UNIVERSIDADE DE CAXIAS DO SUL
Centro de Computação e Tecnologia da Informação
Curso de Bacharelado em Ciência da Computação

Henrique Fiorio

PROTÓTIPO DE UM CONVERSOR DE RELATÓRIOS ORACLE REPORTS PARA JASPERREPORTS

Caxias do Sul

2011

Henrique Fiorio

PROTÓTIPO DE UM CONVERSOR DE RELATÓRIOS ORACLE REPORTS PARA JASPERREPORTS

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Ciência da Computação da Universidade de Caxias do Sul.

Daniel Luís Notari
Orientador

Caxias do Sul

2011

AGRADECIMENTOS

Agradeço a minha família por acreditar e me apoiar durante a minha vida, a minha namorada, Juliana de Lima, por me ajudar e corrigir inúmeras vezes meus textos, e pelo companheirismo.

RESUMO

A empresa NL Informática possui o seu ERP desenvolvido com a tecnologia Oracle Forms e Oracle Reports na versão 6i. Existem, atualmente, aproximadamente 1700 relatórios desenvolvidos nesta tecnologia. Por mudanças nas políticas de comercialização dos produtos da Oracle, a empresa optou pela migração de seus produtos para tecnologia Java. Desta forma, optou-se pela ferramenta de criação de relatórios JasperReports. Este trabalho tem o objetivo de desenvolver um conversor para migrar os relatórios desenvolvidos em Oracle Reports para JasperReports. Serão apresentadas as funcionalidades das ferramentas citadas, bem como, a forma que elas armazenam os relatórios utilizando a estrutura XML. Além disto, será feita a modelagem e implementação do *software* de conversão.

Palavras-chaves: Relatórios Gerenciais, Oracle Reports, JasperReports.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Sigla	Significado em Português	Significado em Inglês
DTD	Definição de Tipo de Documento	*Document Type Definition*
JDBC	Conectividade Java com Bancos de Dados	*Java Database Connectivity*
RDF	Arquivo Fonte do Oracle Reports em Formato Binário	*Oracle Reports Binary Source File*
SGBD	Sistema de Gerenciamento de Banco de Dados	*Database Management System*
URL	Localizador Uniforme de Recursos	*Uniform Resource Locators*
WYSIWYG	O Que Você Vê é o Que Você Obtém	*What You See Is What You Get*
XML	Linguagem de Marcação Extensível	*Extensible Markup Language*
XSD	Definição de Esquema XML	*XML Schema Definition*
UML	Linguagem de Modelagem Unificada	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

A empresa NL Informática[1], fundada em Caxias do Sul há 29 anos tem como principal produto o ERP NL Gestão, que ao longo de sua trajetória passou por inúmeras transformações, dentre elas quatro migrações de tecnologias de desenvolvimento. Atualmente possui o ERP desenvolvido com tecnologia Oracle, como Oracle Forms e Reports na versão 6i.

Relatórios são documentos bem formatados com informações geradas a partir de uma fonte de dados, que são utilizados para o apoio na tomada de decisões gerenciais em uma organização (STAIR & REYNOLDS, 2002).

Oracle Reports Builder é a ferramenta de geração de relatórios da *Oracle Corporation*, distribuído no pacote de desenvolvimento Oracle Developer Suite. Neste trabalho será utilizada versão *10g* que oferece a opção de salvar o arquivo descritor do relatório em formato XML, que será utilizado para o desenvolvimento do trabalho (SNEDECOR, 2005).

Por mudanças nas políticas de comercialização dos produtos Oracle, em especial Forms e Reports, a NL optou pela criação de novas funcionalidades utilizando a tecnologia Java e não mais Oracle, além de migrar gradualmente as já existentes. A opção de utilização do Java deve-se ao fato de não possuir custo de licenciamento, pela sua portabilidade e principalmente pela necessidade de disponibilizar aos clientes seus produtos com acesso via web.

Atualmente a NL possui aproximadamente 1700 relatórios desenvolvidos na tecnologia Oracle Reports. Realizar a migração manual destes relatórios geraria um elevado custo de mão de obra e seria necessário muito tempo de desenvolvimento, além disto, no processo manual, existe a possibilidade de ocorrerem erros. Por estes motivos se torna viável o investimento em uma ferramenta que possibilite a conversão automática.

1 www.nl.com.br

1.1 NOVO CENÁRIO

Com o novo cenário de desenvolvimento utilizando Java, foi escolhido como ferramenta de geração de relatórios o JasperReports, que é uma biblioteca *open source,* inteiramente escrita em Java, que fornece aos desenvolvedores a capacidade de criar, visualizar, imprimir e exportar em uma variedade de formatos de documentos, incluindo HTML, PDF, Excel, Word e OpenOffice (SIDDIQUI, 2010).

1.2 OBJETIVOS

O objetivo deste trabalho é desenvolver um *software* aplicativo *desktop*, que tenha a capacidade de converter relatórios criados a partir da ferramenta Oracle Reports na versão 6i, exportados em formato XML, para relatórios que possam ser executados em Java utilizando a biblioteca *open source* JasperReports.

1.3 ESTRUTURA DO TEXTO

O trabalho está estruturado da seguinte maneira:

No capítulo 2 são apresentados os tipos de relatórios existentes, bem como são apresentadas as ferramentas utilizadas para criação de relatórios: Oracle Reports e JasperReports.

O capítulo 3 apresenta a estrutura dos arquivos descritores dos relatórios. O capitulo 4 apresenta a modelagem do *software.* O capítulo 5 mostra como o software foi desenvolvido, quais as ferramentas utilizadas e como foi implementado.

No capitulo 6 são apresentados quatro casos de uso, com relatórios desenvolvidos pela empresa NL. E, por fim, o capitulo 7 apresenta as conclusões sobre o projeto.

.

2 FERRAMENTAS DE RELATÓRIO

Relatórios são documentos bem formatados com informações geradas a partir de uma fonte de dados, que são utilizados para o apoio na tomada de decisões gerenciais em uma organização (STAIR & REYNOLDS, 2002).

Neste capítulo será apresentada uma visão geral sobre as ferramentas de geração de relatórios que serão estudadas neste trabalho: Oracle Reports Builder e JarperReports. Inicialmente se discorrerá sobre os tipos mais comuns de relatórios.

2.1 TIPOS DE RELATÓRIOS

De acordo com as pesquisas realizadas em ferramentas de desenvolvimento de relatório, com base nos *templates e helps* disponibilizados por elas, pode-se identificar alguns tipos básicos de relatórios (ORACLE, 2010; MICROSOFT, 2010; IREPORT, 2010):

a) tabular
b) formulário
c) mestre/detalhe
d) etiqueta
e) carta
f) matriz

Os tipos de relatório serão explicados e exemplificados a seguir. Os dados utilizados para criação dos exemplos foram retirados do esquema modelo disponibilizado junto da instalação do banco de dados da Oracle.

2.1.1 Relatório Tabular

O formato tabular é o formato mais básico de relatório. Neste formato os dados são apresentados como uma planilha. Os campos são dispostos em colunas, onde o rótulo fica na parte superior e os dados relativos ao campo ficam alinhados

abaixo, onde cada linha representa um registro. A Figura 1 apresentada um exemplo de relatório tabular.

Nome	Sobrenome	Salário
Steven	King	R$ 24.000,00
Neena	Kochhar	R$ 17.000,00
Lex	De Haan	R$ 17.000,00
Alexander	Hunold	R$ 9.000,00
Bruce	Ernst	R$ 6.000,00
David	Austin	R$ 4.800,00
Valli	Pataballa	R$ 4.800,00
Diana	Lorentz	R$ 4.200,00
Nancy	Greenberg	R$ 12.000,00
Daniel	Faviet	R$ 9.000,00

Figura 1: Exemplo de relatório tabular
Fonte: Elaborado pelo autor

2.1.2 Relatório Formulário

O formato formulário ou empilhado apresenta os dados de um registro em grupos, um campo abaixo do outro, geralmente com o rótulo do campo aparecendo à esquerda ou sobre o valor. Este formato é utilizado em casos onde os dados a serem apresentados não cabem em apenas uma linha e precisam estar mais espaçados para uma melhor legibilidade. Na Figura 2 pode-se visualizar como ficaria o mesmo exemplo da Figura 1 em formato de formulário.

Id:	100
Nome:	Steven
Sobrenome:	King
Data de admissão:	17/06/1987
Salário:	R$ 24.000,00
Id:	101
Nome:	Neena
Sobrenome:	Kochhar
Data de admissão:	21/09/1989
Salário:	R$ 17.000,00

Figura 2: Exemplo de relatório formulário
Fonte: Elaborado pelo autor

2.1.3 Relatório Mestre/Detalhe

Relatórios mestre/detalhe são uma variação para os formatos tabular e/ou for-

mulário utilizando agrupadores. Neste formato são apresentados para cada registro (mestre) informações detalhadas referentes a ele, e podem ainda ter informações com totalizadores para cada grupo. Como mostra a Figura 3, onde constam, por exemplo, os funcionários de cada departamento.

ID	NOME	SALÁRIO
10	**Administração**	
200	Jennifer Whalen	R$ 4.400,00
		R$ 4.400,00
20	**Marketing**	
201	Michael Hartstein	R$ 13.000,00
202	Pat Fay	R$ 6.000,00
		R$ 19.000,00
30	**Compras**	
114	Den Raphaely	R$ 11.000,00
115	Alexander Khoo	R$ 3.100,00
116	Shelli Baida	R$ 2.900,00
117	Sigal Tobias	R$ 2.800,00
118	Guy Himuro	R$ 2.600,00
119	Karen Colmenares	R$ 2.500,00
		R$ 24.900,00

Figura 3: Exemplo de relatório com agrupadores
Fonte: Elaborado pelo autor

2.1.4 Relatório Etiqueta

O formato etiqueta é utilizado para criar rótulos informativos, como por exemplo, para endereços de envelopes de cartas, para identificação de mercadorias e para envio de mala direta. Normalmente possui um papel específico para impressão, com tamanho e posições pré-definidos. Cada registro pode ser replicado tanto na vertical quanto na horizontal, como mostra a Figura 4.

Figura 4: Exemplo de relatório etiqueta
Fonte: Elaborado pelo autor

2.1.5 Relatório Carta

O formato carta é utilizado para criação de cartas onde o conteúdo principal é um texto fixo com algumas informações variáveis, como por exemplo, mala direta, uma carta de um comunicado de banco, que apresenta ao longo do conteúdo, informações diferentes de cada cliente, como nome e número de cadastro. Isto é útil, pois permite criar milhares de cartas com conteúdos semelhantes, porém distintos, de uma única vez. A Figura 5 mostra um exemplo de como pode ser feita uma carta.

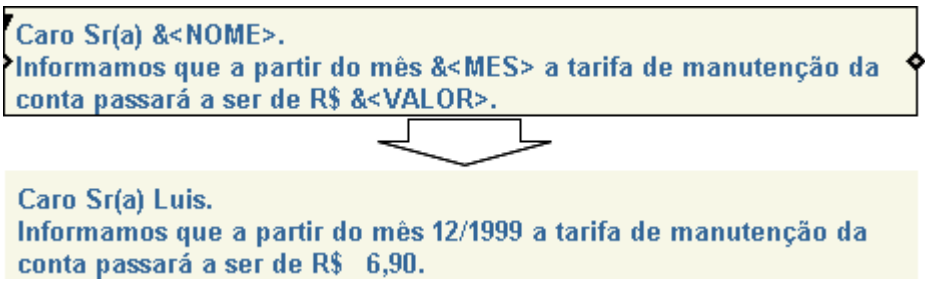


Figura 5: Exemplo de relatório carta
Fonte: Elaborado pelo autor

2.1.6 Relatório Matriz

O formato matriz, também conhecido como tabela de referência cruzada, é utilizado para facilitar a visualização dos dados do resultado do cruzamento de informações. Esse formato pode ser utilizado em situações como, por exemplo, em uma concessionária de veículos, para obter as informações das vendas de cada modelo de carro e as vendas realizadas por cada vendedor.

Por exemplo, usando a abordagem comum com formato tabular e agrupamento precisa-se de dois relatórios: um agrupando por modelos e outro agrupando por vendedores. Enquanto utilizando uma abordagem com relatório matriz obtêm-se todas as informações em um só relatório. Este exemplo é mostrado na Figura 6.

Agrupado por produto

NOME	PREÇO
Civic	
Steven	R$ 61.350,00
Alexander	R$ 62.350,00
Bruce	R$ 60.500,00
	R$ 184.200,00
Corolla	
Alexander	R$ 55.600,00
Steven	R$ 57.900,00
	R$ 113.500,00
Vectra	
Steven	R$ 59.000,00
Bruce	R$ 61.200,00
	R$ 120.200,00
	R$ 417.900,00

Agrupado por vendedor

PRODUTO	PREÇO
Alexander	
Corolla	R$ 55.600,00
Civic	R$ 62.350,00
	R$ 117.950,00
Bruce	
Vectra	R$ 61.200,00
Civic	R$ 60.500,00
	R$ 121.700,00
Steven	
Corolla	R$ 57.900,00
Vectra	R$ 59.000,00
Civic	R$ 61.350,00
	R$ 178.250,00
	R$ 417.900,00

Matriz

	Civic	Corolla	Vectra	Total por Produto
Alexander	R$ 62.350,00	R$ 55.600,00	R$ 0,00	R$ 117.950,00
Bruce	R$ 60.500,00	R$ 0,00	R$ 61.200,00	R$ 121.700,00
Steven	R$ 61.350,00	R$ 57.900,00	R$ 59.000,00	R$ 178.250,00
Total por Vendedor	R$ 184.200,00	R$ 113.500,00	R$ 120.200,00	**R$ 417.900,00**

Figura 6: Exemplo de relatório matriz
Fonte: Elaborado pelo autor

Geradores de relatórios são ferramentas que facilitam a criação de relatórios (BARNFIELD, 1998). Um dos recursos dessas ferramentas é a manipulação do *layout*, que possibilita o manuseio dos elementos com o *mouse*, por exemplo, fazendo o posicionamento do elemento dentro do relatório. Outro recurso destas ferramentas é a criação das fontes de dados, que no momento da execução fornecem os dados para a montagem do documento.

A seguir são apresentadas as ferramentas de geração de relatório da Oracle[2] e da Jaspersoft[3].

2.2 ORACLE REPORTS BUILDER

Para o desenvolvimento desta seção foi utilizada como bibliografia a documentação disponibilizada pela Oracle que pode ser encontrada em seu site[4].

2 www.oracle.com
3 www.jaspersoft.com
4 http://www.oracle.com/technetwork/middleware/reports/documentation/index.html

Oracle Reports Builder é a ferramenta de geração de relatórios da Oracle Corporation, distribuído no pacote de desenvolvimento Oracle Developer Suite. Neste trabalho será utilizada versão *10g* que oferece a opção de salvar o arquivo descritor do relatório em formato XML, que será utilizado para o desenvolvimento do trabalho (SNEDECOR, 2005).

A Figura 7 apresenta a tela principal e as principais ferramentas utilizadas no Reports Builder.

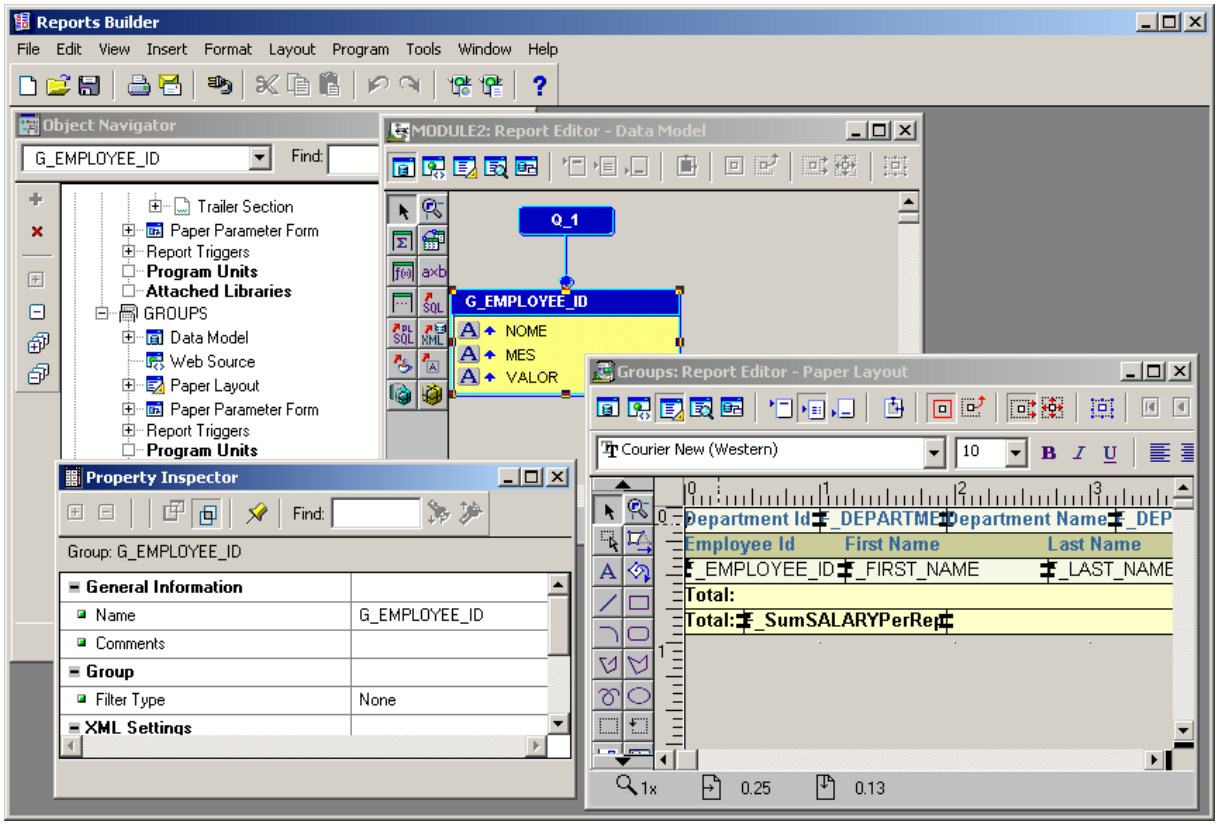


Figura 7: Janela principal do Reports Builder
Fonte: Elaborado pelo autor

No Reports Builder, o relatório é desenvolvido editando os objetos que compõem o relatório. Estes objetos podem ser acessados dentro da ferramenta através do *Object Navigator,* nele pode ser visualizada a hierarquia dos objetos que integram o relatório, dentre eles, os principais são (BARNFIELD, 2008):

a) *Data Model Object*
b) *Layout Object*
c) *Parameter Form Object*
d) *Reports trigger/Program Units*

Além da visão hierárquica dos objetos também é possível visualizar os objetos agrupando-os por tipo. Através do *Object Navigator,* apresentado na Figura 8, pode-se abrir o *Report Editor* e também o *Property Inspector.*

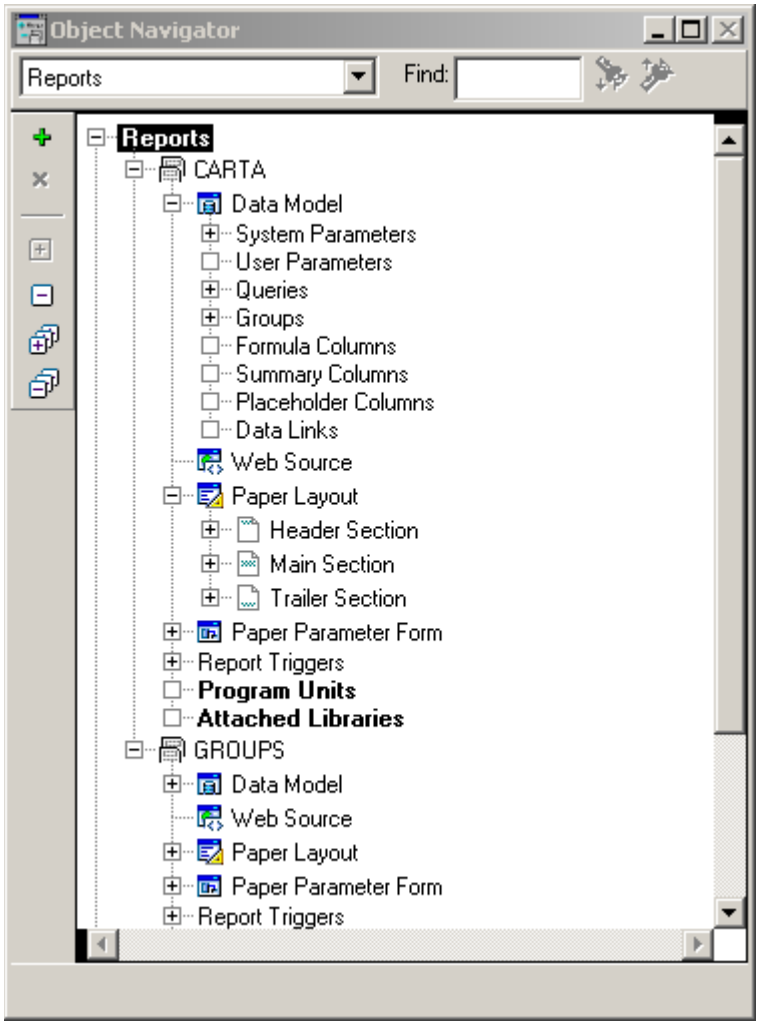


Figura 8: Reports Builder - Object Navigator
Fonte: Elaborado pelo autor

O *Property Inspector* é uma janela onde são apresentadas todas as propriedades de um elemento do relatório. Ela pode ser acessada utilizando o *menu* de contexto do elemento ou usando a tecla de atalho F4. Quando aberta, apresenta automaticamente as propriedades do elemento corrente e conforme o usuário seleciona os elementos ele mostra as propriedades relativas aos mesmos, mas também possui o recurso de fixar a um determinado elemento, isso permite que possa ser utilizado, por exemplo, para comparação de propriedades de dois elementos diferentes (SNEDECOR, 2005). Na Figura 9 pode-se visualizar esta funcionalidade.

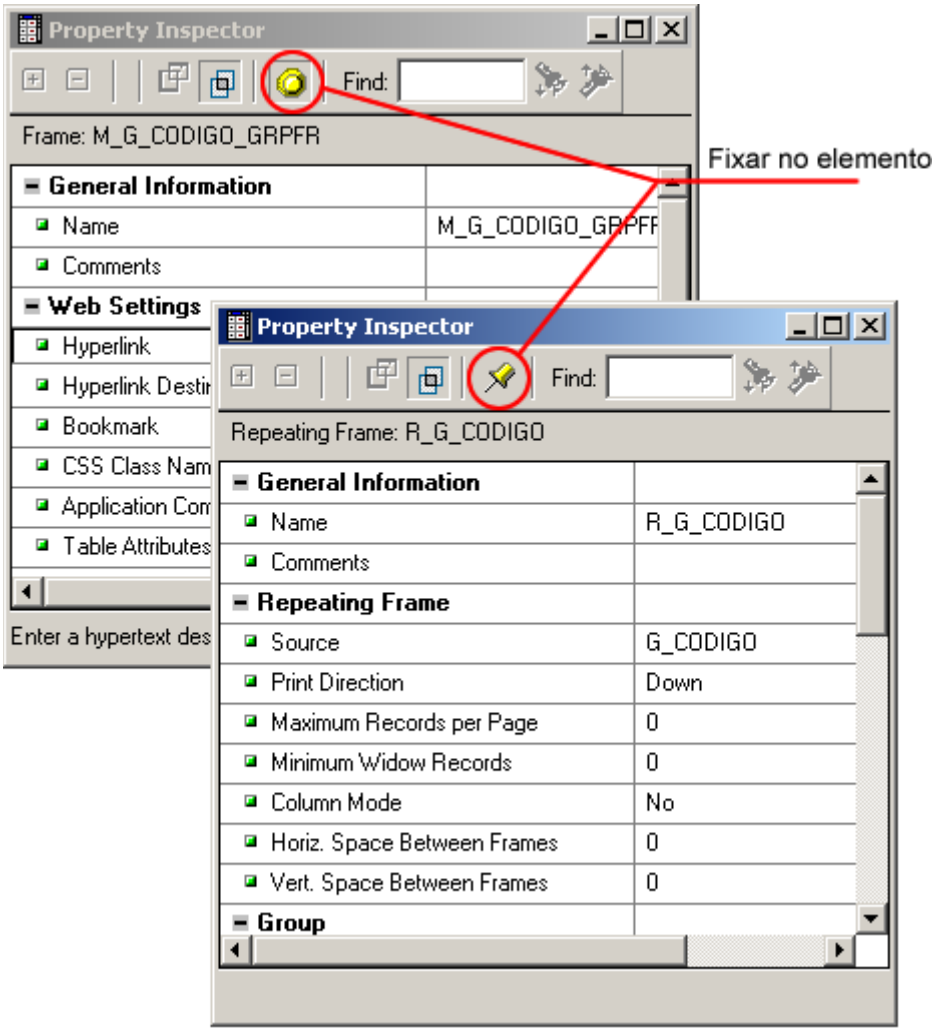


Figura 9: Reports Builder - Property Inspector
Fonte: Elaborado pelo autor

O Report Editor é a janela com a área de trabalho onde pode-se manipular os objetos do relatório diretamente ou alterar suas propriedades utilizando o *Property Inspector.* Nesta janela pode-se editar os objetos do relatório de diferentes pontos de vista. Possui uma visualização para o *Data Model,* uma para o *Layout Model e* uma para o *Parameter Form,* além disso, possui uma visualização em modo WYSIWYG que permite alterar o *layout* do relatório em tempo real com dados, preenchendo o relatório, facilitando o posicionamento de cada elemento no mesmo (SNEDECOR, 2005). Estes itens serão explicados nas próximas subseções.

2.2.1 Data Model Object

O D*ata Model Object* é o local onde são informadas as fontes de dados para o preenchimento das informações do relatório. Estas fontes são adicionadas ao *data model através* de elementos chamados *queries.* Elas podem obter os dados de diferentes maneiras (SNEDECOR, 2005):

- SQL conectado diretamente no banco de dados Oracle;
- arquivo XML;
- arquivo texto com largura variável delimitado por vírgula;
- arquivo texto com largura fixa delimitado por espaço;
- SQL conectado por JDBC;
- etc.

Os relatórios mais comuns possuem apenas uma *query,* que pode ser usada para criar relatórios nos formatos tabular, formulário, etiqueta, carta e com agrupadores. Para criar os agrupadores utiliza-se o elemento chamado *group,* obtendo-se um relatório do tipo mestre/detalhe.

Num mesmo relatório podem ser adicionadas mais de uma *query,* isso é útil quando se deseja (SNEDECOR, 2005):

- Produzir relatórios com múltiplas partes, com dados independentes, que é frequentemente chamado de relatório mestre/mestre;
- Produzir relatórios com múltiplas partes, com dados relacionados, ou seja, relatório mestre/detalhe. Esta relação entre as *queries* é feita pelo elemento *Data Link;*
- Diminuir a complexidade dos comandos SQL para facilitar caso haja a necessidade de manutenção futura. Ao invés de criar comandos SQL complexos para obter os dados e utilizar *groups* para criar os agrupamentos, podem ser criadas várias *queries* relacionadas com comandos SQL simples, obtendo o mesmo efeito.
- Mostrar os mesmo dados com critérios de ordenação ou filtragem diferentes.

Os grupos são criados para organizar as colunas no relatório. Eles possuem duas funcionalidades: separar os dados da *query* em conjuntos e filtrar os dados da

mesma. Ao criar uma *query* o Reports Builder cria automaticamente um grupo que contém todas as colunas selecionadas na *query.* Grupos adicionais podem ser criados para obter níveis de quebra no relatório (mestre/detalhe), isso pode ser realizado manualmente no *Report Editor*, os grupos também são criados quando utilizado o *Report Wizard*. Todos os grupos criados pelo usuário são grupos de quebra (*break groups*), com exceção dos *cross-product groups,* que são utilizados para a criação de relatórios tipo matriz.

Para restringir condicionalmente os registros apresentados por um grupo, usa-se os *Group filters*, que possui três opções de filtros (SNEDECOR, 2005):

- *Fisrt* – são apresentados os primeiros *n* registros da *query;*
- *Last* – são apresentados os últimos *n* registros da *query;*
- PL/SQL – o usuário cria uma função utilizando linguagem PL/SQL, que deve retornar um valor booleano. Essa função é invocada para cada registro do grupo e conforme o retorno, o registro é incluído ou excluído do relatório.

Além das colunas obtidas através da *query,* existe a possibilidade de adicionar colunas calculadas. Estas colunas podem ser, por exemplo, para apresentar subtotais para os grupos, ou fórmulas calculadas com base nas outras colunas da *query.* Estas fórmulas são calculadas através de funções escritas em linguagem PL/SQL.

Nas Figuras 10 e 11 pode-se visualizar a janela do *Report Editor* na visualização *Data Model,* com dois exemplos de definição de dados, um utilizando grupos para realizar as quebras de níveis e outro utilizando duas *queries* com *data link.*

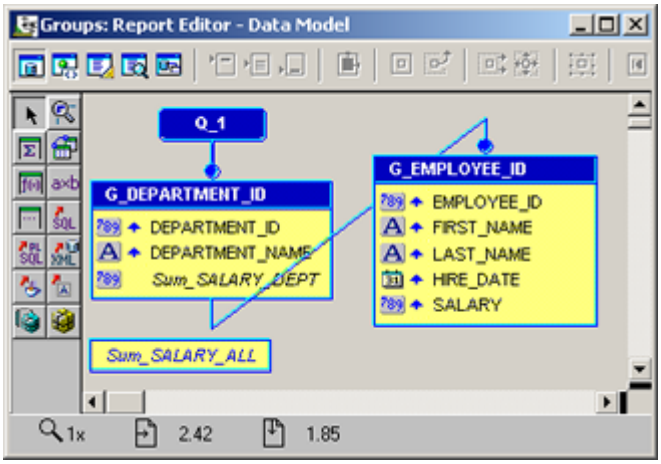


Figura 10: Report Editor - Data Model (*groups*)
Fonte: Elaborado pelo autor

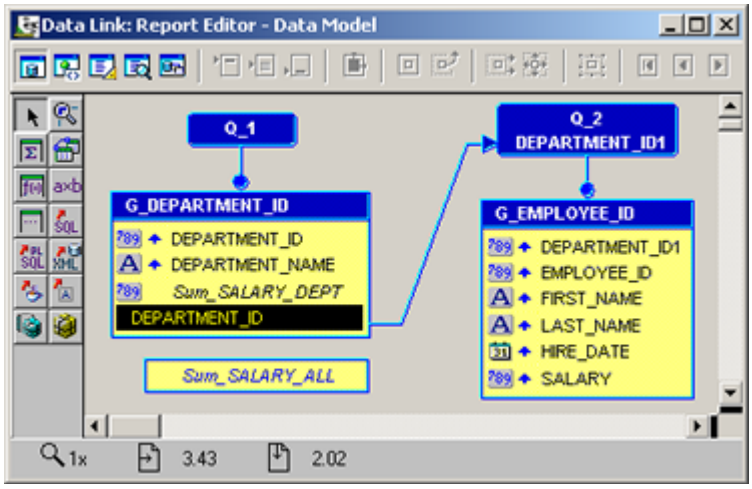


Figura 11: Report Editor - Data Model (*data link*)
Fonte: Elaborado pelo autor

2.2.2 Layout Object

O *layout object* é onde são definidos os componentes visuais apresentados no relatório. Ele é divido em três seções (SNEDECOR, 2005):

- *Header* – é o cabeçalho do relatório, a primeira parte apresentada, uma vez em cada relatório, antes dos dados.
- *Main* – é o corpo do relatório, onde é apresentado o conteúdo, os dados do relatório. É composto por corpo e margem, sendo que no corpo são incluídos os componentes que apresentam os dados do relatório e a margem é apresentada em cada página, utilizada para apresentar, por exemplo, um cabeçalho, ou rodapé de página, ou uma numeração de páginas, ou totalizadores.
- *Trailer* – é o rodapé do relatório, a última parte apresentada, uma vez em cada relatório, após dos dados.

Os componentes são adicionados no relatório em uma estrutura hierárquica, disponibilizados para criação do relatório, que são (SNEDECOR, 2005):

- *Frames*
- *Repeating frames*
- *Fields*
- *Boilerplate objects*
- *Graphics*

Os *frames* são usados para agrupar outros componentes e marcar áreas. Eles têm a função de envolver e proteger para que esses componentes não sejam “empurrados” ou sobrepostos por outros componentes. São usados, por exemplo, para colocar os componentes de um grupo, cabeçalho de colunas ou totalizadores. *Frames* aceitam qualquer outro componente como “filho”.

Os *repeating frames são* atrelados a um grupo e dentro deles são adicionados os campos pertencentes a este grupo. Na execução do relatório ele imprime uma vez para cada registro, o conteúdo disposto dentro dele. Assim com os *frames* aceitam qualquer outro componente como “filho”, inclusive outros *repeating frames. Q*uando aninhados são utilizados para reproduzir as quebras dos grupos criando relatórios mestre/detalhe, para cada registro do *repeating frame* mais *e*xterno, será preenchido o *repeating frame* mais interno com os registros relacionados de acordo com o grupo. O *Report Builder* cria um *repeating frame* automaticamente para cada grupo criado, quando para isso, é utilizado o *wizard.*

O *field* é o componente utilizado para apresentar os dados textuais. Nele são vinculadas as colunas dos grupos, colunas calculadas, parâmetros, valores, como número das páginas, data atual, etc. Para uma melhor visualização, esse componente permite informar uma máscara para formatação.

Os *Boilerplate Objects* são os componentes com conteúdo estático, como formas gráficas (linhas, retângulos, elipses, etc.), textos e imagens.

Com o componente *Graphics* é possível a criação de gráficos dinâmicos com base nos dados em formatos como barras, pizza, linhas, etc.

Na Figura 13 pode-se visualizar o *Report Editor* em modo de edição do *Layout Object, e na Figura 12* em modo WYSIWYG, o que possibilita que o *layout* seja manipulado visualizando o relatório com mesma aparência do resultado final.

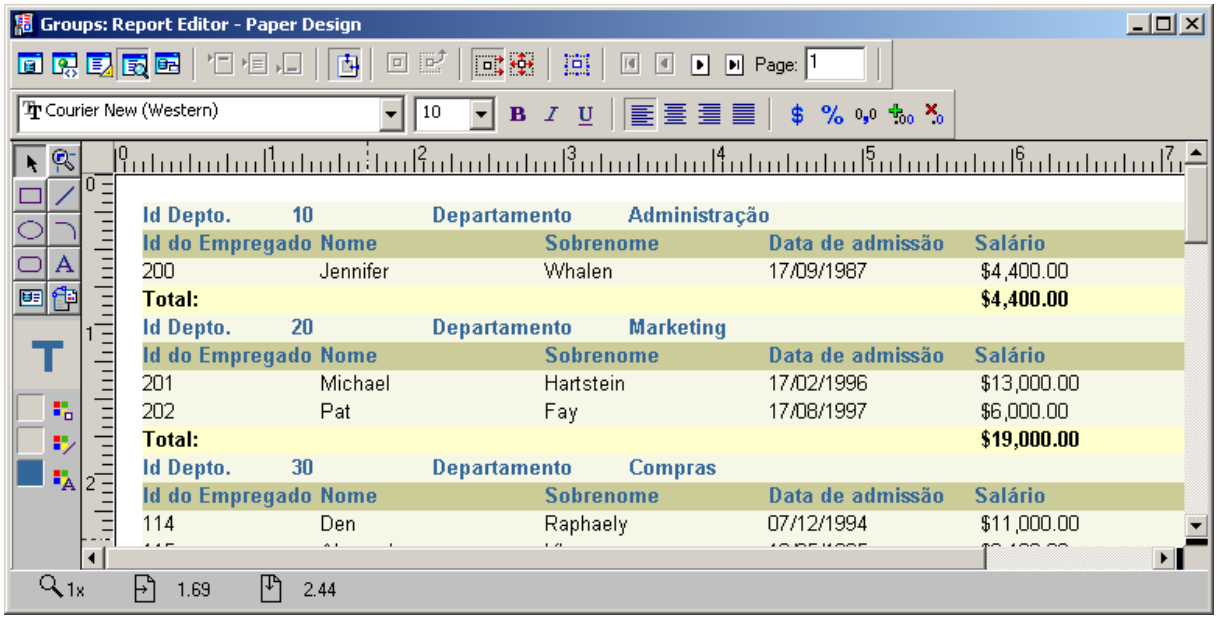


Figura 12: Report Editor - *Layout Object* (WYSIWYG)
Fonte: Elaborado pelo autor

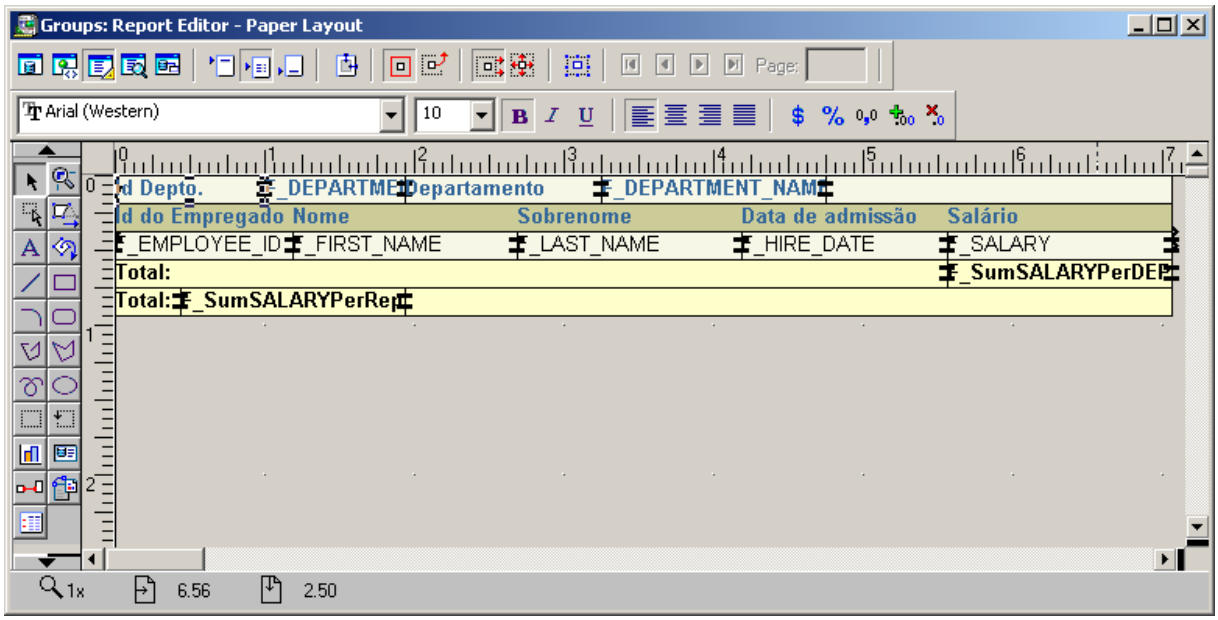


Figura 13: Report Editor - *Layout Object*
Fonte: Elaborado pelo autor

2.2.3 Parameter Form Object

O relatório possui parâmetros, que são variáveis cujo valor pode ser atribuído no momento da execução, esse valor pode ser atribuído por linha de comando, passando como parâmetro na execução, através de outro programa ou criando um formulário no *Parameter Form Object* (SNEDECOR, 2005).

Os parâmetros são utilizados principalmente para modificar o SQL das *queries* no momento da execução e como variáveis nas funções PL/SQL que podem ser utilizadas no relatório (SNEDECOR, 2005). Na Figura 14 e 15 observa-se um exemplo de utilização de parâmetros no SQL e o formulário criado para entrada dos valores nos parâmetros.

```
SELECT e.employee_id
      ,e.first_name
      ,e.last_name
      ,e.hire_date
      ,e.department_id
      ,e.salary
      ,d.department_name
  FROM employees e, departments d
 WHERE e.department_id = d.department_id
   AND (:p_salary IS NULL OR e.salary >= :p_salary)
   AND (:p_hire_date IS NULL OR e.hire_date >= :p_hire_date)
```

Figura 14: Reports Builder - Exemplo de utilização de parâmetros (SQL)
Fonte: Elaborado pelo autor

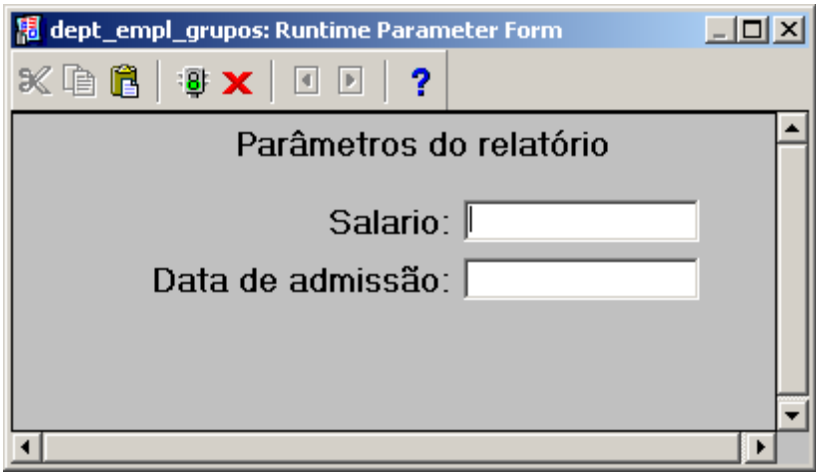


Figura 15: Exemplo de utilização de parâmetros (formulário)
Fonte: Elaborado pelo autor

2.3 JASPERREPORTS

JasperReports[5] é uma biblioteca Java[6] *open-source* para desenvolvimento de relatórios, distribuída pela Jaspersoft Corporation. Ela não é uma aplicação autônoma, ou seja, não roda sozinha, ela deve ser incorporada nas aplicações Java, incluindo sua biblioteca no CLASSPATH da aplicação (HEFFELFINGER, 2006).

5 jasperforge.org/projects/jasperreports
6 www.java.com

O relatório no JasperReports é definido por um arquivo XML, que por padrão a extensão destes arquivos é *.jrxml*. O *jrxml* é compilado, dando origem a um arquivo binário, que é conhecido como arquivo *Jasper* e é normalmente salvo com a extensão *.jasper*. A partir do arquivo *Jasper* o relatório é executado e preenchido, o resultado da execução pode ser visualizado na tela ou exportado em formatos como, por exemplo, PDF, DOC, XLS, CSV, entre outros. A Figura 16 apresenta um *workflow* típico da criação e execução de um relatório *Jasper*.

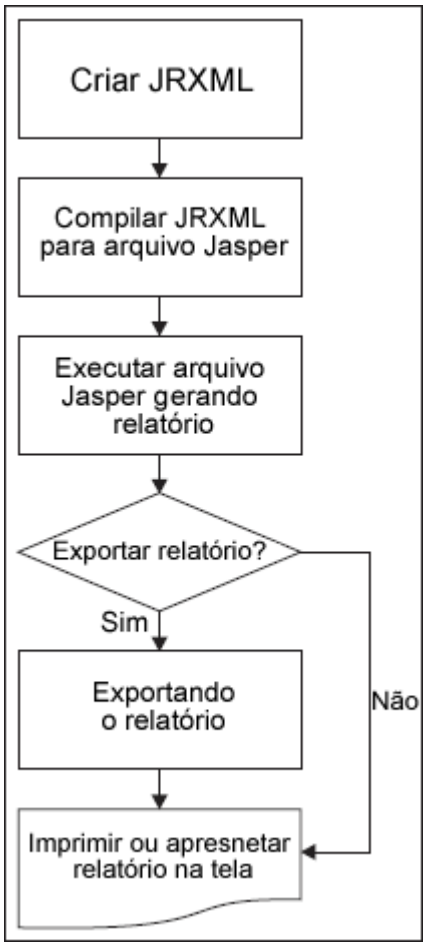


Figura 16: *Workflow* típico de relatório JasperReports
Fonte: HEFFELFINGER, 2006

O arquivo *jrxml* pode ser criado manualmente ou através do uso de ferramentas gráficas. Nesta seção será apresentada a criação do relatório utilizando a ferramenta gráfica oficial da Jaspersoft, o iReport[7], que também é *open-source*. A Figura 17 apresenta a tela principal da ferramenta.

7 jasperforge.org/projects/ireport

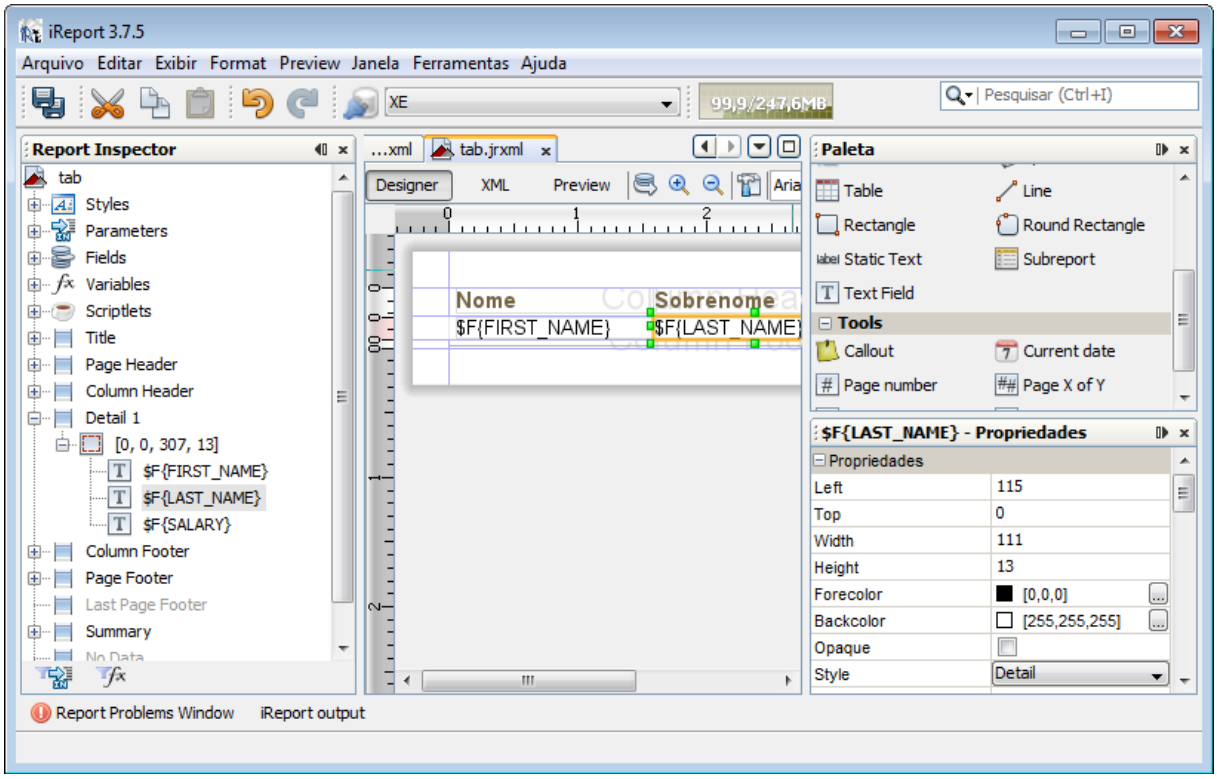


Figura 17: Tela principal do iReport
Fonte: Elaborado pelo autor

A seguir serão apresentados os principais objetos que compõem um relatório *Jasper*.

2.3.1 *Data Source*

O *data source* é a fonte de dados principal do relatório, ou seja, o local de onde são obtidas as informações para o preenchimento do relatório. O JasperReports fornece diferentes opções para obtenção destes *data sources*, podendo ser feito, por exemplo, por conexão a qualquer banco de dados com suporte a JDBC, arquivos XML, arquivos XLS, coleções de objetos JavaBean, qualquer objeto que implementa interface java.util.Map, ou criando uma classe personalizada implementando a interface JRDataSource. A Figura 18 mostra onde, no iReport, pode ser configurado o *data source* e as opções disponíveis (SIDDIQUI, 2010).

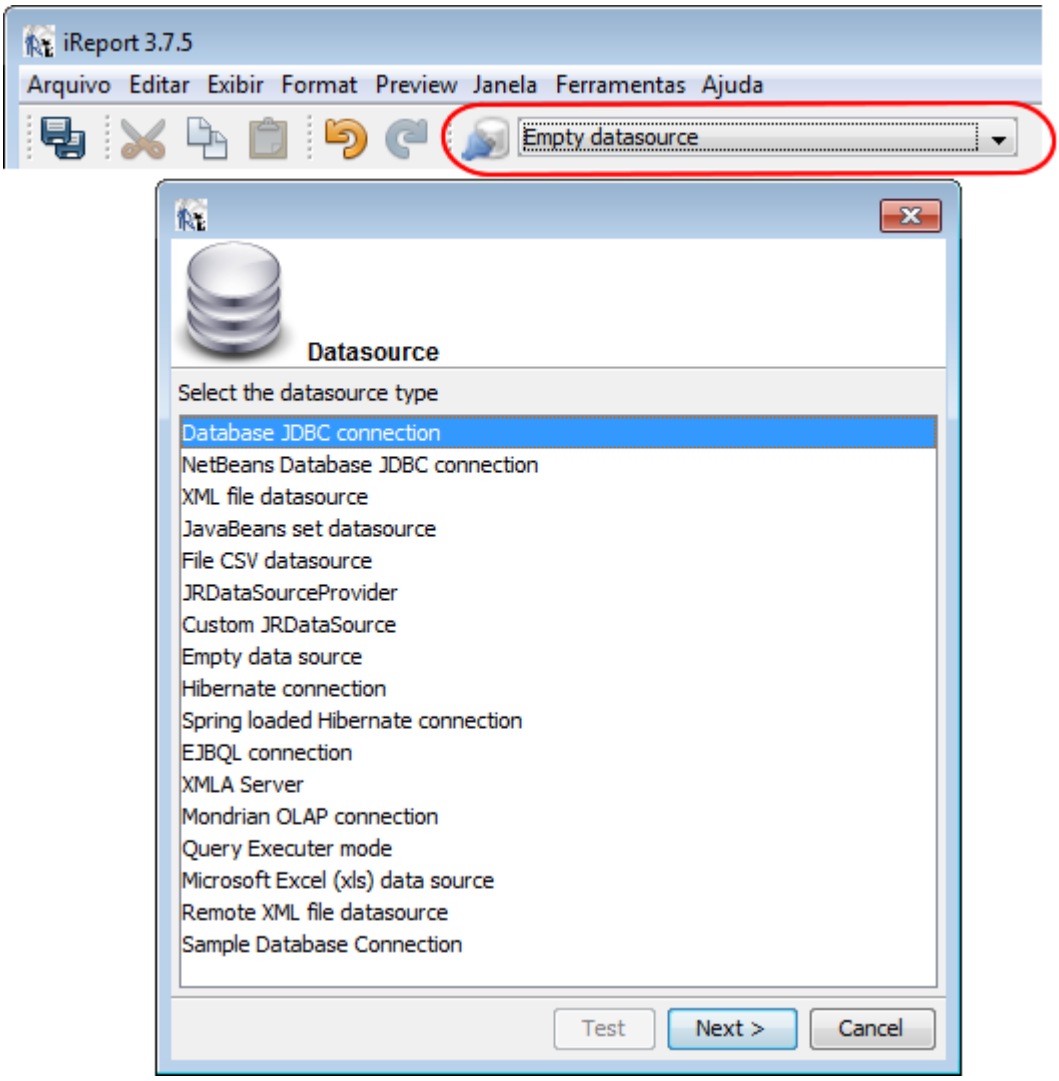


Figura 18: iReport - Fonte de dados
Fonte: Elaborado pelo autor

Para utilizar os dados obtidos pelo *data source*, devem ser criados elementos chamados de *fields,* onde cada um representa um campo retornado pelo *data source*, por exemplo, quando utilizado por JDBC cada *field* deve ser uma coluna do SQL informado.

2.3.2 Dataset

Em alguns casos é necessária a obtenção de um conjunto de dados diferentes da fonte de dados principal, para isso, pode ser usado o elemento *dataset,* sendo que pode ser adicionado mais de um no relatório. É uma fonte de dados independente da principal, e neles podem ser configurados parâmetros, variáveis e grupos.

2.3.3 Parâmetros

Parâmetros são objetos que podem ser passados para o relatório no momento da execução. Eles são úteis para obter informações que normalmente não se tem através do *data source,* como, por exemplo, passar o usuário que está executando o relatório, ou alterar o título dinamicamente. Outra utilização dos parâmetros é para modificação do SQL de consulta com o SGBD, podendo atuar como filtros dinâmicos. Eles são adicionados no relatório informando-se o nome e o tipo e podem ainda possuir uma expressão que defina seu valor padrão. No momento da execução eles são passados para o relatório através de um *java.util.Map,* onde a chave do *Map* é o nome do parâmetro (JASPERFORGE, 2010; AHAMMAD, 2010).

2.3.4 Variáveis

Variáveis são objetos que mantêm o valor durante a execução do relatório, com base em uma expressão. Elas podem ser utilizadas, por exemplo, para apresentar valores resultantes de fórmulas ou para criação de totalizadores.

Para criação de uma variável devem ser considerados os seguintes atributos (AHAMMAD,2010):

- *calculation type* – uma variável pode conter o valor resultante de um cálculo predefinido, conforme cada registro do relatório está sendo processado. Estes cálculos podem ser: soma, media, maior, menor, contagem, desvio padrão, entre outros. Por exemplo, para mostrar o valor total de um pedido, pode ser criada uma variável com tipo soma, com base no campo valor de cada item.
- *reset type* – *é* o momento em que a variável é reinicializada. O valor padrão é *report*, que significa que a variável é inicializada uma vez no início do relatório, e que o cálculo informado será executado até o fim. Este valor pode ser alterado, por exemplo, para *page* ou *group,* o que significa que a variável é reinicializada a cada nova quebra de página ou grupo, respectivamente, criando assim um subtotal.
- *Variable expression* – é a expressão de onde a variável obtém seu va-

lor, e que é utilizada para execução do *calculation type.*

Existem ainda variáveis internas do sistema, que são variáveis mantidas pelo *Jasper*, variáveis que mantém valores como, por exemplo, número atual da página, quantidade de registros processados na página, grupo ou relatório, entre outras.

2.3.5 Expressões

Expressões podem ser usadas para declarar variáveis que executam vários cálculos, para criação dos grupos de dados no relatório, para especificar o conteúdo dos campos texto ou para personalizar a aparência de objetos sobre o relatório.

Elas são basicamente código Java e podem fazer referência a *fields*, parâmetros ou variáveis. Para usar os *fields* coloca-se em torno dos seus nomes, *$F{* e *}* para *field*, *$P{* e *}* para parâmetros e *$V{* e *}* para variáveis. Como mostram os exemplos a seguir:

```
"$F{nome} + " " + $F{sobrenome}"
"Valor total  : R$ " + $V{somaTotal}
(new SimpleDateFormat($P{formato_data})).format($F{data_admissao})
```

O iReport possui uma janela com recursos que facilitam a criação das expressões, como mostra a Figura 19.

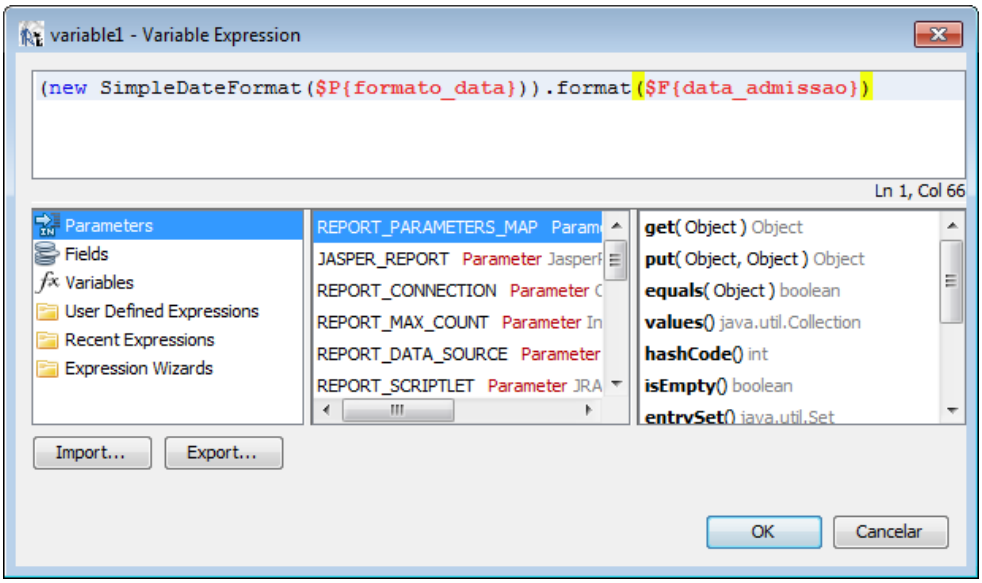


Figura 19: iReport - Janela para criação de expressões
Fonte: Elaborado pelo autor

2.3.6 Grupos

Os grupos são criados para agrupar os dados, levando em conta algo em comum. Os grupos no *Jasper* são criados informando-se uma expressão, na execução do relatório as quebras são criadas no momento em que o valor retornado pela expressão for diferente do valor anterior.

2.3.7 Bandas de visualização

A estrutura do relatório é composta por um conjunto de seções chamadas de bandas. Cada banda tem a sua própria altura, uma posição particular na estrutura e é usada para um objetivo particular. Nelas são adicionados os componentes visuais do relatório. As bandas disponíveis são (AHAMMAD, 2010):

- *Title* – é a primeira banda impressa do relatório, é impressa apenas uma vez.
- *Page Header* – é impressa em cada página do relatório, usada para criar o cabeçalho das páginas.
- *Column Header* – é impressa em cada página, desde que a página possua a banda *detail,* também é utilizada como cabeçalho das colunas.
- *Detail* – esta banda é impressa repetidamente para cada linha do *data source*, pode ser adicionada mais de uma banda *detail*.
- *Column Footer* – é impressa em cada página, desde que a página possua a banda *detail,* também é utilizada como rodapé das colunas.
- *Page Footer* – é impressa ao final de cada página, com exceção da última página se a banda *Last Page Footer* estiver definida.
- *Last Page Footer* – é impressa somente no rodapé da última página.
- *Summary* – é a última banda impressa do relatório, é impressa apenas uma vez.
- *Background* – é impressa em cada página, utilizada para colocar um fundo nas páginas, como, por exemplo, uma marca d'água.
- *No Data* – esta banda é impressa caso nenhum dado tenha sido obtido

e caso a propriedade *When No Data* do relatório tenha sido definida para mostrá-la.

- *Group Header* e *Group Footer* – para cada grupo criado estas bandas podem ser definidas. Elas são impressas a cada quebra do grupo.

O iReport possui uma área de trabalho para configuração destas bandas, que contém o recurso arrastar e soltar. Os componentes são adicionados através da janela Paleta. A Figura 20 mostra como é apresentada a área de trabalho das bandas dentro do iReport e a Figura 21 apresenta a Paleta com os componentes disponíveis.

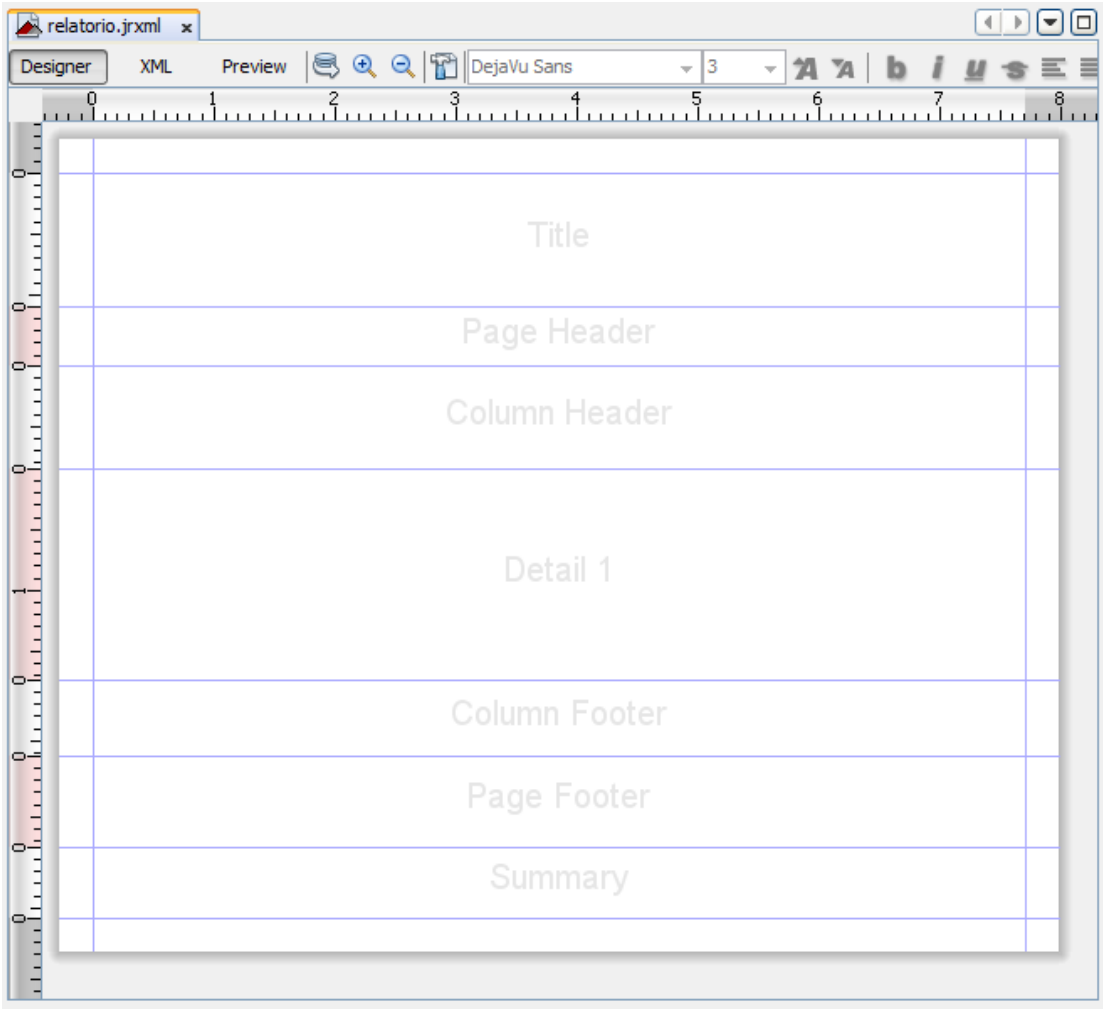


Figura 20: iReport - Área de trabalho das bandas
Fonte: Elaborado pelo autor

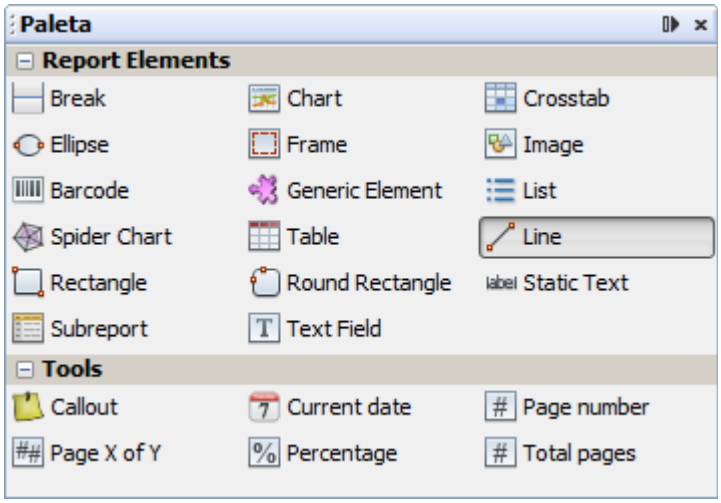


Figura 21: iReport - Paleta
Fonte: Elaborado pelo autor

2.3.8 Componentes visuais

Os componentes disponibilizados para criação do relatório Jasper são (JASPERFORGE, 2010):

- *Static Text* – utilizado para apresentar um texto estático.
- *Line*, *Elipse*, *rectangle* e *round rectangle* – utilizado para criar formas geométricas.
- *Image* – utilizado para apresentar imagens.
- *Text Field* – utilizado para apresentar o conteúdo resultante de uma expressão.
- *Frame* – tem a função de agrupar outros componentes, para criar, por exemplo, uma borda ou fundo em uma determinada área.
- *Chart* e *Spider Chart* – utilizados para criação de gráficos, com base no *data source* ou *dataset.*
- *Crosstab* – componente para criação de relatórios do tipo matriz, podem ser configurados para utilizar os dados do *data source* ou de um *dataset.*
- *Table* – utilizado para criar uma tabela com os dados de um *dataset.* Com este componente é possível criar relatórios tipo mestre/detalhe utilizando como mestre os registros do *data source* e detalhe os registros do *dataset*.

- *List* – utilizado para listar valores de um *dataset.*
- *Subreport* – com este componente é possível incorporar dentro do relatório outros relatórios, isso pode ser utilizado para simplificar relatórios complexos, ou para criar relatórios mestre/detalhe.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo demonstrou-se como são criados os relatórios através das ferramentas, e suas opções.

No próximo capitulo será apresentado como cada ferramenta armazena a definição dos relatórios criados, mostrando a estrutura de cada uma delas.

3 ESTRUTURA DE FUNCIONAMENTO

Neste capítulo será descrita a forma como cada uma das ferramentas armazena as informações da configuração dos relatórios. Tanto no Oracle Reports quanto no JasperReports é possível obter esta configuração em arquivo com formato XML. A seguir será apresentada a estrutura do XML para cada ferramenta.

3.1 ORACLE REPORTS BUILDER

Todos os elementos do arquivo XML do Oracle Reports estão definidos em um arquivo DTD, o *reports.dtd.* Ele pode ser encontrado na instalação do Oracle no seguinte caminho *ORACLE_HOME\reports\dtd*. Nesta seção serão apresentados os principais elementos que compõem a estrutura de um relatório criado no Oracle Reports, que serão utilizadas para o desenvolvimento deste trabalho. O elemento inicial do XML é o *report,* mostrado na Figura 22.

```
<!ELEMENT report (
  xmlSettings        |
  comment?           |
  reportHtmlEscapes  |
  data               |
  layout             |
  parameterForm      |
  programUnits       |
  attachedLibrary    |
  destination*       |
  customize          |
  webSource          |
  colorPalette       |
  reportPrivate      |
  accessibility      |
  documentTaxonomy   |
  userStyle          |
  reportWebSettings
  )* >
```

Figura 22: Elemento *report*
Fonte: Elaborado pelo autor

Os elementos mais importantes são os elementos *data* e *layout.* Estes elementos serão explicados a seguir.

3.1.1 Elemento *data*

A Figura 23 apresenta a estrutura do elemento *data,* onde localiza-se a definição do *Data Model Object* apresentado na subseção 2.2.1. A seguir, serão apresentados os elementos que compõem o elemento data.

```
<!ELEMENT data (
  dataSource        |
  crossProduct      |
  userParameter     |
  systemParameter   |
  summary           |
  formula           |
  placeholder       |
  link
  )* >
```

Figura 23: Elemento *data*
Fonte: Elaborado pelo autor

```
<!ELEMENT dataSource (
  (select | plugin | plsqlStatement),
  comment?,
  displayInfo?,
  formula*,
  group*
  ) >
  <!ATTLIST dataSource
    name                          CDATA   #IMPLIED
    defaultGroupName              CDATA   #IMPLIED
    maxRows                       CDATA   #IMPLIED
  >
```

Figura 24: Elemento *dataSource*
Fonte: Elaborado pelo autor

A Figura 24 mostra a estrutura do *dataSource* que corresponde a cada *query* criada no relatório. Os dois principais elementos desta estrutura são o *select* e o *group.* O elemento *select,* apresentado na Figura 25, define o comando SQL da *query*, enquanto o elemento *group* (Figura 26), define cada grupo criado na *query*.

```
<!ELEMENT select (#PCDATA)>
  <!ATTLIST select
    canParse                       (yes | no) "yes"
    externalQueryFile             CDATA   #IMPLIED
  >
```

Figura 25: Elemento *select*
Fonte: Elaborado pelo autor

```
<!ELEMENT group (
  field        |
  exception    |
  rowDelimiter |
  xmlSettings  |
  displayInfo  |
  dataItem     |
  formula      |
  summary      |
  placeholder  |
  filter       |
  comment
  )* >
  <!ATTLIST group
    name                    CDATA  #IMPLIED
    fillColor               CDATA  #IMPLIED
    lineColor               CDATA  #IMPLIED
    formatTrigger           CDATA  #IMPLIED
  >
```

Figura 26: Elemento *group*
Fonte: Elaborado pelo autor

O elemento *dataItem* deve ser adicionado para cada coluna retornada pela consulta que fazem parte do grupo. Seus principais atributos são explicados na Tabela 1.

Tabela 1: Atributos do *dataItem*
Fonte: autor

Atributo	Descrição
name	O nome da coluna retornada no SQL da query.
datatype	Tipo do dado da coluna, valores possíveis são: *character*, *number*, *date*, entre outros.

Os elementos *formula* e *placeholder* são usados para criar colunas adicionais onde o seu valor é o resultado de uma função PL/SQL, seus principais atributos são apresentados na Tabela 2.

Tabela 2: Atributos do *formula* e *placeholder*
Fonte: autor

Atributo	Descrição
name	Nome da coluna, é o nome que será referenciado por outros elementos.
source	Nome da função PL/SQL.

O elemento *summary* é usado para criar colunas adicionais com base em outras colunas utilizando funções pré-definidas, como, por exemplo, colunas para totais. Os atributos que definem este elemento são explicados na Tabela 3.

Tabela 3: Atributos do *summary*
Fonte: autor

Atributo	Descrição
name	Nome da coluna, é o nome que será referenciado por outros elementos.
source	Nome da coluna que será executada a função .
function	Função que será executada, valores possíveis: • *sum* • *average* • *median* • *minimum* • *maximum* • *count* • *first* • *last* • *percentOfTotal* • *percentile* • *stdDeviation* • *variance* A função padrão é *sum*.

O elemento *filter* é o que define o filtro utilizado no grupo, com os atributos mostrados na Tabela 4.

Tabela 4: Atributos do *filter*
Fonte: autor

Atributo	Descrição
type	Tipo do filtro, pode ser: • plsql • first • last Se for *plsql*, deve ser informado *plsqlFilter*, senão deve-se informar *numberOfRecords*. O padrão para este atributo é *first*.
plsqlFilter	Nome da função PL/SQL que será executada para realizar o filtro.
numberOfRecords	Número de registros que serão retornados.

O elemento *userParameter* apresentado na Figura 27, é utilizado para definição de parâmetros de entrada do relatório, seus principais atributos são explicados na Tabela 5.

```
<!ELEMENT userParameter (
  comment?, listOfValues?
  ) >
  <!ATTLIST userParameter
    name                          CDATA    #REQUIRED
    datatype                      (number | character | date) "number"
    pluginClass                   CDATA    #IMPLIED
    width                         CDATA    "20"
    scale                         CDATA    "0"
    precision                     CDATA    "0"
    initialValue                  CDATA    #IMPLIED
    inputMask                     CDATA    #IMPLIED
    validationTrigger             CDATA    #IMPLIED
    label                         CDATA    #IMPLIED
    defaultWidth                  CDATA    #IMPLIED
    defaultHeight                 CDATA    #IMPLIED
    templateSection               (none | header | main | trailer) "none"
    display                       (yes | no) "yes"
>
```

Figura 27: Elemento *userParameter*
Fonte: Elaborado pelo autor

Tabela 5: Atributos do *userParameter*
Fonte: autor

Atributo	Descrição
name	Nome do parâmetro. Este nome é utilizado para referenciá-lo quando utilizado.
datatype	Tipo do dado do parâmetro, os valores possíveis são: *character*, *number*, *date.*

A Figura 28 apresenta um exemplo simplificado do elemento *data,* na forma de um documento XML.

```
<data>
  <dataSource name="Q_1">
    <select><![CDATA[select d.department_id, d.department_name,
                        e.employee_id, e.first_name, e.salary
                 from employees e, departments d
                where e.department_id = d.department_id ]]>
    </select>
    <group name="G_DEPARTMENT_ID">
      <dataItem name="department_id" datatype="number" />
      <dataItem name="department_name" datatype="character" />
    </group>
    <group name="G_EMPLOYEE_ID">
      <dataItem name="employee_id" datatype="number" />
      <dataItem name="first_name" datatype="character" />
      <dataItem name="salary" datatype="number" />
      <summary name="SUM_SALARY_DEPT" source="SALARY" function="sum" />
    </group>
  </dataSource>
</data>
```

Figura 28: Exemplo elemento *data*
Fonte: Elaborado pelo autor

3.1.2 Elemento *layout*

Para criação do *Layout Object* é informado o elemento *layout,* nele são adicionados elementos *section*. Este elemento é apresentado na Figura 29 e seus atributos na Tabela 6.

```
<!ELEMENT section (
  (tabular              |
   groupAbove           |
   groupLeftInsideGroupAbove |
   groupLeft            |
   matrix               |
   formLike             |
   field)*,
  destination*,
  body?,
  margin?
  ) >
  <!ATTLIST section
    name                          CDATA   #IMPLIED
    width                         CDATA   #IMPLIED
    height                        CDATA   #IMPLIED
    widthInChar                   CDATA   #IMPLIED
    heightInChar                  CDATA   #IMPLIED
    horizontalPanelPerPage        CDATA   #IMPLIED
    verticalPanelPerPage          CDATA   #IMPLIED
    orientation                   (portrait | landscape | default) "default"
    repeatOn                      CDATA   #IMPLIED
  >
```

Figura 29: Elemento *section*
Fonte: Elaborado pelo autor

Tabela 6: Atributos do *section*
Fonte: autor

Atributo	Descrição
name	Este atributo define a qual das três seções corresponde a descrição do elemento, os valores possíveis são: *header, main* e *trailer.*
width	Define a largura da página para a seção.
height	Define a altura da página para a seção.
orietation	Define a orientação da página *portrait ou landscape,* ou seja, retrato ou paisagem.

Cada seção é composta por dois elementos principais: o *body* (Figura 30) que corresponde ao corpo da página e o *margin* (Figura 31) que é a margem da página. Nestes elementos são adicionados os componentes. Existem alguns elementos comuns a todos componentes, os principais são: *geometryInfo, visualSettings e generalLayout,* suas estruturas são apresentadas nas Figuras 32, 33 e 34.

```
<!ELEMENT body (
  location          |  frame            |  repeatingFrame   |
  field             |  text             |  polygon          |
  polyline          |  rectangle        |  line             |
  linkFile          |  arc              |  roundedRectangle |
  image             |  graph            |  matrix           |
  formField
  )* >
```

Figura 30: Elemento *body*

Fonte: Elaborado pelo autor

```
<!ELEMENT margin (
  frame             |  repeatingFrame   |  field            |
  text              |  polyline         |  polygon          |
  rectangle         |  line             |  linkFile         |
  arc               |  image            |  roundedRectangle |
  graph             |  matrix           |  formField
  )* >
```

Figura 31: Elemento *margin*

Fonte: Elaborado pelo autor

```
<!ELEMENT geometryInfo EMPTY>
  <!ATTLIST geometryInfo
    x                                   CDATA   #IMPLIED
    y                                   CDATA   #IMPLIED
    width                               CDATA   #IMPLIED
    height                              CDATA   #IMPLIED
  >
```

Figura 32: Elemento *geometryInfo*

Fonte: Elaborado pelo autor

```
<!ELEMENT visualSettings (
  visualSettingsPrivate
  )* >
  <!ATTLIST visualSettings
    lineWidth                   CDATA   #IMPLIED
    fillPattern                 CDATA   #IMPLIED
    fillForegroundColor         CDATA   #IMPLIED
    fillBackgroundColor         CDATA   #IMPLIED
    linePattern                 CDATA   #IMPLIED
    lineForegroundColor         CDATA   #IMPLIED
    lineBackgroundColor         CDATA   #IMPLIED
    dash                        (solid | dot | shortDash | dashDot | doubleDot
                                | longDash  | dashDoubleDot) "solid"
    capStyle                    (butt | round | projecting) "butt"
    joinStyle                   (mitre| round | bevel ) "mitre"
    hideLeftBorder              (yes | no)  "no"
    hideRightBorder             (yes | no)  "no"
    hideTopBorder               (yes | no)  "no"
    hideBottomBorder            (yes | no)  "no"
  >
```

Figura 33: Elemento *visualSettings*

Fonte: Elaborado pelo autor

```
<!ELEMENT generalLayout (
  conditionalFormat?
  ) >
  <!ATTLIST generalLayout
    pageBreakBefore             (yes | no) "no"
    pageBreakAfter              (yes | no) "no"
    pageProtect                 (yes | no) "no"
    verticalElasticity          (expand | contract | variable | fixed) #IMPLIED
    horizontalElasticity        (expand | contract | variable | fixed) #IMPLIED
    minimumWindowLines              CDATA   #IMPLIED
  >
```

Figura 34: Elemento *generalLayout*

Fonte: Elaborado pelo autor

O elemento *geometryInfo* é utilizado para definir o posicionamento do componente, o *visualSettings* serve para definir algumas opções de visualização, como, por exemplo, cor de fonte, cor do fundo, tamanho da borda, entre outras, e o *generalLayout* tem a função de definir o comportamento do componente durante a execução. Nas Tabelas 7, 8 e 9 são explicados os principais atributos destes elementos.

Tabela 7: Atributos do *geometryInfo*
Fonte: autor

Atributo	Descrição
x	Posição horizontal em relação à seção.
y	Posição vertical em relação à seção.
width	Largura do componente.
height	Altura do componente.

Tabela 8: Atributos do *visualSettings*
Fonte: autor

Atributo	Descrição
lineWidth	Largura das bordas.
fillBackgroundColor	Cor do fundo.
lineForegroundColor	Cor da borda.
hideLeftBorder	Ocultar borda esquerda.
hideRightBorder	Ocultar borda direita.
hideTopBorder	Ocultar borda superior.
hideBottomBorder	Ocultar borda inferior.

Tabela 9: Atributos do *generalLayout*
Fonte: autor

Atributo	Descrição
pageBreakBefore	Quebrar página antes de imprimir componente.
pageBreakAfter	Quebrar página após de imprimir componente.
verticalElasticity	Elasticidade vertical do componente, caso seu conteúdo ocupe mais espaço do que o definido.
horizontalElasticity	Elasticidade horizontal do componente, caso seu conteúdo ocupe mais espaço do que o definido.

A Figura 35 mostra a estrutura do elemento *frame*, que tem como função agrupar outros componentes.

```
<!ELEMENT frame (
  comment           |  geometryInfo      |  visualSettings    |
  webSettings       |  generalLayout     |  advancedLayout    |
  accessibility     |  frame             |  repeatingFrame    |
  field             |  text              |  polyline          |
  polygon           |  rectangle         |  graph             |
  line              |  linkFile          |  arc               |
  roundedRectangle  |  graph             |  image             |
  formField
  )* >
  <!ATTLIST frame
    name                  CDATA   #IMPLIED
    templateSection       (none | header | main | trailer) "none"
  >
```

Figura 35: Elemento *frame*
Fonte: Elaborado pelo autor

A Figura 36 mostra a estrutura do elemento *repeatingFrame*, que é utilizado para repetir os registros dos grupos, e a Tabela 10 apresenta a descrição dos seus atributos.

```
<!ELEMENT repeatingFrame (
  comment           |  geometryInfo      |  visualSettings    |
  webSettings       |  generalLayout     |  advancedLayout    |
  accessibility     |  frame             |  repeatingFrame    |
  field             |  text              |  polyline          |
  polygon           |  rectangle         |  graph             |
  line              |  linkFile          |  arc               |
  roundedRectangle  |  graph             |  image
  )* >
  <!ATTLIST repeatingFrame
    name                                 CDATA   #IMPLIED
    source                               CDATA   #IMPLIED
    printDirection                       CDATA   #IMPLIED
    maxRecordsPerPage                    CDATA   #IMPLIED
    minWidowRecords                      CDATA   #IMPLIED
    columnMode                           CDATA   #IMPLIED
    horizSpaceBetweenFrames              CDATA   #IMPLIED
    vertSpaceBetweenFrames               CDATA   #IMPLIED
    templateSection                     (none | header | main | trailer) "none"
  >
```

Figura 36: Elemento *repeatingFrame*
Fonte: Elaborado pelo autor

Tabela 10: Atributos do *geometryInfo*
Fonte: autor

Atributo	Descrição
source	Nome do grupo vinculado para repetição.
maxRecordsPerPage	Número máximo de registros apresentados em uma mesma página.
horizSpaceBetweenFrames	Espaçamento horizontal entre as repetições.
vertSpaceBetweenFrames	Espaçamento vertical entre as repetições.

O elemento *field,* que é utilizado para apresentar os dados do relatório, tem

sua estrutura apresentada na Figura 37, e seus principais atributos explicados na Tabela 11.

```
<!ELEMENT field (
  (labelAttribute?, exception*) |
  ((font             |   comment          |   geometryInfo     |
   visualSettings    |   webSettings      |   generalLayout    |
   advancedLayout    |   accessibility)*,
   pageNumbering?)
  ) >
```

Figura 37: Elemento *field*
Fonte: Elaborado pelo autor

Tabela 11: Atributos do *field*

Atributo	Descrição
source	Nome da coluna ou parâmetro que será apresentado.
alignment	Alinhamento do texto.
formatMask	Máscara para formatação do campo.

Para criação de textos usa-se o elemento *text* seguido de *textSegment* e *string,* conforme estrutura mostrada na Figura 38.

```
<!ELEMENT text (
  comment        |  textSettings    |  geometryInfo    |
  visualSettings |  webSettings     |  generalLayout   |
  advancedLayout |  accessibility   |  points          |
  textSegment
  )* >

<!ELEMENT textSegment (
  font |  string
  )* >

<!ELEMENT string (#PCDATA)>
```

Figura 38: Elemento *text*
Fonte: Elaborado pelo autor

Para criação de linhas e retângulos, utiliza-se os elementos *line* e *rectangle*, como mostra a Figura 39.

```
<!ELEMENT line (
  comment        |  geometryInfo    |  visualSettings |
  points         |  webSettings     |  generalLayout  |
  advancedLayout |  accessibility
  )* >

<!ELEMENT rectangle (
  comment        |  geometryInfo    |  visualSettings |
  points         |  webSettings     |  generalLayout  |
  advancedLayout |  accessibility
  )* >
```

Figura 39: Elemento *line* e *rectangle*
Fonte: Elaborado pelo autor

A Figura 40 apresenta um exemplo simplificado do elemento *layout,* na forma de um documento XML e na Figura 41 pode-se visualizar como ficaria após a execu-

ção deste relatório.

```
<layout>
  <section name="main">
    <body>
      <frame name="M_1">
        <geometryInfo x="0.18750" y="0.31250" width="3.06250" height="0.37500"/>
        <generalLayout verticalElasticity="variable"/>
        <text name="B_1">
          <geometryInfo x="0.18750" y="0.31250" width="0.33337" height="0.16663"/>
          <textSegment><string><![CDATA[Nome]]></string></textSegment>
        </text>
        <text name="B_2">
          <geometryInfo x="1.75000" y="0.31250" width="0.58337" height="0.16663"/>
          <textSegment><string><![CDATA[Salário]]></string></textSegment>
        </text>
        <line name="B_3">
          <geometryInfo x="0.12500" y="0.37378" width="2.93750" height="0.00122" />
        </line>
        <repeatingFrame name="R_1" source="G_EMPLOYEE_ID">
          <geometryInfo x="0.18750" y="0.50000" width="3.06250" height="0.18750"/>
          <field name="F_1" source="EMPLOYEE_ID" alignment="left">
            <geometryInfo x="0.18750" y="0.50000" width="1.37500" height="0.18750"/>
          </field>
          <field name="F_2" source="SALARY" alignment="start">
            <geometryInfo x="1.75000" y="0.50000" width="1.50000" height="0.18750"/>
          </field>
        </repeatingFrame>
      </frame>
    </body>
  </section>
</layout>
```

Figura 40: Exemplo do elemento *layout*
Fonte: Elaborado pelo autor

Nome	Salário
Steven	24.000.00
Neena	17.000.00
Lex	17.000.00
Alexander	9.000.00
Bruce	6.000.00
David	4.800.00
Valli	4.800.00
Diana	4.200.00
Nancy	12.000.00
Daniel	9.000.00
John	8.200.00
Ismael	7.700.00

Figura 41: Exemplo executado
Fonte: Elaborado pelo autor

3.2 JASPERREPORTS

O arquivo XML que define o relatório Jasper tem por padrão a extensão .jrxml.

O formato JRXML é definido por um XML SCHEMA, o XSD pode ser obtido através da URL http://jasperreports.sourceforge.net/xsd/jasperreport.xsd. Nesta seção serão apresentados os principais elementos deste XML. O elemento raiz é o *jasperReport*, sua estrutura é apresentada na Figura 42 e seus atributos são explicados na Tabela 12.

```
<element name="jasperReport">
 <complexType>
  <sequence>
   <element ref="jr:property" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:import" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:template" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:reportFont" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:style" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:subDataset" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:scriptlet" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:parameter" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:queryString" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:field" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:sortField" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:variable" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:filterExpression" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:group" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:background" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:title" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:pageHeader" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:columnHeader" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:detail" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:columnFooter" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:pageFooter" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:lastPageFooter" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:summary" minOccurs="0" maxOccurs="1"/>
   <element ref="jr:noData" minOccurs="0" maxOccurs="1"/>
  </sequence>
 </complexType>
</element>
```

Figura 42: Elemento *jasperReport*
Fonte: Elaborado pelo autor

Tabela 12: Atributos do *jasperReport*
Fonte: autor

Atributo	Descrição
name	Nome do relatório.
pageWidth	Largura da página.
pageHeight	Altura da página.
orientation	Formado da página, retrato ou paisagem.
columnCount	Número de colunas criadas na página, usado para criar relatórios tipo etiqueta
columnWidth	Largura das colunas
columnSpacing	Espaçamento entre as colunas
leftMargin	Margem esquerda.
rightMargin	Margem direita.
topMargin	Margem superior
bottomMargin	Margem Inferior.
whenNoDataType	Indica qual o comportamento quando o relatório não possui dados.

Para a definição do *data source* utilizando SQL os elementos utilizados são *queryString* e *field*. Onde o *queryString* define o SQL do relatório e o *field* cada um dos campos retornados pelo SQL. As Figuras 43 e 44 apresentam estes elementos e a Tabela 13 explica os atributos do *field*.

```
<element name="field">
 <complexType>
  <sequence>
   <element ref="jr:property" minOccurs="0" maxOccurs="unbounded"/>
   <element ref="jr:fieldDescription" minOccurs="0" maxOccurs="1"/>
  </sequence>
  <attribute name="name" type="string" use="required" />
  </attribute>
  <attribute name="class" type="string" use="optional" default="java.lang.String" />
 </complexType>
</element>
```

Figura 43: Elemento *field*
Fonte: Elaborado pelo autor

```
<element name="queryString">
 <complexType mixed="true">
  <attribute name="language" type="string" use="optional" default="sql" />
 </complexType>
</element>
```

Figura 44: Elemento *queryString*
Fonte: Elaborado pelo autor

Tabela 13: Atributos do *field*
Fonte: autor

Atributo	Descrição
name	Nome do campo. Deve ser o mesmo nome retornado pelo SQL, este nome será utilizado para a criação das expressões.
class	Tipo para o valor do campo. Deve ser informado a classe Java correspondente ao tipo do campo, por exemplo, para um campo VARCHAR informar java.lang.String, para um campo NUMBER informar java.lang.Integer ou java.math.BigDecimal e assim por diante.

A Figura 45 mostra um exemplo de um XML com a definição para um *data source.*

```
<jasperReport name="Exemplo">
    <queryString language="SQL">
        <![CDATA[SELECT employee_id,
                        first_name,
                        email,
                        hire_date,
                        salary,
                    FROM employees]]>
    </queryString>
    <field name="EMPLOYEE_ID" class="java.lang.Integer" />
    <field name="FIRST_NAME" class="java.lang.String" />
    <field name="EMAIL" class="java.lang.String" />
    <field name="HIRE_DATE" class="java.util.Date" />
    <field name="SALARY" class="java.math.BigDecimal" />
</jasperReport>
```

Figura 45: Exemplo de XML para *data source*
Fonte: Elaborado pelo autor

Os parâmetros são adicionados no relatório através do elemento *parameter*, ele possui dois atributos principais o *name*, que define o nome do parâmetro e *class* que define o tipo do parâmetro.

As variáveis são criadas com elemento *variable*, seus atributos são explicados

na Tabela 14.

Tabela 14: Atributos do *variable*
Fonte: autor

Atributo	Descrição
name	Nome da variável.
class	Classe Java do tipo da variável.
calculation	Cálculo executado pela variável. Pode ser , *Count*, *Sum*, *Average,* entre outros.
resetType	Define em qual nível a variável mantém o valor do cálculo. Por exemplo, se informado *Group*, para cada quebra do grupo a variável é reinicializada, se for *Page*, é reinicializada a cada nova página.
resetGroup	Define em qual grupo estará vinculada a reinicialização da variável, quando *resetType*=”*Group*”.

Para a definição das expressões não há apenas um elemento, existe um elemento específico para cada situação, mas todos têm o nome terminado com “Expression”. A Tabela 15 apresenta alguns elementos para definição de expressões.

Tabela 15: Elementos para expressões
Fonte: autor

Atributo	Descrição
variableExpression	Expressão usada pela variável.
groupExpression	Expressão que define a quebra de grupo.
defaultValueExpression	Expressão que define o valor padrão um parâmetro.
textFieldExpression	Expressão do componente *Text Field.*
printWhenExpression	Expressão booleana que indica se a banda ou componente será ou não impresso.

A Figura 46 mostra um trecho de XML com exemplo apresentando o uso de parâmetros, variáveis e expressões.

```
<parameter name="SALARIO_INICIAL" class="java.lang.Double">
    <defaultValueExpression><![CDATA[new Double(0.0)]]></defaultValueExpression>
</parameter>
<parameter name="SALARIO_FINAL" class="java.lang.Double"/>
<variable name="SUM_SALARIO" class="java.lang.Double" calculation="Sum">
    <variableExpression><![CDATA[$F{SALARIO}]]></variableExpression>
</variable>
```

Figura 46: Exemplo com parâmetros, variáveis e expressões
Fonte: Elaborado pelo autor

As bandas de visualização possuem cada uma um elemento. Estes elementos não possuem nenhum atributo e são seguidos de apenas um elemento: o *band*, no qual são adicionados os elementos correspondentes aos componentes visuais.

Os grupos são criados com o elemento *group* seguidos do *groupExpression* e das duas bandas possíveis para cada grupo: *groupHeader* e *groupFooter*.

Os componentes visuais possuem algumas propriedades comuns, e estas informações são centralizadas em um elemento, que é colocado em cada um dos componentes, este elemento é o *reportElement,* seus atributos são explicados na Tabela 16, e é nele que se informa o elemento *printWhenExpression* visto na Tabela 15.

Tabela 16: Atributos do *reportElement*
Fonte: autor

Atributo	Descrição
x	Especifica a coordenada x do componente dentro da banda.
y	Especifica a coordenada y do componente dentro da banda.
width	Define a largura do componente.
height	Define a altura do componente.
forecolor	Cor do primeiro plano, é usada como cor da fonte em componentes de texto, como cor das linhas e das bordas em componentes como retângulos e círculos.
backcolor	Cor do fundo.
mode	Indica se o fundo do componente será transparente ou opaco.
positionType	Define o posicionamento do componente conforme a banda aumenta sua altura.

stretchType	Define o comportamento para elasticidade do componente.
isPrintRepeatedValues	Indica se o componente será impresso repedidas vezes com o mesmo valor.

Para a apresentação dos dados do relatório é utilizado o elemento *textField*, nele é informada a expressão para obtenção do dado através do elemento *textFieldExpression.* Para apresentação de textos estáticos pode ser utilizado o elemento *staticText* seguido do elemento *text.*

Na Figura 47 pode ser visualizado o trecho do XML de um relatório apresentando a estrutura das bandas, e na Figura 48 apresentando o conteúdo de algumas das bandas. Na Figura 49 pode-se visualizar este exemplo após a execução do relatório.

```
<group name="Departamento">
    <groupExpression><![CDATA[$F{DEPARTMENT_ID}]]></groupExpression>
    <groupHeader>
        <band height="20">
    </groupHeader>
    <groupFooter>
        <band height="20">
    </groupFooter>
</group>
<columnHeader>
    <band height="41" splitType="Stretch">
</columnHeader>
<detail>
    <band height="22" splitType="Stretch">
</detail>
<summary>
    <band height="42" splitType="Stretch">
</summary>
```

Figura 47: Exemplo com a estrutura das bandas
Fonte: Elaborado pelo autor

```
<columnHeader>
    <band height="29" splitType="Stretch">
        <staticText>
            <reportElement x="40" y="7" width="100" height="20"/>
            <text><![CDATA[Nome]]></text>
        </staticText>
        <staticText>
            <reportElement x="184" y="9" width="100" height="20"/>
            <text><![CDATA[Salário]]></text>
        </staticText>
        <line>
            <reportElement x="19" y="23" width="488" height="1"/>
        </line>
    </band>
</columnHeader>
<detail>
    <band height="22" splitType="Stretch">
        <textField>
            <reportElement x="40" y="2" width="100" height="20"/>
            <textFieldExpression class="java.lang.String">
                <![CDATA[$F{FIRST_NAME}]]>
            </textFieldExpression>
        </textField>
        <textField>
            <reportElement x="184" y="2" width="100" height="20"/>
            <textFieldExpression class="java.math.BigDecimal">
                <![CDATA[$F{SALARY}]]>
            </textFieldExpression>
        </textField>
    </band>
</detail>
```

Figura 48: Exemplo com conteúdo das Bandas
Fonte: Elaborado pelo autor

	Nome	Salário
Departamento	Administração	
	Jennifer	4400,00
		4400,00
Departamento	Marketing	
	Michael	13000,00
	Pat	6000,00
		19000,00
Departamento	Compras	
	Den	11000,00
	Alexander	3100,00
	Shelli	2900,00

Figura 49: Exemplo executado
Fonte: Elaborado pelo autor

A Figura 50 apresenta a sequência de impressão das bandas no momento da execução do relatório, a banda *Title* é impressa uma vez no inicio no relatório, as bandas *Page Header* e *Page Footer* são impressas a cada quebra de página, *Column Header* e *Column Footer* a cada quebra de coluna, *Group Header* e *Group Footer* são impressas a cada quebra do grupo e por fim a banda *Detail* que é impressa a cada registro processado.

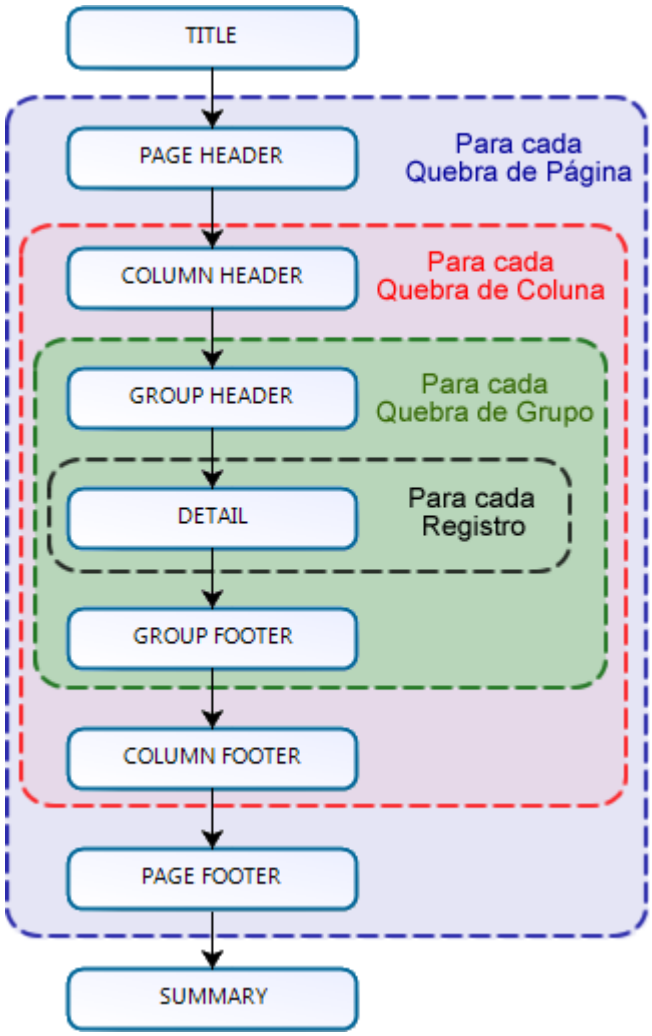


Figura 50: Sequência de impressão das bandas
Fonte: Elaborado pelo autor

3.3 CONSIDERAÇÕES FINAIS

Após estudado o comportamento dos arquivos das ferramentas Oracle Reports e JasperReports, pôde-se observar que a principal diferença na estrutura do XML está na maneira com que elas tratam a questão da criação dos grupos.

No Oracle Reports os grupos são criados junto ao modelo de dados, proporcionando a existência de uma separação do que é dado e do que é *layout,* utilizando para a apresentação dos dados de cada grupo componentes como *frames* e *repeatingFrames,* que são posicionados dentro da página, onde esses componentes fazem referências aos grupos do modelo de dados.

No JasperReports o modelo de dados, ou *data source*, mantém apenas a coleção dos dados que serão apresentados, e o *layout* possui áreas específicas para a apresentação dos dados, as bandas, e os grupos são criados diretamente no *layout* acrescentando a ele duas novas bandas: *groupHeader* e *groupFooter* a cada grupo adicionado.

Levando em consideração que a maioria dos relatórios desenvolvidos pela NL são dos tipos tabular e mestre/detalhe, possuindo apenas uma *query,* utilizando grupos para criação de mestre/detalhe, optou-se por trabalhar estes tipos de relatórios.

4 MODELAGEM DO CONVERSOR DE RELATÓRIOS

Para descrever o *software* proposto, serão apresentados nas seções seguintes, alguns artefatos utilizados para modelagem de *softwares*. Os artefatos envolvidos são: relação de requisitos, diagramas UML tais como, diagrama de casos de uso, diagrama de classes, e diagrama de arquitetura.

Neste capítulo,em que será apresentada a modelagem do *software* proposto, a seção 4.1 apresenta os requisitos do sistema utilizando-se o diagrama de casos de uso, na seção 4.2 são definidas as regras dos mapeamentos entre os elementos que compões os XMLs das duas ferramentas, na seção 4.3 é apresentado o modelo conceitual através do diagrama de classes e a seção 4.4 apresenta a arquitetura do *software.*

4.1 REQUISITOS DO SISTEMA

O sistema tem como objetivo converter relatórios desenvolvidos em Oracle Reports que possuam uma *query,* que são a maioria dos relatórios existentes hoje na NL, para relatórios que possam ser executados em Java utilizando a biblioteca JasperReports.

Os requisitos principais do *software* estão descritos na Tabela 17.

Tabela 17: Requisitos
Fonte: autor

Código	Descrição
R-001	O *software* deve converter relatórios desenvolvidos em Oracle Reports que possuem uma *query* para JasperReports.
R-002	O *software* deve possibilitar a seleção de múltiplos relatórios para conversão.
R-003	O *software* deve permitir a execução e a visualização dos relatórios convertidos.

Na Figura 51 estão relacionados os principais casos de uso do ator “Desen-

volvedor", único ator do sistema. Estes casos de uso refletem os requisitos relacionados na Tabela 17.

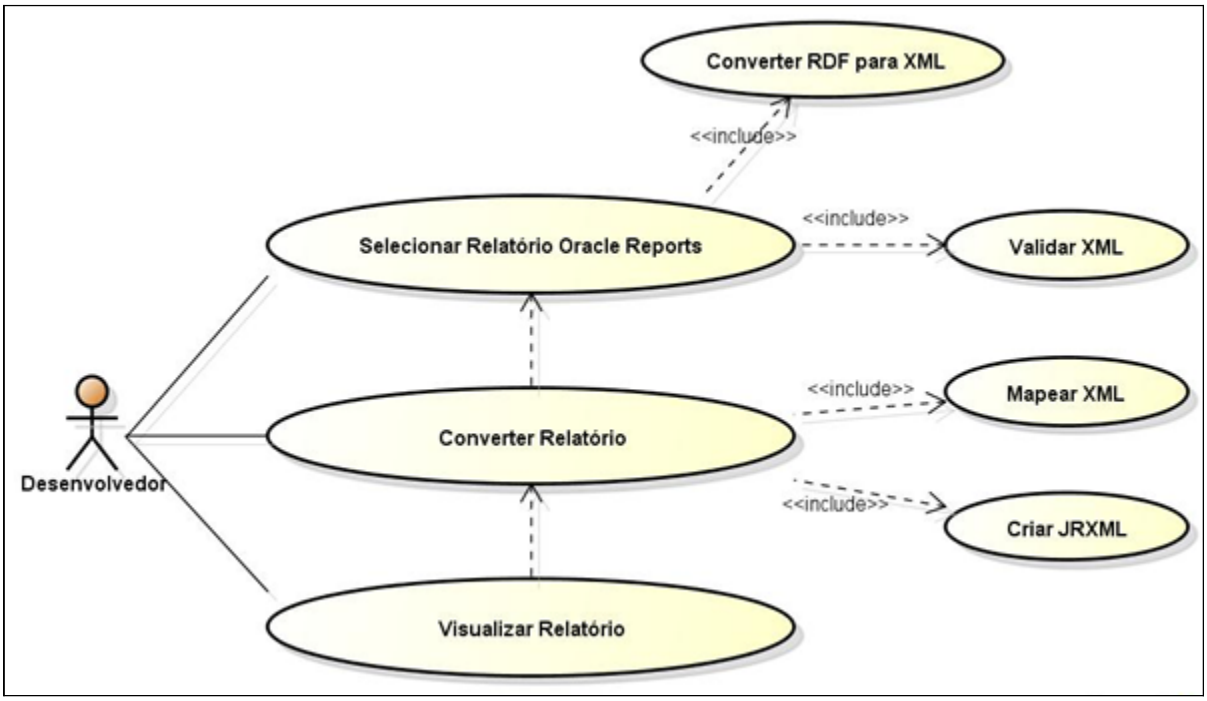


Figura 51: Casos de uso
Fonte: Elaborado pelo autor

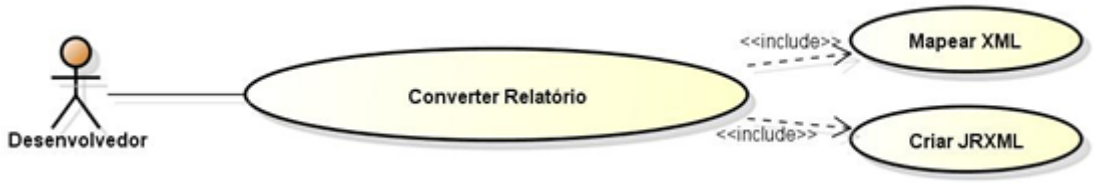
A seguir são explicados os casos de uso apresentados na Figura 51. A Figura 52 apresenta o caso de uso "Selecionar Relatórios Oracle Reports" que faz referência ao requisito R-001 da Tabela 17.

CASO DE USO 001 – Selecionar Relatórios Oracle Reports	
	
Descrição	Seleção dos relatórios do Oracle Reports a serem convertidos.
Ator	Desenvolvedor
Pré-condições	Instalação do Oracle Developer Suite para conversão do relatório em formato RDF (fonte do Oracle Reports em binário) para XML.
Fluxo Principal	1. Sistema apresenta tela de seleção de arquivos padrão do sistema operacional. 2. Desenvolvedor seleciona arquivo correspondente ao relatório, em formato RDF ou XML. 3. Sistema verifica se o arquivo informado é formato RDF, então

	executa comando para conversão de RDF para XML. 4. Sistema verifica se XML é um relatório Oracle Reports. 5. Sistema verifica se relatório possui apenas uma *query*. 6. Sistema apresenta arquivo selecionado em uma lista.
Fluxo alternativo e exceções	**Item 3:** 1. Arquivo selecionado é RDF e não possui programa de conversão. 1.1. Sistema exibe mensagem avisando que não é possível selecionar este tipo de arquivo. **Item 4:** 1. Não é um XML Oracle Reports 1.1. Sistema exibe mensagem informando que é um arquivo inválido. **Item 5:** 1. É um relatório que possui mais de um *query.* 1.1. Sistema exibe mensagem informando que é um relatório não previsto para conversão.
Pós-condições	Lista de relatórios para conversão.

Figura 52: Caso de uso "Selecionar Relatórios Oracle Reports"
Fonte: Elaborado pelo autor

A Figura 53 apresenta o caso de uso “Converter Relatório” que remete ao requisito R-002.

CASO DE USO 002 – Converter Relatório	
	
Descrição	Converter XML Oracle Reports para JRXML.
Ator	Desenvolvedor
Pré-condições	CASO DE USO 001
Fluxo Principal	1. Desenvolvedor solicita conversão dos relatórios 2. Sistema apresenta tela para selecionar o diretório destino. 3. Sistema processa cada relatório. 3.1. Analisa XML 3.2. Mapeia XML origem para XML destino de acordo com as regras definidas. 3.3. Cria JRXML destino 3.4. Cria arquivo de *log* com elementos que não foram convertidos. 3.5. Compila JRXML criando arquivo Jasper. 4. Sistema habilita opção para visualização dos relatórios

	convertidos. 5. Sistema habilita opção para visualização dos *logs*.
Pós-condições	Relatórios convertidos prontos para execução.

Figura 53: Caso de uso "Converter Relatório"
Fonte: Elaborado pelo autor

A Figura 54 apresenta o caso de uso “Visualizar Relatório” que representa o requisito R-003.

CASO DE USO 003 – Visualizar 	
Descrição	Executar e visualizar relatório convertido.
Ator	Desenvolvedor
Pré-condições	CASO DE USO 002
Fluxo Principal	1. Desenvolvedor solicita visualização do relatório. 2. Sistema apresenta tela para configurar conexão com SGBD. 3. Sistema apresenta tela para informar os parâmetros. 4. Sistema executa relatório e apresenta tela para visualização do relatório.
Fluxo alternativo e exceções	**Item 2:** 1. Conexão com SGBD não foi estabelecida. 1.1. Sistema apresenta mensagem e solicita novamente as informações da conexão. **Item 4:** 1. Ocorreram erros na execução do relatórios 1.1. Sistema apresenta mensagem do erro para usuário.
Pós-condições	Relatório apresentado na tela.

Figura 54: Caso de uso "Visualizar"
Fonte: Elaborado pelo autor

Para conversão de arquivo RDF, que é arquivo fonte em formato binário do Oracle Reports, para XML, como mostra o CASO DE USO 001, deverá ser utilizado o programa *rwconverter* disponibilizado na instalação do *Oracle Developer Suite*.

A validação do XML selecionado no CASO DE USO 001, que fará a verificação se o XML é um relatório e que este relatório possui apenas uma *query,* se dará realizando a verificação se o elemento *root* do XML é o elemento *report* e este possui os elementos *data* e *layout*, e dentro do *data* deverá ser encontrado apenas uma ocorrência do elemento *dataSource*.

4.2 REGRAS DE MAPEAMENTO

Para a conversão dos relatórios Oracle para Jasper, como apresentado no CASO DE USO 002, será realizado o mapeamento dos elementos do XML do Oracle Reports para elementos correspondentes no XML do JasperReports. Este mapeamento é apresentado a seguir.

4.2.1 Modelo de dados

O modelo de dados dentro do XML do Oracle Reports é encontrado no elemento *data*, esse elemento será analisado e mapeado para os elementos do Jasper conforme é explicado a seguir.

Dentro deste elemento se encontra os elementos *userParameter*, que são os parâmetros de entrada do relatório. Eles serão mapeados para o Jasper como o elemento *parameter*, levando em conta o tipo do dado que se encontra do atributo *datatype*.

No elemento *dataSource* será procurado o elemento *select*, que contém o SQL do relatório, ele será mapeado para o Jasper como o elemento *querySrtring*, e o comando SQL deverá ser alterado nas partes onde são usados parâmetros para o padrão do Jasper.

O elemento *data* será “varrido” em busca dos elementos *dataItem*, *summary, formula e placeholder,* que podem ser encontrados em diferentes níveis de grupos.

Os elementos *dataItem*, que representam cada campo da consulta, serão mapeados para o Jasper como elemento *field.*

Os elementos *summary* serão mapeados para variáveis do Jasper, onde o atributo *calculation* equivale ao *function*, e os atributos *resetType* e *resetGroup* serão obtidos de acordo com o nível dos grupos onde se encontra o elemento.

Os elementos *formula* e *placeholder* não serão convertidos, pois como são resultados de funções PL/SQL e o JasperReports não executa PL/SQL, a criação de uma *store procedure* no banco seria uma solução, mas para isso seria necessário analisar a fonte PL/SQL verificando as variáveis usadas e colocadas como parâmetros desta *store procedure.* Por isso será criada uma variável com valor fixo, para

que possa ser usada no *layout* e será acrescentado um aviso no arquivo de *log* para que o desenvolvedor possa fazer a funcionalidade manualmente.

A criação dos grupos não poderá ser realizada diretamente a partir de cada elemento *group* encontrado no *dataSource,* pois como os grupos no Jasper são criados junto ao *layout*, com isso, a criação de um grupo no Jasper está associada à existência de um *repeatingFrame* vinculado a um *group.* A criação da expressão que definirá em qual momento ocorrerão as quebras, que se dá através do elemento *groupExpression*, realizar-se-á criando uma expressão concatenando todos *fields* mapeados de cada *dataItem* pertencente ao *group* vinculado no *repeatingFrame.*

A Figura 55 apresenta este mapeamento em um exemplo com a estrutura básica dos XMLs que representam o modelo de dados.

```
<report>
  <data>
    <userParameter name="P_1" />
    <userParameter name="P_2" />
    <dataSource>
      <select />
      <group name="G_1">
        <dataItem name="C_1" />
        <dataItem name="C_2" />
        <summary name="S_1" />
        <formula name="F_1" />
      </group>
      <group name="G_2">
        <dataItem name="C_3" />
        <dataItem name="C_4" />
        <summary name="S_2" />
      </group>
      <summary name="S_3" />
    </dataSource>
  </data>
</report>
```

```
<jasperReport>
  <parameter name="P_1" />
  <parameter name="P_1" />
  <queryString />
  <field name="C_1" />
  <field name="C_2" />
  <field name="C_3" />
  <field name="C_4" />
  <variable name="S_1" />
  <variable name="S_2" />
  <variable name="S_3" />
  <variable name="F_1" />
  <group name="G_1">
    <groupExpression />
  </group>
  <group name="G_2">
    <groupExpression />
  </group>
</jasperReport>
```

Figura 55: Mapeamento XML do modelo de dados
Fonte: Elaborado pelo autor

4.2.2 Layout

A parte da visualização do relatório Oracle Reports é encontrada no XML dentro do elemento *layout*, esse elemento será analisado e mapeado para os elementos do Jasper conforme é explicado a seguir.

O *layout* do Oracle Reports é divido em três setores, são eles: *Header*, *Main* e *Trailer*. Os elementos que estiverem no setor *Header* deverão ser mapeados no Jasper para a banda *Title*, pois nas duas ferramentas estas partes são impressas apenas uma vez, no inicio do relatório. O mesmo ocorre com o Trailer, que é impresso também apenas uma vez, porém no final do relatório, e será mapeado no Jasper para a banda *Summary*.

O conteúdo do setor *Main* será mapeado no Jasper para as bandas, *Page Header, Page Footer, Column Header, Column Footer, Group Header, Group Footer* e *Detail*, de acordo com as regras a seguir:

- Os componentes que estiverem no elemento *margin* e estiverem posicionados acima das coordenadas do elemento *body* serão colocados na banda *Page Header,* já os que estiverem abaixo serão colocados na banda *Page Footer.*
- Os componentes que se encontram no elemento *body* serão distribuídos nas bandas *Column Header, Column Footer, Group Header*, *Group Footer* e *Detail* de acordo com os níveis dos grupos encontrados vinculados aos componentes *repeatingFrame.*
- Os componentes que serão colocados na banda *Detail* serão os componentes encontrados no *repeatingFrame* mais interno, que são os elementos que serão impressos a cada registro processado.
- Os componentes que não estão dentro de um *repeatingFrame* serão colocados no *Column Header* e *Column Footer*, os componentes que estão posicionados acima do primeiro *repeatingFrame* vão no *Column Header* e os que estão abaixo vão no *Column Footer.*
- Os componentes encontrados dentro do *repeatingFrame* serão colocados no *Group Header* e *Group Footer* levando em conta o posicionamento em relação ao próximo *repeatingFrame.*

A Figura 56 demonstra este mapeamento, mostrando onde ficariam os componentes em cada parte do relatório.

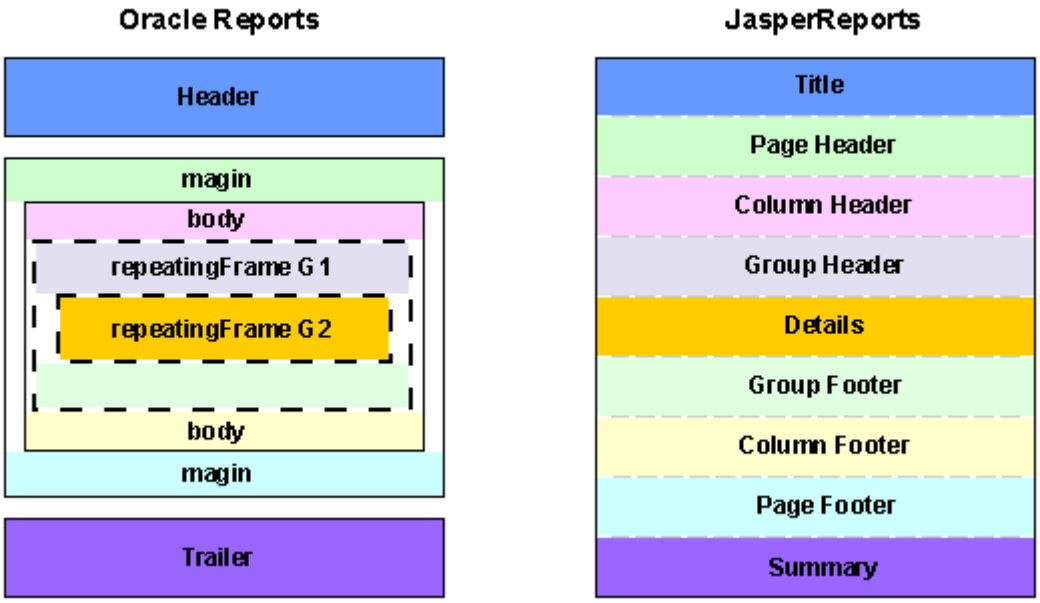


Figura 56: Mapeamento das partes que compõem o *layout*
Fonte: Elaborado pelo autor

Os elementos *text,* que são as caixas de texto estático, serão mapeados no Jasper para *staticText.* Os elementos *field,* que são as caixas de texto com conteúdo dinâmico vinculados a um campo do *dataSource,* serão mapeados para o elemento *textField* e será criada uma expressão através do elemento *textFieldExpression*, correspondente ao atributo *source* do elemento *text.*

As linhas adicionadas através do elemento *line* no Oracle Reports terão o mesmo elemento no Jasper, mas seguindo as suas configurações.

4.3 MODELO CONCEITUAL (DIAGRAMA DE CLASSES)

Como modelo conceitual para o *software* proposto será apresentado o diagrama de classes, onde mostra a interação entre as principais classes do *software.* Este diagrama pode ser visualizado na Figura 57 e na sequência é explicada a função de cada uma das classes.

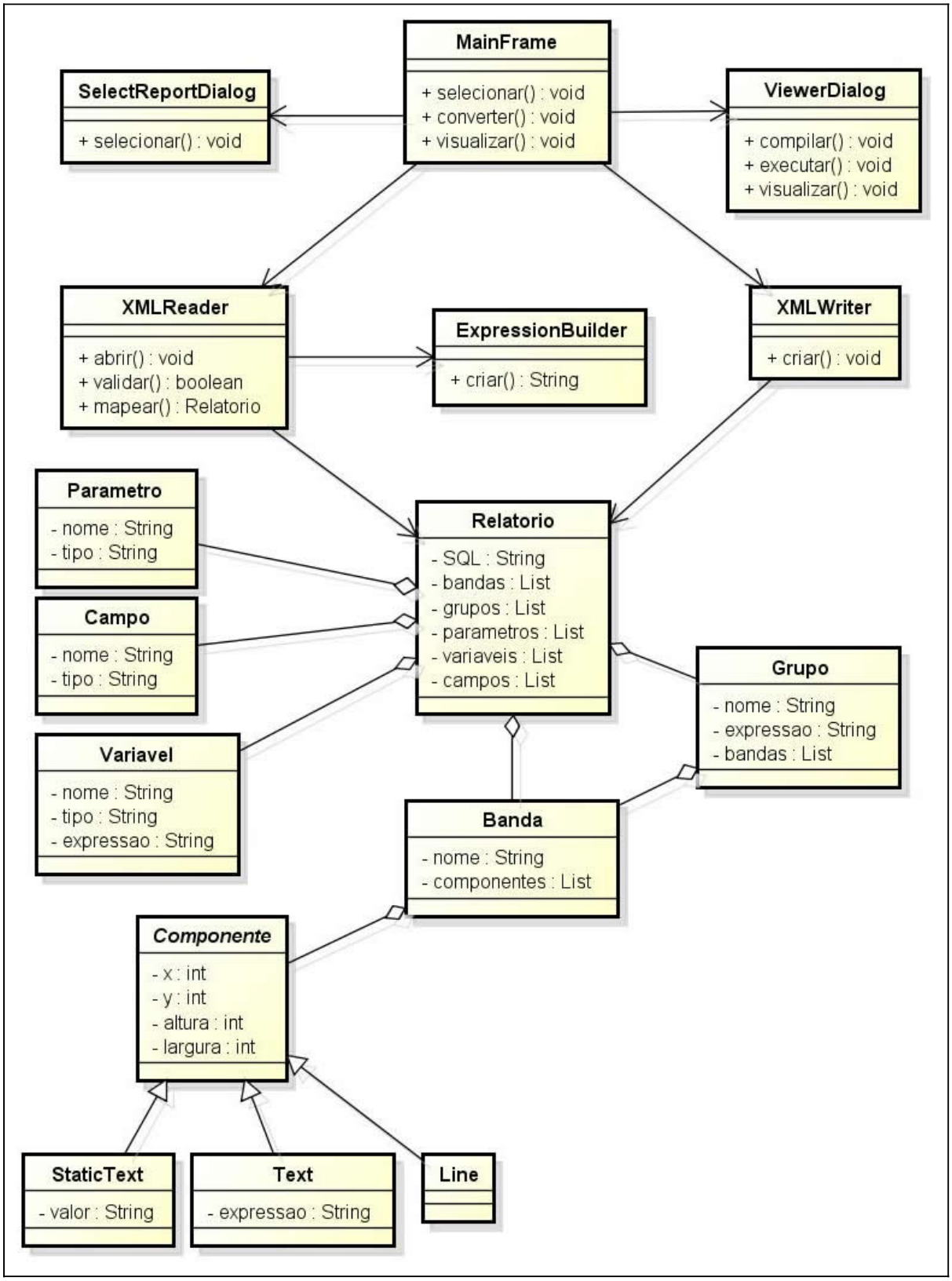


Figura 57: Diagrama de classes
Fonte: Elaborado pelo autor

A classe MainFrame é responsável pelo ponto de entrada do sistema. O usuário terá as opções de selecionar os relatórios que serão convertidos, invocar a execução da conversão, e após a possibilidade de executar e visualizar o relatório já convertido para JasperReports. Estes métodos representam os três casos de uso descritos na seção 4.1 .

A classe SelectReportDialog é a responsável por criar a janela para seleção dos relatórios, e a ViewerDialog será responsável pela visualização do relatório após a execução.

A classe XMLReader, pode ser considerada a principal classe do sistema pois é nela que ficará a responsabilidade de interpretar o XML de entrada e a partir das regras definidas anteriormente poder mapear os elementos. Esse mapeamento será realizado criando um objeto da classe Relatorio que contém as propriedades que compõem o relatório Jasper, que serão preenchidas conforme o XML é analisado. As classes definidas para representar o relatório são:

- Campo – representa cada campo retornado pela consulta.
- Parametro – representa os parâmetros de entrada nos relatórios.
- Variavel – representa as variáveis que serão criadas para suportar as funções.
- Grupo – representa cada grupo criado no relatório.
- Banda – representa cada uma das bandas do relatório Jasper.
- Componente – classe abstrata que representa os componentes visuais do relatório, ela possuirá os atributos comuns a todos componentes, como por exemplo, posicionamento e tamanho.
- StaticText, Text e Line – são os componentes especializados.

A classe ExpressionBuilder será responsável pela criação das expressões utilizadas como, por exemplo, expressões das variáveis, dos grupos, entre outras.

Após realizado todo o mapeamento e criado o objeto Relatorio, a classe XMLWriter será responsável pela criação do XML do JasperReports, o JRXML, finalizando o processo.

4.4 ARQUITETURA DE SOFTWARE

A arquitetura do *software* proposto está baseada na estrutura presente na NL, que hoje possui como SGBD o banco de dados da Oracle, onde são armazenados os dados que são utilizados pelas aplicações desenvolvidas pela NL. A aplicação atualmente considerada como principal produto da empresa, o NL Gestão ERP, que é desenvolvido em Oracle Forms e Oracle Reports está sendo substituída as poucos pela aplicação desenvolvida em Java, que utiliza como ferramenta de geração de relatórios o JasperReports. O *software* proposto neste trabalho entra como facilitador desta migração de tecnologia, buscando os relatórios já desenvolvidos na tecnologia Oracle Reports e os convertendo para tecnologia JasperReports. A Figura 58 apresenta esta arquitetura.

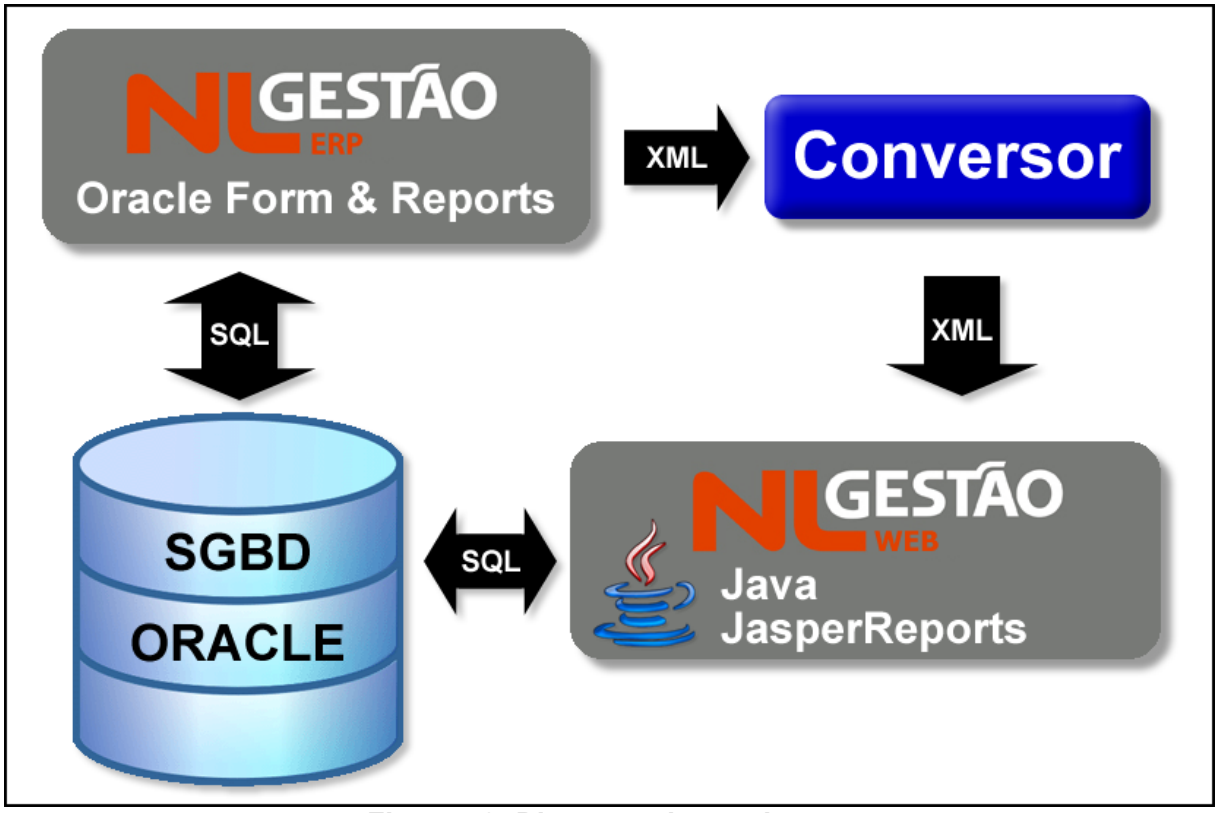


Figura 58: Diagrama de arquitetura
Fonte: Elaborado pelo autor

4.5 CONSIDERAÇÕES FINAIS

Este trabalho apresenta duas ferramentas de geração de relatórios, Oracle Reports e JasperReports, mostrando suas principais funcionalidades e explica como é armazenada a estrutura XML que descreve cada relatório, com o objetivo de analisar e entender como pode ser realizada a conversão de relatórios desenvolvidos em Oracle Reports para JasperReports.

Este capítulo apresentou a modelagem do *software* proposto, mostrando como ele deve se comportar para que possa realizar a tarefa. O próximo capítulo apresenta como o *software* foi desenvolvido, quais as ferramentas utilizadas, descrição dos artefatos criados, explicação de como foi implementado e a apresentação da ferramenta produzida.

5 DESENVOLVIMENTO

Neste capítulo serão apresentadas as ferramentas utilizadas para o desenvolvimento do *software,* as alterações realizadas na modelagem definida no Capítulo 4 apresentando os novos diagramas de classes e pacotes, a descrição de como foi realizada a implementação e a apresentação do *software.*

5.1 FERRAMENTAS UTILIZADAS

Para o desenvolvimento do *software* foi utilizada a linguagem de programação JAVA, como IDE foi escolhido o Eclipse, pela familiaridade, pois é a ferramenta utilizada na NL para o desenvolvimento dos seus produtos JAVA.

A instalação de alguns *softwares* foi necessária para criar o ambiente de desenvolvimento, como *Oracle Database 10g Express Edition*, que é a versão gratuita do SGBD Oracle. Neste SGBD foi criada uma cópia da estrutura do banco de dados utilizado pela NL. Para a realização dos testes, esse SGBD foi instalado em uma máquina virtual, utilizando o *VirtualBox.* Além do SGBD, também foi instalado o *Oracle Developer Suite* para utilização do Oracle Reports para edição dos relatórios testados. Outro *software* instalado foi o iReport, utilizado para a validação dos relatórios convertidos. Algumas bibliotecas foram necessárias para o desenvolvimento, são elas:

- Apache Commons: coleção de bibliotecas desenvolvidas pela Apache Software Foundation, fornecem funções que o Jasper utiliza para a compilação e a geração da visualização do relatório.
- Apache log4j: biblioteca usada para a geração dos *logs* da aplicação.
- OJDBC: implementação do JDBC para conexão com SGBD Oracle.

A interface com o usuário foi desenvolvida utilizando Swing, o parser XML utilizado foi o DOM, utilizando XPath para localização dos elementos. A Figura 59 apresentada essas ferramentas.

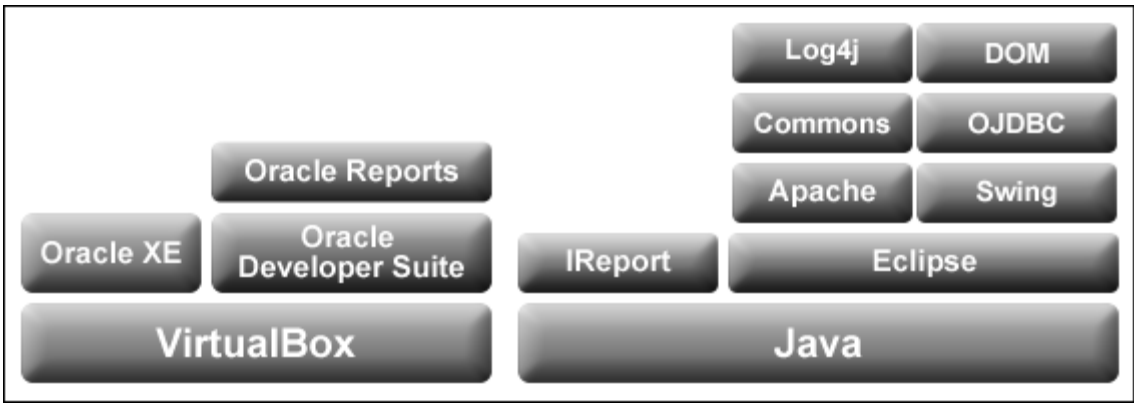


Figura 59: Ferramentas utilizadas
Fonte: autor

5.2 ALTERAÇÕES NA MODELAGEM

Durante o desenvolvimento foram detectadas algumas situações que implicaram na mudança da modelagem descrita no Capítulo 4, estas modificações serão apresentadas na sequência desta seção juntamente com a complementação da modelagem.

5.2.1 Requisitos do sistema

Nos casos de uso apresentados na seção 4.1 a alteração realizada foi quanto a localização dos subprocessos “Converter RDF para XML” e “Validar Xml”, que se encontravam no Caso de uso 1 e foram movidos para o Caso de uso 2. A Figura 60 mostra como o ficaram os casos de uso.

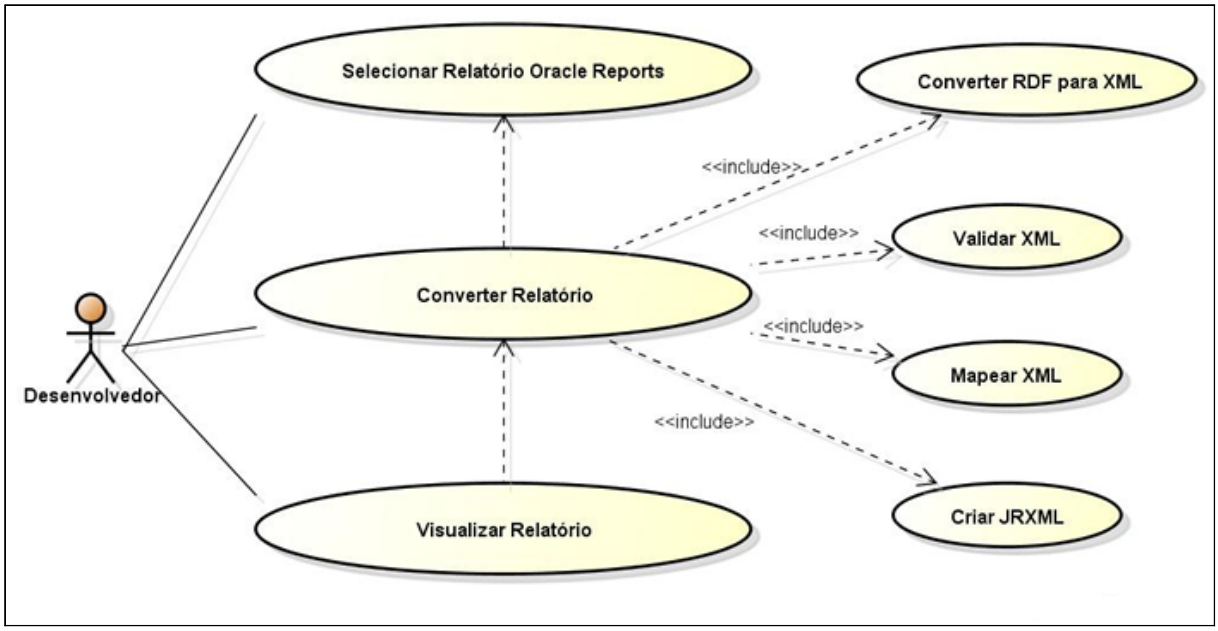


Figura 60: Casos de uso
Fonte: Elaborado pelo autor

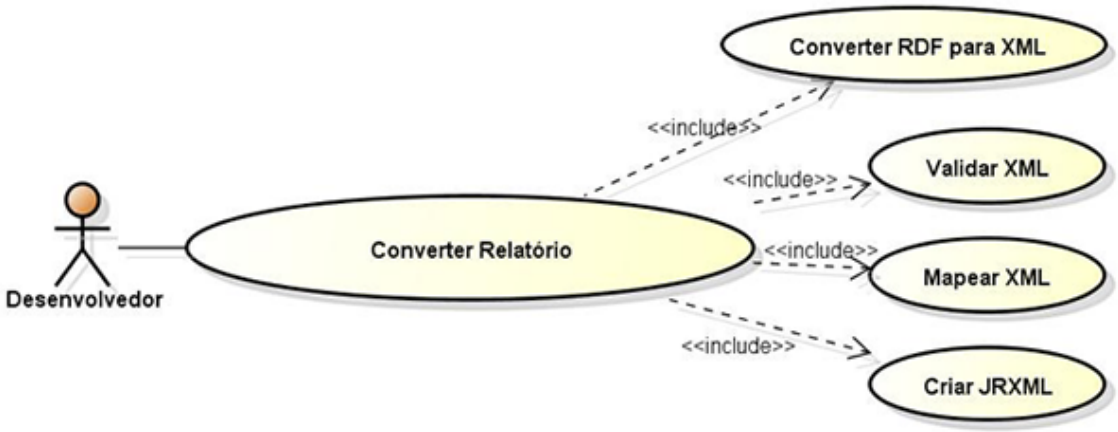
Os dois casos de uso alterados são explicados a seguir. A Figura 61 apresenta o caso de uso “Selecionar Relatórios Oracle Reports” alterado.

CASO DE USO 001 – Selecionar Relatórios Oracle Reports

Descrição	Seleção dos relatórios do Oracle Reports a serem convertidos.
Ator	Desenvolvedor
Pré-condições	Instalação do Oracle Developer Suite para conversão do relatório em formato RDF (fonte do Oracle Reports em binário) para XML.
Fluxo Principal	1. Sistema apresenta tela de seleção de arquivos padrão do sistema operacional. 2. Desenvolvedor seleciona arquivos correspondentes aos relatórios, em formato RDF ou XML. 3. Sistema apresenta arquivos selecionados em uma lista.
Pós-condições	Lista de relatórios para conversão.

Figura 61: Caso de uso "Selecionar Relatórios Oracle Reports"
Fonte: Elaborado pelo autor

A Figura 62 apresenta o caso de uso “Converter Relatório” alterado.

CASO DE USO 002 – Converter Relatório	
	
Descrição	Converter RDF/XML Oracle Reports para JRXML.
Ator	Desenvolvedor
Pré-condições	CASO DE USO 001
Fluxo Principal	1. Desenvolvedor solicita conversão dos relatórios selecionados. 2. Sistema apresenta tela para selecionar diretório destino. 3. Sistema processa cada relatório. 3.1. Sistema verifica se o arquivo informado é formato RDF, então executa comando para conversão de RDF para XML. 3.2. Sistema verifica se XML é um relatório Oracle Reports. 3.3. Sistema verifica se relatório possui apenas uma *query*. 3.4. Analisa XML 3.5. Mapeia XML origem para XML destino de acordo com as regras definidas. 3.6. Cria JRXML destino 3.7. Cria arquivo de *log* com elementos que não foram convertidos. 3.8. Compila JRXML criando arquivo Jasper. 4. Sistema habilita opção para visualização dos relatórios convertidos. 5. Sistema habilita opção para visualização dos *logs*.
Fluxo alternativo e exceções	**Item 3.1:** 1. Arquivo selecionado é RDF e não possui programa de conversão. 1.1. Sistema exibe mensagem e solicita configuração do caminho do programa de conversão. **Item 3.2:** 1. Não é um XML Oracle Reports 1.1. Sistema exibe mensagem informando que é um arquivo inválido. **Item 3.3:** 1. É um relatório que possui mais de um *query.* 1.1. Sistema exibe mensagem informando que é um

	relatório não previsto para conversão. **Todos itens:** 1. Na ocorrência de uma exceção. 1.1. Sistema apresenta mensagem e grava o erro no *log.*
Pós-condições	Relatórios convertidos prontos para execução.

Figura 62: Caso de uso "Selecionar Relatórios Oracle Reports"
Fonte: Elaborado pelo autor

5.2.2 Regras de Mapeamento

Na subseção 4.2.1, em relação à criação dos grupos, a alteração ocorrida foi que a criação do grupo é realizada a partir de cada grupo encontrado no *dataSource,* porém esse grupo só será visualizado se for encontrado o *repeatingFrame* correspondente.

Na subseção 4.2.2, a alteração ocorreu no reposicionamento dos componentes nas bandas, são elas:

- Os componentes que estiverem no elemento *margin* serão colocados na banda *background,* esta banda é impressa em todas as páginas, como se fosse uma estampa inicial, tendo assim o mesmo comportamento do Reports. As bandas *Page Header* e *Page Footer* somente são adicionadas com altura definida com a margem superior e inferior da página.
- Os componentes que não estão dentro de um *repeatingFrame* e estão abaixo do primeiro *repeatingFrame* serão colocados no *Summary*, pois estes elementos, no Reports, são impressos somente uma vez no final do relatório, o que é o comportamento do *Summary.* Enquanto o *Column Footer* é impresso a cada quebra de página.

A Figura 63 apresentada o mapeamento com essas alterações.

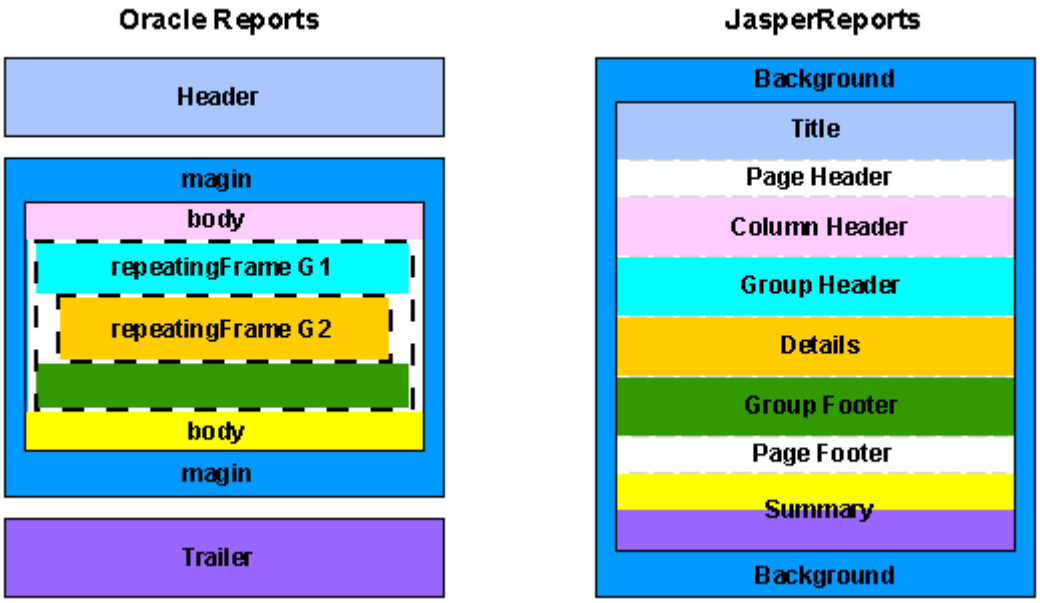


Figura 63: Mapeamento das partes que compõem o *layout*
Fonte: autor

Os *frames* encontrados nos relatórios Oracle Reports, pelo fato de não possuírem uma função que interfira no resultado do relatório criado no Jasper, serão ignorados.

Com relação a unidade de medida, utilizada para o dimensionamento dos componentes no relatório, será necessário realizar a conversão. No Oracle pode ser de três tipos: pontos, centímetros e polegadas, enquanto no Jasper a unidade usada é *pixel.* Para a conversão foram utilizadas as seguintes equivalências:

- 1 ponto = 1 *pixel;*
- 1 polegada = 72 *pixels;*
- 1 centímetro = 1 polegada = 72 / 2.54 *pixels.*

Além disso, no Oracle as medidas podem ser decimais, enquanto no Jasper os valores são inteiros, sendo assim, por conta do arredondamento, alguma diferença no dimensionamento dos componentes poderá ser notada.

5.3 ESTRUTURA DO MODELO CONCEITUAL

Para uma melhor organização do código fonte do projeto, foi criada uma estrutura de pacotes, que pode ser visualizada na Figura 64.

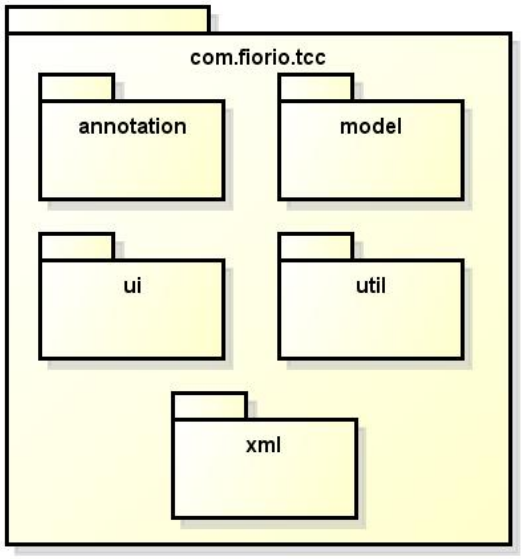


Figura 64: Estrutura de pacotes do projeto
Fonte: autor

Cada um destes pacotes tem uma função distinta:

- annotation**:** neste pacote se encontram as anotações utilizadas para a leitura e a escrita dos XMLs;
- model**:** neste pacote estão localizadas as classes que representam cada um dos elementos dos XMLs;
- ui**:** encontram-se neste pacote as classes responsáveis pelo gerenciamento da interface com o usuário;
- util**:** neste pacote estão as classes com funções utilitárias;
- xml: neste pacote estão as classes responsáveis pela leitura e escrita dos XMLs.

No pacote “model” foi realizada uma subdivisão com dois pacotes (Figura 65):

- data**:** classes que referem-se a elementos do modelo de dados;
- layout**:** classes que representam os elementos do *layout.*

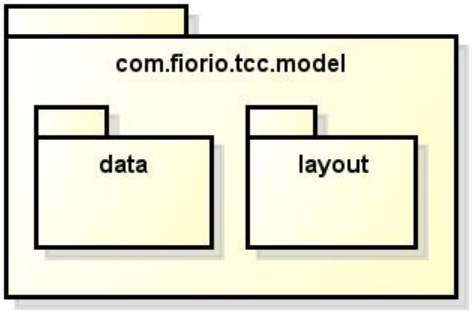


Figura 65: Pacote “model”
Fonte: autor

5.4 MODELO CONCEITUAL

As classes que compõem os pacotes descritos anteriormente serão apresentadas através de diagramas de classes.

A Figura 66 mostra o pacote *annotation* que contém as anotações usadas para a realização da leitura e escrita dos XMLs. A anotação XmlInput é a responsável pela marcação dos métodos de leitura e a XmlOutput é usada para marcar os métodos usados pela escrita. O funcionamento da leitura e escrita dos elementos dos XMLs será explicado na seção 5.5.

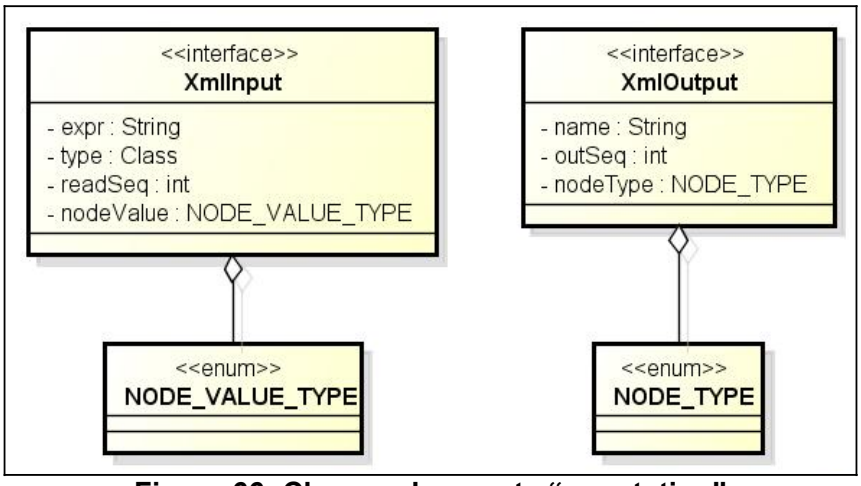


Figura 66: Classes do pacote "annotation"
Fonte: autor

O pacote "model" está representado por dois diagramas de classe. O primeiro com as classes do pacote "data" (Figura 67) e o segundo com as classes do pacote "layout" (Figura 68). As classes Report e Group representam o relatório como um todo e os grupos, respectivamente, encontram-se diretamente no pacote *model*, pois agrupam funções tanto do modelo de dados como de *layout*, aparecendo nos dois diagramas. No entanto, apresenta em cada um dos diagramas, somente os atributos referentes ao contexto. A explicação da função de cada uma das classes está após cada diagrama.

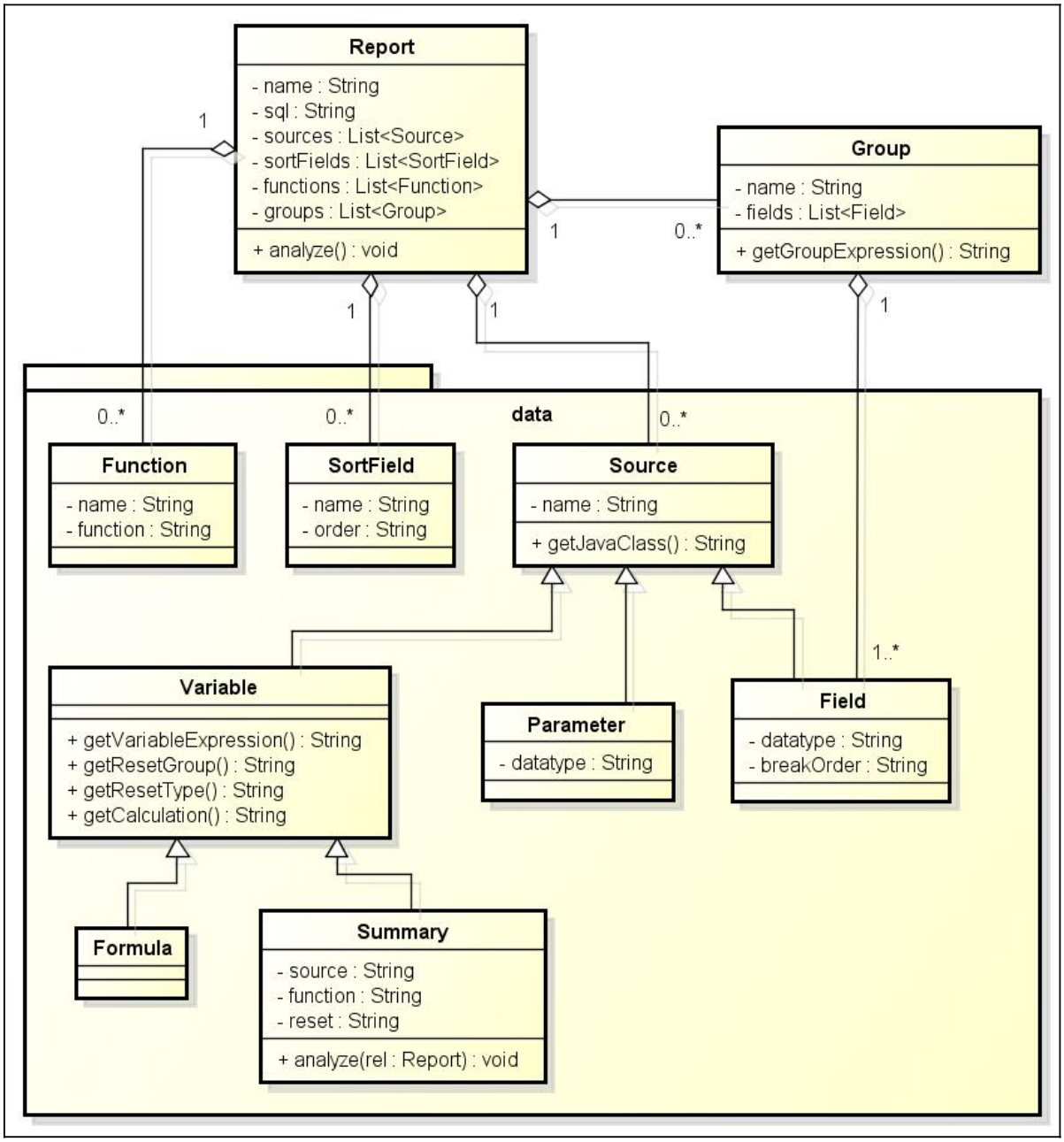


Figura 67: Classes do pacote "model.data"
Fonte: autor

As funções das classes do pacote "model.data" são:

- Report – representa o relatório com todas as informações necessárias para a conversão, o atributo *name* é o nome do relatório. No atributo *sql* fica o comando SQL de consulta executado pelo relatório; os atributos *sources*, *sortFields*, *functions* e *groups* são listas dos objetos utilizados no relatório, conforme cada classe explicada na sequência; o método *analyze* é executado após a leitura de todos elementos mapeados do XML do Oracle, com o objetivo de realizar a análise e preparação

destes objetos para a conversão;

- Function – representa as funções PL/SQL programadas dentro do relatório Oracle. Tem como objetivo apenas apresentar este código através do *log,* para que o desenvolvedor possa visualizar e criar a funcionalidade manualmente no Jasper. O atributo *name* é o identificador da função e o *function* é o código PL/SQL;
- SortField – representa o elemento *sortField* do Jasper*,* que realiza a ordenação dos registros conforme a configuração dos grupos. O atributo *name* é o nome do campo a ser ordenado, e *order* é o sentido da ordenação: “Ascending” ou “Descending”;
- Source – superclasse que representa todos os elementos que são fontes de dados que podem ser utilizados por outros elementos. Por exemplo, campos e variáveis, que podem ser utilizadas em uma expressão de um *textField*. O atributo *name* é o nome do *Source* e o método *getJavaClass* deve ser implementado pelas subclasses que deve retornar o nome classe que será declarada no Jasper;
- Parameter – subclasse de Source que representa os parâmetros do relatório. O atributo *datatype* é o tipo definido no Oracle que servirá para determinar a classe Java do parâmetro, por exemplo, se o *datatype* no Oracle é *number,* no Jasper a classe usada é BigDecimal;
- Field – subclasse de Source que representa cada campo retornado na consulta. É a classe que representa o elemento *dataItem* no Oracle e o *field* do Jasper. O atributo *datatype,* assim como na classe Parameter, se refere ao tipo do dado, enquanto o atributo *breakOrder* é a ordenação definida no Oracle, ascendente ou descendente, que servirá para criação dos objetos SortField;
- Variable – é a superclasse que representa as variáveis do Jasper, essas variáveis são criadas a partir de duas subclasses: Formula e Summary, que devem implementar os métodos definidos:
    - getVariableExpression – retorna a expressão da variável Jasper;
    - getResetType – retorna o atributo resetType da variável;
    - getResetGroup – retorna o atributo resetGroup da variável;

    - getCalculation – retorna o atributo calculation da variável.
- Formula – subclasse de Variable que representa os elementos *formula* e *placeholder* do Oracle, estes elementos são funções PL/SQL e serão mapeados para uma variável com conteúdo fixo, como definido na seção 4.2.1;
- Summary – subclasse de Variable que representa os elementos *summary* do Oracle. O atributo *source* indica qual o campo fonte para criação da expressão, *function* representa a função pré-definida utilizada e *reset* define qual nível de reinicialização usado. O método *analyze* analisa estes dados e realiza os mapeamentos necessários para a criação da variável no Jasper e é executado pelo método *analyze* da classe Report;
- Group – representa cada grupo criado no *dataSource* do Oracle e é usada para criação dos grupos no Jasper. O método getGroupExpression retorna a expressão concatenando os *fields* que pertencem ao grupo.

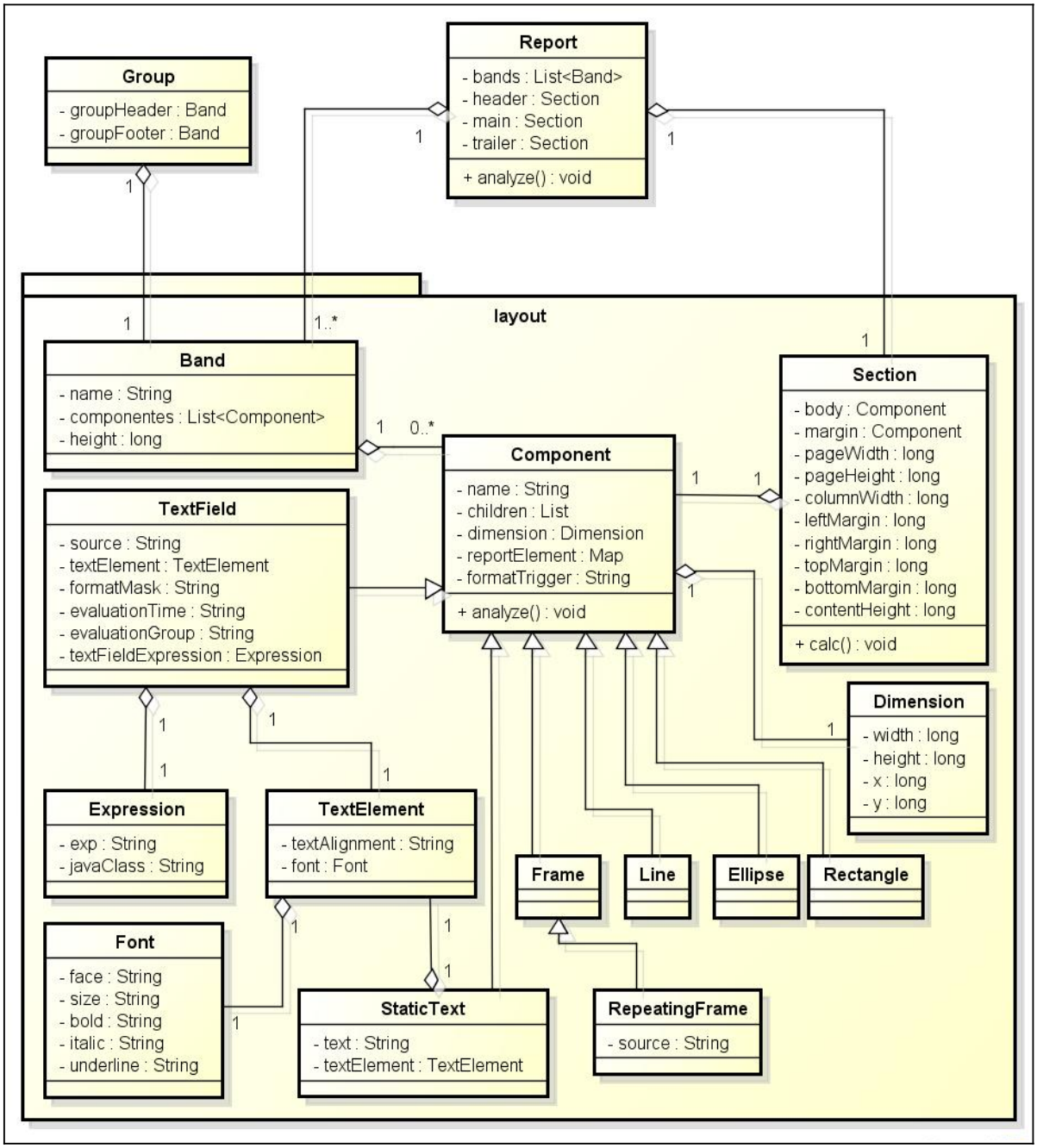


Figura 68: Classes do pacote "model.layout"
Fonte: autor

As funções das classes do pacote "model.layout" são:

- Report – no momento da leitura do XML Oracle são criados os objetos *header*, *main* e *trailer* que representam cada setor do relatório Oracle, na execução do método *analyze* a partir destes objetos são criadas as Bandas, atributo *bands*, com os componentes de acordo com a Figura 63;
- Component – superclasse que representa todos os componentes do re-

latório e possui os atributos comuns a todos os componentes, são eles:

- name – identificador do componentes;
- dimension – tamanho e posição do componente no Oracle;
- reportElement – propriedades comuns adicionados pelo elemento *reportElement* do Jasper;
- childen – componentes filhos, usados no caso das classes Frame e RepeatingFrame e nos atributos *body* e *margin* da classes Section;
- formatTrigger – nome da função PL/SQL utilizada para formatações especiais do componente no Oracle, utilizada para o desenvolvedor ter o conhecimento através do *log.*

- Section – representa cada setor do relatório Oracle. Nos atributos *body* e *margin* são carregados todos componentes lidos do XML, os outros atributos definem o tamanho da página e as margens que são calculados de acordo com a unidade de medida no método *calc;*
- Dimension – classe que carrega os valores de tamanho e posição do relatório;
- Band – representa cada uma das bandas criadas no Jasper, identificada através do atributo *name*. No atributo componentes é colocada a lista dos componentes encontrados no setores de acordo com o mapeamento ilustrado na Figura 63;
- Frame – representa os componentes *frame* do Oracle;
- RepeatingFrame – representa os componentes *repeatingFrame* do Oracle. O atributo *source* é o nome do grupo relacionado a ele;
- Line – representa as linhas presentes no relatório;
- Ellipse – é o componente *arc* no Oracle, que no Jasper cria o elemento *ellipse;*
- Rectangle – representa os componentes *rectangle* e *roundedRectangle* do Oracle, criando *rectangle* no Jasper;
- TextElement – representa o elemento *textElement* do Jasper, possui as informações de formatação dos componentes de texto, como alinhamento e fonte;

- Font – Classe que carrega as informações da fonte usada nos componentes de texto;
- StaticText – representa o componente do *staticText* do Jasper, é criado a partir da leitura do elemento *text* no Oracle;
- TextField – é o componente *textField* do Jasper criado a partir do elemento *field* do Oracle, onde, através do atributo *source* lido do Oracle, que é a fonte de dados do componente, é criada a expressão corresponde no Jasper, usando a classe Expression. O atributo *formatMask* declarado no Oracle, contém a máscara de formatação do campo para valores numéricos e data e é convertido no Jasper para o atributo *pattern*. Os atributos *evaluationTime* e *evaluationGroup* definem o momento em que o componente irá avaliar a expressão na fase de impressão, são usados quando o *source* é uma variável do tipo *Summary;*
- Expression – definem as expressões dos componentes *textFields.*

O pacote “ui” que contém as classes que controlam a interface com o usuário está apresentado na Figura 69.

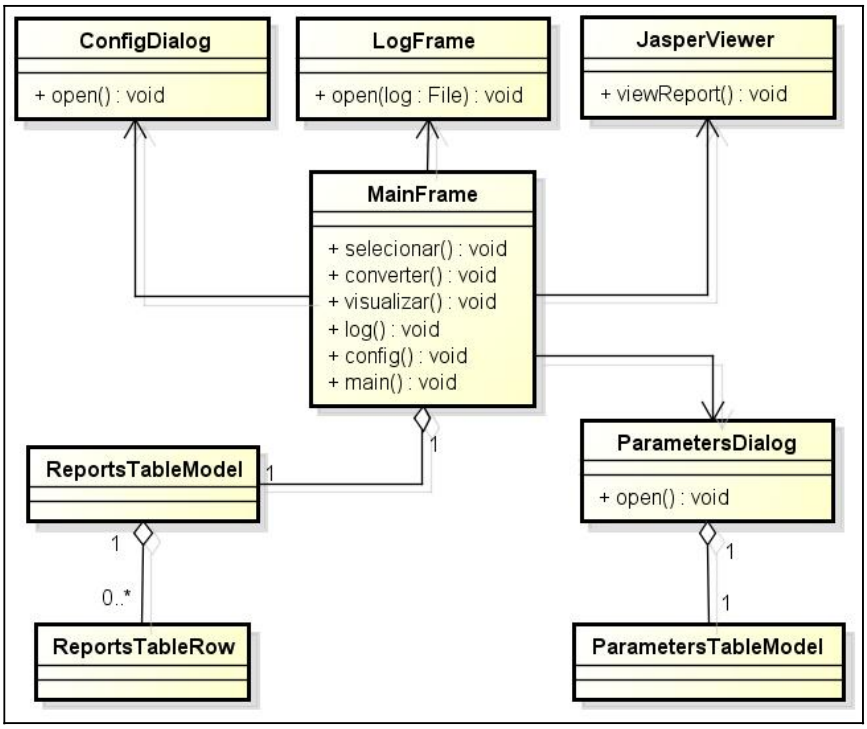


Figura 69: Classes do pacote “ui”
Fonte: autor

As funções das classes do pacote “ui” são:

- MainFrame – Classe que controla a tela principal do *software*, nela são

realizadas as principais operações: selecionar arquivos, converter e visualizar, a partir dela também é possível abrir o *log* gerado pela conversão e a tela de configuração;

- ReportsTableModel – esta classe implementa as funções de gerenciamento da tabela onde são adicionados os relatórios selecionados;
- ReportsTableRow – classe que contém as informações de cada linha da tabela;
- ParametersDialog – controla a tela apresentada para informar os parâmetros de entrada do relatório quando o usuário solicita a visualização;
- ParametersTableModel – implementa as funções de gerenciamento da tabela montada para o usuário informar os parâmetros do relatório;
- ConfigDialog – controla a tela com as opções de conversão dos relatórios;
- LogFrame – controla a tela onde são apresentados os *logs* da execução;
- JasperViewer – classe da biblioteca jasperreports que possibilita para o usuário a visualização do relatório.

A Figura 70 apresenta as classes do pacote “xml”.

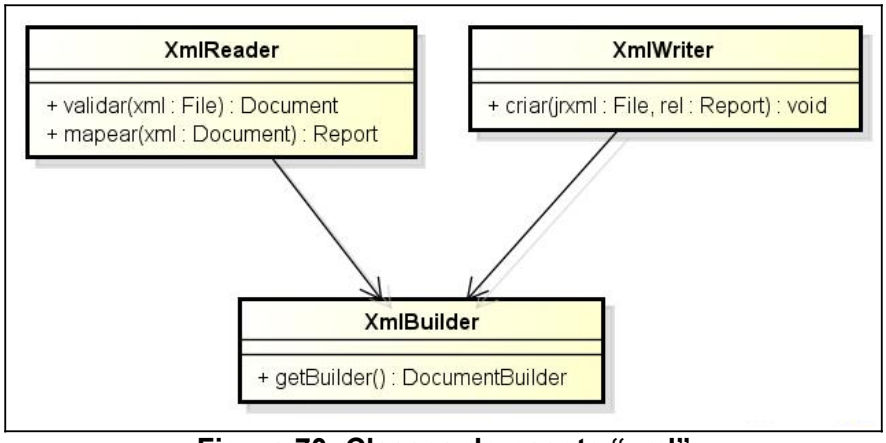


Figura 70: Classes do pacote “xml”
Fonte: autor

As funções das classes do pacote “xml” são:

- XmlBuilder – classes responsável pela criação do DocumentBuilder utilizado para leitura e escrita dos XMLs;
- XmlReader – classe que realiza a leitura do XML do Oracle, têm dois métodos:

- validar – recebe o arquivo XML e realiza a validação definida, se o XML for válido retorna o objeto Document do parser DOM;
- mapear – recebe o objeto Document e realiza a leitura do XML criando os objetos do pacote model detalhados anteriormente.

- XmlWriter – classe que realiza a criação do XML do Jasper, o jrxml, o método criar recebe por parâmetro o arquivo onde será gravado o XML e o objeto Report criado pela classe XmlReader.

Este processo de leitura e escrita dos XMLs será melhor detalhado na seção 5.5 .

A Figura 71 apresenta as classes do pacote “util”.

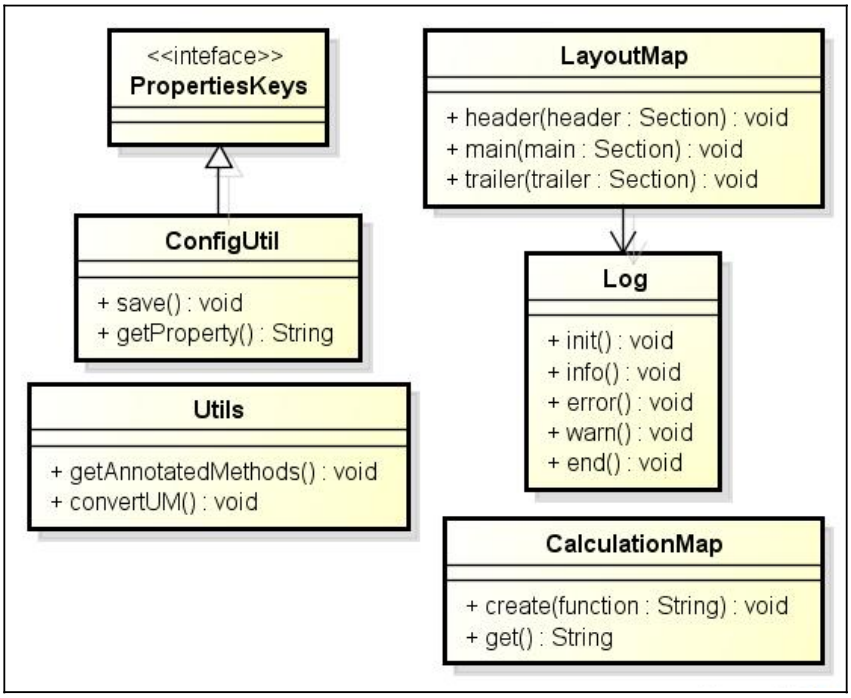


Figura 71: Classes do pacote “util”
Fonte: autor

As funções das classes do pacote “xml” são:

- ProperiesKeys – interface com constantes que identificam os opções da configuração;
- ConfigUtil – classe responsável por armazenar e recuperar as opções configuradas pelo usuário;
- CalculationMap – classe responsável pelo mapeamento das funções das variáveis do tipo *Summary,* recebe o nome da função utilizada no Oracle e retorna a equivalente no Jasper;

- LayoutMap – realiza o mapeamento dos componentes analisando os setores e adicionando os componentes nas bandas correspondentes do Jasper;
- Log – classe responsável pela criação dos arquivos de *log;*
- Utils – possui dois métodos utilitários:
    - getAnnotatedMethods – busca os métodos de uma classe que possui uma determinada anotação;
    - convertUM – converte um valor das unidades de medidas usadas no Oracle para a utilizada no Jasper.

5.5 IMPLEMENTAÇÃO

Nesta seção é descrita como foi implementada a conversão do relatório. O processo de conversão é realizado em três etapas (Figura 72):

1. leitura – cria os objetos e os preenche com os valores encontrados no XML do Oracle;
2. análise – analisa os objetos criados, gerando novos objetos e novos atributos necessários;
3. escrita – percorre os objetos criando os elementos do XML Jasper.

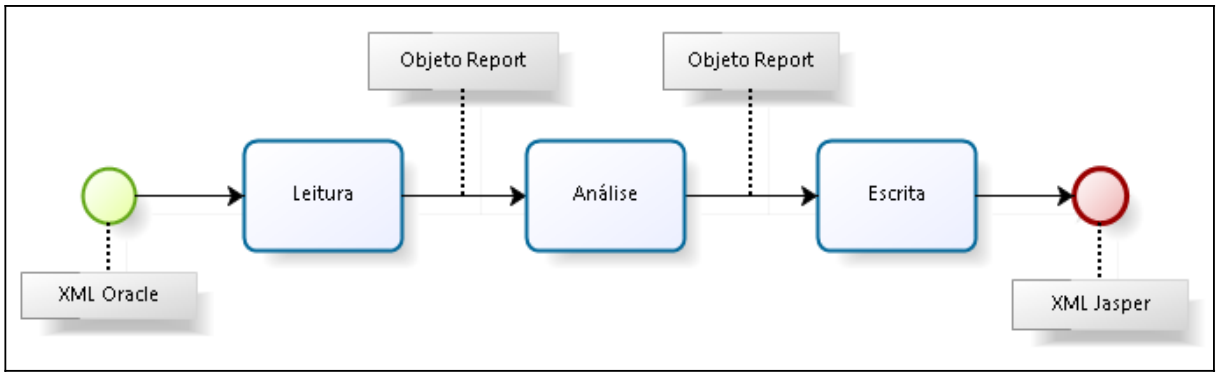


Figura 72: Processo de conversão
Fonte: autor

5.5.1 Etapa de leitura

Para implementação da leitura e escrita do XML são utilizadas as duas anotações, XmlInput e XmlOutput, marcando os métodos das classes do pacote “model” que são executados, de acordo com informações da anotação.

A primeira etapa, a leitura do XML do Oracle (Figura 73), é realizada através de um método recursivo da classe XmlReader que recebe por parâmetro uma classe e um nodo do XML, nesta classe são procurados todos os métodos que possuem a anotação XmlInput, usando-se o método getAnnotatedMethods da classe Utils. Nesta anotação são declarados os seguintes atributos:

- expr– define a expressão XPath a ser executada;
- type – define a nova classe que será passada para a próxima execução recursiva;
- nodeValue – define como pegar o valor do nodo;
- readSeq – usado para ordenar a execução dos métodos.

É definido *type* quando se deseja, com o nodo encontrado na expressão, criar um novo objeto e *nodeValue* quando este nodo representa um valor final.

A execução inicia com a criação de uma instância do objeto da classe recebida, que é retornado ao final do processo. Para cada método anotado encontrado executa-se a expressão XPath. Com o resultado da expressão obtém-se um novo nodo, se for um valor final (ex: atributo do XML) o método é invocado passando-se este valor por parâmetro, prosseguindo a execução para o próximo método. Caso tenha sido definido atributo o *type,* com a classe definida juntamente com o novo nodo encontrado é realizada a chamada recursiva, e com o novo objeto retornado é invocado o método passando-o por parâmetro.

A chamada inicial da leitura do XML é feita passando por parâmetro a classe Report e o nodo raiz do XML do Oracle o elemento *report*.

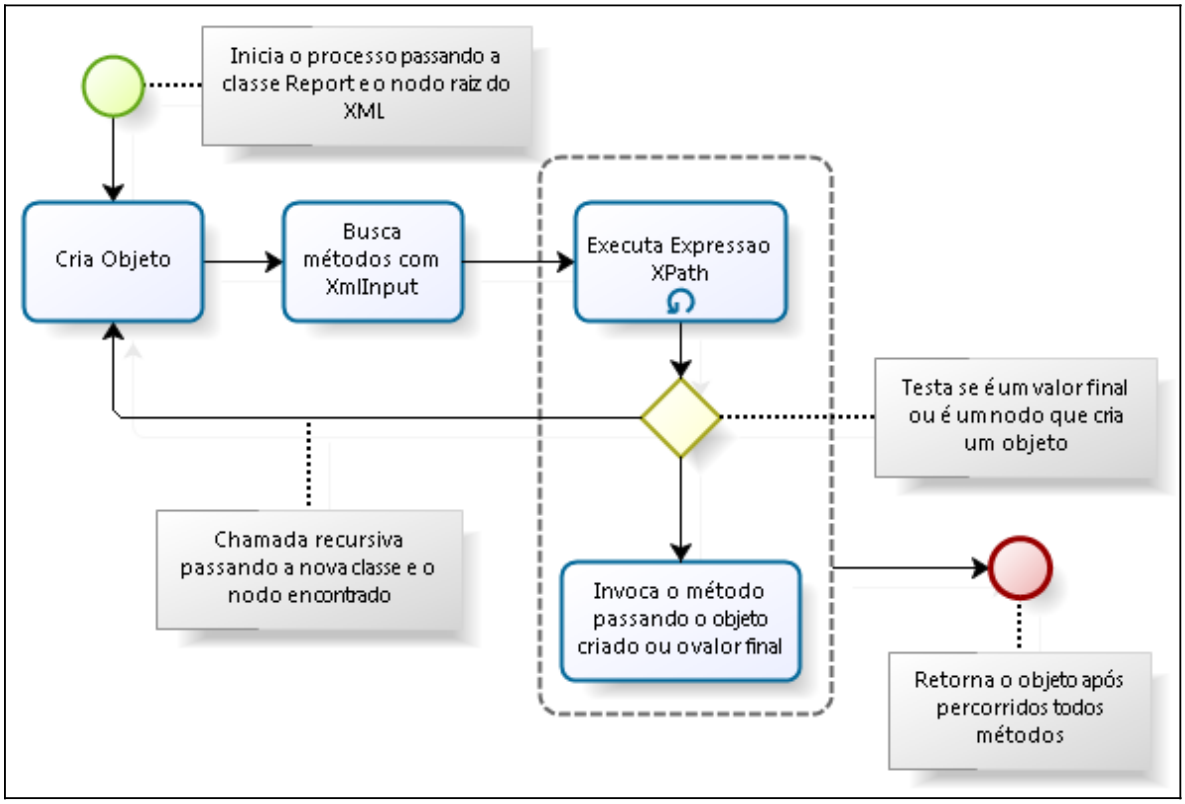


Figura 73: Processo de leitura
Fonte: autor

A Figura 74 mostra o código fonte com a utilização da anotação marcando o método que recebe um valor final, neste caso é o valor do atributo *name* do elemento *report*, exemplificado na Figura 75. Outra forma de se obter o valor final é quando este valor é o conteúdo do elemento, como é o caso do elemento *select* que contém o comando SQL da consulta, a Figura 76 mostra como é declarada a anotação neste caso e a Figura 77 mostra um exemplo do XML.

```
@XmlInput(expr = "/report/@name"
          ,nodeValue = NODE_VALUE_TYPE.VALUE
          ,readSeq = 1)
public void setName(String name) {
       ...
}
```

Figura 74: Exemplo de fonte para leitura de atributos
Fonte: autor

```
<report name="relatorio">
       ...
</report>
```

Figura 75: Exemplo de XML para leitura de atributos
Fonte: autor

```
@XmlInput(expr = "/report/data/dataSource/select"
          ,nodeValue = NODE_VALUE_TYPE.TEXT
          ,readSeq = 2)
public void setSql(String sql) {
      ...
}
```

Figura 76: Exemplo de fonte para leitura de elementos onde o valor é o conteúdo
Fonte: autor

```
<report>
      <data>
            <dataSource>
                  <select>
                   <![CDATA[SELECT * FROM TABELA
                              WHERE campo = &param
                              ORDER BY campo DESC]]>
                  </select>
            </dataSource>
      </data>
</report>
```

Figura 77: Exemplo de XML para leitura de elementos onde o valor é o conteúdo
Fonte: autor

A Figura 78 apresenta um exemplo do código usado quando o elemento encontrado pela expressão XPath define a criação de um novo objeto. Neste exemplo a expressão resulta no elemento *userParameter* que são os parâmetros do usuário definidos no relatório e a classe que representa este elemento, a classe Parameter. A cada nodo encontrado pela expressão é realizada a chamada recursiva passando este nodo e a classe Parameter, ao executar irá se obter um novo objeto que será passado paro o método *addParameter*. A Figura 79 apresenta um exemplo de XML.

```
@XmlInput(expr = "/report/data/userParameter"
          ,type = Parameter.class
          ,readSeq = 5)
public void addParameter(Parameter parameter) {
      ...
}
```

Figura 78: Exemplo de fonte para leitura de elementos que criam um novo objeto
Fonte: autor

```
<report>
      <data>
            <userParameter name="PARAM1" datatype="character" />
            <userParameter name="PARAM2" datatype="number" />
      </data>
</report>
```

Figura 79: Exemplo de XML para leitura de elementos que criam um novo objeto
Fonte: autor

5.5.2 Etapa de análise

Na segunda etapa do processo é executa a análise dos objetos criados pela processo de leitura criando a estrutura necessária para representar o relatório Jasper. Este processo é executado através do método *analyze* da classe Report, que executa as seguintes ações:

1. Preparação das variáveis, criação da expressão, verificação da função equivalente e o nível da reinicialização;
2. Alteração dos parâmetros contidos no comando SQL para a forma usada no Jasper;
3. Criação dos objetos SortOrder, de acordo com as ordenações declaradas nos campos de cada grupo;
4. Criação das bandas e posicionamento dos componentes através da classe utilitária LayoutMap, onde é realizada a conversão dos unidades de medida.

A Figura 80 apresenta o fluxo do processo de análise.

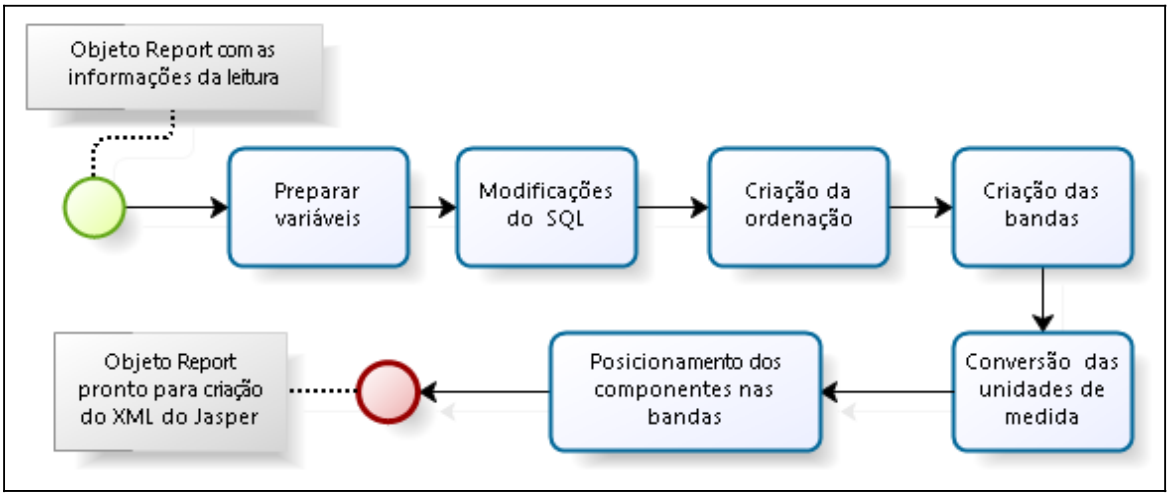


Figura 80: Processo de análise
Fonte: autor

5.5.3 Etapa de escrita

Após a etapa de análise o objeto Report está pronto para iniciar a terceira etapa, descrita na Figura 81, o processo de criação e escrita do XML do Jasper.

O funcionamento do processo de escrita é semelhante ao processo de leitura,

através de um método recursivo da classe XmlWriter invocando métodos marcados com uma anotação. Neste caso a anotação utilizada é a XmlOutput. Nela são declarados os seguintes atributos:

- name – define o nome do nodo a ser criado;
- nodeType – define o tipo do nodo: atributo ou elemento;
- outSeq – define a sequência de execução.

Este método recursivo recebe por parâmetro um objeto e o elemento corrente do XML, onde serão adicionados novos elementos e atributos, a primeira chamada é realizada passando o objeto Report e elemento raiz do XML Jasper, o elemento *jasperReport*, que é previamente criado.

Através da classe Utils getAnnotatedMethods obtêm-se os métodos presentes no objeto recebido por parâmetro que possuem a anotação XmlOutput. Cada método encontrado deverá retornar o objeto com as informações para criação do novo nodo.

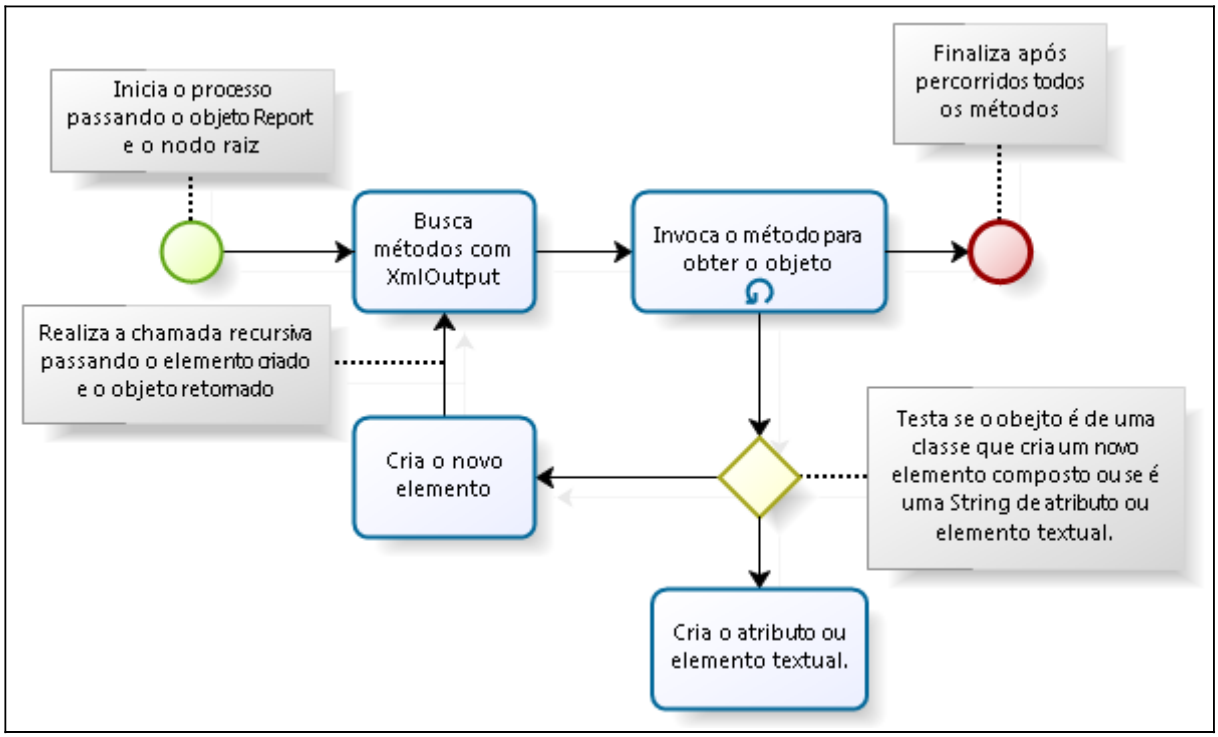


Figura 81: Processo de escrita
Fonte: autor

Quando o nodo for um atributo ou um elemento com conteúdo textual esse método deve retornar uma String, como mostra o exemplo na Figura 82, referente a classe Group, onde no primeiro método cria-se o atributo *name* e no segundo cria-se o elemento *groupExpression* com conteúdo textual, o XML gerado neste caso é

apresentado na Figura 83.

```
@XmlOutput(name = "name", nodeType = NODE_TYPE.ATTR)
public String getName() {
	...
}

@XmlOutput(name = "groupExpression", nodeType = NODE_TYPE.CDATA)
public String getGroupExpression() {
	...
}
```

Figura 82: Exemplo de utilização da anotação XmlOutput
Fonte: autor

```
<group name="grupo_1">
	<groupExpression>
		<![CDATA[$F{CAMPO_1}+$F{CAMPO_2}]]>
	</groupExpression>
</group>
```

Figura 83: XML gerado com atributo e elemento textual
Fonte: autor

Quando o objeto retornado pelo método com a anotação é de uma classe que cria um elemento composto é criado o novo elemento XML com o nome definido no atributo *name* da anotação e realizada a chamada recursiva do processo. A Figura 84 mostra um exemplo da utilização desta anotação para criação do elemento *textElement* da classe TextField, o método retorna um objeto da classe TextElement que contém as informações desse elemento, a Figura 85 mostra a declaração de alguns dos métodos dessa classe e a Figura 86 apresenta um exemplo de XML gerado.

```
@XmlOutput(name = "textElement", nodeType = NODE_TYPE.NODE, outSeq = 15)
public TextElement getTextElement() {
	...
}
```

Figura 84: Exemplo de utilização da anotação XmlOutput
Fonte: autor

```
@XmlOutput(name = "textAlignment", nodeType = NODE_TYPE.ATTR)
public String getTextAlignment() {
	...
}

@XmlOutput(name = "font", nodeType = NODE_TYPE.NODE)
public Font getFont() {
	...
}
```

Figura 85: Exemplo de utilização da anotação XmlOutput
Fonte: autor

```
<textField>
      <textElement textAlignment="Left">
            <font fontName="Arial" isBold="false" isItalic="false"
                  isUnderline="false" size="10" />
      </textElement>
</textField>
```

Figura 86: XML gerado a partir de um objeto
Fonte: autor

Com esta abordagem a implementação de novas funcionalidades na conversão fica facilitada, precisando somente criar as classes e métodos com as anotações para leitura e escrita. Além disso esse mesmo conceito pode ser facilmente utilizado em outras aplicações que manipulam XML.

5.6 FERRAMENTA DESENVOLVIDA

Nesta seção será apresentada como é realizada a utilização e as funcionalidades da ferramenta desenvolvida. A Figura 87 apresenta a tela principal da aplicação.

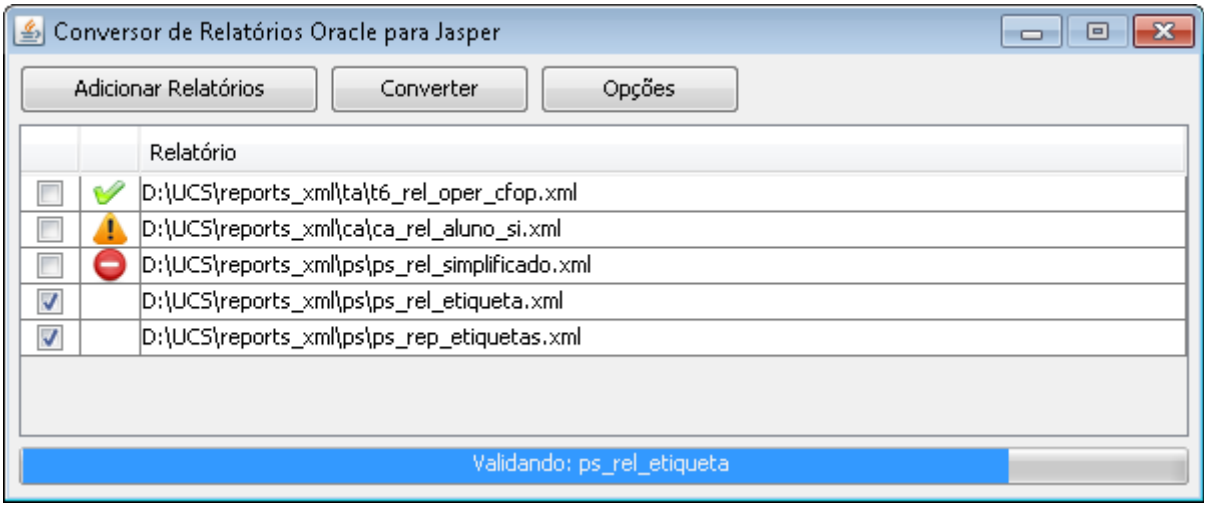


Figura 87: Tela principal
Fonte: autor

Os botões no topo da tela têm as seguintes funções:

- Adicionar Relatórios – abre a janela de seleção de arquivos padrão do sistema operacional para que o usuário selecione os relatórios que deseja converter;
- Converter – executa a conversão dos relatórios selecionados;
- Opções – abre a tela para configurar as opções.

Na base da tela é apresentada uma barra de progresso, que vai sendo completada conforme a conversão é executada mostrando o processo executado no momento.

No centro da tela apresenta-se uma tabela onde são listados os relatórios que foram adicionados. Na primeira coluna da tabela há um *checkbox* para marcar os relatórios que serão convertidos. A segunda apresenta um ícone ilustrando ao usuário o que ocorreu durante a conversão. Enquanto a terceira coluna apresenta o caminho do relatório. Ao clicar em um linha da tabela com o botão direito do mouse, é apresentado um menu de contexto (Figura 88) com as opções:

- Visualizar – executa o relatório já convertido abrindo a tela de visualização do Jasper;
- Log – abre a tela apresentando o *log* gerado na conversão do relatório;
- iReport – abre o relatório convertido com o iReport;
- Abrir XML – abre o XML do relatório Oracle no editor configurado.

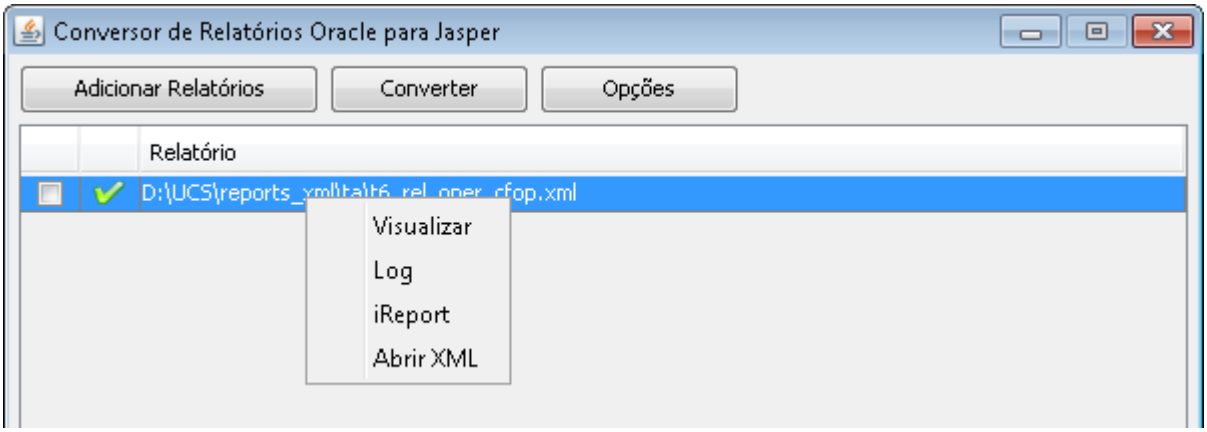


Figura 88: Menu de contexto com opções
Fonte: autor

A tela de configuração (Figura 89) possui as seguintes opções:

- Diretório destino – diretório onde serão salvos os arquivos gerados na conversão;
- RDF converter – caminho do programa da Oracle de conversão de arquivos RDF para XML;
- iReport – caminho do iReport;
- Editor XML – caminho do editor de texto que irá abrir o XML do Oracle;
- Converter – opções de marcar para converter os setores *header* e *trailer* do relatório Oracle;

- Conexão com banco de dados – opções para configurar a conexão com banco de dados que é utilizada na visualização do relatório.

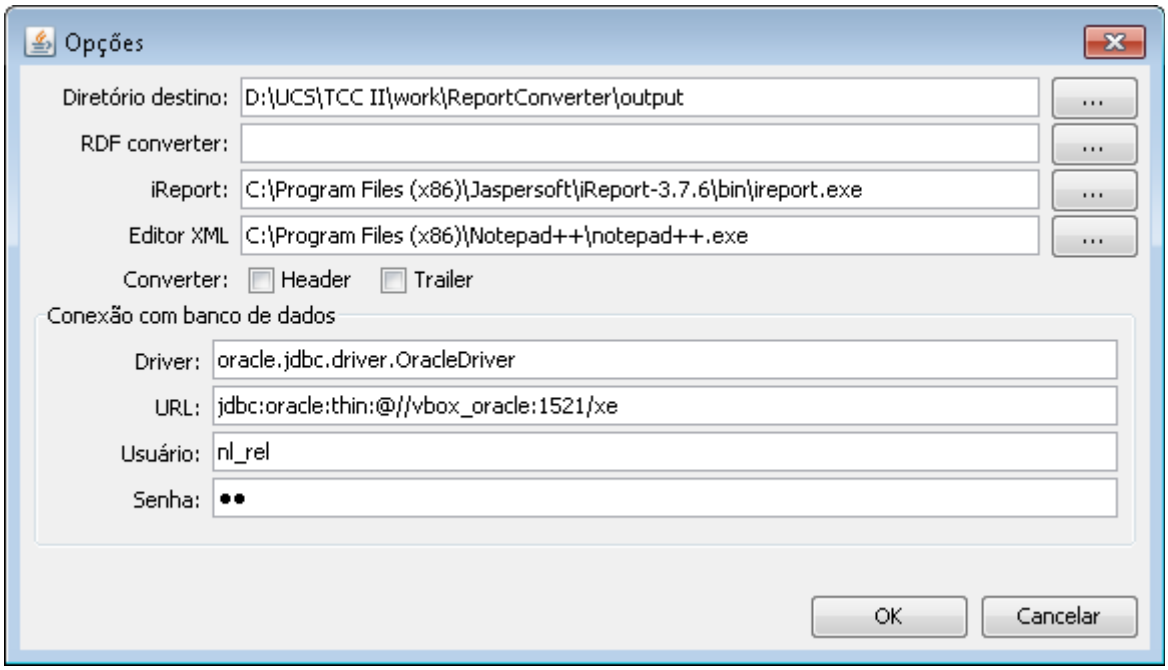


Figura 89: Tela de configuração das opções
Fonte: autor

Quando o usuário solicita a visualização do relatório é apresentada a tela para que seja informado os parâmetros de entrada do relatório (Figura 90) e em seguida é apresentada a tela de visualização do Jasper (Figura 91).

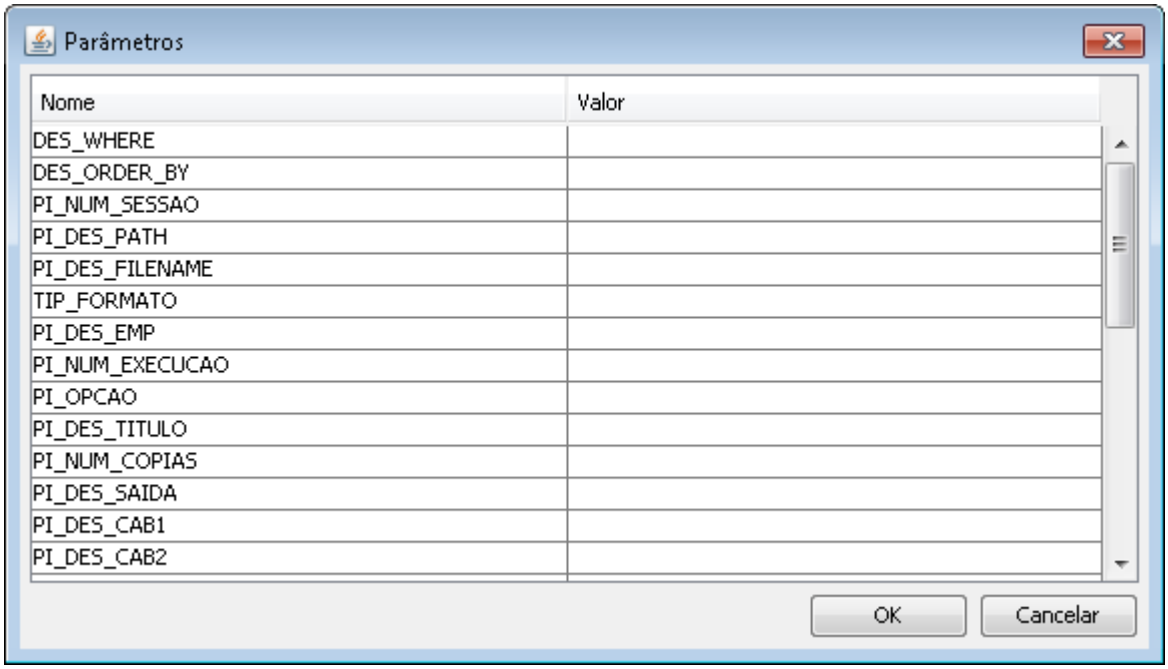
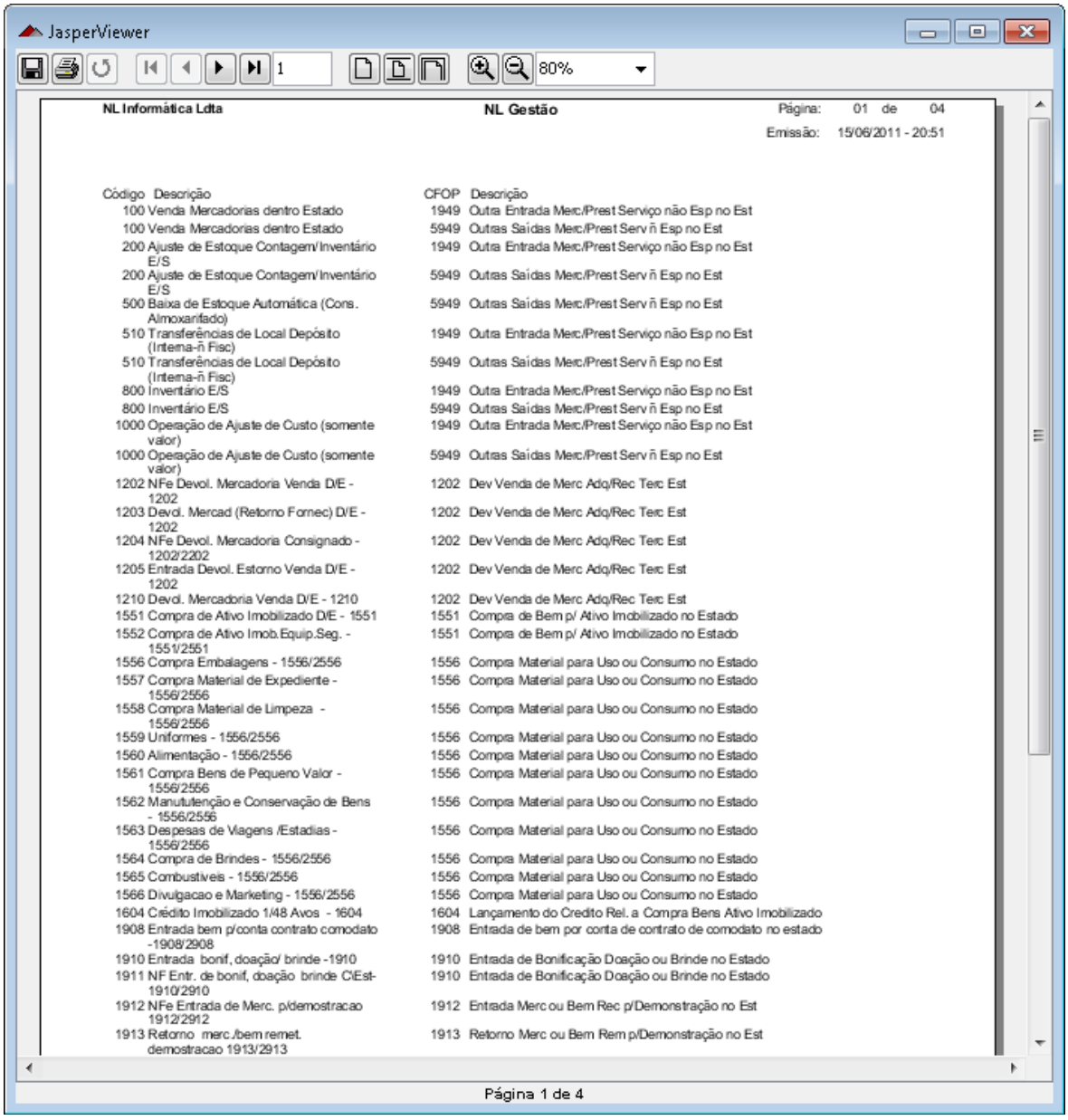


Figura 90: Tela para informar os parâmetros do relatório
Fonte: autor



Código	Descrição	CFOP	Descrição
100	Venda Mercadorias dentro Estado	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
100	Venda Mercadorias dentro Estado	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
200	Ajuste de Estoque Contagem/Inventário E/S	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
200	Ajuste de Estoque Contagem/Inventário E/S	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
500	Baixa de Estoque Automática (Cons. Almoxarifado)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
510	Transferências de Local Depósito (Interna-ñ Fisc)	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
510	Transferências de Local Depósito (Interna-ñ Fisc)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
800	Inventário E/S	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
800	Inventário E/S	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
1000	Operação de Ajuste de Custo (somente valor)	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
1000	Operação de Ajuste de Custo (somente valor)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
1202	NFe Devol. Mercadoria Venda D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1203	Devol. Mercad (Retorno Fornec) D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1204	NFe Devol. Mercadoria Consignado - 1202/2202	1202	Dev Venda de Merc Adq/Rec Terc Est
1205	Entrada Devol. Estorno Venda D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1210	Devol. Mercadoria Venda D/E - 1210	1202	Dev Venda de Merc Adq/Rec Terc Est
1551	Compra de Ativo Imobilizado D/E - 1551	1551	Compra de Bem p/ Ativo Imobilizado no Estado
1552	Compra de Ativo Imob.Equip.Seg. - 1551/2551	1551	Compra de Bem p/ Ativo Imobilizado no Estado
1556	Compra Embalagens - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1557	Compra Material de Expediente - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1558	Compra Material de Limpeza - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1559	Uniformes - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1560	Alimentação - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1561	Compra Bens de Pequeno Valor - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1562	Manututenção e Conservação de Bens - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1563	Despesas de Viagens /Estadias - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1564	Compra de Brindes - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1565	Combustiveis - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1566	Divulgacao e Marketing - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1604	Crédito Imobilizado 1/48 Avos - 1604	1604	Lançamento do Credito Rel. a Compra Bens Ativo Imobilizado
1908	Entrada bem p/conta contrato comodato -1908/2908	1908	Entrada de bem por conta de contrato de comodato no estado
1910	Entrada bonif, doação/ brinde -1910	1910	Entrada de Bonificação Doação ou Brinde no Estado
1911	NF Entr. de bonif, doação brinde C/Est-1910/2910	1910	Entrada de Bonificação Doação ou Brinde no Estado
1912	NFe Entrada de Merc. p/demostracao 1912/2912	1912	Entrada Merc ou Bem Rec p/Demonstração no Est
1913	Retorno merc./bem remet. demostracao 1913/2913	1913	Retorno Merc ou Bem Rem p/Demonstração no Est

Figura 91: Tela de visualização do relatório
Fonte: autor

5.7 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a realização do desenvolvimento do *software,* em que foram encontradas algumas dificuldades listadas abaixo.

A primeira tentativa de leitura do XML foi feita utilizando um método sequencial, definindo uma expressão para cada elemento do XML criando um dos objetos da

classe Report. Porém foi identificado que havia um padrão nas chamadas podendo ser realizada dinamicamente, além disso essa abordagem implicaria na criação de um bloco completo de busca e criação dos objetos para cada elemento, o que resultaria em um código fonte relativamente grande. A solução encontrada foi a criação de um método recursivo que lê a expressão definida dentro das classes com as anotações. A partir do ponto que este método estava pronto a necessidade de leitura de novas informações do XML se tornou simples. Quando foi realizada a escrita do XML Japer foi utilizada diretamente esta abordagem.

No momento da conversão do setor *header* foi identificado que o Jasper imprime sempre as bandas *pageHeader* e *pageFooter,* inclusive na impressão da banda *title*, com isso, ao colocar os elementos na banda *title* do Jasper, a altura dela mais a altura das banda *pageHeader* e *pageFooter* resultavam em um tamanho maior que a altura da página, fazendo com que ocorresse um erro na compilação do relatório. Tendo em vista que na NL este setor era usado como folha de rosto para separação nas filas de impressão e nos novos relatórios esta funcionalidade não estava mais sendo utilizada. Foi colocada uma opção para o usuário decidir se deseja ou não converter este setor. Foi adicionada também a opção para converter o setor *trailer*, tendo em vista que nenhum relatório da NL o utiliza.

Nos elementos do Oracle do tipo *field,* quando vinculados a campos numéricos ou de data, podem ter definidos uma máscara de formatação. No Jasper esta funcionalidade também existe, porém esta máscara possui um padrão diferente no Oracle e no Jasper. Nem sempre esta conversão é simples, com isso foi decidido converter as máscara mais utilizadas. Realizando uma busca nos XMLs do Oracle somente pelo atributo *formatMask* foi possível identificar as máscaras mais utilizadas. Trabalhando com expressões regulares, realizando substituições e fazendo testes com caracteres específicos, foi possível tratar uma grande parte destas máscaras. As máscaras que, por algum motivo, não forem convertidas, o sistema adicionará esta informação no *log* para conhecimento do usuário.

6 ESTUDOS DE CASO

Neste capítulo serão apresentados quatro estudos de caso, utilizando relatórios desenvolvidos pela NL. A seleção dos relatórios para os estudos de caso foi definida com a ajuda de um analista de sistemas da empresa, que apontou alguns relatórios que continham funcionalidades frequentemente utilizadas nos mesmos. Também realizou-se um pesquisa nos relatórios existentes, criando uma estatística que pode ser visualizada na seção 6.1.

6.1 ESTATÍSTICAS

Foi desenvolvido um programa que lê os arquivos XMLs de todos os relatórios da NL procurando especificamente pelo elemento *dataSource*, classificando-os pelo número de *queries* que eles possuem. Foram analisados 1706 relatórios. A Tabela 18 mostra estes dados, apresentando a quantidade de relatórios de acordo com o número de *queries.*

Tabela 18: Quantidade de relatórios pelo número de *queries*
Fonte: autor

Número de *queries*	Número de relatórios	Percentual
0	15	0,88%
1	953	55,86%
2	354	20,75%
mais de 2	384	22,51%
Total	1706	100%

De acordo com estes dados os relatórios que poderiam ser convertidos pelo programa, por possuírem apenas uma *query*, seriam 953. Porém, como mostra o estudo de caso 4, muitos dos relatórios que possuem mais de uma *query* foram criados para suportar *layouts* diferentes. Realizando uma separação dos *layouts* em relatórios distintos e elaborando-os com apenas uma *query* é possível a conversão.

6.2 ESTUDO DE CASO 1 - RELATÓRIO CFOP

Para o primeiro estudo de caso foi escolhido o relatório das operações do sistema com relação à CFOP (Código Fiscal de Operações e Prestações). É um relatório tabular que apresenta quatro campos:

- Código da operação no sistema;
- Descrição da operação no sistema;
- Código de CFOP;
- Descrição da CFOP.

A Figura 92 mostra a configuração do *data source* no Oracle Reports e a Figura 93 mostra como foi definido o *layout*.

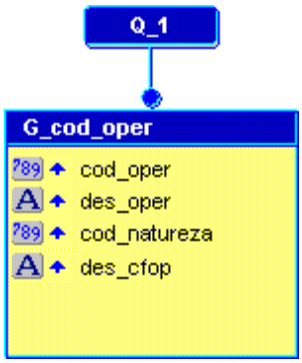


Figura 92: Retatório CFOP - data source
Fonte: autor

Figura 93: Relatório CFOP - layout
Fonte: autor

Na conversão deste relatório nenhum aviso é reportado no *log*. A Figura 94 mostra a tela com o resultado da conversão.

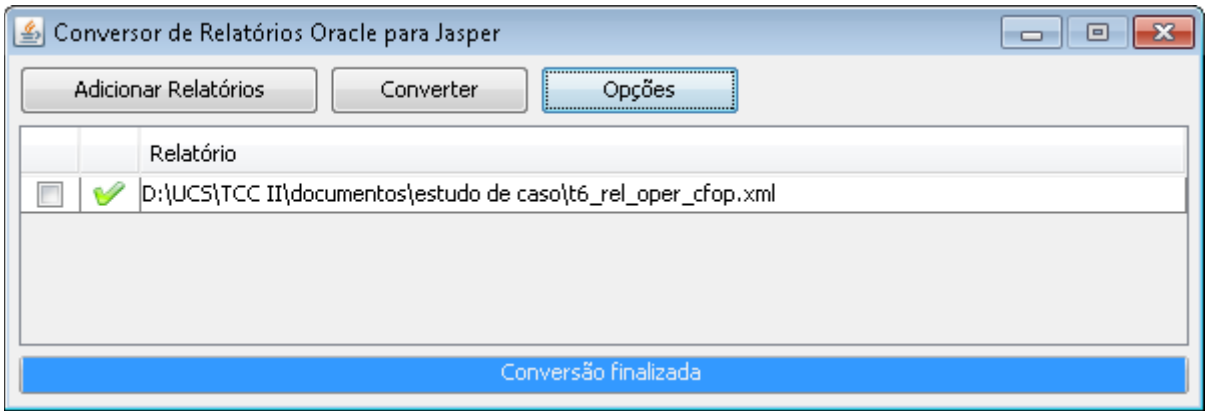


Figura 94: Relatório CFOP - tela da conversão
Fonte: autor

Após a execução do relatório pode ser notada a diferença na quantidade de registros relacionados por página, por causa da diferença na conversão das unidades de medida, conforme foi descrito na seção 5.2.2. No Anexo A pode ser visualizada a primeira página do relatório executado pelo Oracle (Figura 99) e pelo Jasper (Figura 100). A Figura 95 mostra como ficou a estrutura do relatório após a conversão, visualizando-se através do iReport.

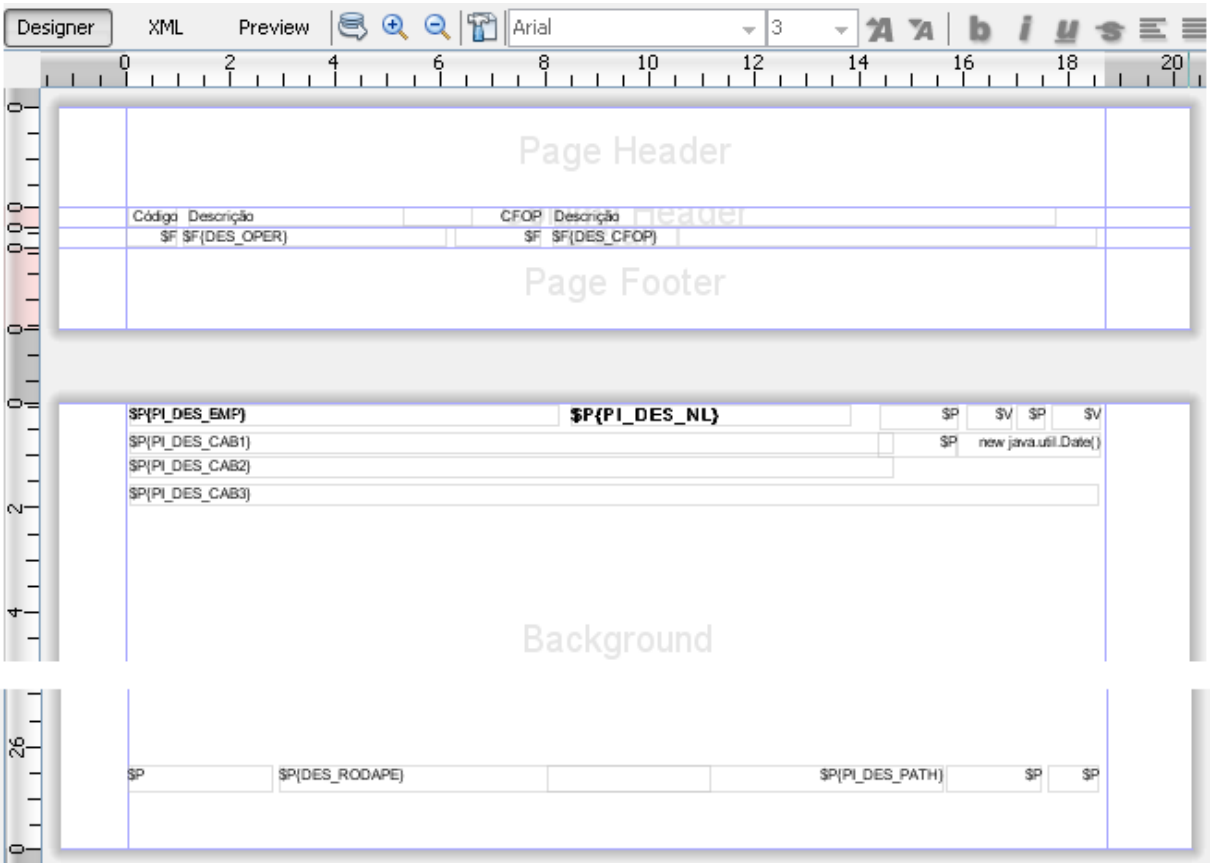


Figura 95: Relatório CFOP - visualizado no iReport
Fonte: autor

6.3 ESTUDO DE CASO 2 - RELATÓRIO DE DISTRIBUIÇÃO

Neste estudo de caso foi escolhido o relatório de distribuição de ordens de compra, pois ele mostra a distribuição dos itens das ordens de compra nas unidades, seleciona as ordens de compra que ainda não foram recebidos por nota fiscal de entrada e/ou transferidos por nota fiscal de transferência.

É um relatório mestre/detalhe que possui duas quebras: um pela ordem de compra e outra pela unidade de destino. A Figura 96 mostra o *data source* deste relatório com a configuração dos grupos de quebra.

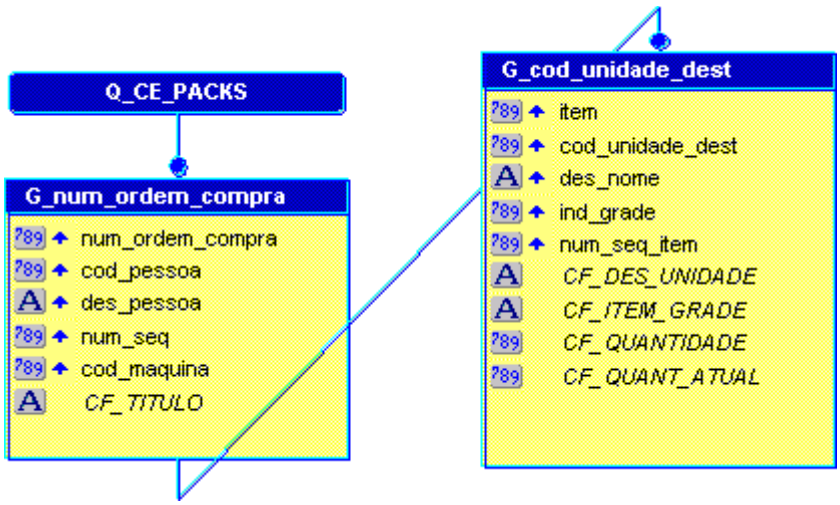


Figura 96: Relatório de distribuição - data source
Fonte: autor

Ao converter este relatório o *software* adiciona no *log* os avisos que os cinco campos do tipo fórmula presentes no relatório não foram convertidos. Dois deles (CF_TITULO e CF_DES_UNIDADE) são usados apenas para concatenação de texto com o código e a descrição. Usando como exemplo, temos o CF_TITULO, que no Oracle possui o seguinte código:

```
function CF_TITULOFormula return Char is
begin
  RETURN('Ordem de compra: '||:num_ordem_compra
         ||' - Fornecedor: '||:cod_pessoa
         ||' - '||:des_pessoa);
end;
```

Ele pode ser convertido, editando a variável criada, usando o iReport, para a seguinte expressão no Jasper:

```
"Ordem de compra: " + $F{NUM_ORDEM_COMPRA}
+ " - Fornecedor: " + $F{COD_PESSOA}
+ " - " + $F{DES_PESSOA}
```

Os campos CF_ITEM_GRADE, CF_QUANTIDADE e CF_QUANT_ATUAL são funções mais complexas, possuem testes e executam comandos SQL para retornar seus valores. Para a conversão poderiam ser criadas funções no banco de dados para executar estas operações e colocá-las diretamente no comando SQL retornando campos do tipo *field*. Para simulação foram editadas as variáveis colocando-se valores fixos para se executar o relatório e obter a visualização. O Anexo B possui o relatório executado a partir do Oracle, nas figuras 101 e 102, e a partir do Jasper, nas figuras 103 e 104.

6.4 ESTUDO DE CASO 3 - RELATÓRIO DE DISCIPLINAS

Este relatório é utilizado para listar os alunos matriculados em cada disciplina por turma. É um relatório mestre/detalhe com três quebras: unidade, disciplina e turma. A Figura 97 mostra a configuração do *data source* do relatório Oracle.

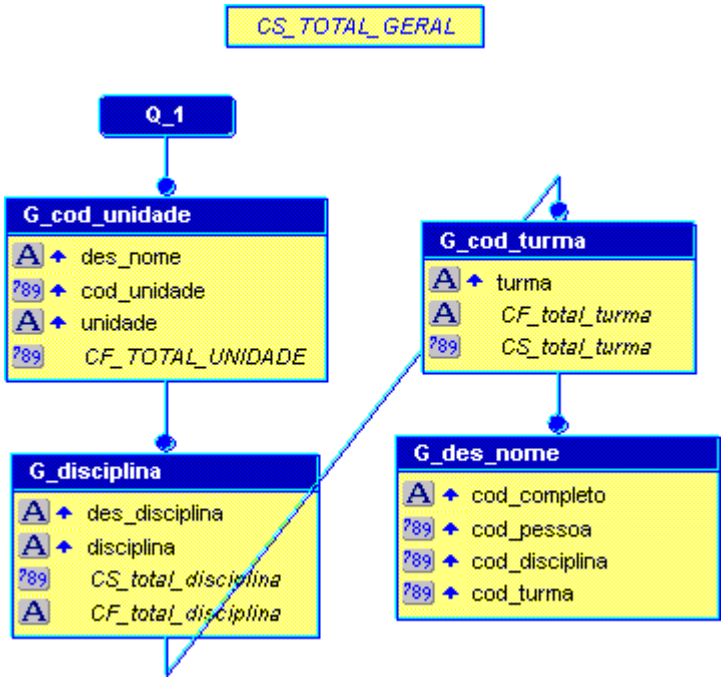


Figura 97: Relatório de disciplinas - data source
Fonte: autor

Ao converter esse relatório os avisos reportados *log* são a respeito dos campos do tipo fórmula, que, assim como no estudo de caso anterior, possui dois casos

que são utilizados para concatenação de texto e valor (CF_total_disciplina e CF_total_turma). Enquanto o campo CF_TOTAL_UNIDADE executa um comando SQL para calcular o total de alunos na unidade, independente de quantas disciplinas eles cursam. No Jasper isso pode ser feito com uma variável utilizando a função *Distinct Count.*

Após a realização destas alterações, pode-se executar e visualizar o relatório através do Jasper. Porém comparando com a execução gerada pelo Oracle, percebe-se que alguns campos não apareciam, por conta dos arredondamentos, uma vez que a altura destes campos estava 1 pixel menor do que a necessária. Além disso, de acordo com o posicionamento dos campos disciplina e turma, durante a conversão eles foram adicionados à banda *Group Footer,* o que faz com que eles apareçam no final da impressão do grupo, movendo-os para a banda *Detail* e marcando a opção de não repetir para valores iguais, obteve-se o mesmo resultado da execução no Oracle. As visualizações deste relatório estão apresentadas no Anexo C. As figuras 105 e 106 mostram o relatório executado pelo Oracle, a Figura 107 mostra a primeira execução pelo Jasper e as figuras 108 e 109 a execução após os ajustes.

6.5 ESTUDO DE CASO 4 - RELATÓRIO DE PRATOS

Para este estudo de caso o relatório escolhido foi o relatório de pratos. Ele lista os pratos servidos pelos restaurantes, listando os insumos utilizados em cada prato. Este relatório possui duas *queries* para imprimir *layouts* diferentes de acordo com alguns parâmetros de entrada. Se tentasse converter diretamente este relatório pelo *software* seria apresentado para o usuário que ele não pode ser convertido, pois possui mais de uma *query*. Porém, pode-se editar este relatório pelo Oracle Reports, antes da conversão, criando-se relatórios diferentes, fazendo a separação das *queries* e dos *layouts*. Para simulação foi escolhido um dos *layouts* e realizada esta separação. Este *layout* possui quatro quebras de grupo: grupo de restaurantes, restaurante, grupo de pratos e prato. Na Figura 98 pode-se visualizar como ficou configurado o *data source* deste relatório.

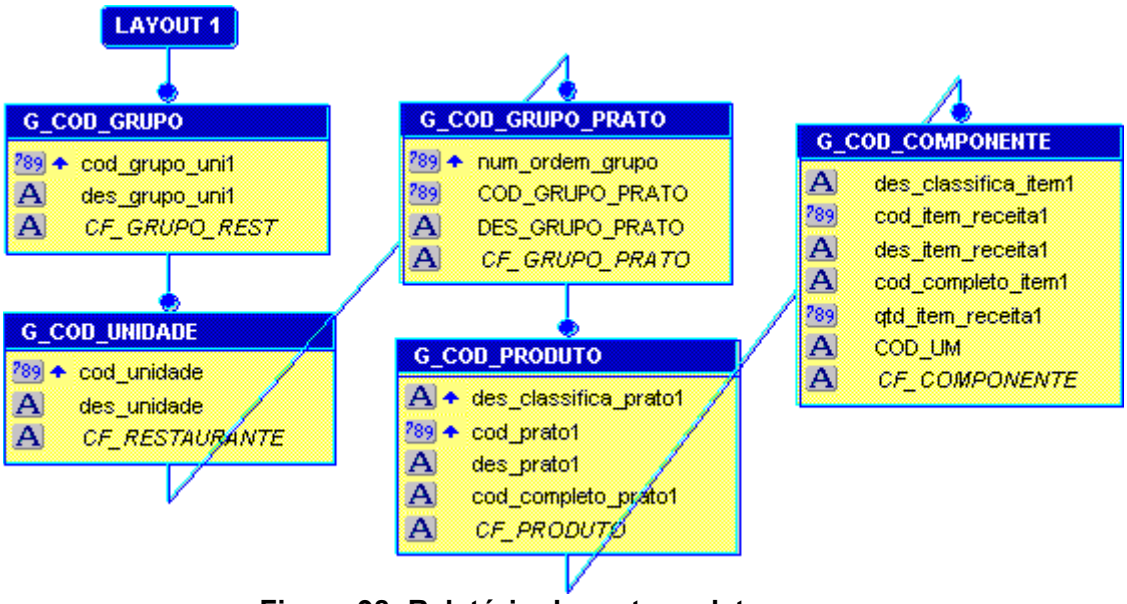


Figura 98: Relatório de pratos - data source
Fonte: autor

Ao converter este relatório, os avisos reportados no *log* são a respeito dos campos tipo fórmula, que, assim como no estudo de caso anterior, são usadas para concatenação dos valores e descrição.

Após a realização destes ajustes e execução do relatório, o que pode ser notado é a diferença no espaçamentos entre os campos, por conta dos arredondamentos das unidades de medida. No anexo D pode-se visualizar o relatório executado pelo Oracle, figuras 110 e 111, e pelo Jasper, figuras 112 e 113.

7 CONCLUSÃO

A empresa NL Informática, que completa em 2011, 30 anos de atuação no mercado de desenvolvimento de *softwares* para gestão empresarial, está atualmente em uma fase de migração da sua tecnologia de desenvolvimento. Possui seu sistema consolidado desenvolvido em tecnologia Oracle, porém, nos últimos anos vem realizando a migração para a tecnologia Java.

Como ferramentas de criação de relatórios a empresa utiliza, na plataforma Oracle, a Oracle Reports e na plataforma Java, a JasperReports. Diante do exposto, surgiu a necessidade de converter os relatórios já existentes para a nova tecnologia.

Este trabalho teve como objetivo o desenvolvimento de um protótipo de aplicativo que fosse capaz de realizar a conversão dos relatórios desenvolvidos em Oracle para Jasper. Para atingir este objetivo realizou-se um estudo sobre a forma de funcionamento das duas ferramentas e de armazenamento das informações dos relatórios no formato XML. Neste trabalho identificou-se como sendo a principal diferença entre as duas ferramentas, a forma como elas tratam a questão da criação dos grupos de quebras, principalmente no que diz respeito ao *layout* do relatório.

Para o desenvolvimento do *software* criou-se a restrição de que o relatório possuísse apenas uma *query*, com o foco na conversão de relatórios com *layout* do tipo tabular e mestre/detalhe, tendo em vista que a maioria dos relatórios existentes na NL é deste tipo.

Acredito que este trabalho possa agregar à empresa no sentido de diminuir o tempo que seria utilizado para a conversão manual dos relatórios, bem como a redução de possíveis erros que esta conversão manual poderia gerar. Sendo assim, este projeto contribuirá para a efetiva migração da tecnologia e também, para o melhoramento nos processos de conversão dos relatórios em questão. No entanto, para o uso efetivo desta ferramenta alguns ajustes podem ser realizados, no que diz respeito aos novos padrões adotados para o desenvolvimento dos relatórios na tecnologia JasperReports, como por exemplo, a forma como os relatórios são executados a partir do *software* desenvolvido pela NL.

Pretende-se apresentar o projeto finalizado para a empresa, que já demons-

trou interesse em fazer uma análise do *software* desenvolvido e colocá-lo em pratica com os devidos ajustes.

A realização deste trabalho me possibilitou adquirir conhecimento amplo sobre o funcionamento das duas ferramentas, além de novas funcionalidades presentes na programação Java e formas de trabalhar com arquivos XML, diferentemente das que fazem parte da minha rotina de trabalho na empresa.

Sugiro para trabalhos futuros, verificar a possibilidade de conversão dos relatórios que possuem mais de uma *query.* Outra questão a ser discutida seria com relação a identificar-se uma maneira de converter as funções PL/SQL executadas. Durante a realização de teste utilizando os relatórios existentes percebeu-se que há uma ocorrência muito grande de utilização de campos tipo fórmula, principalmente para concatenação de valores, nestes casos poderiam ser detectadas estas situações e automaticamente criadas as variáveis equivalentes no Jasper.

8 REFERÊNCIAS

AHAMMAD, Shamsuddin. **IReport 3.7.** Packt Publishing, 2010.

BARNFIELD, Louise. **Developer/2000: Build Reports**. Redwood Shores: Oracle Education, 1998.

HEFFELFINGER, David R.. **JasperReports for Java Developers.** Packt Publishing, 2006

IREPORT – **IReport Tutorials & Help 2010**. Disponível em http://jasperforge.org/website/ireportwebsite/IR Website/ir_documentation.html. Acesso em 05/10/2010.

JASPERFORGE – **JasperReports - Schema Reference (version 3.7.6)**. Disponível em: http://jasperforge.org/uploads/publish/jasperreportswebsite/trunk/schema.reference.html, Acesso em 31/10/2010.

JASPERFORGE – **JasperReports Tutorial**. Disponível em http://jasperforge.org/website/jasperreportswebsite/trunk/documentation.html Acesso em 31/10/2010.

MICROSOFT – **Ajuda e instruções do Access 2010**. Disponível em http://office.microsoft.com/pt-br/access-help. Acesso em 05/10/2010.

ORACLE – **Oracle Reports Documentation**. Disponível em http://www.oracle.com/technetwork/middleware/reports/documentation/index.html. Acesso em 05/10/2010.

SIDDIQUI, Bilal. **JasperReports 3.6 Development Cookbook.** Packt Publishing, 2010.

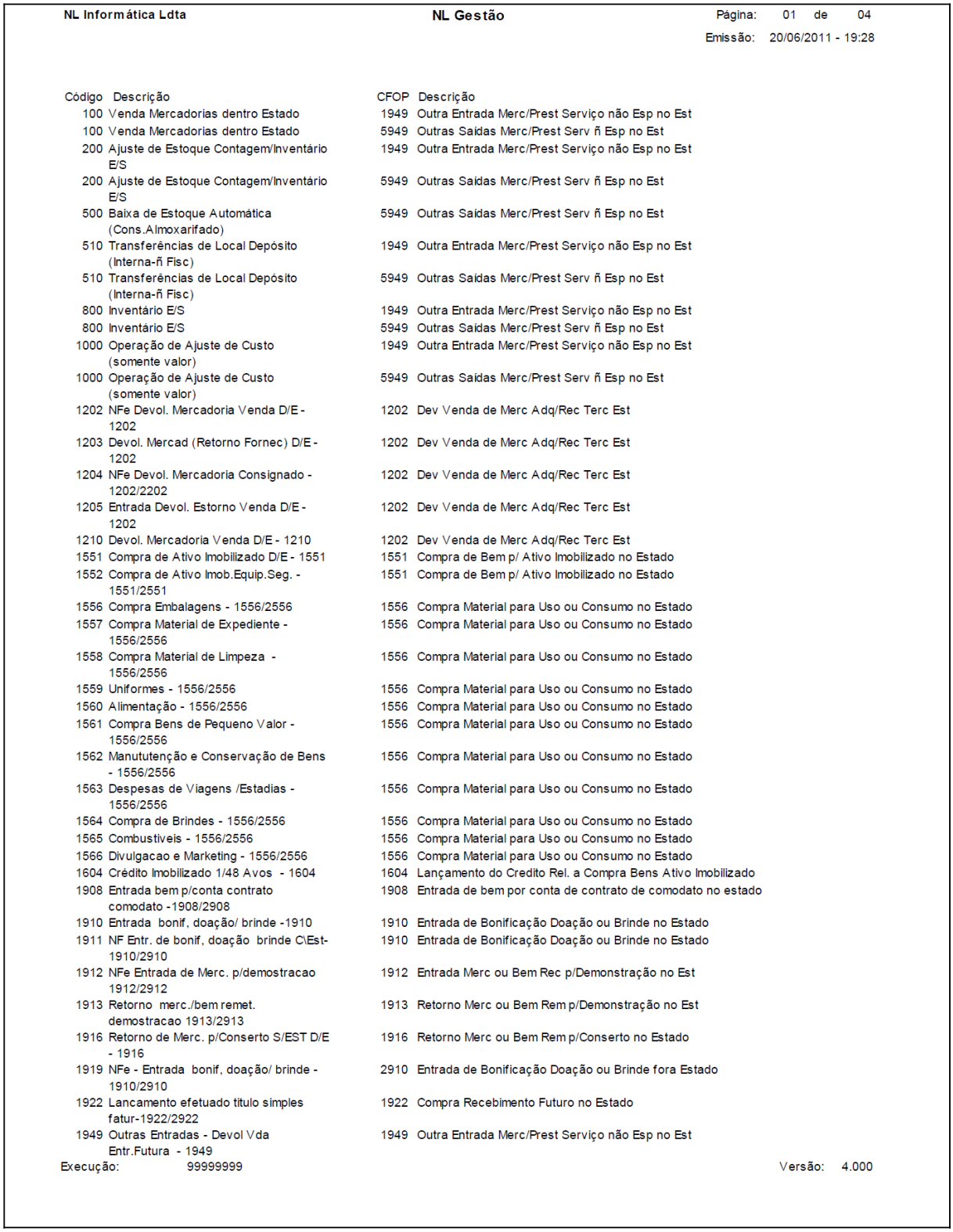
SNEDECOR, Ingrid. **Oracle Reports Building Reports 10g Release 2**. 2005. Disponível em: http://download.oracle.com/docs/cd/B14099_17/bi.1012/b13895.pdf.

SNEDECOR, Ingrid. **Oracle Reports Tutorial, 10g Release 2**. 2005. Disponível em http://download.oracle.com/docs/cd/B14099_17/bi.1012/b14364.pdf.

STAIR, Ralph M.; REYNOLDS, George W.. **Princípios de Sistemas de Informação: uma abordagem gerencial**. 4.ed. Rio de Janeiro: LTC, 2002.

ANEXO A

Execução do relatório CFOP apresentado pelo estudo de caso 1 na seção 6.2. A Figura 99 mostra o relatório executado pelo Oracle Reports e a Figura 100 mostra o relatório executado pelo Jasper.



NL Informática Ldta — NL Gestão — Página: 01 de 04

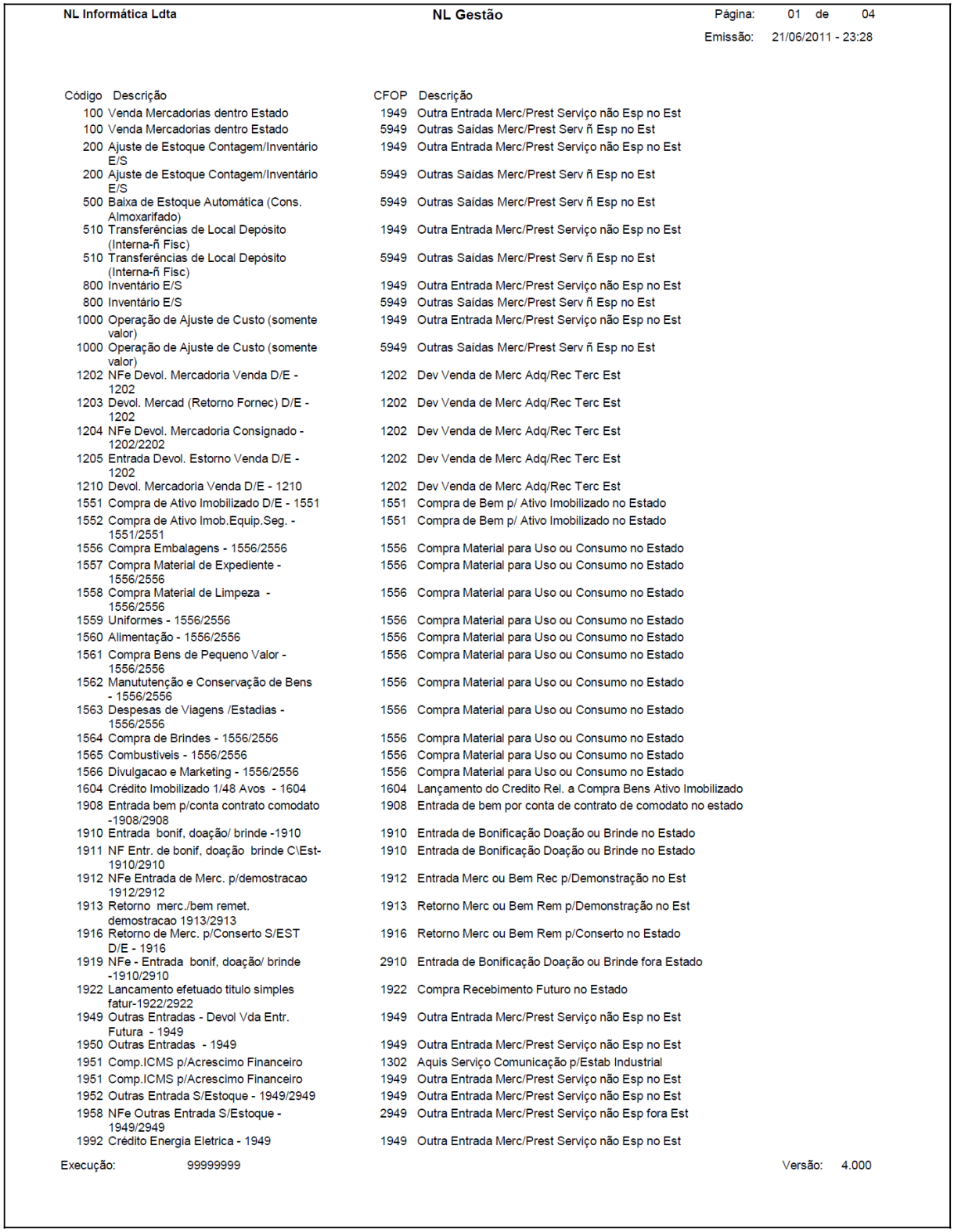
Emissão: 20/06/2011 - 19:28

Código	Descrição	CFOP	Descrição
100	Venda Mercadorias dentro Estado	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
100	Venda Mercadorias dentro Estado	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
200	Ajuste de Estoque Contagem/Inventário E/S	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
200	Ajuste de Estoque Contagem/Inventário E/S	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
500	Baixa de Estoque Automática (Cons.Almoxarifado)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
510	Transferências de Local Depósito (Interna-ñ Fisc)	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
510	Transferências de Local Depósito (Interna-ñ Fisc)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
800	Inventário E/S	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
800	Inventário E/S	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
1000	Operação de Ajuste de Custo (somente valor)	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
1000	Operação de Ajuste de Custo (somente valor)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
1202	NFe Devol. Mercadoria Venda D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1203	Devol. Mercad (Retorno Fornec) D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1204	NFe Devol. Mercadoria Consignado - 1202/2202	1202	Dev Venda de Merc Adq/Rec Terc Est
1205	Entrada Devol. Estorno Venda D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1210	Devol. Mercadoria Venda D/E - 1210	1202	Dev Venda de Merc Adq/Rec Terc Est
1551	Compra de Ativo Imobilizado D/E - 1551	1551	Compra de Bem p/ Ativo Imobilizado no Estado
1552	Compra de Ativo Imob.Equip.Seg. - 1551/2551	1551	Compra de Bem p/ Ativo Imobilizado no Estado
1556	Compra Embalagens - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1557	Compra Material de Expediente - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1558	Compra Material de Limpeza - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1559	Uniformes - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1560	Alimentação - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1561	Compra Bens de Pequeno Valor - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1562	Manututenção e Conservação de Bens - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1563	Despesas de Viagens /Estadias - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1564	Compra de Brindes - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1565	Combustiveis - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1566	Divulgacao e Marketing - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1604	Crédito Imobilizado 1/48 Avos - 1604	1604	Lançamento do Credito Rel. a Compra Bens Ativo Imobilizado
1908	Entrada bem p/conta contrato comodato -1908/2908	1908	Entrada de bem por conta de contrato de comodato no estado
1910	Entrada bonif, doação/ brinde -1910	1910	Entrada de Bonificação Doação ou Brinde no Estado
1911	NF Entr. de bonif, doação brinde C\Est-1910/2910	1910	Entrada de Bonificação Doação ou Brinde no Estado
1912	NFe Entrada de Merc. p/demostracao 1912/2912	1912	Entrada Merc ou Bem Rec p/Demonstração no Est
1913	Retorno merc./bem remet. demostracao 1913/2913	1913	Retorno Merc ou Bem Rem p/Demonstração no Est
1916	Retorno de Merc. p/Conserto S/EST D/E - 1916	1916	Retorno Merc ou Bem Rem p/Conserto no Estado
1919	NFe - Entrada bonif, doação/ brinde - 1910/2910	2910	Entrada de Bonificação Doação ou Brinde fora Estado
1922	Lancamento efetuado titulo simples fatur-1922/2922	1922	Compra Recebimento Futuro no Estado
1949	Outras Entradas - Devol Vda Entr.Futura - 1949	1949	Outra Entrada Merc/Prest Serviço não Esp no Est

Execução: 99999999 — Versão: 4.000

Figura 99: Relatório CFOP executado pelo Oracle

Fonte: autor



NL Informática Ldta — NL Gestão — Página: 01 de 04

Emissão: 21/06/2011 - 23:28

Código	Descrição	CFOP	Descrição
100	Venda Mercadorias dentro Estado	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
100	Venda Mercadorias dentro Estado	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
200	Ajuste de Estoque Contagem/Inventário E/S	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
200	Ajuste de Estoque Contagem/Inventário E/S	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
500	Baixa de Estoque Automática (Cons. Almoxarifado)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
510	Transferências de Local Depósito (Interna-ñ Fisc)	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
510	Transferências de Local Depósito (Interna-ñ Fisc)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
800	Inventário E/S	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
800	Inventário E/S	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
1000	Operação de Ajuste de Custo (somente valor)	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
1000	Operação de Ajuste de Custo (somente valor)	5949	Outras Saídas Merc/Prest Serv ñ Esp no Est
1202	NFe Devol. Mercadoria Venda D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1203	Devol. Mercad (Retorno Fornec) D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1204	NFe Devol. Mercadoria Consignado - 1202/2202	1202	Dev Venda de Merc Adq/Rec Terc Est
1205	Entrada Devol. Estorno Venda D/E - 1202	1202	Dev Venda de Merc Adq/Rec Terc Est
1210	Devol. Mercadoria Venda D/E - 1210	1202	Dev Venda de Merc Adq/Rec Terc Est
1551	Compra de Ativo Imobilizado D/E - 1551	1551	Compra de Bem p/ Ativo Imobilizado no Estado
1552	Compra de Ativo Imob.Equip.Seg. - 1551/2551	1551	Compra de Bem p/ Ativo Imobilizado no Estado
1556	Compra Embalagens - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1557	Compra Material de Expediente - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1558	Compra Material de Limpeza - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1559	Uniformes - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1560	Alimentação - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1561	Compra Bens de Pequeno Valor - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1562	Manututenção e Conservação de Bens - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1563	Despesas de Viagens /Estadias - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1564	Compra de Brindes - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1565	Combustiveis - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1566	Divulgacao e Marketing - 1556/2556	1556	Compra Material para Uso ou Consumo no Estado
1604	Crédito Imobilizado 1/48 Avos - 1604	1604	Lançamento do Credito Rel. a Compra Bens Ativo Imobilizado
1908	Entrada bem p/conta contrato comodato -1908/2908	1908	Entrada de bem por conta de contrato de comodato no estado
1910	Entrada bonif, doação/ brinde -1910	1910	Entrada de Bonificação Doação ou Brinde no Estado
1911	NF Entr. de bonif, doação brinde C\Est-1910/2910	1910	Entrada de Bonificação Doação ou Brinde no Estado
1912	NFe Entrada de Merc. p/demostracao 1912/2912	1912	Entrada Merc ou Bem Rec p/Demonstração no Est
1913	Retorno merc./bem remet. demostracao 1913/2913	1913	Retorno Merc ou Bem Rem p/Demonstração no Est
1916	Retorno de Merc. p/Conserto S/EST D/E - 1916	1916	Retorno Merc ou Bem Rem p/Conserto no Estado
1919	NFe - Entrada bonif, doação/ brinde -1910/2910	2910	Entrada de Bonificação Doação ou Brinde fora Estado
1922	Lancamento efetuado titulo simples fatur-1922/2922	1922	Compra Recebimento Futuro no Estado
1949	Outras Entradas - Devol Vda Entr. Futura - 1949	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
1950	Outras Entradas - 1949	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
1951	Comp.ICMS p/Acrescimo Financeiro	1302	Aquis Serviço Comunicação p/Estab Industrial
1951	Comp.ICMS p/Acrescimo Financeiro	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
1952	Outras Entrada S/Estoque - 1949/2949	1949	Outra Entrada Merc/Prest Serviço não Esp no Est
1958	NFe Outras Entrada S/Estoque - 1949/2949	2949	Outra Entrada Merc/Prest Serviço não Esp fora Est
1992	Crédito Energia Eletrica - 1949	1949	Outra Entrada Merc/Prest Serviço não Esp no Est

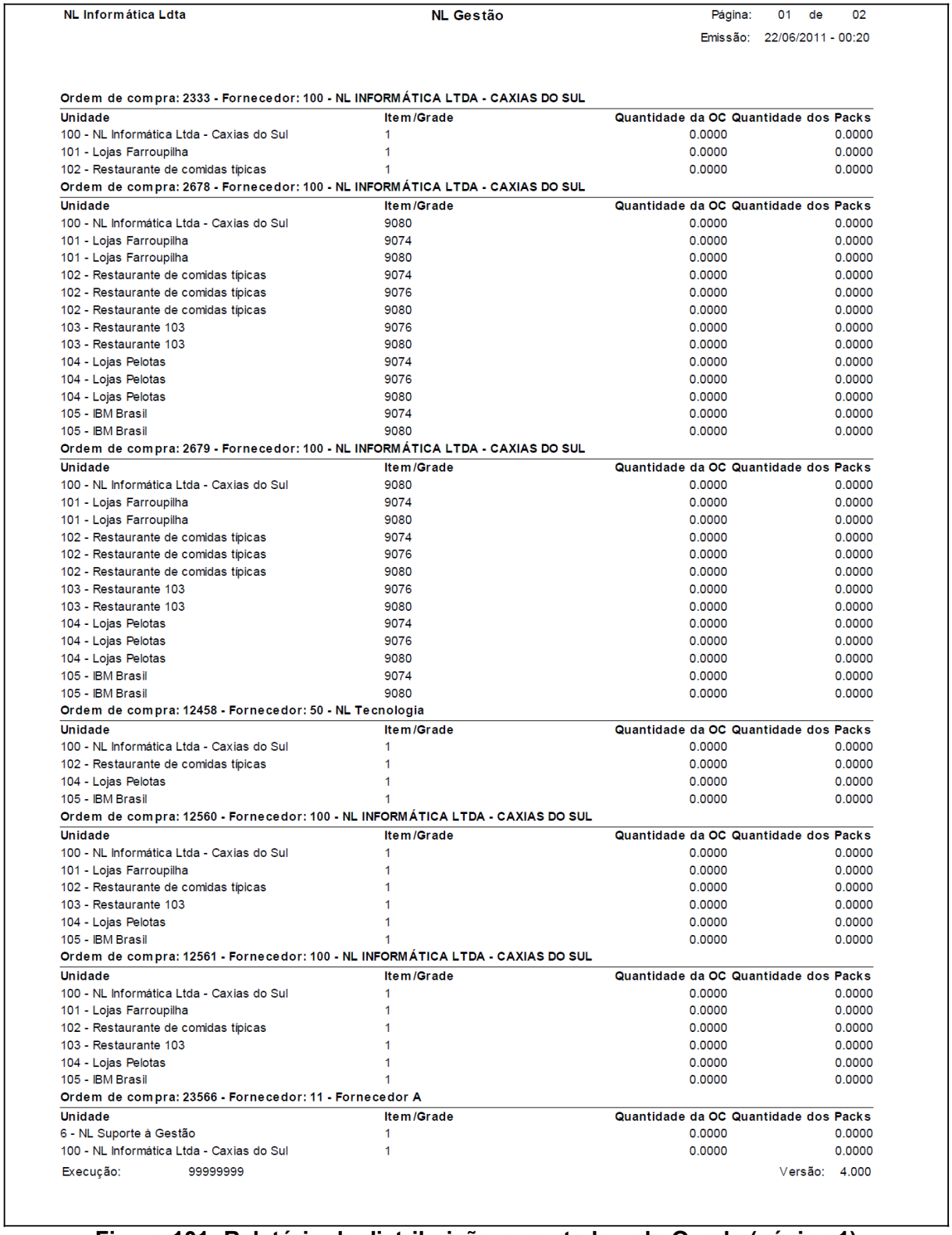
Execução: 99999999 — Versão: 4.000

Figura 100: Relatório CFOP executado pelo Jasper

Fonte: autor

ANEXO B

Execução do relatório de distribuição apresentado pelo estudo de caso 2 na seção 6.3. As figuras 101 e 102 mostram o relatório executado pelo Oracle Reports e as figuras 103 e 104 mostram o relatório executado pelo Jasper.



NL Informática Ldta | NL Gestão | Página: 01 de 02
Emissão: 22/06/2011 - 00:20

Ordem de compra: 2333 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0.0000	0.0000
101 - Lojas Farroupilha	1	0.0000	0.0000
102 - Restaurante de comidas típicas	1	0.0000	0.0000

Ordem de compra: 2678 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	9080	0.0000	0.0000
101 - Lojas Farroupilha	9074	0.0000	0.0000
101 - Lojas Farroupilha	9080	0.0000	0.0000
102 - Restaurante de comidas típicas	9074	0.0000	0.0000
102 - Restaurante de comidas típicas	9076	0.0000	0.0000
102 - Restaurante de comidas típicas	9080	0.0000	0.0000
103 - Restaurante 103	9076	0.0000	0.0000
103 - Restaurante 103	9080	0.0000	0.0000
104 - Lojas Pelotas	9074	0.0000	0.0000
104 - Lojas Pelotas	9076	0.0000	0.0000
104 - Lojas Pelotas	9080	0.0000	0.0000
105 - IBM Brasil	9074	0.0000	0.0000
105 - IBM Brasil	9080	0.0000	0.0000

Ordem de compra: 2679 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	9080	0.0000	0.0000
101 - Lojas Farroupilha	9074	0.0000	0.0000
101 - Lojas Farroupilha	9080	0.0000	0.0000
102 - Restaurante de comidas típicas	9074	0.0000	0.0000
102 - Restaurante de comidas típicas	9076	0.0000	0.0000
102 - Restaurante de comidas típicas	9080	0.0000	0.0000
103 - Restaurante 103	9076	0.0000	0.0000
103 - Restaurante 103	9080	0.0000	0.0000
104 - Lojas Pelotas	9074	0.0000	0.0000
104 - Lojas Pelotas	9076	0.0000	0.0000
104 - Lojas Pelotas	9080	0.0000	0.0000
105 - IBM Brasil	9074	0.0000	0.0000
105 - IBM Brasil	9080	0.0000	0.0000

Ordem de compra: 12458 - Fornecedor: 50 - NL Tecnologia

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0.0000	0.0000
102 - Restaurante de comidas típicas	1	0.0000	0.0000
104 - Lojas Pelotas	1	0.0000	0.0000
105 - IBM Brasil	1	0.0000	0.0000

Ordem de compra: 12560 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0.0000	0.0000
101 - Lojas Farroupilha	1	0.0000	0.0000
102 - Restaurante de comidas típicas	1	0.0000	0.0000
103 - Restaurante 103	1	0.0000	0.0000
104 - Lojas Pelotas	1	0.0000	0.0000
105 - IBM Brasil	1	0.0000	0.0000

Ordem de compra: 12561 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

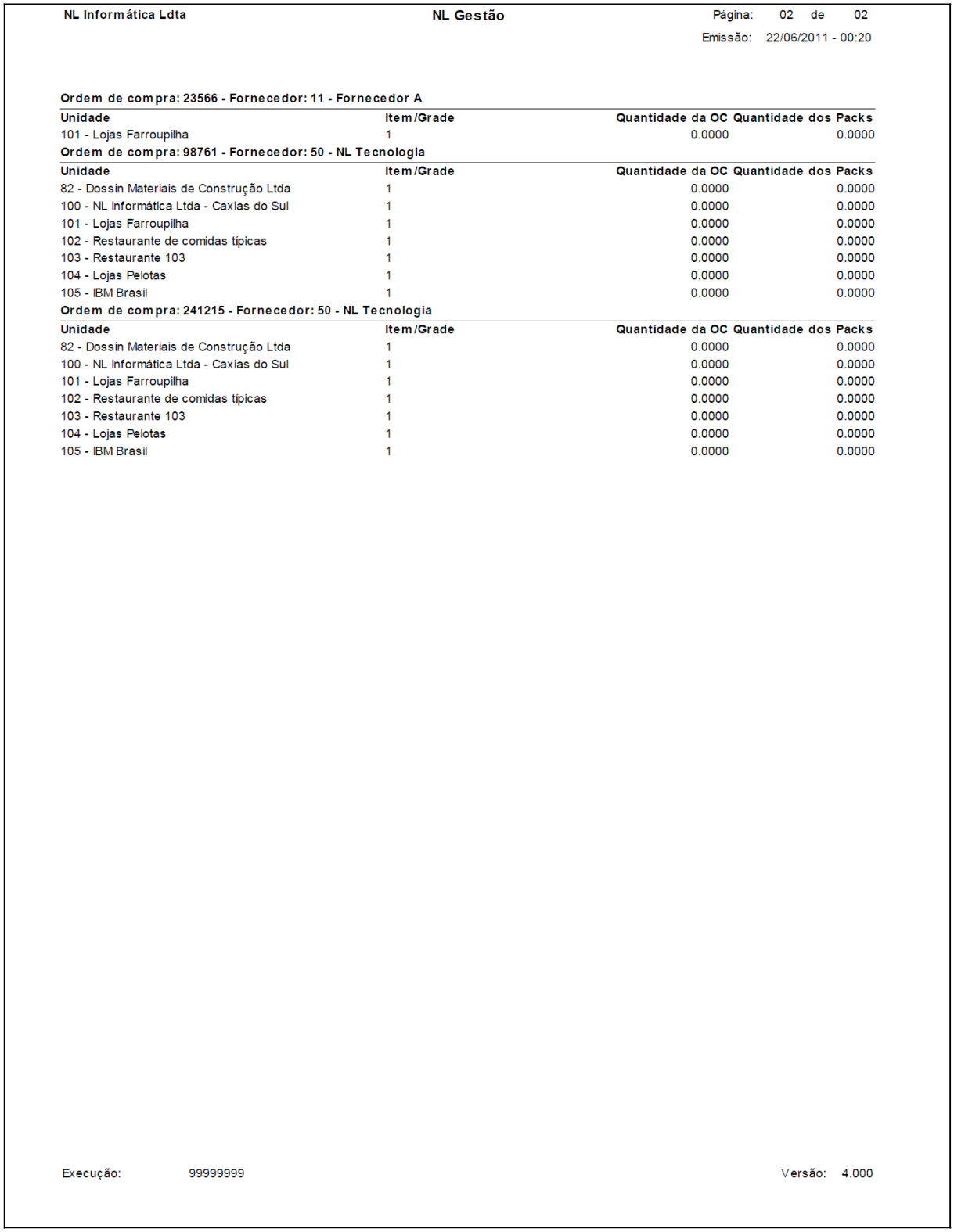
Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0.0000	0.0000
101 - Lojas Farroupilha	1	0.0000	0.0000
102 - Restaurante de comidas típicas	1	0.0000	0.0000
103 - Restaurante 103	1	0.0000	0.0000
104 - Lojas Pelotas	1	0.0000	0.0000
105 - IBM Brasil	1	0.0000	0.0000

Ordem de compra: 23566 - Fornecedor: 11 - Fornecedor A

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
6 - NL Suporte à Gestão	1	0.0000	0.0000
100 - NL Informática Ltda - Caxias do Sul	1	0.0000	0.0000

Execução: 99999999 Versão: 4.000

Figura 101: Relatório de distribuição executado pelo Oracle (página 1)
Fonte: autor



NL Informática Ldta | NL Gestão | Página: 02 de 02
Emissão: 22/06/2011 - 00:20

Ordem de compra: 23566 - Fornecedor: 11 - Fornecedor A

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
101 - Lojas Farroupilha	1	0.0000	0.0000

Ordem de compra: 98761 - Fornecedor: 50 - NL Tecnologia

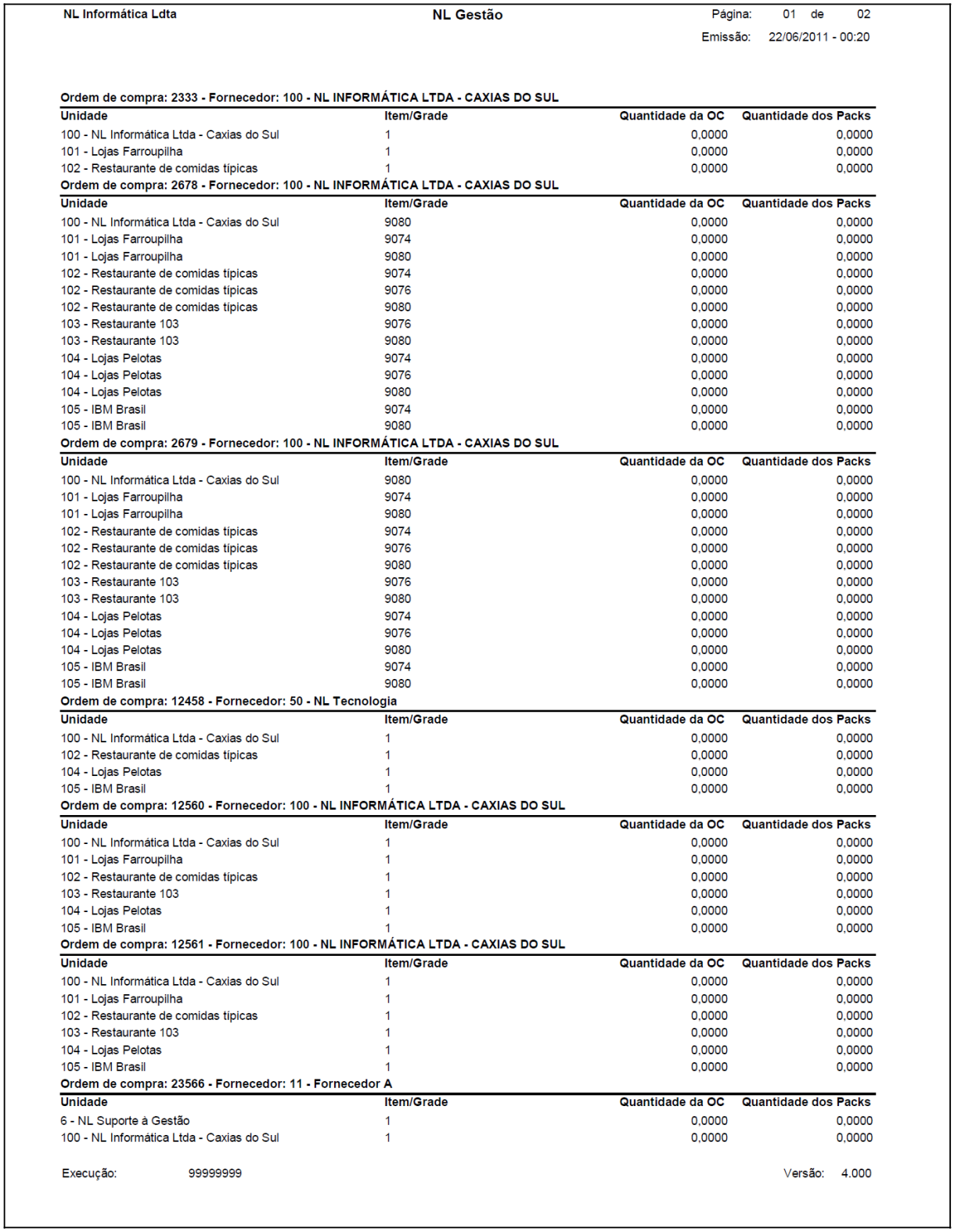
Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
82 - Dossin Materiais de Construção Ltda	1	0.0000	0.0000
100 - NL Informática Ltda - Caxias do Sul	1	0.0000	0.0000
101 - Lojas Farroupilha	1	0.0000	0.0000
102 - Restaurante de comidas típicas	1	0.0000	0.0000
103 - Restaurante 103	1	0.0000	0.0000
104 - Lojas Pelotas	1	0.0000	0.0000
105 - IBM Brasil	1	0.0000	0.0000

Ordem de compra: 241215 - Fornecedor: 50 - NL Tecnologia

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
82 - Dossin Materiais de Construção Ltda	1	0.0000	0.0000
100 - NL Informática Ltda - Caxias do Sul	1	0.0000	0.0000
101 - Lojas Farroupilha	1	0.0000	0.0000
102 - Restaurante de comidas típicas	1	0.0000	0.0000
103 - Restaurante 103	1	0.0000	0.0000
104 - Lojas Pelotas	1	0.0000	0.0000
105 - IBM Brasil	1	0.0000	0.0000

Execução: 99999999 Versão: 4.000

Figura 102: Relatório de distribuição executado pelo Oracle (página 2)
Fonte: autor



NL Informática Ldta — NL Gestão — Página: 01 de 02

Emissão: 22/06/2011 - 00:20

Ordem de compra: 2333 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0,0000	0,0000
101 - Lojas Farroupilha	1	0,0000	0,0000
102 - Restaurante de comidas típicas	1	0,0000	0,0000

Ordem de compra: 2678 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	9080	0,0000	0,0000
101 - Lojas Farroupilha	9074	0,0000	0,0000
101 - Lojas Farroupilha	9080	0,0000	0,0000
102 - Restaurante de comidas típicas	9074	0,0000	0,0000
102 - Restaurante de comidas típicas	9076	0,0000	0,0000
102 - Restaurante de comidas típicas	9080	0,0000	0,0000
103 - Restaurante 103	9076	0,0000	0,0000
103 - Restaurante 103	9080	0,0000	0,0000
104 - Lojas Pelotas	9074	0,0000	0,0000
104 - Lojas Pelotas	9076	0,0000	0,0000
104 - Lojas Pelotas	9080	0,0000	0,0000
105 - IBM Brasil	9074	0,0000	0,0000
105 - IBM Brasil	9080	0,0000	0,0000

Ordem de compra: 2679 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	9080	0,0000	0,0000
101 - Lojas Farroupilha	9074	0,0000	0,0000
101 - Lojas Farroupilha	9080	0,0000	0,0000
102 - Restaurante de comidas típicas	9074	0,0000	0,0000
102 - Restaurante de comidas típicas	9076	0,0000	0,0000
102 - Restaurante de comidas típicas	9080	0,0000	0,0000
103 - Restaurante 103	9076	0,0000	0,0000
103 - Restaurante 103	9080	0,0000	0,0000
104 - Lojas Pelotas	9074	0,0000	0,0000
104 - Lojas Pelotas	9076	0,0000	0,0000
104 - Lojas Pelotas	9080	0,0000	0,0000
105 - IBM Brasil	9074	0,0000	0,0000
105 - IBM Brasil	9080	0,0000	0,0000

Ordem de compra: 12458 - Fornecedor: 50 - NL Tecnologia

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0,0000	0,0000
102 - Restaurante de comidas típicas	1	0,0000	0,0000
104 - Lojas Pelotas	1	0,0000	0,0000
105 - IBM Brasil	1	0,0000	0,0000

Ordem de compra: 12560 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0,0000	0,0000
101 - Lojas Farroupilha	1	0,0000	0,0000
102 - Restaurante de comidas típicas	1	0,0000	0,0000
103 - Restaurante 103	1	0,0000	0,0000
104 - Lojas Pelotas	1	0,0000	0,0000
105 - IBM Brasil	1	0,0000	0,0000

Ordem de compra: 12561 - Fornecedor: 100 - NL INFORMÁTICA LTDA - CAXIAS DO SUL

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
100 - NL Informática Ltda - Caxias do Sul	1	0,0000	0,0000
101 - Lojas Farroupilha	1	0,0000	0,0000
102 - Restaurante de comidas típicas	1	0,0000	0,0000
103 - Restaurante 103	1	0,0000	0,0000
104 - Lojas Pelotas	1	0,0000	0,0000
105 - IBM Brasil	1	0,0000	0,0000

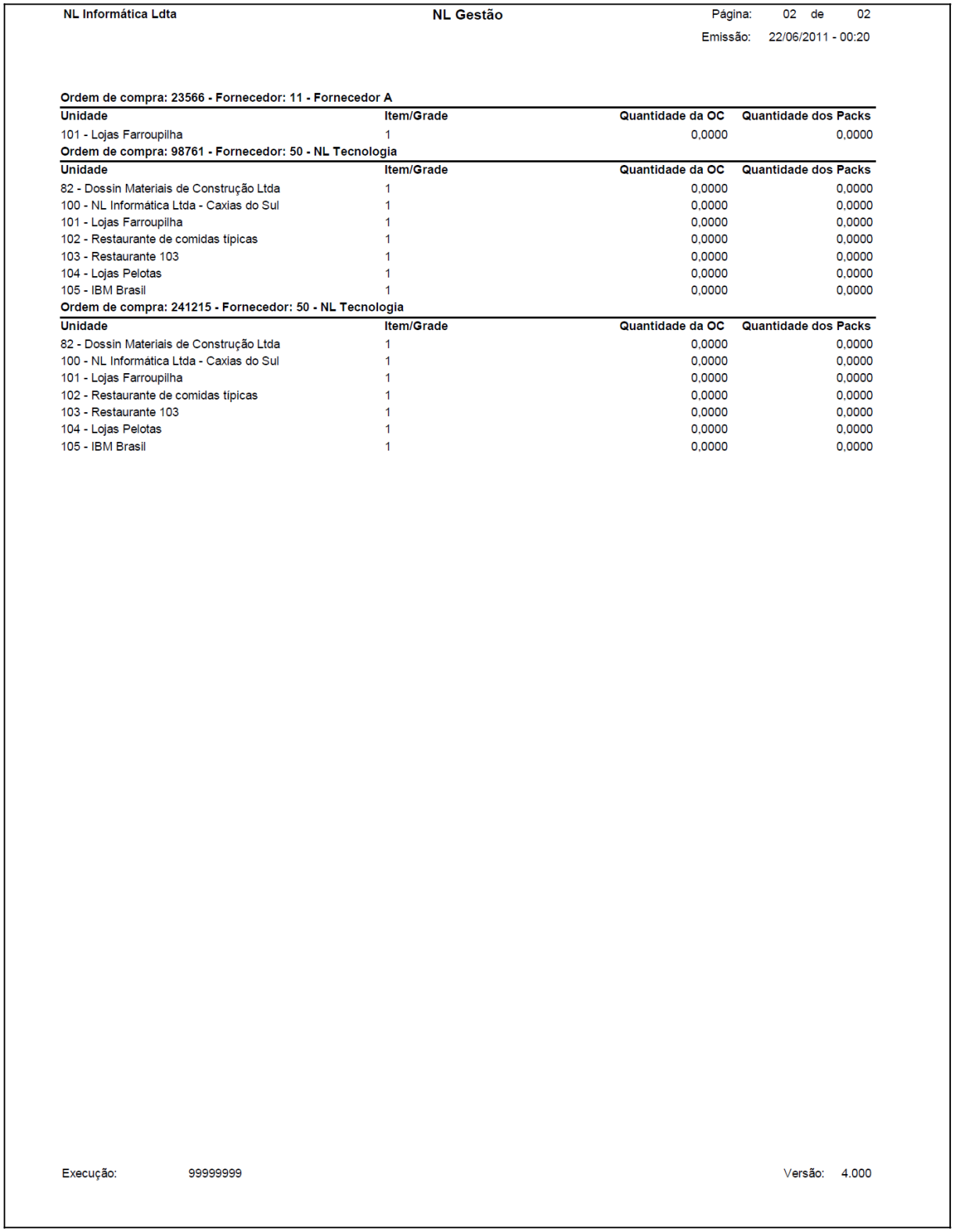
Ordem de compra: 23566 - Fornecedor: 11 - Fornecedor A

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
6 - NL Suporte à Gestão	1	0,0000	0,0000
100 - NL Informática Ltda - Caxias do Sul	1	0,0000	0,0000

Execução: 99999999 — Versão: 4.000

Figura 103: Relatório de distribuição executado pelo Jasper (página 1)

Fonte: autor



NL Informática Ldta | NL Gestão | Página: 02 de 02
Emissão: 22/06/2011 - 00:20

Ordem de compra: 23566 - Fornecedor: 11 - Fornecedor A

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
101 - Lojas Farroupilha	1	0,0000	0,0000

Ordem de compra: 98761 - Fornecedor: 50 - NL Tecnologia

Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
82 - Dossin Materiais de Construção Ltda	1	0,0000	0,0000
100 - NL Informática Ltda - Caxias do Sul	1	0,0000	0,0000
101 - Lojas Farroupilha	1	0,0000	0,0000
102 - Restaurante de comidas típicas	1	0,0000	0,0000
103 - Restaurante 103	1	0,0000	0,0000
104 - Lojas Pelotas	1	0,0000	0,0000
105 - IBM Brasil	1	0,0000	0,0000

Ordem de compra: 241215 - Fornecedor: 50 - NL Tecnologia

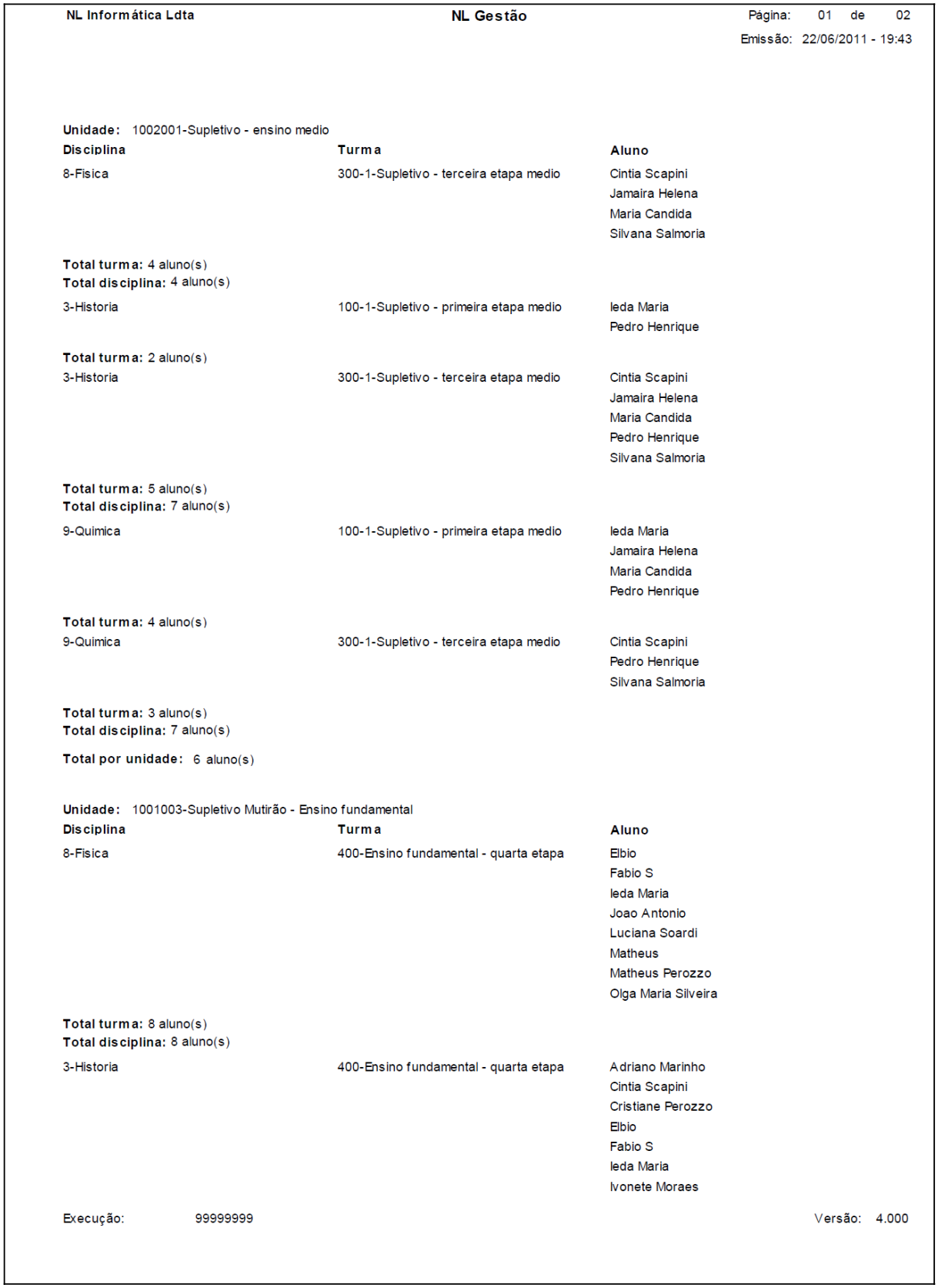
Unidade	Item/Grade	Quantidade da OC	Quantidade dos Packs
82 - Dossin Materiais de Construção Ltda	1	0,0000	0,0000
100 - NL Informática Ltda - Caxias do Sul	1	0,0000	0,0000
101 - Lojas Farroupilha	1	0,0000	0,0000
102 - Restaurante de comidas típicas	1	0,0000	0,0000
103 - Restaurante 103	1	0,0000	0,0000
104 - Lojas Pelotas	1	0,0000	0,0000
105 - IBM Brasil	1	0,0000	0,0000

Execução: 99999999 Versão: 4.000

Figura 104: Relatório de distribuição executado pelo Jasper (página 2)
Fonte: autor

ANEXO C

Execução do relatório de disciplinas apresentado pelo estudo de caso 3 na seção 6.4. As figuras 105 e 106 mostram o relatório executado pelo Oracle Reports, a Figura 107 mostra a primeira execução no Jasper e as figuras 108 e 109 mostram a execução após os ajustes.



NL Informática Ldta — NL Gestão — Página: 01 de 02
Emissão: 22/06/2011 - 19:43

Unidade: 1002001-Supletivo - ensino medio

Disciplina	Turma	Aluno
8-Fisica	300-1-Supletivo - terceira etapa medio	Cintia Scapini
		Jamaira Helena
		Maria Candida
		Silvana Salmoria
Total turma: 4 aluno(s)		
Total disciplina: 4 aluno(s)		
3-Historia	100-1-Supletivo - primeira etapa medio	Ieda Maria
		Pedro Henrique
Total turma: 2 aluno(s)		
3-Historia	300-1-Supletivo - terceira etapa medio	Cintia Scapini
		Jamaira Helena
		Maria Candida
		Pedro Henrique
		Silvana Salmoria
Total turma: 5 aluno(s)		
Total disciplina: 7 aluno(s)		
9-Quimica	100-1-Supletivo - primeira etapa medio	Ieda Maria
		Jamaira Helena
		Maria Candida
		Pedro Henrique
Total turma: 4 aluno(s)		
9-Quimica	300-1-Supletivo - terceira etapa medio	Cintia Scapini
		Pedro Henrique
		Silvana Salmoria
Total turma: 3 aluno(s)		
Total disciplina: 7 aluno(s)		

Total por unidade: 6 aluno(s)

Unidade: 1001003-Supletivo Mutirão - Ensino fundamental

Disciplina	Turma	Aluno
8-Fisica	400-Ensino fundamental - quarta etapa	Elbio
		Fabio S
		Ieda Maria
		Joao Antonio
		Luciana Soardi
		Matheus
		Matheus Perozzo
		Olga Maria Silveira
Total turma: 8 aluno(s)		
Total disciplina: 8 aluno(s)		
3-Historia	400-Ensino fundamental - quarta etapa	Adriano Marinho
		Cintia Scapini
		Cristiane Perozzo
		Elbio
		Fabio S
		Ieda Maria
		Ivonete Moraes

Execução: 99999999 — Versão: 4.000

Figura 105: Relatório de disciplinas executado pelo Oracle (página 1)
Fonte: autor

NL Informática Ldta | NL Gestão | Página: 02 de 02

Emissão: 22/06/2011 - 19:43

Unidade: 1001003-Supletivo Mutirão - Ensino fundamental

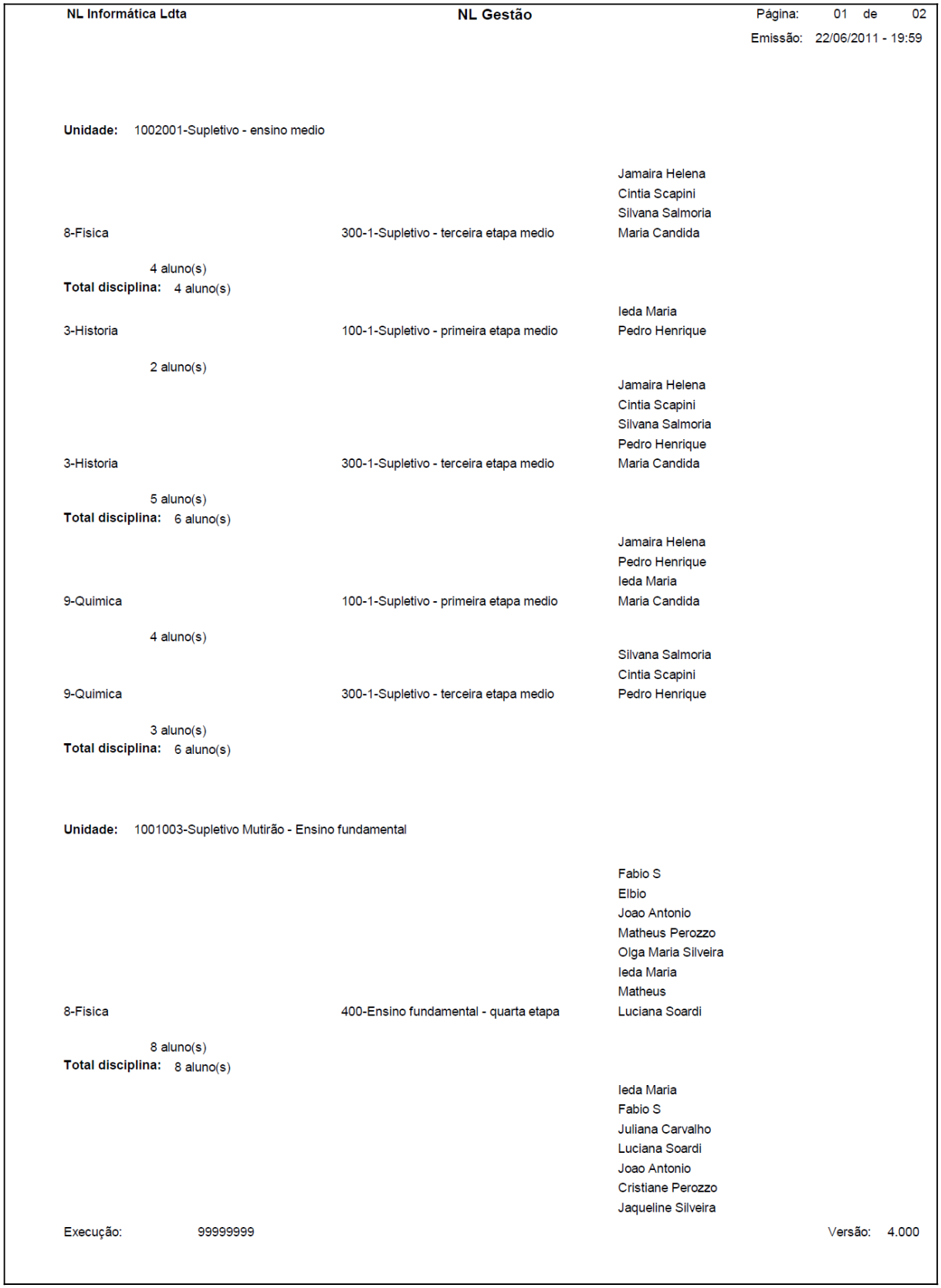
Disciplina	Turma	
3-Historia	400-Ensino fundamental - quarta etapa	Jamaira Helena Janice Ferreira Jaqueline Silveira Joao Antonio Juliana Carvalho Luciana Soardi
Total turma: 13 aluno(s)		
Total disciplina: 13 aluno(s)		
2-Matematica	100-Primeira etapa outra escola	Flavia Terezinha Jamaira Helena Paulo Jose
Total turma: 3 aluno(s)		
2-Matematica	300-Ensino fundamental supletivo	Cristiane Perozzo Flavia Terezinha Luciana Soardi Matheus Silvana Salmoria
Total turma: 5 aluno(s)		
2-Matematica	400-Ensino fundamental - quarta etapa	Adriano Marinho Elbio Ivonete Moraes Juliana Carvalho Maria Candida Pedro
Total turma: 6 aluno(s)		
Total disciplina: 14 aluno(s)		
9-Quimica	200-Ensino fundamental segunda etapa	Ivonete Moraes Jaqueline Silveira Matheus
Total turma: 3 aluno(s)		
9-Quimica	300-Ensino fundamental supletivo	Flavia Terezinha Olga Maria Silveira Silvana Salmoria
Total turma: 3 aluno(s)		
9-Quimica	400-Ensino fundamental - quarta etapa	Adriano Marinho Cintia Scapini Flavia Terezinha Jamaira Helena Maria da Silva Pedro
Total turma: 6 aluno(s)		
Total disciplina: 12 aluno(s)		

Total por unidade: 22 aluno(s)

Total geral: 28 aluno(s)

Execução: 99999999 Versão: 4.000

Figura 106: Relatório de disciplinas executado pelo Oracle (página 2)
Fonte: autor



NL Informática Ldta | NL Gestão | Página: 01 de 02
Emissão: 22/06/2011 - 19:59

Unidade: 1002001-Supletivo - ensino medio

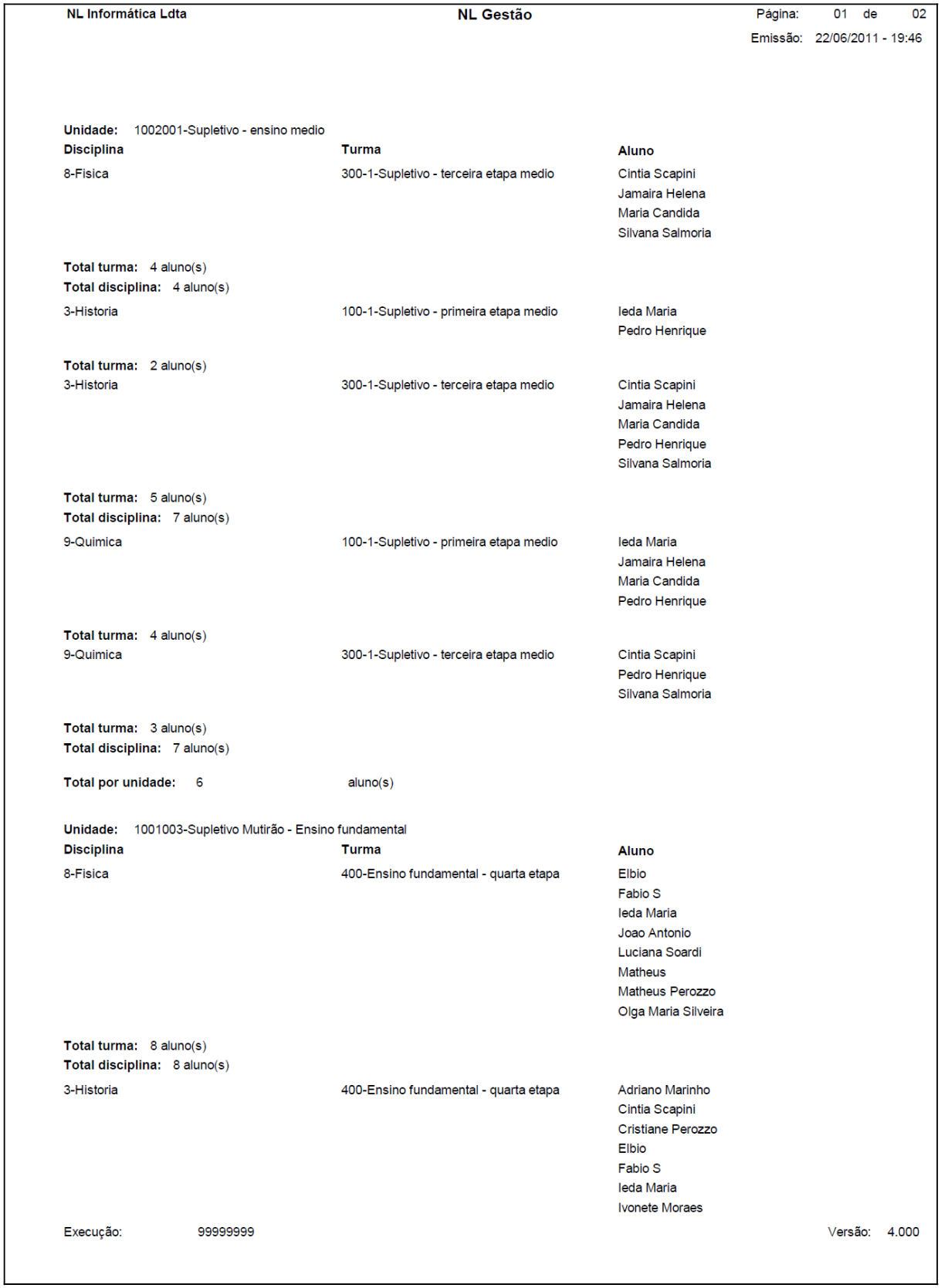
		Jamaira Helena Cintia Scapini Silvana Salmoria
8-Fisica	300-1-Supletivo - terceira etapa medio	Maria Candida
4 aluno(s)		
Total disciplina: 4 aluno(s)		
		Ieda Maria
3-Historia	100-1-Supletivo - primeira etapa medio	Pedro Henrique
2 aluno(s)		
		Jamaira Helena Cintia Scapini Silvana Salmoria Pedro Henrique
3-Historia	300-1-Supletivo - terceira etapa medio	Maria Candida
5 aluno(s)		
Total disciplina: 6 aluno(s)		
		Jamaira Helena Pedro Henrique Ieda Maria
9-Quimica	100-1-Supletivo - primeira etapa medio	Maria Candida
4 aluno(s)		
		Silvana Salmoria Cintia Scapini
9-Quimica	300-1-Supletivo - terceira etapa medio	Pedro Henrique
3 aluno(s)		
Total disciplina: 6 aluno(s)		

Unidade: 1001003-Supletivo Mutirão - Ensino fundamental

		Fabio S Elbio Joao Antonio Matheus Perozzo Olga Maria Silveira Ieda Maria Matheus
8-Fisica	400-Ensino fundamental - quarta etapa	Luciana Soardi
8 aluno(s)		
Total disciplina: 8 aluno(s)		
		Ieda Maria Fabio S Juliana Carvalho Luciana Soardi Joao Antonio Cristiane Perozzo Jaqueline Silveira

Execução: 99999999 Versão: 4.000

Figura 107: Relatório de disciplinas - primeira execução pelo Jasper
Fonte: autor



NL Informática Ldta | NL Gestão | Página: 01 de 02
Emissão: 22/06/2011 - 19:46

Unidade: 1002001-Supletivo - ensino medio

Disciplina	Turma	Aluno
8-Fisica	300-1-Supletivo - terceira etapa medio	Cintia Scapini Jamaira Helena Maria Candida Silvana Salmoria
Total turma: 4 aluno(s)		
Total disciplina: 4 aluno(s)		
3-Historia	100-1-Supletivo - primeira etapa medio	Ieda Maria Pedro Henrique
Total turma: 2 aluno(s)		
3-Historia	300-1-Supletivo - terceira etapa medio	Cintia Scapini Jamaira Helena Maria Candida Pedro Henrique Silvana Salmoria
Total turma: 5 aluno(s)		
Total disciplina: 7 aluno(s)		
9-Quimica	100-1-Supletivo - primeira etapa medio	Ieda Maria Jamaira Helena Maria Candida Pedro Henrique
Total turma: 4 aluno(s)		
9-Quimica	300-1-Supletivo - terceira etapa medio	Cintia Scapini Pedro Henrique Silvana Salmoria
Total turma: 3 aluno(s)		
Total disciplina: 7 aluno(s)		
Total por unidade: 6	aluno(s)	

Unidade: 1001003-Supletivo Mutirão - Ensino fundamental

Disciplina	Turma	Aluno
8-Fisica	400-Ensino fundamental - quarta etapa	Elbio Fabio S Ieda Maria Joao Antonio Luciana Soardi Matheus Matheus Perozzo Olga Maria Silveira
Total turma: 8 aluno(s)		
Total disciplina: 8 aluno(s)		
3-Historia	400-Ensino fundamental - quarta etapa	Adriano Marinho Cintia Scapini Cristiane Perozzo Elbio Fabio S Ieda Maria Ivonete Moraes

Execução: 99999999 Versão: 4.000

Figura 108: Relatório de disciplinas executado pelo Jasper (página 1)
Fonte: autor

NL Informática Ldta | NL Gestão | Página: 02 de 02

Emissão: 22/06/2011 - 19:46

Unidade: 1001003-Supletivo Mutirão - Ensino fundamental

Disciplina	Turma	Aluno
3-Historia	400-Ensino fundamental - quarta etapa	Jamaira Helena Janice Ferreira Jaqueline Silveira Joao Antonio Juliana Carvalho Luciana Soardi
Total turma: 13 aluno(s)		
Total disciplina: 13 aluno(s)		
2-Matematica	100-Primeira etapa outra escola	Flavia Terezinha Jamaira Helena Paulo Jose
Total turma: 3 aluno(s)		
2-Matematica	300-Ensino fundamental supletivo	Cristiane Perozzo Flavia Terezinha Luciana Soardi Matheus Silvana Salmoria
Total turma: 5 aluno(s)		
2-Matematica	400-Ensino fundamental - quarta etapa	Adriano Marinho Elbio Ivonete Moraes Juliana Carvalho Maria Candida Pedro
Total turma: 6 aluno(s)		
Total disciplina: 14 aluno(s)		
9-Quimica	200-Ensino fundamental segunda etapa	Ivonete Moraes Jaqueline Silveira Matheus
Total turma: 3 aluno(s)		
9-Quimica	300-Ensino fundamental supletivo	Flavia Terezinha Olga Maria Silveira Silvana Salmoria
Total turma: 3 aluno(s)		
9-Quimica	400-Ensino fundamental - quarta etapa	Adriano Marinho Cintia Scapini Flavia Terezinha Jamaira Helena Maria da Silva Pedro
Total turma: 6 aluno(s)		
Total disciplina: 12 aluno(s)		

Total por unidade: 22 aluno(s)

Total geral: 28 aluno(s)

Execução: 99999999 Versão: 4.000

Figura 109: Relatório de disciplinas executado pelo Jasper (página 2)
Fonte: autor

ANEXO D

Execução do relatório de pratos apresentado pelo estudo de caso 4 na seção 6.5. As figuras 110 e 111 mostram o relatório executado pelo Oracle Reports e as figuras 112 e 113 mostram o relatório executado pelo Jasper.

NL Informática Ldta | NL Gestão | Página: 01 de 02

Emissão: 23/06/2011 - 11:12

Grupo restaurantes: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Restaurante: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Grupo de pratos: 1 - ARROZ

Pratos

Prato: 108 - ARROZ A GREGA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.4000
6492 - ARROZ PARBOLIZADO	KG	0.0790
310 - CENOURA	KG	0.0150
326 - ERVILHA EM CONSERVA	KG	0.0030
1644 - MORTADELA FATIADA	KG	0.6000
6478 - OLEO DE SOJA 900 ML	LT	0.0020
171 - SAL MOIDO	KG	0.0020
1647 - UVA PASSA PRETA	KG	0.0020

Prato: 10886 - ARROZ C/GERGELIM

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.0000
6492 - ARROZ PARBOLIZADO	KG	0.0400
1659 - GERGELIM	KG	0.0020
6478 - OLEO DE SOJA 900 ML	LT	0.0013
171 - SAL MOIDO	KG	0.0020

Prato: 6 - ARROZ BRANCO

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.0700
290 - ALHO NACIONAL	KG	0.0400
170 - ARROZ POLIDO	KG	0.0800
5053 - LOURO FRESCO	KG	0.0200
6478 - OLEO DE SOJA 900 ML	LT	0.0020
370 - OREGANO	KG	0.0200
171 - SAL MOIDO	KG	0.0020

Grupo de pratos: 2 - FEIJAO

Pratos

Prato: 5305 - FEIJAO PRETO C/CALABRESA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.2200
5768 - ALHO EM PASTA	KG	0.0002
177 - CEBOLA	KG	0.0020
174 - FEIJAO PRETO	KG	0.0400
178 - LINGUICA CALABRESA	KG	0.0050
6478 - OLEO DE SOJA 900 ML	LT	0.0001
370 - OREGANO	KG	0.0001
171 - SAL MOIDO	KG	0.0020

Prato: 5306 - FEIJAO PRETO C/BACON

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.2200
5768 - ALHO EM PASTA	KG	0.0002
292 - BACON	KG	0.0050
177 - CEBOLA	KG	0.0020
174 - FEIJAO PRETO	KG	0.0400
351 - LOURO INDUSTRIALIZADO	KG	0.0001
6478 - OLEO DE SOJA 900 ML	LT	0.0001
370 - OREGANO	KG	0.0001
171 - SAL MOIDO	KG	0.0020

Prato: 6561 - FEIJAO CARIOCA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.2200
5768 - ALHO EM PASTA	KG	0.0002
177 - CEBOLA	KG	0.0020
417 - FEIJAO CARIOCA	KG	0.0400
351 - LOURO INDUSTRIALIZADO	KG	0.0001

Execução: 99999999 Versão: 4.000

Figura 110: Relatório de pratos executado pelo Oracle (página 1)

Fonte: autor

Grupo restaurantes: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Restaurante: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Grupo de pratos: 2 - FEIJAO

Pratos

Prato: 6561 - FEIJAO CARIOCA

Item receita	UM	Quantidade
6478 - OLEO DE SOJA 900 ML	LT	0.0001
370 - OREGANO	KG	0.0001
171 - SAL MOIDO	KG	0.0020

Prato: 6645 - FEIJAO MEXIDO

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.1200
5768 - ALHO EM PASTA	KG	0.0003
177 - CEBOLA	KG	0.0070
183 - FARINHA DE MANDIOCA	KG	0.0070
174 - FEIJAO PRETO	KG	0.0450
351 - LOURO INDUSTRIALIZADO	KG	0.0001
6478 - OLEO DE SOJA 900 ML	LT	0.0001
370 - OREGANO	KG	0.0001
6646 - SAL TEMPERADO SUBPRODUTO	KG	0.0020
412 - TEMPERO VERDE	KG	0.0003

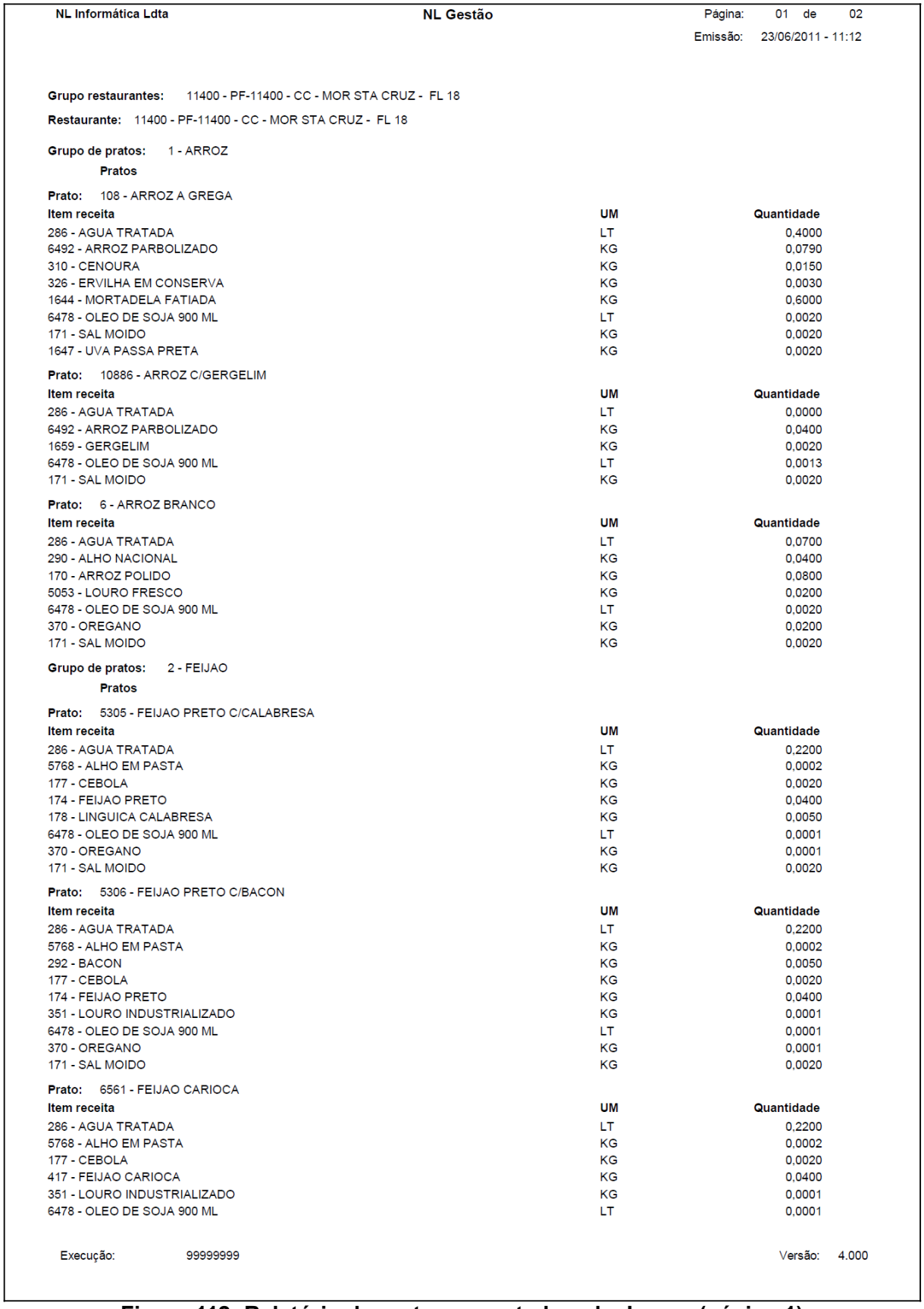
Prato: 7 - FEIJAO PRETO

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.2200
5768 - ALHO EM PASTA	KG	0.0010
177 - CEBOLA	KG	0.0010
174 - FEIJAO PRETO	KG	0.0390
6478 - OLEO DE SOJA 900 ML	LT	0.0001
370 - OREGANO	KG	0.0000
171 - SAL MOIDO	KG	0.0020

Prato: 8 - LENTILHA C/CALABRESA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0.2200
5768 - ALHO EM PASTA	KG	0.0003
293 - BATATA INGLESA BRANCA	KG	0.0060
177 - CEBOLA	KG	0.0023
175 - LENTILHA	KG	0.0400
178 - LINGUICA CALABRESA	KG	0.0008
6478 - OLEO DE SOJA 900 ML	LT	0.0020
171 - SAL MOIDO	KG	0.0020

Figura 111: Relatório de pratos executado pelo Oracle (página 2)
Fonte: autor



NL Informática Ldta — NL Gestão — Página: 01 de 02

Emissão: 23/06/2011 - 11:12

Grupo restaurantes: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Restaurante: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Grupo de pratos: 1 - ARROZ

Pratos

Prato: 108 - ARROZ A GREGA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,4000
6492 - ARROZ PARBOLIZADO	KG	0,0790
310 - CENOURA	KG	0,0150
326 - ERVILHA EM CONSERVA	KG	0,0030
1644 - MORTADELA FATIADA	KG	0,6000
6478 - OLEO DE SOJA 900 ML	LT	0,0020
171 - SAL MOIDO	KG	0,0020
1647 - UVA PASSA PRETA	KG	0,0020

Prato: 10886 - ARROZ C/GERGELIM

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,0000
6492 - ARROZ PARBOLIZADO	KG	0,0400
1659 - GERGELIM	KG	0,0020
6478 - OLEO DE SOJA 900 ML	LT	0,0013
171 - SAL MOIDO	KG	0,0020

Prato: 6 - ARROZ BRANCO

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,0700
290 - ALHO NACIONAL	KG	0,0400
170 - ARROZ POLIDO	KG	0,0800
5053 - LOURO FRESCO	KG	0,0200
6478 - OLEO DE SOJA 900 ML	LT	0,0020
370 - OREGANO	KG	0,0200
171 - SAL MOIDO	KG	0,0020

Grupo de pratos: 2 - FEIJAO

Pratos

Prato: 5305 - FEIJAO PRETO C/CALABRESA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,2200
5768 - ALHO EM PASTA	KG	0,0002
177 - CEBOLA	KG	0,0020
174 - FEIJAO PRETO	KG	0,0400
178 - LINGUICA CALABRESA	KG	0,0050
6478 - OLEO DE SOJA 900 ML	LT	0,0001
370 - OREGANO	KG	0,0001
171 - SAL MOIDO	KG	0,0020

Prato: 5306 - FEIJAO PRETO C/BACON

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,2200
5768 - ALHO EM PASTA	KG	0,0002
292 - BACON	KG	0,0050
177 - CEBOLA	KG	0,0020
174 - FEIJAO PRETO	KG	0,0400
351 - LOURO INDUSTRIALIZADO	KG	0,0001
6478 - OLEO DE SOJA 900 ML	LT	0,0001
370 - OREGANO	KG	0,0001
171 - SAL MOIDO	KG	0,0020

Prato: 6561 - FEIJAO CARIOCA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,2200
5768 - ALHO EM PASTA	KG	0,0002
177 - CEBOLA	KG	0,0020
417 - FEIJAO CARIOCA	KG	0,0400
351 - LOURO INDUSTRIALIZADO	KG	0,0001
6478 - OLEO DE SOJA 900 ML	LT	0,0001

Execução: 99999999 — Versão: 4.000

Figura 112: Relatório de pratos executado pelo Jasper (página 1)

Fonte: autor

Grupo restaurantes: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Restaurante: 11400 - PF-11400 - CC - MOR STA CRUZ - FL 18

Grupo de pratos: 2 - FEIJAO

Pratos

Prato: 6561 - FEIJAO CARIOCA

Item receita	UM	Quantidade
370 - OREGANO	KG	0,0001
171 - SAL MOIDO	KG	0,0020

Prato: 6645 - FEIJAO MEXIDO

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,1200
5768 - ALHO EM PASTA	KG	0,0003
177 - CEBOLA	KG	0,0070
183 - FARINHA DE MANDIOCA	KG	0,0070
174 - FEIJAO PRETO	KG	0,0450
351 - LOURO INDUSTRIALIZADO	KG	0,0001
6478 - OLEO DE SOJA 900 ML	LT	0,0001
370 - OREGANO	KG	0,0001
6646 - SAL TEMPERADO SUBPRODUTO	KG	0,0020
412 - TEMPERO VERDE	KG	0,0003

Prato: 7 - FEIJAO PRETO

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,2200
5768 - ALHO EM PASTA	KG	0,0010
177 - CEBOLA	KG	0,0010
174 - FEIJAO PRETO	KG	0,0390
6478 - OLEO DE SOJA 900 ML	LT	0,0001
370 - OREGANO	KG	0,0000
171 - SAL MOIDO	KG	0,0020

Prato: 8 - LENTILHA C/CALABRESA

Item receita	UM	Quantidade
286 - AGUA TRATADA	LT	0,2200
5768 - ALHO EM PASTA	KG	0,0003
293 - BATATA INGLESA BRANCA	KG	0,0060
177 - CEBOLA	KG	0,0023
175 - LENTILHA	KG	0,0400
178 - LINGUICA CALABRESA	KG	0,0008
6478 - OLEO DE SOJA 900 ML	LT	0,0020
171 - SAL MOIDO	KG	0,0020

Figura 113: Relatório de pratos executado pelo Jasper (página 2)
Fonte: autor