt){
                                                array.removeItemAt(i);
                                                break;
                                        }
                                }
                        }
                        else{
                                lbTeste.text = parc.id.toString()+"//"+parc.label;
                                array.addItem(txt);
                        }
                }

                public function gerarRelatorio():void{

                        ManipulaMDX.arraySelecionado(array, novoMDX);
                }

                public function confereArray():void{

                        inputArray.text = "";
                        for(var i:int = 0; i<array.length ;i++){

                                inputArray.text = inputArray.text+ "-" +array[i];
                        }
                }
```

```
            ]]>
        </mx:Script>

        <mx:Style>

        </mx:Style>

</mx:Application>
```

Apêndice B – Código fonte da lógica de negócios da aplicação.

```
package controle;

import modelo.Dimensao;
import modelo.Medida;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 *
 * @author Alex
 */
public class AnalisaMDX {


    private manipulaXML2 m1 = new manipulaXML2();
    Dimensao d;
    private ArrayList<Dimensao> dimensoes = new ArrayList<Dimensao>();
    Medida m;
    private ArrayList<Medida> medidas = new ArrayList<Medida>();
    String mdx = new String();

    public void capturaMdx(String mdx){
       this.mdx = mdx;
    }


    public void analiseMdx(String mdx){
        dimensoes = m1.retornaDimensoes();
        medidas=m1.retornaMedidas();

        System.out.println("Dimensoes...:");
        for(int i=0; i<dimensoes.size();i++){
            if(mdx.contains("["+dimensoes.get(i).getNome()+"]")){
                d=dimensoes.get(i);
                System.out.println("\n"+d.getNome());
                for(int j=0; j<d.getCampos().size();j++){
```

```
                    System.out.println("Campo:
"+d.getCampos().get(j).getNome());
                }
            }

        }

        System.out.println("\nMeasures:");
        for(int i=0; i<medidas.size();i++){
            if(mdx.contains(medidas.get(i).getNome())){
                System.out.println(medidas.get(i).getNome());
                medidas.get(i).setContenNaAnalise(true);
            }

        }

    }

    public List capturaDimensoes(String mdx){

        dimensoes = m1.retornaDimensoes();
        List lista_dimensoes = new ArrayList();
        System.out.println("Dimensoes:");
        for(int i=0; i<dimensoes.size();i++){
            if(mdx.contains("["+dimensoes.get(i).getNome()+"]")){
                d=dimensoes.get(i);

                System.out.println("\n"+d.getNome());
                for(int j=0; j<d.getCampos().size();j++){
                     HashMap hashmap_dimensoes = new HashMap();
                    hashmap_dimensoes.put("dim_nome",d.getNome());
                     hashmap_dimensoes.put("dim_campo",
d.getCampos().get(j).getNome());
                    System.out.println("Campo:
"+d.getCampos().get(j).getNome());
                    lista_dimensoes.add(hashmap_dimensoes);
                }
            }
```

```
         }
         return lista_dimensoes;
    }

    public List capturaMedidas(String mdx){
       List lista_medidas = new ArrayList();
       medidas=m1.retornaMedidas();
       System.out.println("\nMedidas:");
        for(int i=0; i<medidas.size();i++){
            if(mdx.contains(medidas.get(i).getNome())){
              HashMap hashmap_medidas = new HashMap();
              hashmap_medidas.put("med_nome", "Measures");
              hashmap_medidas.put("med_campo", medidas.get(i).getNome());
                System.out.println(medidas.get(i).getNome());
                medidas.get(i).setContenNaAnalise(true);
                lista_medidas.add(hashmap_medidas);
            }
         }
       return lista_medidas;
    }
package controle;

import java.io.FileNotFoundException;
import controle.desenhaJrxml;
import java.io.IOException;
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.swing.JOptionPane;

public class ManipulaMDX {
       private ArrayList<String> dim = new ArrayList<String>();
       private ArrayList<String> med =  new ArrayList<String>();
```

```
//o mdx deve ser passado por parametro emviado pelo componente
public void arraySelecionado(ArrayList<String> list, String mdx){
	for(int i = 0; i< list.size(); i++){

		String sp[] = list.get(i).split("-");
		if(sp[0].equals("Measures")){

			med.add(list.get(i));
		}
		else{
			dim.add(list.get(i));
		}
		//System.out.println(arr[i]);
	}

	System.out.println("DIMENSAO");
	for(int i=0;i<dim.size();i++){
		System.out.println(dim.get(i));
	}

	System.out.println("MEDIDA");
	for(int i=0;i<med.size();i++){
		System.out.println(med.get(i));
	}
	try {
		desenhaJrxml jrxml = new desenhaJrxml(mdx,dim,med);
	} catch (FileNotFoundException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
}

public String remontaMDX(ArrayList<String> med, String mdx){
	String[] sufix = mdx.split("ON COLUMNS");
    int count = 0;
```

```
            Matcher matcher = Pattern.compile("\\b" + "Children" +
            "\\b").matcher(mdx);
             while (matcher.find()) {
                 count++;
                 System.out.println(count);
             }

             return montaSelectOnColumns(med)+ sufix[1];
        }

        public String montaSelectOnColumns(ArrayList<String> lst){
               String select = "select {";
               for (int i=0; i<lst.size(); i++){
                      String[] item = lst.get(i).split("-");
                      String it1 = "["+item[0]+"].";
                      String it2;
                      if(i==lst.size()-1)
                              it2 = "["+item[1]+"]";
                      else
                              it2 = "["+item[1]+"],";
                      String it =it1+it2;
                      select=select+it;

               }
               select=select+"} ON COLUMNS, ";
               System.out.println(select);
               return select;
        }
}


package controle;
import modelo.Dimensao;
import modelo.Medida;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

```
/**
 *
 * @author Alex
 */
public class AnalisaMDX {
    private manipulaXML2 m1 = new manipulaXML2();
    Dimensao d;
    private ArrayList<Dimensao> dimensoes = new ArrayList<Dimensao>();
    Medida m;
    private ArrayList<Medida> medidas = new ArrayList<Medida>();
    String mdx = new String();
    public void capturaMdx(String mdx){
       this.mdx = mdx;
    }
    public void analiseMdx(String mdx){
        dimensoes = m1.retornaDimensoes();
        medidas=m1.retornaMedidas();

        System.out.println("Dimensoes...:");
        for(int i=0; i<dimensoes.size();i++){
            if(mdx.contains("["+dimensoes.get(i).getNome()+"]")){
                d=dimensoes.get(i);
                System.out.println("\n"+d.getNome());
                for(int j=0; j<d.getCampos().size();j++){
                    System.out.println("Campo:
"+d.getCampos().get(j).getNome());
                }
            }
        }
        System.out.println("\nMeasures:");
        for(int i=0; i<medidas.size();i++){
            if(mdx.contains(medidas.get(i).getNome())){
                System.out.println(medidas.get(i).getNome());
                medidas.get(i).setContenNaAnalise(true);
            }
        }
    }
    public List capturaDimensoes(String mdx){
```

```
        dimensoes = m1.retornaDimensoes();
        List lista_dimensoes = new ArrayList();
        System.out.println("Dimensoes:");
        for(int i=0; i<dimensoes.size();i++){
            if(mdx.contains("["+dimensoes.get(i).getNome()+"]")){
                d=dimensoes.get(i);
                System.out.println("\n"+d.getNome());
                for(int j=0; j<d.getCampos().size();j++){
                     HashMap hashmap_dimensoes = new HashMap();
                    hashmap_dimensoes.put("dim_nome",d.getNome());
                     hashmap_dimensoes.put("dim_campo",
d.getCampos().get(j).getNome());
                    System.out.println("Campo:
"+d.getCampos().get(j).getNome());
                    lista_dimensoes.add(hashmap_dimensoes);
                }   }  }
        return lista_dimensoes;
    }
    public List capturaMedidas(String mdx){
       List lista_medidas = new ArrayList();
       medidas=m1.retornaMedidas();
       System.out.println("\nMedidas:");
        for(int i=0; i<medidas.size();i++){
            if(mdx.contains(medidas.get(i).getNome())){
              HashMap hashmap_medidas = new HashMap();
              hashmap_medidas.put("med_nome", "Measures");
              hashmap_medidas.put("med_campo", medidas.get(i).getNome());
                System.out.println(medidas.get(i).getNome());
                medidas.get(i).setContenNaAnalise(true);
                lista_medidas.add(hashmap_medidas);
            }
        }     return lista_medidas;
    }
    package controle;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.util.List;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.Namespace;
import org.jdom.output.XMLOutputter;

/**
 *
 * @author Alex
 */
public class desenhaJrxml {

    Namespace nsp =
Namespace.getNamespace("http://jasperreports.sourceforge.net/jasperreports");

    public desenhaJrxml(String mdx, List<String> dim,List<String> med) throws
FileNotFoundException, IOException{
        Document doc = new Document();
        Writer out = new OutputStreamWriter(new
FileOutputStream("C:/workspace/NovoRelatorio/src/relatorioTesteEclipse.jrxml")
);
        montaExpressoes(med, 1);
        montaExpressoes(dim, 2);
        //monta tah especial para ciração de campo onde a Medida é relacionada
com o campo
        String specialField = relacionaMedidaCampo(med.get(0), dim.get(1));

        addJasperRoot(doc);
        addQueryString(doc.getRootElement(),mdx);
        addBackground(doc.getRootElement());
        addTitle(doc.getRootElement());
        addPageHeader(doc.getRootElement());
        addDetail(doc.getRootElement());
        addpageFooter(doc.getRootElement());
        //fazer metodo com replace na String
```

```
        //construindo o Title
        addImgOnTitle(doc.getRootElement(),'"'+ "C:"+'\\'+'\\'+
"Users"+'\\'+'\\'+"Alex"+'\\'+'\\'+"Documents"+'\\'+'\\'+"NetBeansProjects"+'\
\'+'\\'+"Reports"+'\\'+'\\'+"src"+'\\'+'\\'+"schemas"+'\\'+'\\'+"img.jpg"+'"')
;
        addText(doc.getRootElement(), "Relatorio de Departamentos", "155",
"0", "title", "25", "220", "60");
        addCurrentDateOnTitle(doc.getRootElement());
        //pageHeader
        addText(doc.getRootElement(), "Total: ", "181", "0", "pageHeader",
"20", "65","30");
        addText(doc.getRootElement(), "Family: ", "0", "0", "pageHeader",
"20", "65","30");
        addField(doc.getRootElement(), "Rows"+dim.get(1), "java.lang.String",
dim.get(1), "58", "0", "pageHeader");
        addField(doc.getRootElement(), "RowsAndMeasure", "java.lang.String",
specialField, "245", "10", "pageHeader");
        //detail
        addField(doc.getRootElement(), "Columns-"+med.get(0),
"java.lang.Number", med.get(0), "215", "20", "detail");
        addField(doc.getRootElement(), "Rows"+dim.get(0), "java.lang.String",
dim.get(0), "0", "15", "detail");
        addText(doc.getRootElement(), "Total", "165", "20", "detail", "15",
"45","15");

        XMLOutputter xout = new XMLOutputter();

xout.setFormat(org.jdom.output.Format.getCompactFormat().setEncoding("ISO-
8859-1"));
        xout.output(doc, out);

        System.out.println("XML criado com sucesso!");


    }

    private String relacionaMedidaCampo(String medida, String campo){
       String dtM1 = medida.substring(0, 5);
```

```
    String dtM2 = medida.substring(5, medida.length());

    String specialField  = dtM1+campo+dtM2;
    System.out.print(specialField);
    return specialField;
}

private void montaExpressoes(List<String> list,int flag){
   if(flag==1){

   for(int i=0;i<list.size();i++){
        String[] item = list.get(i).split("-");
          String it1 = "["+item[0]+"].";
          String it2 = "["+item[1]+"]";
          String it =it1+it2;
          String exp="Data("+it+",?)";

          list.remove(i);
          list.add(i,exp);
          System.out.println(list.get(i)+ "pos: "+ i);
        }}
   else{
    for(int i=0;i<list.size();i++){
        String[] item = list.get(i).split("-");
          String it1 = "["+item[0]+"].";
          String it2 = "["+item[1]+"]";
          String it =it1+it2;
          String exp="Rows"+it;

          list.remove(i);
          list.add(i, exp);
        }}
}
private void addJasperRoot(Document doc){

    Element jasperReport = new Element("jasperReport",nsp );
```

```
        Namespace nsp2 =
Namespace.getNamespace("xsi","http://www.w3.org/2001/XMLSchema-instance");
        Namespace nsp3 =
Namespace.getNamespace("schemaLocation","http://jasperreports.sourceforge.net/
jasperreports http://jasperreports.sourceforge.net/xsd/jasperreport.xsd");

        jasperReport.setNamespace(nsp);
        jasperReport.addNamespaceDeclaration(nsp2);
        jasperReport.addNamespaceDeclaration(nsp3);
        jasperReport.setAttribute("name", "report name");
        jasperReport.setAttribute("pageWidth", "595");
        jasperReport.setAttribute("pageHeight", "842");
        jasperReport.setAttribute("columnWidth", "535");
        jasperReport.setAttribute("leftMargin", "20");
        jasperReport.setAttribute("rightMargin", "20");
        jasperReport.setAttribute("topMargin", "20");
        jasperReport.setAttribute("bottomMargin", "20");
        doc.setRootElement(jasperReport);


    }

    private void addQueryString (Element root, String query){
        Element queryString = new Element("queryString",nsp );
        queryString.setAttribute("language", "mdx");
        queryString.setText(query);
        root.addContent(queryString);
    }

    private void addBackground (Element root){

        Element background = new Element("background",nsp);
        Element band = new Element("band",nsp);
        band.setAttribute("splitType", "Stretch");
        background.addContent(band);

        root.addContent(background);
    }
```

```
private void addTitle (Element root){

    Element title = new Element("title",nsp);
    Element band = new Element("band",nsp);
    band.setAttribute("height", "79");
    band.setAttribute("splitType", "Stretch");
    title.addContent(band);

    root.addContent(title);
}

private void addPageHeader (Element root){

    Element pageHeader = new Element("pageHeader",nsp);
    Element band = new Element("band",nsp);
    band.setAttribute("height", "35");
    band.setAttribute("splitType", "Stretch");
    pageHeader.addContent(band);

    root.addContent(pageHeader);
}

private void addDetail (Element root){

    Element detail = new Element("detail",nsp);
    Element band = new Element("band",nsp);
    band.setAttribute("height", "125");
    band.setAttribute("splitType", "Stretch");
    detail.addContent(band);

    root.addContent(detail);
}
private void addpageFooter (Element root){

    Element pageFooter = new Element("pageFooter",nsp);
    Element band = new Element("band",nsp);
    band.setAttribute("height", "54");
```

```
        band.setAttribute("splitType", "Stretch");
        pageFooter.addContent(band);

        root.addContent(pageFooter);
    }


    private void addText(Element root,String texto, String posx, String posy,
String place, String size, String width, String height){
        Element staticText = new Element("staticText", nsp);
        Element reportElement = new Element("reportElement", nsp);
        Element textElement = new Element("textElement", nsp);
        Element font = new Element("font", nsp);
        Element text = new Element("text", nsp);

        reportElement.setAttribute("x",posx);
        reportElement.setAttribute("y",posy);
        reportElement.setAttribute("width",width);
        reportElement.setAttribute("height",height);
        font.setAttribute("fontName","Arial");
        font.setAttribute("size",size);
        text.setText(texto);

        textElement.addContent(font);
        staticText.addContent(reportElement);
        staticText.addContent(textElement);
        staticText.addContent(text);

        root.getChild(place, nsp).getChild("band",
nsp).addContent(staticText);
    }
    private void addImgOnTitle(Element root, String caminho){
        Element image = new Element("image", nsp);
        Element reportElement = new Element("reportElement", nsp);
        Element imageExpression = new Element("imageExpression", nsp);

        reportElement.setAttribute("x","0");
        reportElement.setAttribute("y","0");
```

```
        reportElement.setAttribute("width","125");
        reportElement.setAttribute("height","60");

        imageExpression.setAttribute("class","java.lang.String");
        imageExpression.setText(caminho);

        image.addContent(reportElement);
        image.addContent(imageExpression);

        root.getChild("title", nsp).getChild("band", nsp).addContent(image);
    }

    private void addCurrentDateOnTitle(Element root){
        Element textField = new Element("textField", nsp);
        Element reportElement = new Element("reportElement", nsp);
        Element textElement = new Element("textElement", nsp);
        Element textFieldExpression = new Element("textFieldExpression", nsp);

        textField.setAttribute("pattern","EEEEE dd MMMMM yyyy");
        reportElement.setAttribute("y","59");
        reportElement.setAttribute("x","455");
        reportElement.setAttribute("width","100");
        reportElement.setAttribute("height","20");
        textFieldExpression.setAttribute("class","java.util.Date");
        textFieldExpression.setText("new java.util.Date()");


        textField.addContent(reportElement);
        textField.addContent(textElement);
        textField.addContent(textFieldExpression);


        root.getChild("title", nsp).getChild("band",
nsp).addContent(textField);
    }

    private void addField(Element root,String nome,String classe, String
expression, String posx,String posy, String place){
```

```
        //especificacoes
         Element field = new Element("field", nsp);
         Element fieldDescription = new Element("fieldDescription", nsp);
       field.setAttribute("name", nome);
         field.setAttribute("class", classe);
         fieldDescription.setText(expression);
         field.addContent(fieldDescription);
         root.addContent(1,field);
         //campo
         Element textField = new Element("textField", nsp);
         Element reportElement = new Element("reportElement", nsp);
         Element textElement = new Element("textElement", nsp);
         Element textFieldExpression = new Element("textFieldExpression", nsp);

         reportElement.setAttribute("x",posx);
         reportElement.setAttribute("y",posy);
         reportElement.setAttribute("width","100");
         reportElement.setAttribute("height","20");
         textFieldExpression.setAttribute("class",classe);
         textFieldExpression.setText("$F{"+nome+"}");
         textField.addContent(reportElement);
         textField.addContent(textElement);
         textField.addContent(textFieldExpression);
                 root.getChild(place, nsp).getChild("band",
nsp).addContent(textField);
         }}
```