

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA
COMPUTAÇÃO**

CRISTIANO MARCOS LUNKES

**XML PARA CRIAÇÃO DE RELATÓRIOS
DINÂMICOS EM BANCOS DE DADOS RELACIONAIS**

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Rogério Cid Bastos, Dr.
Orientador

Florianópolis, Março de 2003

XML PARA CRIAÇÃO DE RELATÓRIOS DINÂMICOS EM BANCOS DE DADOS RELACIONAIS

CRISTIANO MARCOS LUNKES

Esta dissertação foi julgada adequada para a obtenção do título de mestre em Ciência da Computação Área de Concentração: Sistemas de Conhecimento e aprovada em sua forma final pelo Curso de Pós-Graduação em Ciência da Computação – CPGCC.

Prof. Fernando A. Ostuni Gauthier, Dr. – Coordenador.

Banca Examinadora:

Prof. Rogério Cid Bastos, Dr. – Orientador.

Prof. Raul Sidnei Wazlawick, Dr.

Prof. Luiz Fernando Jacinto Maia, Dr.

A todos que
souberam entender a
minha ausência durante o período de estudos...

Agradecimentos

À Faculdade de Ciências Sociais
Aplicadas de Cascavel.

Ao orientador Prof. Rogério Cid Bastos Dr.Eng,
pelo acompanhamento e profissionalismo no trabalho de orientação.

A Datacooper Software, que me incentivou e tornou possível o
acompanhamento das disciplinas e permitiu minha ausência neste período.

Ao colega de trabalho e sobretudo amigo, Adriano Jedi, que me acompanhou
no desenvolvimento dos trabalhos na empresa.

A Banda Jam Session, que me tira o stress do dia a dia, onde encontrei
amigos reais, entre os shows e viagens.

A minha família, Dulce, Adelino, Juliana, Sandra, Marco e Guilherme, que
sempre me deram todo apoio do mundo.

A minha eterna Fabinha, por me trazer uma nova visão do mundo, e me
iluminar com seus olhos.

Aos MAFIC's, Orlando e Fernando, que me acompanharam durante toda esta
caminhada, passando pelas mesmas pedras, e que foram decisivos para o NOSSO
sucesso e me fizeram acreditar a realização deste trabalho possível.

A Deus e meu Anjo da Guarda, por me permitirem estar, e sempre se
fazerem presentes nesta caminhada...

*“Todos os dias quando acordo, não tenho mais o tempo que passou, mas
tenho muito tempo. Temos todo tempo do mundo....”*
(Renato Russo)

SUMÁRIO

RESUMO	IX
ABSTRACT	X
LISTA DE FIGURAS	XI
DEFINIÇÕES, ACRÔNIMOS E ABREVIações	XII
1 INTRODUÇÃO.....	1
1.1 Considerações Iniciais.....	1
1.2 O Contexto da Pesquisa	2
1.3 O Problema de Pesquisa.....	2
1.4 Objetivos	3
1.4.1 Objetivo geral.....	3
1.4.2 Objetivos específicos.....	4
1.5 Importância do Trabalho.....	4
1.6 Delimitação deste Estudo.....	6
1.7 Estrutura do Trabalho	6
2 FUNDAMENTAÇÃO TEÓRICA	8
2.1 Metadados.....	8
2.2 XML, SQL e Bancos de Dados.....	10
2.3 Entendendo o XML	12
2.3.1 Documentos XML Bem Formados.....	12
2.4 Padronização de Documentos da Linguagem XML.....	14
2.4.1 Document Type Definition – DTD	14
2.4.2 DTD Interno	15
2.4.3 DTD Externo.....	15
2.4.4 XML Schema	15
2.5 Engenharia de Software.....	17
2.5.1 O que é Engenharia de Software ?	17
2.5.2 Por que usar Engenharia de Software ?.....	18

	2.5.3	Paradigmas da Engenharia de Software	19
2.6		Gerenciamento de Projetos	19
	2.6.1	Requisitos e Análise	20
	2.6.2	Projeto e Implementação.....	21
	2.6.3	Teste de Software	22
	2.6.4	Manutenção.....	22
	2.6.5	Qualidade de Software	23
	2.6.5.1	Ferramentas CASE.....	23
	2.6.5.2	Orientação a Objetos.....	23
	2.6.6	Por que usar Orientação a Objetos ?	24
	2.6.7	Considerações Finais	25
2.7		Tecnologias de Desenvolvimento para a Web.....	26
	2.7.1	O que é uma Aplicação Web ?.....	27
	2.7.2	Tecnologias para a Interface	28
	2.7.2.1	HTML.....	28
	2.7.2.2	XML	28
	2.7.2.3	Servlets.....	29
3		PROPOSTA DE GERAÇÃO DE RELATÓRIOS A PARTIR DE BANCOS DE DADOS RELACIONAIS COM XML.....	30
	3.1	Propósito.....	30
	3.2	Escopo.....	30
	3.3	Representação da Arquitetura.....	31
	3.4	Objetivos e Restrições da Arquitetura.....	32
	3.5	Visão do Usuário	33
	3.6	Visão Lógica	33
	3.6.1	WReport	33
	3.6.2	WRelat	34
	3.7	Visão do Processo	37
	3.8	Visão dos Dados.....	37
	3.9	Instalação	38

3.10	A Ferramenta XSLEditor	39
	3.10.1 Propósito	39
	3.10.2 Escopo	39
3.11	Representação da Arquitetura.....	40
	3.11.1 Objetivos e Restrições da Arquitetura	40
	3.11.2 Visão do Usuário	40
	3.11.3 Visão Lógica	41
3.12	Visão do Processo	47
4	CONCLUSÕES E RECOMENDAÇÕES.....	49
4.1	Sugestões para Trabalhos Futuros	50
	REFERÊNCIAS BIBLIOGRÁFICAS	52

RESUMO

LUNKES, Cristiano Marcos. **XML para Criação de Relatórios Dinâmicos em Bancos de Dados Relacionais**. Florianópolis, 2003, 100f. Dissertação em Mestrado – Programa de Pós-Graduação em Ciência da Computação, UFSC, 2003.

Portabilidade, padronização e facilidade de interpretação das estruturas utilizadas para a disponibilização de informações em diferentes plataformas, são os fatos em evidência no que diz respeito aos problemas encontrados na distribuição de documentos entre sistemas ou processos, principalmente nas incompatibilidades existentes nas diversas tecnologias utilizadas e suas particularidades.

O presente trabalho de pesquisa foca a utilização da linguagem XML como meio de disponibilização de relatórios criados a partir de Bancos de Dados Relacionais, de forma independente da plataforma operacional na qual as informações devem ser disponibilizadas ou da qual serão obtidas, assim como do Banco de Dados utilizado, trazendo compatibilidade entre os diversos ambientes de distribuição destas informações.

Centralizar as soluções de extração de informações a partir de Bancos de Dados relacionais com a geração de relatórios, utilizando-se uma linguagem padrão de manipulação de dados, a SQL, e gerando um único documento que pode ser utilizado em diversas plataformas, através de arquivos XML, e ainda, apresentar uma proposta de solução com estas características é o problema descrito nesta pesquisa.

Palavras Chaves: Portabilidade, Padronização, Bancos de Dados Relacionais, Linguagem XML, Estruturação dos Dados.

ABSTRACT

Portability, standardization and comprehension facility of the structures used to publish data in different platforms, are the main issues about the concerns of documents distribution among processes or systems, mostly about the incompatibilities found in various technologies used and its particularities.

In this research work, the mainstream is the use of XML as a way to present reports created from Relational Databases, independent of the operational platform where the information should be shown or where its extracted from, as the Database used, creating a compatibility among the various distributions environments of this information.

Centralize the solution of information extraction from Relational Databases with report generation, using the standard manipulation language, the SQL, and generating a single documento that can be used in many platforms, through the XML files, and more, presents a solution purpose with those characteristics is the subject described in this research

Key Words: Portability, Padronization, Relational DataBases, XML Language, Data Organization.

LISTA DE FIGURAS

Figura 01 - Representação da Arquitetura	43
Figura 02 - Parâmetros do Relatório.....	46
Figura 03 - Relatório em HTML.....	46
Figura 04 - Relatório em TXT.....	47
Figura 05 - Consultas SQL	48
Figura 06 - Representação da Arquitetura – XSL Editor	51
Figura 07 - XSL Editor	53

DEFINIÇÕES, ACRÔNIMOS E ABREVIações

Freeware - Um arquivo, ou mais frequentemente um programa, que é disponibilizado ao público sem custos, também conhecido como de Domínio Público. É um shareware que abertamente distribuído sem que exista uma taxa para o seu registro. É um software livre, do qual podemos fazer o download e utilizá-lo para fazer parte de um processo ou aplicação sem custo algum. Observe contudo, que o freeware é registrado pelo fabricante, o que significa que você não poderá modificá-lo e vendê-lo como se fosse seu próprio software.

XML - Extensible Markup Language (Linguagem de Marcação Extensível), é um padrão emergente na Internet para criação de documentos customizados, com base em tags, similar ao HTML, provendo a construção de blocos para criação de arquivos para publicação on-line e comunicação entre aplicações em rede. O XML pode ser usado por desenvolvedores, grupos ou empresas que quiserem compartilhar informações de uma maneira consistente. Define quais dados serão visualizados e a sua consistência, sem se preocupar com o modo em que serão apresentados para o usuário.

XSL - eXtensible Style Language (Linguagem de Estilos Extensível), uma especificação para preparar a moldura de conteúdo, na criação de páginas XML ou HTML. As especificações funcionam como templates, permitindo aos desenvolvedores aplicar um único estilo em múltiplas páginas. Permite definir como as páginas serão impressas e a maneira de transferir documentos XML através de diferentes plataformas. O XSL define como será o layout da página, ou seja, a forma na qual os dados serão visualizados, sendo que o conteúdo não é levado em consideração.

Parser - Programa que divide as instruções de uma linguagem natural ou de programação em partes menores e mais facilmente interpretadas pelo computador. O parser determina como a frase pode ser contruída pela

gramática da linguagem, produzindo uma saída. O parser também pode verificar se todos os dados de entrada foram passados e se estão descritos de forma correta. Neste caso em particular, irá receber como entrada o arquivo XML e o XSL, e irá processá-los juntos, de modo a compor o resultado final na forma de um documento HTML, PDF ou TXT.

J2EE - Java 2 platform, Enterprise Edition, (Plataforma Java 2, Edição para Aplicações Distribuídas), é uma plataforma da SUN para construção de aplicações distribuídas. Os serviços J2EE são executados na camada do meio entre o navegador do usuário e os bancos de dados distribuídos e as informações dos sistemas legados. J2EE é composto de uma especificação, referência para implementação e um conjunto de regras para teste.

Java - Linguagem de programação independente de plataforma inventada pela SUN, mais conhecida pelo seu uso em grande escala na Web. Diferente das outros softwares, os programas escritos em Java podem ser executados em qualquer plataforma que esta possua uma JVM, ou seja, teoricamente todas as plataformas atuais.

JVM - Java Virtual Machine (Máquina Virtual Java), é um "motor de execução" de software, que executa com segurança e compatibilidade os arquivos de classes Java no processador, independente da plataforma. Significa que as classes java que compõem um sistema, podem ser executadas em qualquer sistema operacional onde a JVM possa ser executada.

Java Servlets - Um programa java que estende a funcionalidade do servidor Web, gerando conteúdo dinâmico e interagindo com os clientes web através da utilização de navegadores que implementam o paradigma request-response (pedido-resposta).

Servidor Web (Tomcat) - Uma máquina que disponibiliza ou serve páginas WEB. Cada servidor WEB tem um endereço IP e possibilita um

nome para o domínio tendo a capacidade de enviar informações a browsers remotos. Provê serviços para acesso a internet, intranet, ou extranet. Um servidor web pode hospedar web sites, suportar HTTP e outros protocolos, e executar programas do lado servidor.

SQL - Structured Query Language (Linguagem de Consulta Estruturada). Linguagem de programação estruturada para enviar consultas a bancos de dados relacionais ou que possuam compatibilidade com este tipo de estrutura para o armazenamento dos dados. Cada aplicação específica tem sua própria versão de SQL, que implementa capacidades diferenciadas, sendo que todos os bancos de dados capazes de compreender o SQL suportam um sub-conjunto padrão do SQL.

Deployment - Processo pelo qual o software é instalado em um ambiente operacional, permitindo que a partir de então seja distribuído e utilizado pelos usuários.

PDF – Portable Document Format (Formato de documento portátil). Um tipo de arquivo criado pela Adobe que combina diversos elementos em um arquivo (texto, gráficos, fontes embutidas, etc) e que pode ser visualizado em qualquer computador de forma muito próxima ao documento original. Tornou-se amplamente popular como meio de publicação de trabalhos e livros na Web.

TXT - Text File (arquivo de texto), sem formatação, contendo apenas os caracteres ASCII. Pode ser lido por vários editores de textos em diversas plataformas. Estes arquivos possuem uma extensão alternativa, .ASC, indicando um formato de arquivo ASCII.

URL - Uniform Resource Locator (Localizador Uniforme de Recursos), é o esquema utilizado na web para localizar uma determinada página ou arquivo.

Plug-in - Software que é acoplado a um aplicativo para ampliar suas funções, como manipular um tipo de arquivo para o qual não existe suporte nativo. Dessa forma os programas vão sendo implementados de acordo com as atualizações que os fabricantes fornecem.

XslEditor- Ferramenta desenvolvida especificamente o objetivo de formatar os arquivos Xsl previamente gerados a partir do componente de relatório, de modo a facilitar e agilizar o processo de edição do layout de relatórios que terão a sua saída em formato PDF ou TXT;

Wrelat (XMLReport)- Principal componente em Java (JBuilder), desenvolvido para prover o mecanismo de geração de relatórios, através do Xml gerado dinamicamente o do Xsl formatado, o que é feito através da invocação aos parsers, retornando o arquivo final diretamente para o browser (visualizador) do usuário.

Wreport- Projeto de um Java Servlet que implementa as consultas SQL que irão compor o relatório, através da utilização do componente Wrelat (XMLReport) , estabelecendo desta forma a comunicação entre o Client e o Server por meio das requisições e respostas (request-response) e dos serviços do browser;

1 INTRODUÇÃO

1.1 Considerações Iniciais

Portabilidade, padronização e facilidade de interpretação das estruturas utilizadas para a disponibilização de informações em diferentes plataformas, são os fatos em evidência no que diz respeito aos problemas encontrados na distribuição de documentos entre sistemas ou processos, principalmente nas incompatibilidades existentes nas diversas tecnologias utilizadas e suas particularidades.

A criação de um formato padronizado e que pudesse ser estendido para atender aos mais variados requisitos, veio através da linguagem XML, que obteve sucesso onde outras tentativas de solução propostas anteriormente falharam, devido a dois principais fatores: auto-descrição, onde a própria informação descreve o seu conteúdo, e portabilidade, fazendo com que não existam fronteiras entre plataformas ou tecnologias. XML é a sigla para eXtensible Markup Language (Linguagem de Marcação eXtensível) desenvolvida pelo World Wide Web Consortium(W3C), para lidar com uma parte não bem resolvida do HTML (Hiper Text Markup Language).

O seu desenvolvimento ocorreu, pelo fato de que, a medida que novas funcionalidades eram incluídas no HTML para resolver os novos e diversos requisitos dos usuários da internet, esta linguagem foi gradativamente se expandindo, aumentando demasiadamente sua complexidade e mesmo assim, não mais atendendo satisfatoriamente aos desenvolvedores. A necessidade de criação de uma linguagem de marcação específica, que não tivesse em si contida todo o peso do HTML se tornou cada vez mais evidente e necessária, sendo então desenvolvida a linguagem XML.

O presente trabalho de pesquisa foca a utilização da linguagem XML como meio de disponibilização de relatórios criados a partir de Bancos de Dados Relacionais, de forma independente da plataforma operacional na qual as informações devem ser disponibilizadas ou da qual serão obtidas, assim como do Banco de Dados utilizado, trazendo compatibilidade entre os diversos ambientes de distribuição destas informações.

1.2 O Contexto da Pesquisa

Quando os primeiros sistemas para gerenciamento de dados foram implementados, ainda antes dos Bancos de Dados Relacionais, sendo estes desenvolvidos sobre Gerenciadores de Arquivos, e a linguagem SQL ainda estava em sua fase de criação, a maior preocupação dos desenvolvedores era de como extrair e disponibilizar as informações obtidas através dos dados registrados, sem que se levasse em consideração a plataforma ou tecnologia na qual esta informação deveria ser apresentada. No momento em questão, os sistemas eram presos tanto na solução utilizada para a persistência dos dados quanto na plataforma na qual iriam ser utilizados.

Com o desenvolvimento e pesquisa de novas tecnologias, o advento da Internet e a crescente necessidade de integração entre sistemas criados em diferentes plataformas, as informações a serem disponibilizadas passaram a ser necessárias nos mais diferentes formatos. Verificando-se que a integração de sistemas e independência de plataforma são requisitos cada vez mais observados ao se pensar um projeto de Banco de Dados e devido a grande diversidade destas tecnologias, o XML foi inicialmente proposto e desde então vem satisfatoriamente cumprindo o seu papel, criando uma ponte entre esses diferentes ambientes.

Com o intuito de propor uma solução de integração, o XML surge com algumas premissas das mais importantes na questão de portabilidade, sendo dentre elas algumas discutidas nesta pesquisa: independência de plataforma, sem que seja necessário se preocupar com o sistema operacional do usuário final; formato facilmente legível tornando mais fácil para desenvolvedores localizar e consertar erros; e extensividade, que é uma maneira dos desenvolvedores adicionarem informações extras sem invalidar os documentos criados anteriormente.

1.3 O Problema de Pesquisa

Com o crescente desenvolvimento e disseminação de tecnologias e novos padrões para o intercâmbio de informações, assim como a disponibilização das mesmas nos mais diversos ambientes, foi paralelamente sendo criada uma complexidade cada vez maior para conseguir a devida portabilidade necessária para a distribuição dos respectivos documentos.

O intercâmbio de informações geradas a partir de Bancos de Dados, em

ambientes híbridos, tornou-se um requisito muito importante como parte das soluções apresentadas pelos desenvolvedores, que tem a missão de tornar possível a distribuição dos dados através das mais diversas tecnologias.

A reimplementação das mesmas funcionalidades de um sistema, como os seus relatórios para cada plataforma na qual as informações devem ser disponibilizadas, tem um custo elevado, desgaste dos desenvolvedores por ser um trabalho repetitivo e redundante, além de aumentar os custos de manutenção, controle de versão e padronização das aplicações.

A dificuldade encontrada para permitir esta ponte entre os diferentes ambientes, foi bruscamente reduzida com a chegada do XML, que de maneira simples, faz a descrição dos dados contidos em um documento em seu próprio arquivo, facilitando o desenvolvimento, reduzindo o tempo com a manutenção, que então é feita de forma centralizada, e possibilitando a criação de extensões para as novas necessidades do sistema, que normalmente surgem durante e após o seu desenvolvimento.

Centralizar as soluções de extração de informações a partir de Bancos de Dados relacionais com a geração de relatórios, utilizando-se uma linguagem padrão de manipulação de dados, a SQL, e gerando um único documento que pode ser utilizado em diversas plataformas, através de arquivos XML, e ainda, apresentar uma proposta de solução com estas características é o problema descrito nesta pesquisa.

1.4 Objetivos

1.4.1 Objetivo geral

O presente trabalho tem como foco principal a investigação sobre as formas de extração e disponibilização de documentos armazenados em B.D's relacionais. Uma das maiores dificuldades encontradas para implementação deste tipo de funcionalidade em uma aplicação é a padronização do modelo de desenvolvimento, que deve ser definida antes de dar início a codificação, uma vez que o código fonte torna-se bastante complexo ao envolver este tipo de geração de documentos. Para isso, foram utilizadas as técnicas de Engenharia de Software e Projetos de Sistemas, conhecimento imprescindível para obter como resultado um bom produto de software.

Neste trabalho é apresentado a geração dos documentos em ambientes operacionais distintos, sua evolução de acordo com o desenvolvimento de novas tecnologias e uma proposta de solução para extração de relatórios em Banco de Dados Relacionais e sua distribuição de forma independente de plataforma.

1.4.2 Objetivos específicos

- ✓ Realizar estudo como base de referência;
- ✓ Ressaltar o uso do XML como solução para apresentação de dados;
- ✓ Propor uma solução para disponibilização de dados entre plataformas distintas;

1.5 Importância do Trabalho

A estrutura de funcionamento e utilização de um Banco de Dados, consiste em disponibilizar um grande poder de processamento de transações para inserção, manutenção e principalmente extração de informações, que é o propósito final de aplicações que se utilizam de recursos desta natureza. Nos últimos cinco anos, contudo, houve uma explosão na demanda para acesso a aplicações de Banco de Dados, tanto em redes locais quanto na própria internet, envolvendo diferentes sistemas operacionais e plataformas. Ficou também evidente a necessidade de suporte a operações de negócios eletrônicos, divulgação de resultados, dados gerenciais, relatórios, gráficos e todo tipo de documento que possa ser disponibilizado rápida, segura e facilmente através de ambientes computacionais distintos., incluindo transações entres sistemas distribuídos, comércio Business to Bussines, e diretamente com o consumidor.

A reação inicial da maioria das empresas que trabalha, direta ou indiretamente, com este tipo de negócios, foi de integrar e disponibilizar este tipo de informação ou processamento, construindo ou adquirindo software como “Servidores de Aplicação” que utilizam protocolos como DCOM ou CORBA para executar tal integração e distribuição.

Mais recentemente, o XML ofereceu a opção de realizar a integração necessária por meio de troca de dados padronizados, de forma muito mais simplificada e barata, sem que seja necessário grandes investimentos tanto para

adquirir as ferramentas quanto para a treinamento e certificação ou contratação de mão de obra qualificada para sua manutenção.

A linguagem de consulta SQL define uma forma padrão de encontrar e extrair dados de um Banco de Dados Relacional, sendo desta forma, é possível escrever uma aplicação middleware que possa extrair estes dados de um Banco e colocá-los em outro ou fazer uma publicação destas informações. Estas soluções são em sua maioria, relativamente complexas e tem uma preocupação muito grande com os esquemas dos Bancos de Dados, diferentes em cada um, e que é necessária para que a implementação funcione adequadamente.

O XML, por outro lado, é simplesmente um “meta-formato” padronizado que pode representar praticamente qualquer tipo de dado, sendo que para tal, esquemas precisos de sua definição são opcionais. Desta forma, o XML se tornou amplamente utilizado para integrar soluções, informações e aplicações locais ou para internet, pelo fato de possuir algumas características que o possibilitam fornecer tal poder de integração e portabilidade, dentre elas:

Padronização: Muitos produtos de vários fornecedores são disponíveis de forma a implementar da forma mais próxima possível da recomendação XML do W3C (Word Wide Web Consortium)

Simplicidade: A tecnologia é fácil de aprender e implementar, com uma estrutura de funcionamento que pode ser estendida e adequada a cada implementação.

Auto Descritivo: A troca direta de dados entre aplicações pode ser escrita sem a referência detalhada de formatos ou esquemas, de forma a não engessar a solução e ao mesmo tempo descrever informações sobre os próprio dados.

Desta forma o XML está redefinindo a estrutura de distribuição de dados e comunicação entre aplicações, inclusive entre Banco de Dados, plataformas, sistemas operacionais, protocolos de rede e de segurança diferentes, servido como a “cola” que liga diversas aplicações umas as outras e entre a internet.

O presente trabalho vem de encontro a crescente necessidade de integração de dados entre as diversas plataformas, descrevendo uma proposta de implementação para geração de relatórios, que pode ser utilizada por aplicações de Banco de Dados, independente de qual seja, para mostrar as funcionalidades e portabilidade que o formato de arquivo XML possibilitou.

1.6 Delimitação deste Estudo

O estudo em questão foi limitado a mostrar uma aplicação dos arquivos em formato XML, como proposta de solução para geração de relatórios criados a partir de Bancos de Dados Relacionais e sua distribuição, sem levar em consideração todas as possibilidades ou requisitos que um relatório pode conter, como gráficos ou customização pelo usuário do sistema onde for utilizado, não incluindo também uma solução completa para todos os tipos de aplicações que necessitam de relatórios.

O trabalho foi desenvolvido com base nas necessidades básicas e mais relevantes para criação e disponibilização de relatórios, não apresentado um estudo completo sobre todos os recursos, assim como delimitado a apresentar as características mais importante do XML, utilizadas para tal proposta, sem se aprofundar em todos os recursos relativos a este formato de arquivos.

Desta forma, a proposta aqui apresentada não resolve todos os problemas ou situações relativas a criação e distribuição de relatórios, mas sim, procura apresentar uma solução básica para o problema, demonstrando as principais funcionalidade que o uso do XML como forma de disponibilização de documentos possibilita.

1.7 Estrutura do Trabalho

O trabalho foi dividido em 5 partes, sendo a estrutura feita de acordo com a seguinte disposição:

O primeiro capítulo mostra as considerações utilizadas como base para desenvolvimento da pesquisa no tema escolhido, mostrando o problema em questão, assim como contextualizando o trabalho e indicando as bases nas quais foi apresentado, e ainda, os objetivos geral e específico, a importância, as limitações e a estrutura de desenvolvimento, propondo de forma prática uma solução para a geração de relatórios independentes de Banco e Plataforma através da utilização do padrão de arquivos XML.

O segundo capítulo apresenta, na forma de uma revisão bibliográfica, o formato de arquivos XML, com sua padronização e extensividade, assim como contextualiza a Engenharia de Software e o desenvolvimento de aplicações Web, descrevendo as práticas utilizadas para a apresentação da proposta de solução com o uso de tais princípios.

No terceiro capítulo, é apresentada uma proposta de solução para o problema levantado, demonstrando a arquitetura criada, assim como uma visão de como foram estruturados os componentes da implementação e também a descrição de seu funcionamento, permitindo desta forma, que o leitor compreenda o processo de criação de relatórios com a utilização destes componentes.

No capítulo 4 são feitas as conclusões e considerações finais sobre o desenvolvimento da pesquisa e também descritas as sugestões para trabalhos futuros relativos a proposta apresentada.

2 FUNDAMENTAÇÃO TEÓRICA

Quando os primeiros sistemas de Banco de Dados foram implementados e a Web estava em processo de criação, não se imaginava que os padrões seriam modificados de forma tão radical, sendo que em seu início era somente possível a apresentação de textos, de forma estática e sem quaisquer outros recursos. Obviamente, isso mudou, e conhecemos não só as páginas estáticas, como também sites dinâmicos e até portais e megaportais. Este fato ocorreu devido a inúmeras ferramentas e linguagens que foram lançadas para este fim sendo que o XML foi quem realmente trouxe uma nova visão e um novo formato à Web e ao intercâmbio de informações entre sistemas. Com todas as possibilidades que proporciona aos desenvolvedores e usuários de sistemas de Bancos de Dados, assim como da Internet, não foi difícil para o XML se tornar uma tendência e rapidamente se espalhar no mercado.

O XML é uma linguagem aberta para a troca de dados entre aplicações e que permite a utilização de schemas, garantindo a sua consistência. Sua grande utilização está na troca de informações entre empresas que o adotam como padrão de comunicação nas aplicações Business-to-Business. Seu grande problema está na velocidade de transferência de dados quando estes são, por exemplo, uma imagem em formato binário, devido ao código se tornar muito extenso e a velocidade ser então penalizada no momento da transferência dos dados. Esta fato já está sendo resolvido, pois está surgindo no mercado um aliado fortíssimo para o XML, o DIME, criado especialmente para a troca pesada de dados.

Uma pesquisa recente, realizada pela Evans Data Corp., confirma a tendência, sendo que, segundo ela., o uso de XML entre os desenvolvedores de todo o mundo cresceu 48,6% desde novembro do ano passado. A pesquisa afirma ainda, que 53,1% dos entrevistados mostraram clara intenção de utilizar a linguagem em 2003.

2.1 Metadados

Metadados são normalmente conhecidos como "informação que descreve informação". Em um ambiente como o da Web, isto pode ser entendido como os dados que podem auxiliar na organização, identificação, segurança, descrição,

localização, recuperação e controle de acesso assim como a proteção dos direitos autorais e de propriedade, de documentos eletrônicos e não eletrônicos. Um relatório extraído de um banco de dados relacional é um exemplo típico de metadados pois, nele existem várias informações que descrevem os recursos armazenados, auxiliando os desenvolvedores em sua utilização

Os metadados tem importantes funções que não estão somente associadas à localização do recurso, podendo abranger também outras propriedades como verificação, controle e gerenciamento do recurso, contando ainda com a padronização de um formato para utilização entre processos(MILSTEAD,1999).

No caso do intercâmbio de dados obtidos para geração de um relatório, sua importância está ligada principalmente a descrição das dados contidos e a estruturação dos mesmos. Fato importante a ser observado no estudo de metadados é que estes são, em geral, extremamente específicos para a utilização de determinados grupos de trabalho. Assim existem metadados para descrição de acervos, localização de recursos, descrição de dados, estruturação das informações. Muitos projetos envolvendo metadados tem sido desenvolvidos nos últimos anos (W3C, 2000).

2.2 XML, SQL e Bancos de Dados.

A grande flexibilidade que o XML disponibiliza, inaugura uma gama de aplicações para este padrão, muito mais amplo do que as oferecidas pelo HTML. O XML não somente pode processar informações complexas e hierárquicas, mas também ser utilizado para transações comerciais. Ele é para a camada de aplicação o que o protocolo TCP/IP é para camada de comunicação. Este fato cria uma base que permite que a Internet seja utilizada para negócios eletrônicos, troca de informações gerenciais, assim como divulgação de documentos de forma a serem compartilhados em ambientes distintos a partir do formato XML.

Este cenário exige alguns requisitos sejam satisfeitos, contemplando a infra estrutura necessária para a padronização e possibilidade de compartilhamento e compatibilidade na troca de informações:

- A informação no formato XML precisa ser disponibilizada de maneira rápida e confiável, e se necessário, em grandes quantidades e fazendo parte de transações.
- Deve ser integrada com dados corporativos existentes e possibilitar que a divulgação e manutenção de informações obtidas através dos dados armazenados seja feita de forma flexível.

O atual estado da arte com relação a divulgação de informações padronizadas através de plataformas distintas, é utilizar Bancos de Dados cuja funcionalidade, consistência, capacidade de reinicialização, segurança e recuperação de dados possa ser utilizada por dados em XML. A forma mais simples de armazenar objetos XML em um Banco de Dados, seria sob forma de “objetos grandes de caracteres”.

Quando os dados estão armazenados no formato XML, pela própria estrutura de armazenamento oferecida por este padrão, é permitido que os dados sejam acessados tanto pela sua estrutura quanto pelo seu conteúdo. Toda informação relevante neste caso deve ser gerenciada em campos ou tabelas separadas. Por outro lado, não seria possível usar este método para mapear dados que estivessem armazenados como textos planos. Se assim fosse, não seria possível extrair informações ou metadados destes arquivos, tornando a consulta ou manipulação

das informações disponibilizadas mais difícil e muito improvável de ser feita de forma automatizada.

Um das melhores formas para integração de XML e sua utilização como forma intermediária para criação e divulgação de documentos, a partir de Bancos de Dados Relacionais, é implementar estruturas XML com o suporte dos modelos de dados que são de uso comum, como o modelo relacional. Assim como estruturas de dados podem ser utilizadas em uma interface XML, um objeto XML pode ser convertido em uma estrutura de dados relacional.

Não é problemático converter dados existentes em Bancos de Dados Relacionais utilizando SQL em arquivos XML. Embora a ausência de tipos de dados no XML ainda constitua um obstáculo no momento, é provável que uma solução esteja disponível em breve na forma de um esquema XML, considerando-se a velocidade de implementação e disponibilização de novas tecnologias, principalmente as utilizadas com propósito de integração e facilidade de intercâmbio de informações entre ambientes distintos.

2.3 Entendendo o XML

A linguagem XML - eXtensible Markup Language, ou Linguagem de Marcação Extendida, pode ser definida como um subconjunto da linguagem SGML -Standard Generalized Markup Language, que permite a criação de uma marcação própria com intuito de especificar idéias e compartilhá-las na grande rede, sendo estabelecida pelo consórcio W3C - World Wide Web Consortium. [Furgeri, 2001]

A linguagem XML também pode ser considerada uma meta-linguagem, o que significa que ela provê recursos para a definição de gramáticas que caracterizam linguagens para classes de documentos específicos, com conjunto de elementos, atributos e regras de composição bem determinadas. [SBC2000, 2000]

A especificação da linguagem XML primou pelos seguintes objetivos [Décio, 2000]:

- Dever ser claro utilizar a estrutura da linguagem XML na Internet;
- A linguagem XML deve suportar uma grande variedade de aplicações;
- A linguagem XML dever ser compatível com a linguagem SGML;
- Os documentos da linguagem XML devem ser legíveis, formais e concisos.

A linguagem XML fornece uma maneira padronizada para descrever a estrutura e propriedades de dados/informação, de forma que possa ser transmitida usando a Internet. Isto significa que os dados podem ser disponibilizados para aplicativos como browsers, bem como outras aplicações que precisam de dados de diferentes fontes, sendo chamado de documento a definição do dado e seu conteúdo. Um documento em linguagem XML apresenta um visual semelhante a um documento na linguagem HTML, sendo também as marcas que caracterizam os elementos delimitadores, iniciadas com o caracter "<" e terminadas com ">".

Ao contrário da linguagem HTML, o XML não se limita apenas a um conjunto fixo de elementos, embora algumas regras sintáticas devam ser cumpridas, sendo que as informações de marcação incluem comentários, referências a caracteres e entidades, delimitadores de seção CDATA, elementos, instruções de processamento e DTD's - Documente Type Definitions, [Furgeri, 2001]

2.3.1 Documentos XML Bem Formados

Além das regras já introduzidas nos itens de descrição de marcações, um

documento bem formado deve respeitar o seguinte:

- Se presente a instrução de processamento da declaração XML deve iniciar o documento;

- Todo documento deve ter um elemento raiz, que inclui todos os demais;

É comum elementos de um documento incluírem outros elementos, formando uma estrutura de aninhamento de elementos. Exemplo:

<i>itálico e negrito</i>

Portanto, um documento XML bem formado é aquele cujas tags estão distribuídas corretamente, isto é, para toda tag de abertura existe sua correspondente de encerramento. Existe uma tag mais externa (a raiz de todo o documento) na qual todas as outras tags e dados estão aninhados. Um documento será bem formado quando um software XML (como um browser) puder interpretá-lo como uma estrutura hierárquica (em forma de árvore). A linguagem XML não é apenas mais uma linguagem de marcação como a linguagem HTML, pois ela possibilita a utilização de vários recursos importantes. A alternativa do desenvolvedor definir marcadores personalizados torna o documento mais inteligente, uma vez que dá significado ao texto armazenado entre os marcadores.

Enquanto a linguagem HTML apenas trata de especificar a formatação de uma palavra ou um trecho de texto, a linguagem XML trata de fornecer seu significado, sendo esse o aspecto mais importante da linguagem XML. [Furgeri, 2001]

O objetivo maior do desenvolvimento da linguagem XML foi possibilitar que os criadores de páginas Web descrevessem suas próprias tags, superando as limitações impostas pela linguagem HTML. Da mesma forma que a linguagem HTML, a linguagem XML é independente da plataforma e não é uma linguagem de formatação, sendo que nenhuma empresa pode monopolizá-la. Os documentos criados em XML pertencem ao seu criador. Uma vez padronizada a estrutura do documento, é possível com a utilização de linguagens de programação, interpretar e manipular o conteúdo do documento.

Um documento XML é composto de 3 elementos:

- Conteúdo dos Dados: são as informações armazenadas entre as tags;
- Estrutura: organização dos elementos dentro do documento;
- Apresentação: forma como as informações são apresentadas.

2.4 Padronização de Documentos da Linguagem XML

A linguagem XML por ser extensível permite a definição de novas tags para documentos dependendo da necessidade de descrever a informação. Se por um lado isso proporciona flexibilidade, por outro lado é certo que conflitos acontecerão quando forem combinados documentos de diversas origens. Além disso, a sintaxe da linguagem XML não foi projetada para determinar em que ordem os elementos poderão aparecer no documento, se existem elementos opcionais, e quais os tipos desses elementos. O mesmo se pode dizer dos atributos. Para ajudar a solucionar esse problema, criou-se o DTD - Document Type Definition

2.4.1 Document Type Definition – DTD

O DTD - Document Type Definition é um conjunto de regras que define quais tipos de dados e entidades farão parte de um documento XML. Estas regras são utilizadas para que o analisador sintático possa verificar se o documento foi gerado de forma correta ou se apresenta erro. [Bender, 2001].

O DTD pode estar definido dentro do próprio arquivo XML ou em um arquivo à parte com extensão .dtd, que deve ser mencionado no código XML. A utilização do DTD pode padronizar um documento XML estabelecendo padrões de marcação para as mais diversas áreas, possibilitando que todas as empresas de um mesmo ramo de atividade (ou até mesmo de um outro), possam se comunicar de maneira unificada.

Para a construção de um DTD, é necessário conhecer todas as características que os documentos XML vão conter, embora nada impeça que futuras alterações possam ser realizadas quando necessário. O DTD deve reconhecer basicamente blocos de construção como [Furgeri, 2001]:

- Elementos: São definidos como blocos de construção utilizados tanto em HTML quanto em XML. Alguns exemplos de elementos em HTML são <BODY>, <TABLE>, sendo que em XML pode se referir a qualquer tag criada.
- Tags: São os elementos marcadores que possuem abertura (“<”) e encerramento (“>”).
- Atributos: Fornecem informações extras sobre os elementos e são inseridos nas tags iniciais de um elemento. Possuem um nome e conteúdo, um exemplo de atributo pode ser

- Entidade: São variáveis designadas para conter textos ou documentos;
- Notações: Existem duas formas de notação, o PCDATA, que possibilita o armazenamento de texto que será interpretado pelo analisador do documento e o CDATA, que armazena o texto não interpretado pelo analisador.

As declarações de elementos e entidades no DTD iniciam-se com os caracteres <! (menor e exclamação) e armazenam as regras que devem ser seguidas na construção dos documentos XML.

2.4.2 DTD Interno

O DTD interno, apesar de não ser a forma mais utilizada, pode ser útil quando surgir a necessidade de se estabelecer regras para um único documento sem que seja preciso utilizar as mesmas regras para elaborar outros documentos. A declaração do tipo de documento interno pode definir todas as regras ou parte delas, podendo também atuar em conjunto com um outro DTD, sendo este externo. O DTD interno é mais maleável do que o DTD externo, pois mesmo que um elemento não seja declarado ele pode ser utilizado no documento, fato que não ocorre em um DTD externo, cujas regras de construção são severamente analisadas. [Décio, 2000]

2.4.3 DTD Externo

Caso o DTD esteja armazenado em um arquivo, ou seja, o DTD é externo ao documento XML.

2.4.4 XML Schema

A linguagem XML Schema (Esquema XML), é uma linguagem nova criada no intuito de permitir que os desenvolvedores definam através de regras, a estrutura, o conteúdo e a semântica de um documento XML. Possui a mesma função que o DTD, porém contempla novas características, o que a torna muito mais poderosa que a DTD.

Em resumo DTD's e XML Schemas possuem a mesma função, descrevendo uma estrutura para documentos XML. A descrição deste tipo de estrutura permite a automação de vários tipos de processos, que envolvem desde da edição de um

simples documento XML por um editor, até a validação de um pedido de compras em uma aplicação e-commerce baseada em XML. A descrição da estrutura de um documento XML pode carregar informações de como os elementos de um documento devem se relacionar, como eles devem ser utilizados e quais atributos eles podem possuir. Desta forma um parser pode determinar através desta estrutura a validade de um documento XML ou seja, a utilização dessa estrutura poderá determinar se um documento XML foi escrito seguindo as regras de uma determinada gramática. [Liberty e Kraley, 2000]

Este processo de determinar se um documento XML está escrito de acordo com as regras de uma DTD ou um XML Schema é denominado de validação. Diante das infinidades de aplicações do processo de validação para DTD's e XML Schemas destacam-se pela praticidade destas linguagens aspectos como:

- A utilização dessas duas linguagens para fundamentar a transformação de um documento XML em outro documento XML,

- Aplicações de e-commerce podem utilizar DTD e/ou XML Schemas para validarem transações efetuadas através de troca de mensagens baseadas em XML. Por exemplo, a linguagem ebXML, é uma linguagem que permite desenvolver aplicações e-business de forma que empresas possam conduzir negócios através da Web, utilizando troca de mensagens baseadas em XML. Dessa forma duas empresas podem comunicar-se através de aplicações baseadas em ebXML, e utilizando-se do processo de validação, as aplicações entre as empresas podem determinar se as mensagens trocadas estão corretas e conseqüentemente determinar se completarão uma transação financeira, validarão um pedido de compra ou enviarão mensagens de erro.

Um editor XML pode utilizar uma DTD ou XML Schema para a criação de todas as estruturas implicitamente necessárias em um documento XML, facilitando assim a edição de um determinado documento XML por parte do desenvolvedor.

Como já citado, o DTD possui uma série de limitações, que o XML Schema não possui. Uma dessas limitações que torna impossível o desenvolvimento de uma série de aplicações é a falta quase completa de tipos de dados, especialmente para determinar o tipo de conteúdo de um elemento. [Liberty e Kraley, 2000] [Décio, 2000]

Através de uma DTD não podemos dizer, que um elemento com o nome PREÇO deva conter um número, muito menos que ele deva ser inteiro, ou com dois

dígitos decimais de precisão. Já com XML Schema pode-se definir estas características, além de se poder definir os tipos para o conteúdo de seus elementos e atributos, como boolean, string, dentre outros. Outro problema levantado é que DTD's têm uma sintaxe própria. Na verdade precisa-se de parsers e APIs – Application Program Interface, separados para controlar DTD's.

Pode-se observar que a sintaxe das DTD's não é a sintaxe da linguagem XML. Já a linguagem XML Schema possui sua sintaxe baseada no XML, o que permite a utilização de todos os recursos da linguagem XML e também do XML Schema. Dentre as dificuldades com o uso do DTD, destaca-se ainda outros problemas que podemos identificar, que são:

- Difícil compreensão e sintaxe complexa;
- Não possui extensibilidade;
- Não suportar Namespaces (especificação que descreve como se pode associar uma URL – Uniform Resource Locator a uma simples tag e atributos em um documento XML);
- Não suportar herança;

XML Schemas é uma tentativa de resolver todos estes problemas, definindo uma nova sintaxe baseada na linguagem XML, para descrever as possíveis estruturas e elementos de um documento da linguagem XML. [Décio, 2000]

2.5 Engenharia de Software

O processo de construção de um sistema é uma atividade de engenharia: Engenharia de Software (ES). Para isso, precisa seguir um conjunto de métodos e técnicas para a correta construção do produto, no caso, um software. O objetivo deste capítulo é descrever os principais métodos, ferramentas e procedimentos na ES, destacando os seus principais aspectos, numa tentativa de oferecer uma visão geral sobre esta área.

2.5.1 O que é Engenharia de Software ?

Bauer (1972) define a Engenharia de Software como: "*O estabelecimento e uso de sólidos princípios de engenharia para que se possa obter um software economicamente viável, que seja confiável e que funcione eficientemente em máquinas reais*". De qualquer forma, Pressman (1995) destaca que, ainda que

várias definições tenham sido dadas à ES, todas reforçam a exigência da disciplina de engenharia no desenvolvimento de software.

Abrange um conjunto de três elementos fundamentais: métodos, ferramentas e procedimentos. Os métodos detalham "como fazer" para se construir o software, as ferramentas proporcionam apoio automatizado ou semi-automatizado aos métodos, e os procedimentos constituem o elo de ligação que mantém juntos os métodos e suas ferramentas, e possibilita um processo de desenvolvimento claro, eficiente, visando garantir ao desenvolvedor e seus clientes a produção de um software de qualidade.

2.5.2 Por que usar Engenharia de Software ?

Nos últimos 20 anos, o hardware deixou de ser o item mais caro na implementação de um sistema, enquanto que o custo relacionado ao software cresceu e se tornou o principal item no orçamento da computação. Isso se deve principalmente à crescente complexidade dos problemas a serem resolvidos pelos softwares. Sistemas como os de geração de relatório e sua posterior publicação em ambientes diversos chegam a possuir milhões de linhas de código e geralmente envolvem uma equipe complexa para o seu desenvolvimento (Degoulet e Fieschi, 1997). Aliado a isso, alguns problemas inerentes ao processo de desenvolvimento de um software começaram a surgir (Pressman, 1995): as estimativas de prazo e de custo freqüentemente são imprecisas, a produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços e a qualidade de software às vezes é menor que a adequada, ocorrendo muito freqüentemente a insatisfação do usuário.

A chave para se vencer esses problemas e dificuldades acima relatados é a larga utilização de uma abordagem de engenharia ao desenvolvimento de software, aliada a uma contínua melhoria das técnicas e ferramentas, no intuito também de melhorar a produtividade da equipe (Pressman, 1995; Degoulet e Fieschi, 1997). Dessa forma, pode-se destacar duas tendências para justificar o uso da ES: primeiro, o software é um item de alto custo e em progressivo aumento e, segundo, os softwares têm um importante papel no bem-estar da sociedade (Boehm, 1981). Dessa forma, a ES assume papel crítico para garantir que tarefas, dados, pessoas e tecnologia estejam apropriadamente alinhadas para produzir um sistema efetivo e

eficiente (Murphy, Hanken e Waters, 1999).

2.5.3 Paradigmas da Engenharia de Software

No processo de desenvolvimento de um software, um conjunto de etapas deve ser definido, denominadas de *Paradigmas da Engenharia de Software* (Pressman, 1995); também conhecido como Modelos de Ciclo de Vida de Software. Destacam-se alguns paradigmas (Davis, 1997): o ciclo de vida clássico, o modelo incremental, o evolucionário, o concorrente, a prototipação e o modelo espiral. Deve ser lembrado ainda que podemos combinar os paradigmas, obtendo-se um melhor resultado

Independentemente do paradigma a ser utilizado, três fases genéricas dividem o processo de desenvolvimento (Pressman, 1995):

.-Definição: esta fase focaliza o " *o quê*" (análise do sistema, planejamento do projeto de software e análise de requisitos).

.-Desenvolvimento: focaliza-se o " *como*" (projeto de software, codificação e realização de testes do software).

.-Manutenção: concentra-se nas " *mudanças*" (correção, adaptação e melhoramento funcional).

É importante destacar que existem no mercado diversas metodologias de ES que criaram novos paradigmas, combinando e aproveitando os melhores conceitos das outras metodologias. Nesse ponto deve-se destacar a metodologia *Rational Unified Process* (RUP) da Rational Inc., como sendo uma das principais metodologias utilizadas atualmente no mundo, juntamente com o XP (eXtreme Programming), que também tem sido cada vez mais utilizado, principalmente para projetos curtos que envolvem uma equipe relativamente pequena.

2.6 Gerenciamento de Projetos

O gerenciamento de projetos deve abranger todo o desenvolvimento, sendo praticado em cada etapa do processo. Uma das primeiras atividades de gerenciamento é o chamado Estudo de Viabilidade. Sua proposta é justificar a necessidade para o desenvolvimento do sistema, tanto do ponto de vista técnico e organizacional como financeiro (custos), através do estudo de índices como Retorno sobre Investimento (Davis, 1998). No gerenciamento, deve-se destacar o uso das

métricas de software que são usadas para medir a qualidade dos softwares e controlar a produtividade dos projetos (Ramamoorthy *et al.*, 1984). As métricas podem ser categorizadas como diretas ou indiretas; da produtividade e da qualidade; e ainda em orientadas ao tamanho (LOC), à função (FP-*Function Point*) e às pessoas (Von Mayrhauser, 1990; Abran e Robillard, 1996).

O planejamento, como atividade de gerenciamento, deve ocorrer baseado em estimativas seguras que, em geral, possuem a experiência passada de outros sistemas como único guia. Surge então a necessidade de efetivamente usar-se métricas de software, para se construir uma grande base de informações, para serem utilizadas em projetos futuros. Diversos modelos empíricos de estimativa foram desenvolvidos, destacando-se o CONstructive COst MOdel (COCOMO) (Boehm, 1981) e o modelo de estimativa de Putnam (Murphy, Hanken e Waters, 1999). Há também modelos gráficos para o controle do cronograma do projeto, tais como Gantt e PERT (Murphy, Hanken e Waters, 1999). Entretanto, o uso de métricas é uma atividade complexa e, em geral, não se tem os registros da experiência passada de softwares similares, tornando essa atividade ainda mais difícil e incerta.

2.6.1 Requisitos e Análise

O primeiro passo na construção de um sistema deve ser o entendimento de "o quê" será desenvolvido, através do levantamento dos requisitos e sua análise. Os requisitos se referem às necessidades dos usuários, do sistema, de custos e prazos. A especificação dos requisitos é de suma importância, pois a maior parte dos erros encontrados durante os testes e a operação dos sistemas são derivados de pouco entendimento ou má interpretação dos requisitos (Ramamoorthy *et al.*, 1984; Degoulet e Fieschi, 1997). Com isso, é de fundamental importância a compreensão total dos requisitos para que se obtenha sucesso no desenvolvimento de um software, gerando benefícios como: a aceitação de todos os envolvidos (usuários, desenvolvedores, etc.), servirá de base para estimativas (custos, prazos, equipe, etc.), melhora da usabilidade, melhoria da manutenibilidade e de outros atributos de qualidade do sistema, cumprimento das metas com os recursos previstos, dentre outros (Dorfman, 1997). A análise de requisitos visa também garantir uma estrutura de dados adequada, para que futuras aplicações, tais como ensaios

clínicos, por exemplo, possam ser implementados e contar com todas as informações necessárias (Dolin, 1997).

Após os requisitos, segue-se a análise do problema a ser informatizado. Existem diversas técnicas para análise e modelagem de sistemas, tais como: análise estruturada, análise orientada a objeto (OOA - *Object-Oriented Analysis*), modelagem de dados, dentre outras. Atualmente destaca-se a OOA que introduziu uma série de novos conceitos. A OOA traz vários benefícios, tais como: funcionalidades complexas podem ser desenvolvidas com uma codificação menor e melhor; um rápido desenvolvimento é alcançado comparado a outros métodos e as aplicações são mantidas mais facilmente (Davis, 1998).

2.6.2 Projeto e Implementação

Enquanto as fases de requisito e análise concentram-se no “o quê” a solução fará, o projeto descreve “como” o software será implementado (Von Mayrhauser, 1990). A fase de projeto também pode ser vista como um aprofundamento da análise caminhando em direção a implementação do sistema. É durante a fase de projeto que a estrutura geral e o estilo do sistema, bem como a sua arquitetura, são definidos (Rumbaugh *et al.* , 1994).

Da mesma forma da análise, existem diversos métodos para o projeto do software, cada qual com o seu conjunto de princípios e notações. Dentre vários pode-se citar: projeto orientado ao fluxo de dados, projeto orientado a objeto (OOD - *Object-Oriented Design*), projeto estruturado e o desenvolvimento estruturado de Jackson (Pressman, 1995).

Com o OOAD é possível a utilização de *Design Patterns* (padrões de projetos) e *Frameworks* (modelos de arquiteturas). *Design Patterns* são estruturas que aparecem repetidamente nos projetos orientados a objeto para resolver um determinado problema de forma flexível e adaptável dinamicamente, trazendo vantagens como: aumento da produtividade e da consistência das aplicações. Podem ser combinados para resolverem problemas mais complicados e, a cada dia, novos padrões de projeto surgem, enriquecendo o conjunto de opções para o desenvolvedor. Um *Framework* pode ser considerado como uma infra-estrutura de classes que oferecem o comportamento necessário para implementar aplicações dentro de um domínio específico, funcionando com um molde para as aplicações

(Gamma, 1995).

Após o projeto, segue-se a codificação, também chamada de implementação. Esta fase é uma simples questão de tradução do projeto para um código, já que as decisões mais difíceis já foram tomadas durante a fase de projeto (Rumbaugh *et al.*, 1994). Existem as ferramentas do tipo *Rapid Application Development* (RAD) que permitem ao usuário um rápido desenvolvimento, baseado em conceitos de reusabilidade e componentização. Java, Visual Basic, Delphi e C++ são algumas das linguagens de programação mais usadas atualmente. Além disso, tecnologias específicas para o desenvolvimento de sistemas na Web, tais como o ASP (*Active Server Pages*), têm sido muito utilizadas nos últimos anos.

2.6.3 Teste de Software

Várias estratégias de testes podem ser implementadas para assegurar que o software está em acordo com suas especificações e livre de erros. Teste de unidade, teste de integração, teste de sistema, teste de instalação e teste de aceitação são exemplos de técnicas que podem ser utilizadas (Von Mayrhauser, 1990). Os mais conhecidos são: o *alpha-test*, no qual o software é testado num ambiente controlado por alguns usuários e na presença dos desenvolvedores; e o *beta-test*, no qual o software é testado por um conjunto maior de usuários, que se propõem a dar um *feedback* aos desenvolvedores, caso alguma irregularidade seja encontrada. Muitas ferramentas CASE (*Computer-Aided Software Engineering*) oferecem suporte automatizado ao processo de teste.

2.6.4 Manutenção

Em geral, a manutenção de software usualmente consome mais de 60% do custo no ciclo de vida de um software. Isso ocorre devido ao fato dos programadores, freqüentemente, serem negligentes durante as fases anteriores a implementação do software (Ramamoorthy *et al.* , 1984). Dessa forma, para uma manutenção mais tranqüila e segura, deve-se utilizar extensamente a ES que garantirá um *design* adequado e escalável para futuras modificações. Durante a manutenção, são realizadas atividades corretivas, adaptativas e preventivas (Pressman, 1995).

2.6.5 Qualidade de Software

A ES é a responsável pelo controle da qualidade, fazendo com que o sistema atenda a todos os requisitos e atributos (Ramamoorthy *et al.*, 1984), assumindo assim papel crítico na produção dos sistemas. A garantia de qualidade de software (*Software Quality Assurance – SQA*) é uma atividade que deve ser aplicada ao longo de todo o processo de desenvolvimento; envolvendo revisões técnicas formais, múltiplas fases de teste, controle da documentação de software e das mudanças, procedimentos para garantir a adequação aos padrões e mecanismos de medição e divulgação (Pressman, 1995).

Um dos métodos mais modernos que visam o aumento da qualidade de software é o modelo proposto pelo *Software Engineering Institute* (SEI) da *Carnegie Mellon University* conhecido como *Capability Maturity Model* (CMM) que visa a melhoria da qualidade de serviços de desenvolvimento de programas de computador. Esse modelo divide em cinco níveis a situação de amadurecimento no processo de desenvolvimento de software de uma organização, bem como oferece os meios de se atingir esses estágios. Segundo o SEI, apenas 1% das empresas se encontram no nível 4 ou no 5 (SEI, 1995; Vaz, 2000).

2.6.5.1 Ferramentas CASE

Nos últimos anos, com a crescente complexidade das metodologias da ES, surgiram as ferramentas CASE (*Computer-Aided Software Engineering - Engenharia de Software auxiliada por computador*). Elas ajudam em todo o processo de desenvolvimento, desde o gerenciamento e análise e até mesmo na codificação. Entretanto, o uso dessas ferramentas ainda é pequeno.

2.6.5.2 Orientação a Objetos

Certamente, a forma mais moderna de abordagem no desenvolvimento de sistemas é a Análise e Projeto Orientado a Objetos (*Object-Oriented Analysis and Design - OOAD*). Rumbaugh (1994) define orientação a objetos como: " *uma nova maneira de pensar os problemas utilizando modelos organizados a partir de conceitos do mundo real. O componente fundamental é o objeto que combina estrutura e comportamento em uma única entidade*". Dizer que um software é orientado a objetos significa que ele é organizado como uma coleção de objetos

separados, que incorporam tanto a estrutura como o comportamento dos dados. A Orientação a Objetos (OO) trouxe vários novos conceitos ao desenvolvimento de software, como: Abstração, Encapsulamento, Objeto, Classe, Atributo, Operação, Método, Mensagem, Evento, Interface, Generalização, Herança e Polimorfismo (Jacobson *et al.*, 1996; Furlan, 1998; Fuzion, 1999).

2.6.6 Por que usar Orientação a Objetos ?

Quando bem empregada, a Orientação a Objetos traz diversas vantagens: reutilização, confiabilidade, modelo de sistema mais realístico, facilidade de interoperabilidade e de manutenção, aumento da qualidade, maior produtividade e unificação do paradigma (da análise a implementação) (Martin e Odell, 1992; Jacobson *et al.*, 1996; Fuzion, 1999). Muitos foram os métodos desenvolvidos para a aplicação da orientação a objetos no processo de análise e projeto. Metodologias como a de Booch, OMT, OOSE, Shlaer/Mellor, Coad/Yourdon, Martin/Odell, Wirfs/Brock e Embley/Kurtz são alguns exemplos (Jacobson *et al.*, 1996; Furlan, 1998). Com o decorrer do tempo, as metodologias de Booch, a OMT (de Rumbaugh) e a OOSE (de Jacobson) evoluíram, seus autores se uniram e lançaram uma linguagem de notação unificada, chamada *Unified Modeling Language* (UML) e também lançaram uma metodologia orientada a objetos chamada *Rational Unified Process* (RUP) (Jacobson, Booch e Rumbaugh, 1999), que abrange todo o processo de desenvolvimento de um sistema.

A Orientação a Objetos (OO) é um paradigma que pode ser aplicado ao longo de todo o processo de construção do software. Dessa forma, tem-se as metodologias acima descritas, que atuam no processo de análise e projeto e, no ciclo de implementação, existem as tecnologias de *back-end* (banco de dados) e as de *front-end* (linguagens e ferramentas de programação) (Furlan, 1998). Os Bancos de Dados têm evoluído no sentido de suportar a tecnologia OO. Inicialmente foram lançados bancos objeto-relacional que suportam apenas alguns dos conceitos OO, mantendo a estrutura do modelo relacional. Em seguida, surgiram os bancos de dados realmente OO, tais como o Jasmine da CA Computer Inc., que suportam, efetivamente, os conceitos OO. Entretanto, devido a questões como a falta de habilidade em OO pelas empresas, dentre outras, esses bancos de dados tiveram pouca penetração no mercado (Belloquim, 2000). Com isso, continuou o domínio no

mercado dos bancos de dados relacionais e objeto-relacionais, forçando de certa forma os desenvolvedores a "quebrar" o paradigma da OO no momento de implementar o banco de dados, tendo-se que utilizar técnicas de mapeamento objeto-relacional para acomodar os dois modelos no sistema.

As primeiras linguagens de programação orientadas a objetos apareceram em meados de 1966, como o Simula e, em 1972, o Smalltalk. Linguagens com maior penetração no mercado, tais com Pascal e C, evoluíram e criaram versões OO, como o C++, por exemplo. Outras linguagens, já criadas dentro do conceito da OO, como o Java, por exemplo, possibilitaram uma maior difusão do uso dessa tecnologia pelo mercado. Viu-se também o rápido crescimento de ambientes de desenvolvimento integrados, que permitem a construção visual dos sistemas de forma rápida (RAD -*Rapid Application Development*) e com uso de componentes previamente montados. São exemplos dessas linguagens/ferramentas o Visual Basic e o Delphi.

2.6.7 Considerações Finais

O uso da Engenharia de Software é uma tarefa difícil e extensa, com diversos métodos, que torna a sua utilização uma atividade para especialistas. Entretanto, não se deve desprezar a sua aplicação. A importância da computação na sociedade moderna tem aumentado o significado do conceito de qualidade de software. Com isso, o desenvolvimento de softwares é uma tarefa fundamental e, em muitos casos, de missão crítica. Como foi comentado, para a construção de softwares de qualidade, uma série de etapas precisam ser seguidas. Portanto, um sistema de PEP precisa de vários passos para o seu desenvolvimento, com uma detalhada análise de requisitos, escolha de um modelo adequado, hardware e software para o auxílio do desenvolvimento, projeto de interface bem definido, consideração dos fatores humanos e participação efetiva dos usuários no processo de desenvolvimento. A consideração desses fatores permitirá a produção de um PEP com qualidade, para que assim, obtenha-se sucesso.

Para finalizar, destacam-se alguns pontos importantes no momento de desenvolver qualquer tipo de sistema (adaptado de Von Mayrhauser, 1990):

- 1) particionamento do desenvolvimento em estágios, escolhendo o modelo de ciclo de vida (paradigma) mais adequado às necessidades;

- 2) utilização de técnicas apropriadas e modernas para a análise e projeto, tais como a Orientação a Objetos;
- 3) seleção da linguagem/ferramenta de programação adequada, considerando a experiência da equipe e pontos como curva de aprendizado e novas tecnologias;
- 4) desenvolvimento de um plano de testes, refinado durante a construção do sistema;
- 5) "Não reinvente a roda". A procura por componentes de software que já possuem o comportamento desejado deve ser encorajada;
- 6) utilização de ambientes de desenvolvimento integrados (RAD, CASE, etc.);
- 7) uso da equipe de forma eficiente, distribuindo as atividades e percebendo as habilidades de cada um;
- 8) larga utilização de padrões, tanto para o desenvolvimento como para o conteúdo dos dados e sua representação;
- 9) consideração da fase de manutenção do sistema, desenvolvendo-o de forma a facilitar a sua manutenção no futuro;
- 10) simplicidade. O produto e a documentação devem ser de fácil entendimento;
- 11) envolvimento do usuário no desenvolvimento do sistema;
- 12) equipe para garantir a qualidade do software;
- 13) necessidade de especialista da área para colaborar ou coordenar o desenvolvimento.

2.7 Tecnologias de Desenvolvimento para a Web

O desenvolvimento de soluções para a Internet utiliza várias tecnologias que interagem entre si para fazer a aplicação funcionar. Tais tecnologias envolvem protocolos de rede, *server-side applications* (aplicações servidoras), bancos de dados e programação de interfaces gráficas para os usuários. CGI, ASP e Servlets são exemplos de tecnologia para o processamento no servidor (*server-side programming*); CORBA, DCOM e Enterprise Java Beans, por sua vez, são exemplos de tecnologias para Objetos Distribuídos; XML, HTML, SGML, XSL, CSS e JavaScript são voltadas para a construção da interface com o usuário via o navegador de páginas para a Web (*browser*, tais como Internet Explorer e

Netscape). Estas e outras tecnologias serão brevemente descritas neste capítulo, tentando-se dessa forma dar uma visão geral do processo de desenvolvimento para a Web.

2.7.1 O que é uma Aplicação Web ?

Essencialmente, uma aplicação Web realiza as seguintes tarefas (Frydrych, 2001) :

1. Disponibiliza uma interface para a entrada de dados.
2. Transmite os dados informados pelo usuário para o Web server.
3. Recebe os dados enviados utilizando algum conjunto de " *middlewares*".
4. Realiza o processamento no servidor (*Server Side Processing*).
5. Transmite os resultados de volta ao cliente.
6. Realiza o processamento no cliente dos dados enviados, mostrando-os ao usuário.

Dessa forma, considera-se as aplicações Web multicamadas. Primariamente, três camadas se destacam, estando sempre presentes em qualquer aplicação Web (Safran e Goldberg, 2000). São elas: camada de apresentação (interface com o usuário); 2) camada *middleware* (objetos e programas *server side*) e 3) camada de banco de dados.

A primeira camada utiliza, em geral, um *Web browser* para interpretar as páginas HTML oriundas do servidor.

A segunda camada, que pode separar camadas de objetos com finalidades específicas, como objetos que tratam das regras de negócio, é a responsável pelo processamento do sistema, recebendo as solicitações do usuário, interagindo com o banco de dados e remetendo as respostas ao usuário na forma de páginas HTML.

A terceira camada é o banco de dados, no qual estão armazenadas as informações do sistema. Acrescenta-se ainda a camada de comunicação, fundamental para que a aplicação funcione. Entretanto, esta camada é transparente para o desenvolvedor, visto que *browser* e o *Web server* (servidor) se encarregam de utilizá-la, sem a necessidade da interferência do desenvolvedor. É na camada de comunicação onde os protocolos de rede atuam. O protocolo utilizado na Internet é o TCP/IP.

2.7.2 Tecnologias para a Interface

A aplicação Web utiliza-se de uma página em HTML, interpretada pelo *browser*, para interagir com o usuário, formando a Camada de Apresentação. Outras tecnologias podem ser misturadas ao HTML para a construção de uma interface mais poderosa, com um visual mais adequado, além de proporcionar recursos que o HTML isoladamente não é capaz. A seguir, um breve resumo sobre essas tecnologias, as quais são responsáveis pela construção da interface com o usuário (Frydrych, 2001):

2.7.2.1 HTML

O *HyperText Markup Language* (HTML) utiliza os conceitos do HyperTexto e da Hipermídia para apresentar, num mesmo ambiente: dados, imagens e outros tipos de mídia, como vídeos, sons e gráficos. O HTML é um subconjunto do *Standard Generalized Markup Language*

2.7.2.2 XML

Extensible Markup Language (XML) é uma linguagem de marcação, tal como o HTML. O XML lida com rótulos (*tags*) sendo possível definir conjuntos de *tags* próprios. A definição do padrão de *tags*, possibilita a criação de documentos num formato XML que podem ser facilmente interpretados pelo *Browser*. Diferentemente do HTML, no XML não há *tags* para a aparência dos dados. O XML é também muito utilizado para padronizar a troca de informações entre sistemas.

Na camada *middleware* (software intermediário), ocorre realmente o trabalho de programação do aplicativo Web, sendo esta camada a responsável por processar a informação enviada pelo cliente (*browser*), processar a regra de negócio (que pode estar em outra camada), interagir com o banco de dados, preparar a resposta (quase sempre na forma de uma página HTML) e enviá-la ao cliente. Os componentes dessa camada estão no *Web Server* e são capazes de utilizar os recursos desses servidores e dos demais recursos conectados para realizar o processamento.

É importante perceber que a forma com que todas essas tecnologias trabalham é similar: recebem uma solicitação do cliente, processam essa solicitação e respondem na forma de uma página HTML. Existem várias tecnologias para a

construção dessa camada. São elas (Zoltán, 2001):

2.7.2.3 Servlets

É um tipo de aplicativo Java que, executado no *Web Server*, permitem um funcionamento similar ao CGI. Os Servlets Java são multiplataforma e oferecem bom desempenho.

3 PROPOSTA DE GERAÇÃO DE RELATÓRIOS A PARTIR DE BANCOS DE DADOS RELACIONAIS COM XML

Como proposta de utilização do XML para geração de relatórios criados a partir de Bancos de Dados Relacionais, foi implementada uma solução independente de fornecedor. baseada em conceitos de ferramentas freeware, reutilização de código, orientação a objetos, internet e tecnologias XML e XSL, e que se integra a um sistema de Banco de Dados, independentemente da tecnologia utilizada para sua implementação, provendo ferramentas com as quais os relatórios do sistema poderão ser criados e emitidos.

3.1 Propósito

“Facilitar e viabilizar a criação e emissão de relatórios independente da arquitetura definida para implementação de um sistema de Banco de Dados e demonstrar a utilização do XML como ponte entre as diversas tecnologias envolvidas entre as partes cliente, servidores e mesmo o middleware .”

Considerando a ausência de uma ferramenta para criação de relatórios que atenda de forma padrão estes requisitos, foi identificada a possibilidade de se propor e implementar uma solução que propiciasse uma forma prática e ágil de desenvolver este tipo de relatórios.

Desta forma os relatórios serão criados em alto nível, deixando de lado a preocupação com a complexidade que envolve a arquitetura existente em camadas inferiores, de forma análoga a outras ferramentas que tem por objetivo a geração de relatórios.

3.2 Escopo

Iremos considerar como tecnologia na qual serão baseadas as descrições a seguir, a J2EE, que é uma arquitetura complexa, sendo que utilizando-a como base, outras arquiteturas mais simples certamente irão ser compatíveis com a solução proposta.

J2EE - Java 2 platform, Enterprise Edition, (Plataforma Java 2, Edição para aplicações distribuídas), é uma plataforma da SUN MICROSYSTEMS para construção de aplicações distribuídas. Os serviços J2EE são executados na camada

do meio entre o navegador do usuário e os bancos de dados distribuídos e as informações dos sistemas legados. J2EE é composto de uma especificação, uma referência para implementação e um conjunto de regras para teste.

A arquitetura da solução proposta envolve diversas camadas interdependentes, que juntas formam a estrutura necessária para o processo de geração e consultas aos relatórios do sistema. A arquitetura definida é baseada em Java e XML, que provê a flexibilidade necessária em termos de plataforma, banco de dados, browser e protocolos de comunicação, possibilitando assim a utilização do resultado final em qualquer ambiente que possua uma JVM.

A ferramenta faz a criação dinâmica dos dados em formato XML e a criação básica do layout em formato XSL, para cada tipo específico de relatório (através do componente WReport), a partir de consultas SQL escritas pelo desenvolvedor. Estas consultas são implementadas em Java Servlets que ficam armazenados no servidor web. O arquivo básico de formatação XSL, é editado através de uma ferramenta de edição visual apropriada (XSLEditor), que dispõe e formata as templates relativas a cada consulta SQL definida. Esta ferramenta foi criada para possibilitar a edição de layout para o formato XSL/FO que é capaz de gerar o resultado final em formato PDF/TXT ou HTML, podendo o mesmo ser visualizado através de um plug-in no browser web. Para abrir o relatório, o usuário precisa acessar a URL correspondente ao Servlet do relatório, passando-se os parâmetros necessários para o mesmo.

Para a visualização final do relatório, o XML e o XSL são mesclados através de um parser, antes de ser enviado para o browser, tornando assim a visualização independente da plataforma e do browser. Quando visualizado, o relatório pode ser impresso diretamente pelo browser ou salvo em disco para referências futuras.

3.3 Representação da Arquitetura

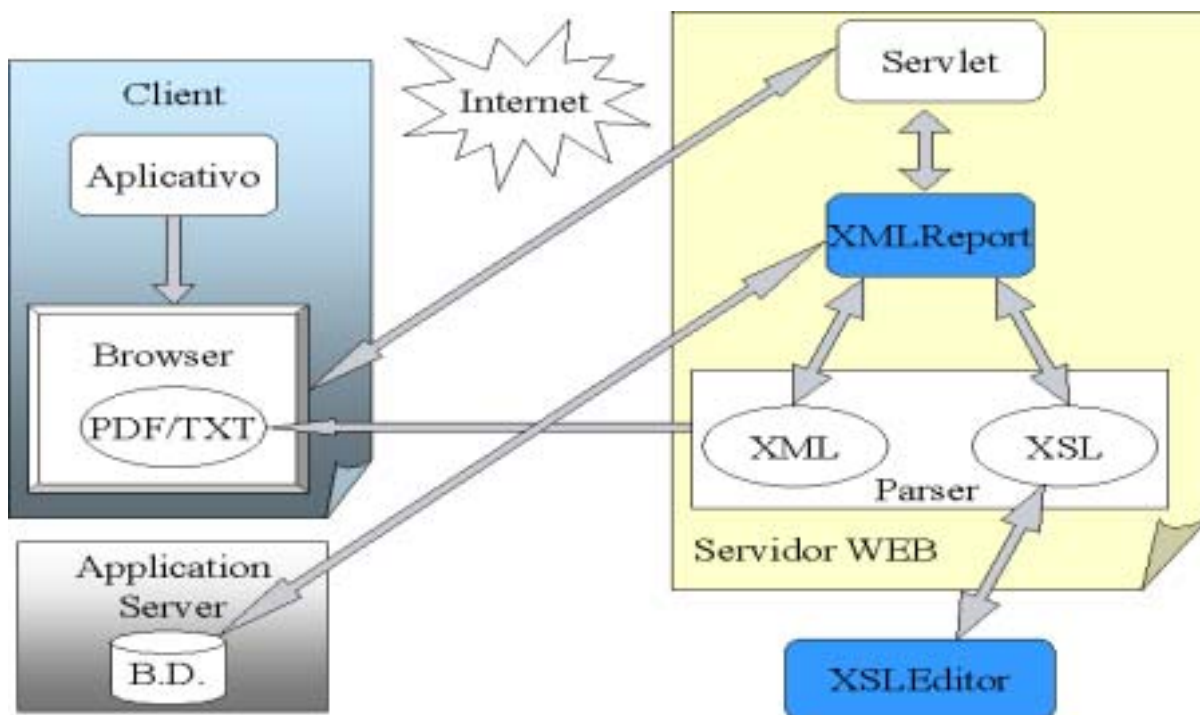


Figura 01 - Representação da Arquitetura

3.4 Objetivos e Restrições da Arquitetura

Com o objetivo inicial de atender as características de portabilidade, segurança e independência de banco de dados, e ainda de forma a tornar prático o desenvolvimento de relatórios, a arquitetura está configurada como no diagrama acima.

O modelo prevê a troca de informações entre o servidor e o cliente, sem a necessidade de utilizar as funções browser onde for feita a visualização, pois o parser separadamente do browser (no servidor), realiza todo o trabalho de mesclagem dos arquivos XML/XSL, retornado apenas o resultado final para o usuário na forma de um arquivo PDF, TXT ou HTML.

O modelo se restringe inicialmente aos tipos de arquivos mencionados, bem como necessita de uma infra-estrutura que dê suporte aos serviços Web (Java Servlets) através do protocolo http, tais como o servidor java da Apache (Jakarta Tomcat), juntamente com os parsers Fop e Saxon. As tecnologias até aqui envolvidas são de distribuição livre, ou seja, não acarretam custos adicionais em termos de desenvolvimento e podem ainda ser substituídas ou aperfeiçoadas em função da evolução da implementação.

Para prover o desenvolvimento necessário dos componentes Wreport (*Servlet*) e *XmlReports* (*Java Bean*), foi utilizada a ferramenta JBuilder da Borland, e para o desenvolvimento da ferramenta XSLEditor foi utilizado o Delphi, também da Borland.

3.5 Visão do Usuário

Do ponto de vista do usuário final, o sistema dispara um evento passando alguns dados como parâmetro para o servidor Web, através de um navegador, que chama o Servlet responsável pela geração do relatório. O Servlet realiza a consulta necessária no banco, gerando um arquivo XML, que será mesclado com a XSL correspondente através do parser, finalmente retornando os dados para o usuário no formato definido (PDF, TXT ou HTML). Em caso de não haver dados relativos à consulta solicitada, o retorno será apenas uma página contendo uma mensagem explicativa.

3.6 Visão Lógica

Sobre a etapa desenvolvida no Jbuilder, temos como principais componentes:

3.6.1 WReport

O projeto definido como *wreport* está estruturado de forma a conter a definição de uma única classe, o servidor de relatórios, implementado na forma de um Servlet. A compilação deste pacote cria um arquivo *.jar* e um arquivo *.war* através do qual será possível fazer a instalação no servidor Web do Application Server.

Esta classe foi criada para testar a funcionalidade do componente *XmlReport*, sendo que para cada relatório a ser criado, será necessário a definição de classes equivalentes, mas que de forma geral será muito próxima desta, onde a diferença será as consultas realizadas. Como regra geral, a classe precisa criar a conexão com o banco de dados e criar uma instância do componente *XmlReport*, passando para este a conexão. Em seguida o desenvolvedor do relatório necessita definir suas consultas e adicioná-las ao componente.

Além disso, esta classe será a responsável por fornecer a conexão com o banco de dados através da qual os dados serão obtidos, e também por criar uma instância da classe *XmlReport*

O Servlet precisa receber o parâmetro *TpXSL*, que especifica o nome do arquivo XSL que será usado na formatação do relatório e o *tpSaida*, que especifica o tipo do arquivo gerado (PDF/HTM). Além destes, outros parâmetros opcionais relacionados ao relatório em questão (Códigos, intervalo de datas, setores, filiais, etc...) poderiam ser informados.

3.6.2 WRelat

O projeto definido como *wrelat* está estruturado de forma a conter a definição de duas classes. A classe *XMLReport* é a principal delas, e que se utiliza da classe *RegNo*, que controla a disposição hierárquica das consultas SQLs relacionadas ao relatório bem como a criação de campos totalizadores e a inter-relação entre as mesmas.

O trecho de código a seguir mostra como seria a utilização do componente bem como as declarações necessárias ao seu funcionamento. Neste caso o código implementa o corpo de um Servlet.

Para invocar este Servlet o usuário precisa chamar o browser de navegação passando para o mesmo uma URL com a seguinte definição:

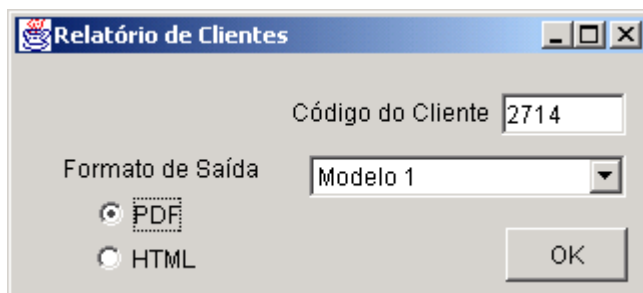
<http://192.168.0.112:8080/WebApp/servlet/wreport.ServSaxon?CodCliente=2714&TpXSL=formato.xml&tpSaida=pdf>

Esta URL é composta de três partes principais, detalhadas a seguir:

http://192.168.0.112:8080/ - especifica o host e a porta onde o servidor web (TomCat) está ativo, a espera de chamadas aos seus serviços.

WebApp/servlet/wreport.ServSaxon – especifica o caminho do pacote e a classe correspondente ao Servlet em questão.

?CodCliente=2714&TpXSL=formato.xml&tpSaida =pdf – especifica os parâmetros passados para o Servlet juntamente com os respectivos valores associados. No caso, o código do cliente, o nome do arquivo XSL que será usado para formatar o relatório e a especificação do tipo de arquivo gerado (no caso, PDF).



Relatório de Clientes

Código do Cliente: 2714

Formato de Saída: Modelo 1

PDF

HTML

OK

Figura 02 - Parâmetros do Relatório

A URL será gerada a partir de uma interface com o usuário, semelhante à figura acima, onde todos os parâmetros necessários serão informados. A partir daí, o relatório será visualizado conforme figura 3, considerando-se que o Browser não necessariamente precisa ser o MS Internet Explorer, desde que exista um plug-in para o formato PDF, caso a opção de saída tenha sido esta. O plug-in não será necessário quando o formato de saída especificado for o padrão HTML.

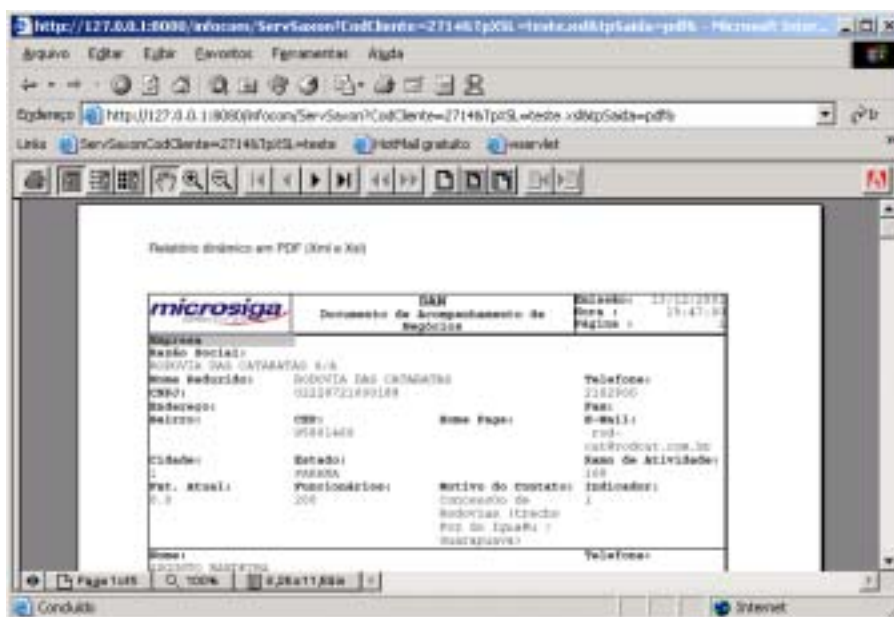


Figura 03 - Relatório em HTML:

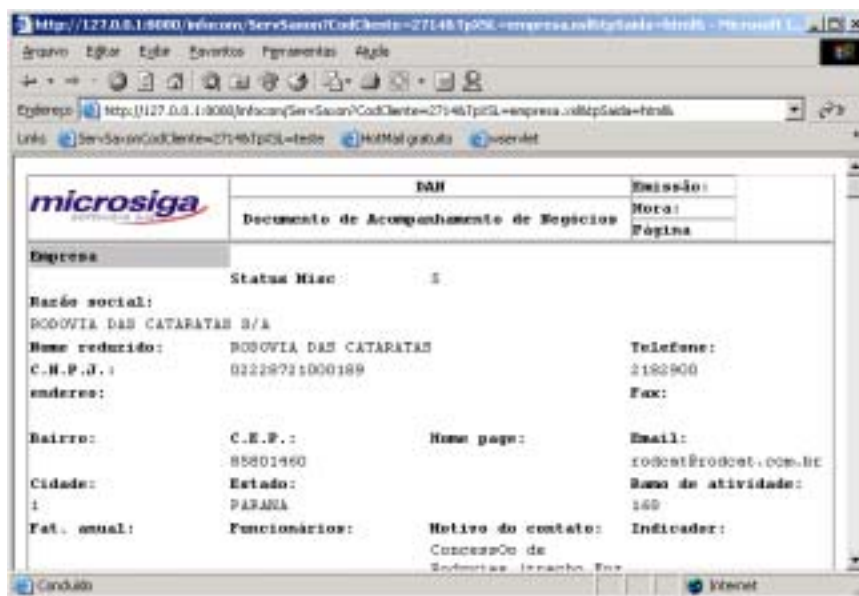


Figura 04 - Relatório em TXT

Para ambos os casos o relatório pode ser imediatamente impresso ou salvo em disco.

```

try {
    //Obtém a conexão com o banco de dados
    Connection conn;
    Context context = new InitialContext();
    javax.sql.DataSource cpds = (javax.sql.DataSource)
context.lookup("serial://jdbc/WebReport");
    conn = cpds.getConnection();
    //Cria a instância do componente XmlReport
    XMLReport xmlReport = new XMLReport (conn);
    //A descrição está na legenda abaixo
    xmlReport.addBand("1", "Select a.cd_empresa, a.cd_cliente, c.nm_uf
from dan011 a, dan003 b, dan002 c "+where a.cd_cliente = "+CodCliente+" and "
+"a.cd_empresa = 1 and a.cd_cidade = b.cd_cidade and b.cd_uf = c.cd_uf Order By
nm_razaosocial");
    xmlReport.addBand("1.1", "select * from dan026 where cd_empresa = ?
and cd_cliente = ?", "[{1}{2}]", "[{1}{2}]");
    //Gera o relatório em XML
    xmlReport.apply(req, res);
    //Gera os arquivos em disco. Esta rotina é chamada para gerar o
arquivo XSL básico a ser formatado no XSLEditor para relatórios em formato PDF, ou a
ser formatado em algum editor HTML para este formato.
    xmlReport.generateXslPdf("c:\\temp\\Pdf_Saida.xml");
    xmlReport.generateXslHtml("c:\\temp\\Html_Saida.xml");
    //Esta rotina é utilizada para gerar o arquivo XML em disco para
propósitos gerais, como testes ou validações.
    xmlReport.SaveXmlToFile("c:\\temp\\Xml_Saida.xml");
    . . .
}

```

O detalhamento do processo de como as consultas estão relacionados, está explicado na legenda abaixo.

```

xslReport.addBand("1", "Select a.cd_empresa, a.cd_cliente, c.na_uf from dan011 a, dan003 b, dan002 c "+
"where a.cd_cliente = "+{cd_cliente}+" and "+
"a.cd_empresa = 1 and a.cd_cidade = b.cd_cidade and b.cd_uf = c.cd_uf Order By na_razaosocial");
xslReport.addBand("1.1", "select * from dan026 where cd_empresa = ? and cd_cliente = ?, " + "{(1)}{(2)}", "{(1)}{(2)}");

```

Estabelece a hierarquia entre as consultas SQL, indicando qual consulta depende de outra dentro das SQLs informadas para o relatório. No exemplo acima, irá ocorrer uma consulta 1.1 para cada ocorrência da consulta 1.

O valor indicado neste parâmetro, relaciona o índice do campo especificado na consulta da hierarquia acima, cujo valor será substituído nos pontos de interrogação da mesma. No exemplo acima, o valor 1 refere-se ao primeiro campo especificado na consulta hierarquicamente acima (cd_empresa), cujo valor será substituído no ponto de interrogação correspondente. Digamos que o valor retornado em cd_empresa fosse 5 na consulta 1, este seria o valor atribuído ao ponto de interrogação correspondente na consulta 1.1.

Este parâmetro é passado diretamente pelo componente que implementa o relatório (Servlet), que será utilizado para fazer a filtragem dos dados retornados na consulta a qual ele está associado. São os valores que podem ser passados como parâmetros pelo usuário quando este for chamar um relatório.

Os valores especificados neste parâmetro, referem-se aos índices dos campos da consulta atual sobre os quais será criada uma totalização. No exemplo acima, os dois primeiros campos retornados no select da consulta 1.1, serão totalizados no arquivo XSL.

Figura 05 – Consultas SQL

3.7 Visão do Processo

O modelo proposto estabelece que o processamento pesado ocorre no lado Server, sendo o Servlet o responsável por gerar os dados necessários, bem como combinar os arquivos Xml e Xsl para gerar um único arquivo que será retornado para o Client através do Browser.

Em termos de processamento, o Client realiza apenas a chamada ao método e fica aguardando o arquivo de retorno. Assim sendo, o modelo exige pouco processamento da máquina do usuário e permite que vários relatórios sejam disparados em seqüência. O tempo de resposta para o usuário dependerá da configuração do servidor, do tamanho da base de dados, dos filtros especificados da complexidade do arquivo XSL usado para formatar os dados, bem como dos processos atualmente em execução no servidor e o tráfego na rede.

3.8 Visão dos Dados

O processo de criação dos relatórios não implica na gravação de dados no sistema, a não ser para um eventual controle de log. No entanto para prover facilidades em termos de testes e definição de layout para os relatórios, ou mesmo para fins de exportação dos dados gerados, foram criados também mecanismos para geração dos arquivos Xml e Xsl (html ou Pdf) em disco. Isto se dá através da

invocação de métodos específicos do componente.

3.9 Instalação

A configuração necessária para realizar a instalação será a mesma exigida pelo Servidor de Aplicações, uma vez que os servlets e os pacotes necessários serão instalados junto ao servidor Web (TomCat), através dos arquivos .war e .jar. Da mesma forma, será necessário fazer a instalação das bibliotecas do componente, bem como dos parser utilizados (Saxon e Fop). Este trabalho pode ser realizado remotamente, a partir de uma rede TCP-IP.

A escalabilidade e portabilidade do componente tornou-se possível através do desenvolvimento em camadas onde cada uma delas possui características próprias e que possibilitam maior facilidade para a evolução do produto, como descrito a seguir.

O Servidor Web no qual é feita a instalação dos componentes, pode eventualmente ser substituído por qualquer outro que tenha suporte a Java Servlets. Este foi o servidor escolhido por se tratar de uma ferramenta gratuita e também por ser amplamente difundido como servidor Web, fazendo parte da distribuição do servidor de aplicações da Borland, o Borland Application Server (BAS), além de estar disponível também para a plataforma Unix.

Os Parsers utilizados FOP e SAXON, também são independentes de plataforma e distribuídos gratuitamente, sendo que poderão ser substituídos por outros ou por novas versões que forem sendo implementadas. A escolha se deu também pelo fato dos mesmos passarem por constantes evoluções, o que possibilita a implementação de novas facilidades nas ferramentas de criação e edição dos arquivos, assim como traz maior segurança, sendo um produto em constante evolução. Os parsers exercem um papel fundamental na definição do componente, pois são capazes de manipular os arquivos Xml/Xsl tirando este papel do browser e tornando mais rápida a sua visualização, além de possibilitar a visualização em qualquer browser, de qualquer plataforma na qual o cliente estiver rodando. Se a visualização for feita no formato PDF, o mesmo precisa possuir um mecanismo de visualização compatível com este formato (como o Acrobat Reader).

3.10 A Ferramenta XSLEditor

3.10.1 Propósito

Facilitar a edição dos arquivos XSL gerados automaticamente e que já possuem as tags especiais para seleção dos campos no arquivo XML, através da manipulação de componentes visuais, tal qual acontece em outras ferramentas de implementação de relatórios específicas (Quick Report, Crystal Reports). A partir de recursos como *Drag-And-Drop*, criação e exclusão de componentes, edição dos atributos visuais dos campos e dos valores associados ao conteúdo do arquivo XML do relatório, o desenvolvedor pode criar o layout do relatório de forma visual, sem a necessidade de intervenção manual no arquivo XSL.

3.10.2 Escopo

A partir da criação básica do layout em formato XSL, que é feita no componente XMLReport, para os relatórios que deverão ser disponibilizados em formato PDF ou TXT, foi identificada a necessidade de criar uma ferramenta para a edição visual do layout destes relatórios, o XSLEditor, uma vez que modificar diretamente o XSL é uma tarefa muito complexa e demorada. O arquivo XSL é editado através dos recursos implementados nesta ferramenta, que permite dispor e formatar os componentes relativos as templates, posicionar, dimensionar, inserir e excluir os labels e campos correspondentes a mesma e salvar o arquivo dentro dos padrões estabelecidos pelo componente XMLReport, assim como pelos parsers utilizados.

3.11 Representação da Arquitetura

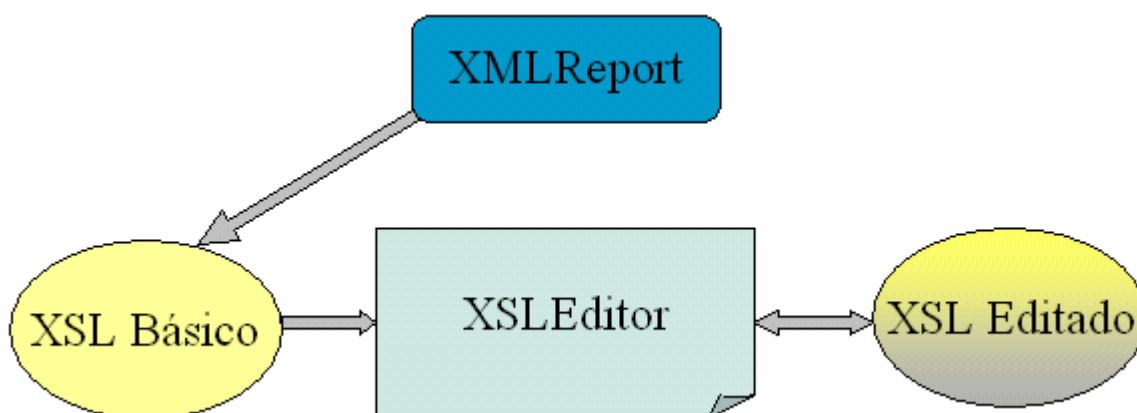


Figura 06 - Representação da Arquitetura – XSL Editor

3.11.1 Objetivos e Restrições da Arquitetura

Com o objetivo inicial de possibilitar a edição visual do relatório e também permitir que o desenvolvedor crie rapidamente o layout necessário para a apresentação do relatório, foi desenvolvida a ferramenta XSLEditor. A ferramenta está limitada à edição apenas dos arquivos gerados inicialmente através do componente XMLReport, permitindo a edição visual das *tags* especiais do padrão XSL FO, que possibilitam apresentar o resultado final em PDF/TXT, sendo que o desenvolvedor não tem a necessidade de conhecer a sintaxe associada a linguagem XSL. O XSLEditor realiza este trabalho, mapeando para a estrutura XSL do arquivo associado, as modificações realizadas visualmente através dos recursos implementados na ferramenta. Para o desenvolvimento da ferramenta, foi utilizado o Delphi, o que restringe o seu uso a plataforma windows, sendo o seu uso restrito a equipe de desenvolvimento, o que não impede que o arquivo resultante seja utilizado em qualquer plataforma.

3.11.2 Visão do Usuário

O desenvolvedor, que é o usuário da ferramenta, deverá localizar e abrir o arquivo XSL padrão, gerado previamente pelo componente XMLReport. Enquanto o arquivo está sendo aberto pelo XSLEditor, o mesmo faz a interpretação das *tags* nele contidas, estabelecendo a disposição dos componentes, criando abas

separadas para cada template, cria uma área abaixo das templates para contemplar os campos somatórios e disponibiliza uma lista com os campos existentes em cada template. O arquivo pode então ser editado e salvo pelo desenvolvedor, estando desta forma pronto para ser utilizado pelo componente XMLReport, para ser mesclado com o XML correspondente, gerando o relatório.

3.11.3 Visão Lógica

O XSLEditor está estruturado em quatro unidades principais, contidas em um único projeto, da seguinte forma:

- 1) **UMain** – Esta é a unidade principal do projeto que está estruturada em *procedures* e *functions*, utilizando-se da estrutura de *chunks*, que dá o tratamento necessário a cada identificador correspondente as tags encontradas no arquivo XSL. Também é responsável pela abertura do arquivo, disposição dos componentes na tela, assim como salvar as modificações feitas pelo desenvolvedor. Compreende a interface principal da ferramenta XSLEditor com o desenvolvedor, e suas funcionalidades estão descritas com base na legenda da figura abaixo:

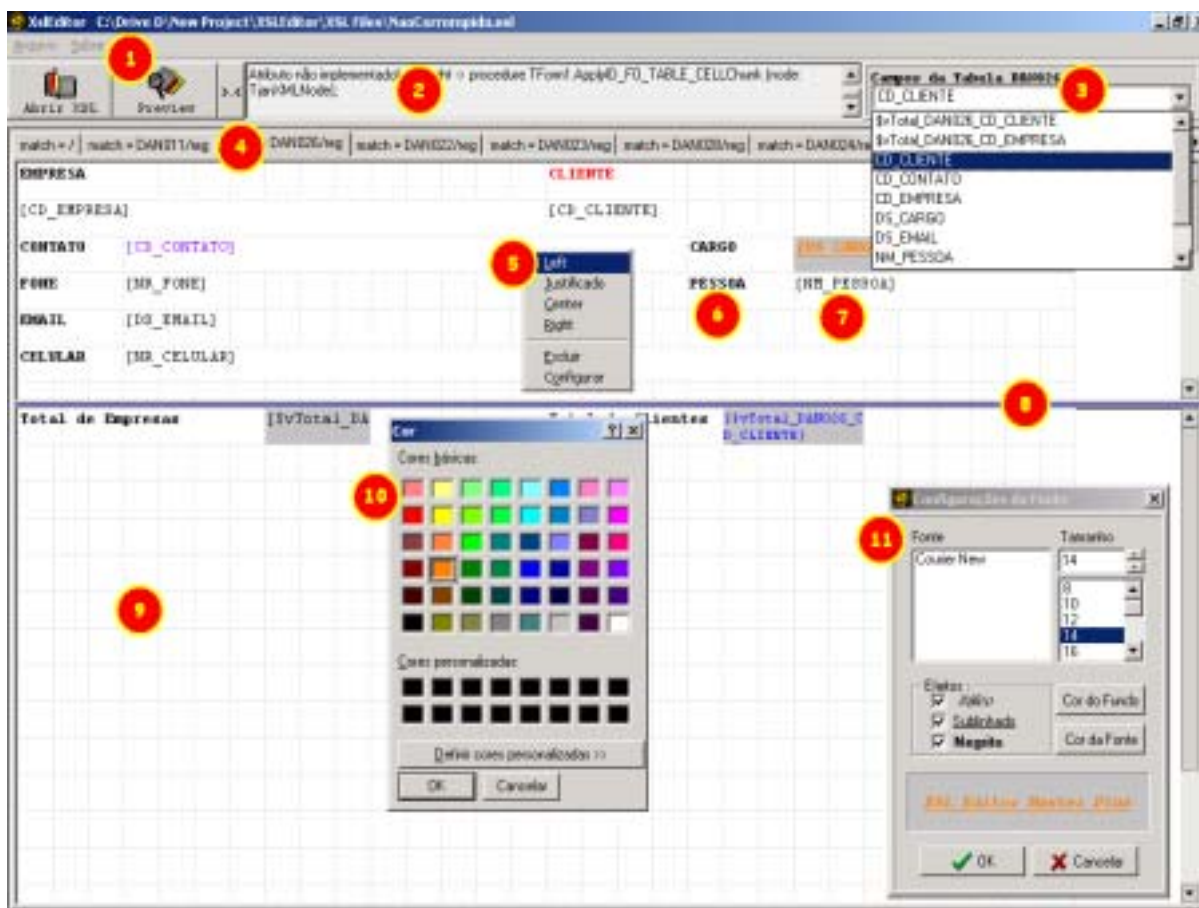


Figura 07 - XSL Editor

Cada uma das partes acima destacadas por números, que são as que compõem as principais funcionalidade da ferramenta XSLEditor são:

1.01 - Menu da Aplicação » Apresenta as opções para Abrir, Salvar, Salvar Como e Sair do programa. Para a função de abertura do arquivo, existe também um atalho representado pelo botão 'Abrir XSL'. Através dele o desenvolvedor irá acessar o arquivo a ser aberto, assim como poderá salvar versões do mesmo. Existe também o botão 'Preview', que dispara o relatório utilizando o arquivo que está sendo editado e o botão '>.<' que serve para tirar o foco do componente que estiver sendo manipulado, para que se possa incluir um novo componente label ou mesmo um campo da tabela.

1.02 - Caixa de Mensagens ou Warnings » Mostra mensagens indicando que existem tags ou atributos no documento que ainda não foram implementados pela ferramenta ou que não foram atendidas pelos procedimentos implementados. Caso a mensagem estiver relacionada a uma tag, será mostrado o nome da mesma: **Tag não implementada!** -> `fo:page-number`; caso estiver relacionada a um

atributo não implementado, irá mostrar o nome do atributo e o nome da procedure onde o mesmo deverá ser implementado: **Atributo não implementado!** -> **height** -> **procedure TForm1.ApplyID_FO_TABLE_CELLChunk (node: TjanXMLNode);**.

1.03 - ComboBox » Lista todos os campos relativos a template que está sendo editada. São listados em ordem alfabética, sendo que aqueles que iniciam com '\$vTotal_', se existentes, se posicionarão abaixo do *splitter* (Item 08). Estes campos são os que aparecem por primeiro na lista e a sua nomenclatura é padronizada. Considerando-se um campo chamado **\$vTotal_DAN026_CD_EMPRESA**, sua composição é constituída por três partes: **\$vTotal_**, indica que o valor deste campo será obtido com base na totalização de outros campos da template, **DAN026_**, representa a tabela a qual o campo pertence e **CD_EMPRESA** que indica qual é o campo que será selecionado. Os campos do tipo somatório para existirem, precisam ser definidos nas consultas SQL que compõem o relatório, no componente XMLReport. Os demais campos representam valores associados diretamente ao campo correspondente e são posicionados acima do *splitter*. (Item 08). Sobre o ComboBox, existe ainda a informação de qual tabela está associado a template em questão, sendo este muito parecido com o nome da própria template (Item 04).

1.04 – Templates » Cada tabela utilizada na composição do relatório está separada em uma template própria, que contém os respectivos campos e campos de somatória. A template é a representação de como os campos daquela tabela estarão dispostos no relatório assim como as propriedades dos mesmos (cor, fonte,...), sendo que o nome da template aparece na aba de seleção. Quando uma template é selecionada, no ComboBox (Item 03) que está acima dela, irão aparecer somente os campos e campos de somatória correspondentes. Cada template pode ou não conter os campos de somatória, sendo que estes sempre aparecerão separados dos demais por um *splitter* (Item 08).

1.05 – Menu Pop-Up » Este tipo de menu é acionado com o botão contrário do mouse e dispõe opções da mesma forma que um menu normal. No componente XSLEditor, existem dois menus deste tipo, conforme descrição abaixo:

- a) Este menu é acionado, quando o desenvolvedor clica com o botão contrário sobre o Grid (item 09), não importando se for acima ou abaixo do

splitter (*item 08*) e apresenta as seguintes opções: Novo Campo – Insere no Grid o campo que estiver selecionado no ComboBox (*Item 03*), que irá aparecer na parte correspondente do Grid, ou seja, se iniciar com '\$vTotal_' (do tipo somatório), irá aparecer na parte superior do Grid e caso contrário irá para a parte inferior; Novo Label – Insere um novo label na parte superior do splitter, logo abaixo do Label (*Item 06*) mais inferior; Novo Label Total – Insere um novo label na parte inferior do Grid, logo abaixo do Label mais inferior.

- b) Este menu é acionado, quando o desenvolvedor clica com o botão contrário sobre um Campo ou Label, independentemente do mesmo estar acima ou abaixo do Grid. Apresenta opções para configurar as propriedades assim como o alinhamento do Label ou Campo em questão, com as seguintes opções Left – Faz o alinhamento a esquerda; Justificado – Faz o alinhamento justificado do texto; Center – Faz o alinhamento no centro; Right – Faz o alinhamento a direita; Excluir – Remove o Campo ou Label selecionado e finalmente Configurar, que abre a caixa de ferramentas Configurações de Fonte (*Item 11*), onde é possível definir a aparência dos mesmos.

1.06 – Label – Os Labels são utilizados para que o desenvolvedor possa inserir textos informativos ao usuário, informando o que os Campos (*Item 07*), colunas, ou áreas do relatório significam. Clicando-se sobre um Label, o cursor estará indicando que o texto do mesmo está pronto para ser editado, sendo que se o espaço para o texto não for suficiente, parte do texto poderá ficar escondida. Se isso ocorrer, é possível redimensionar o tamanho do label, clicando na sua borda quando o cursor do mouse se apresentar em forma de seta, alterando o seu tamanho. Quando o arquivo XSL padrão (gerado pelo XMLReport), for aberto, os Labels aparecerão na primeira coluna de cada template, em negrito, ao lado do respectivo Campo.

1.07 - Campo – Os Campos são utilizados para que o desenvolvedor possa referenciar os campos da tabela, que serão representados pelos valores dos registros existentes no arquivo XML. Os Campos não podem ter o seu texto alterado diretamente pelo desenvolvedor, o que pode ser feito somente com os Labels, sendo possível apenas alterar as suas propriedades. É possível, entretanto, trocar o campo que está representado pelo componente, se ele estiver selecionado, e for

escolhido um novo Campo através do ComboBox (*Item 03*). É possível redimensionar o tamanho do Campo, clicando na sua borda quando o cursor do mouse se apresentar em forma de seta, alterando o seu tamanho. A largura ocupada pelo Campo no Grid (*Item 09*), é o espaço máximo que o valor que vier do arquivo XML poderá ocupar, sendo que se a largura não for suficiente para todo o conteúdo, o Campo irá automaticamente fazer a quebra de linha, sempre respeitando a largura definida. Quando o arquivo XSL padrão (gerado pelo XMLReport), for aberto, os Campos aparecerão na segunda coluna de cada template, marcados através de colchetes '[]', ao lado do respectivo Label.

1.08 – Splitter (Divisor) – O splitter aparece somente nas templates que apresentam campos totalizadores, dividindo a área relativa aos campos da própria template (parte superior) e a área relativa aos campos somatórios (parte inferior). Quando o arquivo é aberto o splitter ocupa a posição imediatamente abaixo do último componente Label ou Campo da template. O splitter pode ser usado para redimensionar a área da template, simplesmente clicando e arrastando-o no sentido vertical.

1.09 – Grid - O grid serve para estabelecer a posição dos componentes (Labels e Campos) em colunas e linhas em relação a um Snap definido, e que coincide como o número de colunas da tabela no arquivo XSL gerado. O Grid facilita o alinhamento dos componentes em colunas e linhas, podendo ser movido verticalmente através de uma barra de rolagem no canto direito da tela. A rolagem também obedece ao Snap definido, de modo que a posição relativa do componente não seja alterada incorretamente no Grid. As interseções das linhas tracejadas verticais e horizontais desenhadas no Grid, definem o “ponto de atração gravitacional”, ou seja, o ponto para onde o componente será atraído, quando arrastado no Grid. Isso significa que todos os componentes contidos no Grid, estarão sempre alinhados exatamente com as linhas e colunas desenhadas.

1.10 – Cor – Esta pequena tela permite a seleção de cor de fonte ou de fundo a ser aplicada em determinado componente, sendo que para isso o desenvolvedor deverá chamar primeiramente a janela de configuração da fonte (*Item 11*) e clicar em “Cor da Fonte” ou “Cor do Fundo”.

1.11 – Configurador de Fontes - Configura as fontes dos Campos e Labels e seus atributos, como negrito, itálico, sublinhado e cores. Quando é chamado, apresenta os atributos do componente selecionado, podendo ser acionado pelo meu

Pop-Up ou com um duplo clique sobre o componente que se queira configurar.

2) **Uconst** – Esta unidade possui a definição de todas as constantes utilizadas pelo processo de leitura do arquivo no qual estão definidas as tags principais e relevantes a serem interpretadas. Para gerar os valores para as respectivas tags, foi desenvolvido também um processo de geração de código automático (Code Generator), que através de uma função Hash, estabelece valores automaticamente para cada constante. Essa necessidade se deu pelo fato de que a estrutura de controle CASE da linguagem Pascal não permite instruções comparativas com Strings, o que prejudicaria a performance de construção da estrutura interna.

Existem dois tipos de constantes definidas: tags e atributos;

- Constantes tipo tag: apresentam o prefixo ID_ seguido do nome da tag no padrão XSL.

Exemplo:

```

case GetHashCode(node.Name) of
    ID_ROOT : begin
        ApplyID_ROOTChunk (node);
    end;
end;

```

- Constantes tipo atributo: apresentam o prefixo AT_ seguido do nome do atributo associado a determinada tag.

Exemplo:

```

case GetHashCode(nodeAttrib.Name) of
    AT_XMLNS_XSL : begin
        ...
    end;
end;

```

3) **UFontConfig** – Esta unidade, tem a função de possibilitar a edição dos atributos dos componentes inseridos no layout do relatório. O formulário de configurações permite modificar a fonte, cor, cor do fundo, tamanho, negrito, sublinhado, itálico além de mostrar automaticamente um preview do texto. Pode ser chamado de dentro do XSLEditor a partir do menu pop-up ou com um clique duplo sobre o componente que se deseja configurar.

4) **JanXMLTree** – Este é um pacote freeware, com diversos componentes

visuais, que possibilita ler a estrutura do arquivo XSL para a memória, na forma de objetos hierarquicamente dispostos, mapeando cada tag e respectivos atributos em uma árvore, dispondo ainda de mecanismos para salvar esta estrutura. Este pacote foi utilizado para facilitar a implementação da leitura e salvamento dos arquivos XSL utilizados e por implementar rotinas de baixo nível para realizar esta tarefa, inclusive com códigos em assembly. Tem como função principal interpretar e identificar as tags específicas do arquivo para que estes componentes possam ser mapeados para a representação do layout do relatório, na forma de componentes da VCL do Delphi. Desta forma, a complexidade da ferramenta é feita de forma transparente para o desenvolvedor, que precisa apenas preocupar-se com a diagramação do relatório.

3.12 Visão do Processo

O processamento do arquivo XSL ocorre somente em dois momentos: quando este é aberto ou salvo através da ferramenta XSLEditor. Ele é editado através dos mecanismos mencionados no item 5, sendo que todo o processamento é executado localmente na máquina do desenvolvedor,

Quando o arquivo XSL é aberto, algumas de suas principais tags são mapeadas para a representação do layout do relatório, na forma de componentes da VCL do Delphi. Enquanto os componentes estão sendo editados, o arquivo não sofre qualquer tipo de modificação, o que será feito somente quando o arquivo for salvo através das opções do menu **salvar** ou **salvar como**.

4 CONCLUSÕES E RECOMENDAÇÕES

Através do trabalho de revisão da literatura sobre os diversos assuntos relacionados ao projeto, com o conhecimento prático em termos do processo de projeto e desenvolvimento de software, e com o desenvolvimento do protótipo para geração de relatórios dinâmicos, foi possível entender a maioria dos fatores relacionados ao assunto, as suas dificuldades, os obstáculos e as perspectivas, sendo estes experimentos realizados de forma prática, através do desenvolvimento do XMLReport e do XSL Editor.

Devido a diversidade de ambientes computacionais e o rápido avanço das tecnologias, o XML sofrerá um evolução natural e poderá trazer novos conceitos que facilitem ainda mais a sua utilização para padronização, extração e distribuição de documentos eletrônicos, assim como servir de base para a troca mais padronizada de dados. Sendo isto feito de forma independente de plataforma computacional, o XML proporciona ainda a portabilidade necessária para este tipo de arquivo, e que o mesmo tenha sido escolhido para ser utilizado para a base de pesquisa, projeto e implementação das ferramentas que tem como atributo o armazenamento das informação de forma estruturada.

A arquitetura baseada na Web é, sem dúvida, a grande tendência mundial para os sistemas de informação, independentemente da área na qual este estiver implantado. Na geração de informações, a partir de dados armazenados em B.Ds, que via consulta SQL cria simultaneamente a forma de visualizar e formatar estas informações, o protótipo apresentado apresenta vários benefícios. Outra tendência observada é a crescente utilização da Web como interface para sistemas, e maior ainda sobre sistemas que tem alguma forma de conexão através dela. Dessa forma, considerando tais tendências e apostando que é a melhor solução, a construção do protótipo e projeto baseados na Web e nos formatos XML e XSL de arquivos, deve ser considerada como uma forte opção em se tratando deste tipo de aplicação.

O modelo proposto tem como pontos a serem destacados, a portabilidade oferecida em termos de Banco de Dados e plataforma para apresentação do resultado final, além de disponibilizar os arquivos que contêm as informações em diferentes formatos, de acordo com a necessidade de cada usuário. Por tratar-se de uma proposta, o modelo não oferece ao desenvolvedor a possibilidade de recursos

avançados, como gráficos ou customização via programação, para geração de relatórios que não contemplam os componentes implementados, sendo para isso necessário uma evolução do próprio componente.

A abordagem utilizada para solução do problema proposto, preocupa-se em fazer uso das tecnologias mais atuais no que diz respeito a apresentação de documentos, buscando sempre separar o conteúdo da forma na qual os mesmos serão apresentados, fazendo assim com que a proposta apresente-se de forma componentizada, e assim, mais maleável para futuras modificações ou evoluções.

As técnicas de Engenharia de Software foram abordadas para que exista uma base sólida onde a implementação pudesse ser baseada, de acordo com as melhores práticas para a implementação dos componentes.

Esta pesquisa se constitui numa contribuição para busca de uma forma padrão de apresentação de documentos portáteis, abrindo caminho para uma implementação que contemple os diferentes ambientes computacionais onde as informações são, e cada vez mais terão que ser disponibilizadas de forma padrão, sem que para isso seja necessário uma reimplementação para cada ambiente distinto onde a mesma tenha que ser apresentada.

Apesar de toda a complexidade existente em termos tecnológicos e todas as sub-camadas envolvidas, o desenvolvimento de relatórios através dos componentes propostos e implementados no presente trabalho de pesquisa, é feito de maneira muito simples, sendo que o desenvolvedor necessita basicamente de conhecimento em Linguagem de Consulta SQL padrão, além das ferramentas de edição de layout para os arquivos Xsl, especialmente para este propósito.

Desta forma, o presente trabalho de pesquisa realizou um estudo, propos uma modelo e realizou sua implementação como forma de criar um modelo que possa ser efetivamente utilizado, levando-se em consideração as restrições descritas, mostrando a viabilidade e utilidade dos componentes propostos.

4.1 Sugestões para Trabalhos Futuros

Com a crescente demanda por relatórios que possam estar disponíveis em ambientes cada vez mais heterogêneos, é imprescindível que exista uma forma padrão para prover tal recurso e que satisfaça o requisito de portabilidade.

Tendo isto em vista, o componente para geração de relatórios proposto e

implementado no presente trabalho, a partir de suas restrições, poderiam sofrer modificações a fim de contemplar mais alguns recursos interessantes no que diz respeito a relatórios, principalmente os gerenciais, como geração de gráficos.

Outra funcionalidade que poderia ser estendida, é a de fazer a integração dos componentes com um ambiente de desenvolvimento, facilitando ainda mais a compreensão e utilização dos mesmos pelos desenvolvedores. Desta forma, com um ambiente integrando o desenvolvimento do aplicativo de acesso a dados em si, e a geração de seus relatórios, facilita e reúne a solução de implementação de um sistema de Banco de Dados e respectivos relatório de forma conjunta.

REFERÊNCIAS BIBLIOGRÁFICAS

"Application integration with XML" <http://www.oasis-open.org/cover/xmlIntro.html> 05/05/2002

Banerjee, S. V. K., Krishnaprasad, M. , Murthy, R., "Oracle8i – The XML Enabled Data Management System", Proceedings of the International Conference on Data Engineering (ICDE), California, March 2000.

Berg, C., Advanced Java 2 Development for Enterprise Applications, Prentice Hall, 1999.

Birdeck, M., Kay, M., Anderson., R., Professional XML. Editora: Ciência Moderna - ISBN: 8573931167 - Ano: 2001

Bourett, R., XML and Databases. June 2001. <http://www.rpbouret.com/xml/XMLAndDatabases.htm>. 10/01/2003

Bourett, R., XML Database Products.. <http://www.rpbouret.com/xml/XMLAndDatabases.htm>. 22/10/2002

CDROM The XML StarterKit – 2nd Edition – Software AG The XML Company

Chamberlin, D., et al. XQuery 1.0:An XML Query Language (Working Draft).7 June 2001. <http://www.w3.org/TR/2001/WD-xquery-20010607>. 15/12/2003

Charset Standards <http://czyborra.com/charsets/iso8859.html#ISO-8859-1>. 02/09/2002

Commerce XML, <http://www.cxml.org>. 24/09/2002

Cover, R., The XML Cover Pages, <http://www.oasis-open.org/cover/sgml-xml.html>. 05/08/2002

Cover, R., The XML Cover Pages: Extensible Markup Language (XML). 22 March 2001. <http://oasis-open.org/cover/xml.html>. 08/08/2001

Desing and Authoring Tool <http://www.xslfast.com/products/features.htm> 06/06/2002

Deutsch, A., Fernandez, F. D. M., Levy, D. A., "XML-QL: A Query Language for XML", Proceedings of the International World Wide Web (WWW) Conference, Toronto, May 1999.

Dumbill, E., XML protocol technology reference. 1 November 2000. <http://www.xml.com/pub/a/2000/11/01/protocols/quickref.html>. 05/07/2002

Espósito, D., XML Database Applications http://www.xml.com.br/msdn/pdf/XML_DB_Apps.pdf. 08/08/2002

Fernandez, M., Morishima, A., Suciú, D., "Efficient Evaluation of XML Middle-ware Queries", Proceedings of the ACM SIGMOD Conference on the Management of

Data, Santa Barbara, California, May 2001.

Fernandez, M., Tan, W., Suciu, D., "SilkRoute: Trading Between Relations and XML", Proceedings of the International World Wide Web (WWW) Conference, May 2000.

Hamilton, D. O., Faculdade de Tecnologia de Americana. Introdução a XML e suas Aplicações <http://www.fatec.br/americana/trabalhos/docs/douglas01.pdf>. 10/01/2002

http://gopher.lib.virginia.edu/speccol/scdc/articles/alcts_brief.html. 09/09/2002

<http://tigger.uic.edu/~cmsmcq/talks/teidlf1.html>. 09/09/2002

Hunter, D., Beginning XML. Birmingham: Wrox Press, 2000. ISBN 1-861003-4-12

IBM <http://www-106.ibm.com/developerworks/java/library/j-jdom/>. 03/03/2002

Jakarta-TomCat WebServer <http://jakarta.apache.org/tomcat/index.html>. 06/08/2002

Laddad., R., Sztokbant. C., XML and how it will change the Web <http://www-106.ibm.com/developerworks/library/jw-xmlapis/index.html?dwzone=xml>. 06/09/2002

Maloney, M., Rubinski, Y., SGML on the Web: Small Steps Beyond XML, Prentice Hall, 1999.

Maruyama, H., Uramoto, N., Tamura, K., The Power of XML:XML's Purpose and Use in Web Applications <http://www6.ibm.com/developerworks/xml/library/xmlpower/xmlpower.html?dwzone=xml>. 14/03/2001

Microsoft Corporation, <http://www.microsoft.com/xml>. 09/09/2002.

Monberg. J., Wynholds. M., Introduction to XSLT <http://builder.cnet.com/webbuilding/0-3882-8-7033236-1.html?tag=txt>. 09/10/2001

Parser FOP <http://xml.apache.org/fop/index.html>. 09/08/2002

Parser Saxon <http://users.iclway.co.uk/mhkay/saxon/saxon5-5-1/>. 09/08/2002

Programacion XML/XSL http://html.programacion.net/taller/tw_xml_y_xslt.php. 04/07/2002

Real Estate Transaction Standard, <http://www.rets-wg.org>. 05/07/2002

Shanmugasundaram, J., Tufte, K., He, G., Zhang, C., DeWitt, D., Naughton, J., "Relational Databases for Querying XML Documents: Limitations and Opportunities", Proceedings of the Very Large Data Bases (VLDB) Conference, Scotland, England, September 1999.

Westphal, R., Building an XML-based metasearch engine on the server. O'Reilly & Associates, Inc., 2000.

World Wide Web Consortium <http://www.w3.org/>. 03/07/2002

World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Second Edition) ", W3C Recommendation, October 2000. <http://www.w3c.org/TR/REC-xml> 03/01/2002.

World Wide Web Consortium, "XML Schema Parts 0, 1, 2", W3C Candidate Recommendation, October 2000. <http://www.w3c.org/TR/xmlschema-0>, 1, 2. 02/10/2000

World Wide Web Consortium, "XQuery: A Query Language for XML", W3C Working Draft, February 2001. <http://www.w3c.org/TR/xquery>. 06/02/2001

XML Journal - <http://www.sys-con.com/xml>. 06/02/2003

xml.com <http://www.xml.com>. 30/12/2002

XSL Standards http://www.cuesoft.com/docs/cuexsl_activex/xsl_stylesheet_commands.htm#xsl:value-of. 06/06/2002

XSLT DocBook <http://www.dpawson.co.uk/xsl/sect2/generic.html>. 06/06/2002

YENGAR, S & BRODSKY, S., XML Metadata Interchange (XML), 1998, disponível na INTERNET via <ftp://ftp.omg.org/pub/docs/ad/98-10-17.pdf>. 02/02/2001

Zoltán, E. Server -side scripting languages - PHP, Perl, Java servlets -- Which one's right for you ?. <http://www.ibm.com/developerworks/Web/library/wa-sssl.html?dwzone=Web>. 06/02/2002