Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

Sara Raquel Oliveira da Silva

Sistema para gestão de repositórios XML para bibliotecas digitais

Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

Sara Raquel Oliveira da Silva

Sistema para gestão de repositórios XML para bibliotecas digitais

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Ciências da Engenharia de Computadores e Telemática, realizada sob a orientação científica do Dr. Joaquim Arnaldo Martins, Professor catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

presidente

Doutora Maria Beatriz Alves de Sousa Santos
Professora Associada com Agregação da Universidade de Aveiro

Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático da Universidade de Aveiro

Doutor Joaquim Manuel Henriques de Sousa Pinto
Professor Auxiliar da Universidade de Aveiro

Doutor Fernando Joaquim Lopes Moreira
Professor Auxiliar do Departamento de Inovação, Ciência e Tecnologia da Universidade Portucalense

agradecimentos

Gostaria de agradecer ao orientador Prof. Dr. Joaquim Arnaldo Martins e ao Prof. Dr. Joaquim Sousa Pinto por me terem proporcionado esta experiência, e aos colaboradores Marco Fernandes e Pedro Almeida, pela sua disponibilidades e paciência em me ajudar.
Um obrigado muito grande à minha mãe, Fátima, pelo apoio que me deu, pela compreensão por ter estado mais ausente e por nunca me deixar desistir.
Ao meu padrinho, José Luís, quero agradecer pelos conselhos e indicações que me deu.
Aos meus colegas e amigos, que directamente ou indirectamente contribuiram para o meu trabalho, estou muito agradecida.

palavras-chave

XML, optimização de bases de dados, indexação, performance, bases de dados XML nativas.

resumo

Tradicionalmente, os sistemas de informação usam bases de dados relacionais para armazenar a descrição dos registos. Nos últimos anos, a XML tornou-se no standard de facto para descrição documental, transporte de informação e interoperabilidade entre sistemas heterogéneos. Recentemente surgiram alguns casos de bibliotecas digitais que usam a XML para descrever os documentos armazenados. Neste trabalho, é descrita uma solução de gestão de base de dados que integra uma base de dados XML nativa e um motor de indexação para melhorar a performance da pesquisa no repositório dos ficheiros XML.Também é apresentada uma aplicação de gestão *web-based* que permite a administração, local ou remota, da base de dados.

keywords XML, database optimization, indexing, performance, XML native database.

abstract Traditionally, most of the information systems use relational databases to store the information of their records. In the last few years, XML has become the standard to documental description, information transportation and interoperability between heterogeneous systems. Recently a considerable number of digital libraries were developed using XML documents to describe their resources. In this project, we describe a database management solution which integrates an XML native database and a search engine to improve query performance in XML file repository. A web based management application is also presented which allows local and remote administration of the database.

ÍNDICE

Lista de Figuras

Lista de Tabelas

1. INTRODUÇÃO

1.1. Enquadramento e Descrição do Problema

Nos últimos anos, a XML tornou-se no standard de facto para descrição documental, transporte de informação e interoperabilidade entre sistemas heterogéneos.

Recentemente, surgiram alguns casos de bibliotecas digitais que usam a XML para descrever os documentos armazenados, como é o caso do SInBAD [24], Sistema Integrado de Biblioteca e Arquivo Digital, desenvolvido pela Universidade de Aveiro, que permite o acesso, pesquisa e manipulação de documentos de natureza diversa, que vão desde as tradicionais referências bibliográficas, teses de mestrado e doutoramento, até documentação de natureza multimédia, incluindo áudio e vídeo, estendido quando possível à comunidade exterior, de modo a ter uma verdadeira biblioteca digital.

Ao contrário do que acontece nas bibliotecas digitais que usam bases de dados relacionais (SQL Server, Oracle, etc.) para armazenar a descrição dos registos, o modelo de armazenamento de dados do SInBAD utiliza documentos XML guardados no sistema de ficheiros, que por sua vez são organizados em diferentes colecções (livros, fotografias, etc.).

Numa primeira abordagem foi adoptado um sistema simples de ficheiros, em que cada descrição XML correspondia a um documento. No entanto, com o aumento de requisitos, tornou-se necessário incorporar igualmente as vantagens de uma base de dados tradicional, tais como a robustez, segurança e estabilidade. Isto levou a que se adoptasse uma base de dados XML nativa, Sedna (*open-source*) [1][12], para armazenar os documentos.

1.1. Objectivos

O objectivo principal deste trabalho foi construir uma aplicação/sistema para efectuar, de forma simples, a gestão deste repositório XML.

A primeira fase consistiu no desenvolvimento de uma API flexível que permitisse a interoperabilidade com outros componentes. Tornou-se também necessária uma aplicação de gestão *web-based* de forma a facilitar a sua administração remota, já que o repositório do SInBAD armazena documentos de diferentes departamentos da Universidade. Esta aplicação representa o equivalente a uma aplicação de administração de uma base de dados relacional mas para ficheiros XML. O sistema permite efectuar a gestão de colecções e documentos, pesquisar todo o repositório ou apenas uma colecção, e fazer a edição dos documentos XML.

Observou-se ainda que o desempenho da pesquisa que a plataforma oferecia não era suficiente. De forma a melhorar a rapidez do sistema, tornou-se necessário incorporar métodos de indexação para que a solução final fosse robusta e de resposta rápida. A segunda fase deste trabalho foi integrar indexação eficiente e que essa integração oferecesse aos utilizadores a possibilidade de poderem configurar como e quais os elementos e/ou atributos XML deveriam ser indexados. Para alcançar este objectivo, foi utilizado o Lucene (*open-source*) [23][34] que é bastante flexível e que oferece uma integração bastante simples com outros componentes e de fácil manipulação.

1.2. Estrutura do Trabalho

Este relatório encontra-se dividido por capítulos com objectivos bem distribuídos.

O capítulo 2, ENQUADRAMENTO E ESTADO DA ARTE, engloba uma breve descrição do que é a XML, as suas vantagens, de como pode servir de armazenamento de informação e que soluções existem hoje em dia para utilizar a XML como repositório de dados de forma eficiente. Também neste capítulo, faz-se o estudo das bases de dados XML nativas, as suas vantagens, desvantagens e arquitectura.

O capítulo 3, ESTUDO DE CASO, aborda a preparação efectuada para o desenvolvimento deste projecto bem como as tecnologias utilizadas, limitações destas tecnologias e o que foi necessário contornar.

A explicação da implementação da aplicação e a sua utilização encontra-se no capítulo 4, APLICAÇÃO DESENVOLVIDA, onde estará a descrição mais ou menos detalhada como utilizar a aplicação desenvolvida, tanto na visão de um utilizador final bem como numa perspectiva menos abstracta para programadores.

No capítulo 5, AVALIAÇÃO E RESULTADOS, será realizada uma avaliação final de como decorreu o trabalho, as limitações encontradas e como foram resolvidas bem como as que ficaram por resolver ou que podem ser melhoradas.

Por último no capítulo 6, CONCLUSÕES E TRABALHO FUTURO, serão apresentadas algumas sugestões para desenvolvimento futuro, tais como melhorias que se pensam ser interessantes e algumas indicações do que ficou pendente.

2. ENQUADRAMENTO E ESTADO DA ARTE

No âmbito deste trabalho tornou-se essencial adquirir um vasto conhecimento sobre bases de dados (BD) e como foram evoluindo ao longo do tempo para que se pudesse observar as diferentes características e benefícios de cada uma. Só desta forma se conseguiu verificar em que casos é que cada tipo de base de dados deverá ser utilizada para obter uma melhor performance de um sistema.

De seguida, serão apresentados alguns conceitos fundamentais para que o conhecimento das base de dados XML nativas seja introduzido progressivamente e, de forma a clarificar a sua utilização, tentou-se sempre comparar algumas noções com termos de bases de dados relacionais que são mais conhecidas.

2.1. Conceitos sobre Base de Dados

Podemos dizer que uma base de dados é um conjunto de dados que normalmente descreve as actividades de uma ou mais organizações relacionadas. O objectivo de criarmos e mantermos uma BD é a de poder obter e utilizar os dados lá armazenados [4]. Embora sendo possível usar esta definição genérica, o termo base de dados é aplicado hoje em dia principalmente para fazer referência a bases de dados informáticas, isto é, conjuntos de dados estruturados, manipulados usando um Sistema de Gestão de Bases de Dados (SGBD) ou *Database Management System* (DBMS).

Para permitir ao utilizador atingir os objectivos referidos acima, um SGBD disponibiliza linguagens [25] de:

- **definição de dados:** para criação e alteração da estrutura da BD (*DDL - Data Definition Language*);
- **consulta de dados:** obter e processar os dados armazenados (*DQL - Data Query Language*);

- **manipulação de dados:** para acrescentar dados novos e modificar dados existentes (*DML - Data Manipulation Language*).

Alguns exemplos de SGBD de grande porte são ORACLE, Informix, Adabas, SQL Server e DB2. Para PCs temos o MySQL, Dbase, FoxPro e Access. Os primeiros têm mais capacidade e são mais fiáveis do que os últimos. Estes são adequados para uso doméstico, em pequenas empresas ou como forma de aceder a partir de PCs a BDs instaladas em sistemas de grande porte, através de uma aplicação acessível ao utilizador não especialista em informática.

A evolução dos sistemas de gestão das base de dados foi sempre impulsionada pela procura de novas formas de modelar dados cada vez mais complexos [26].

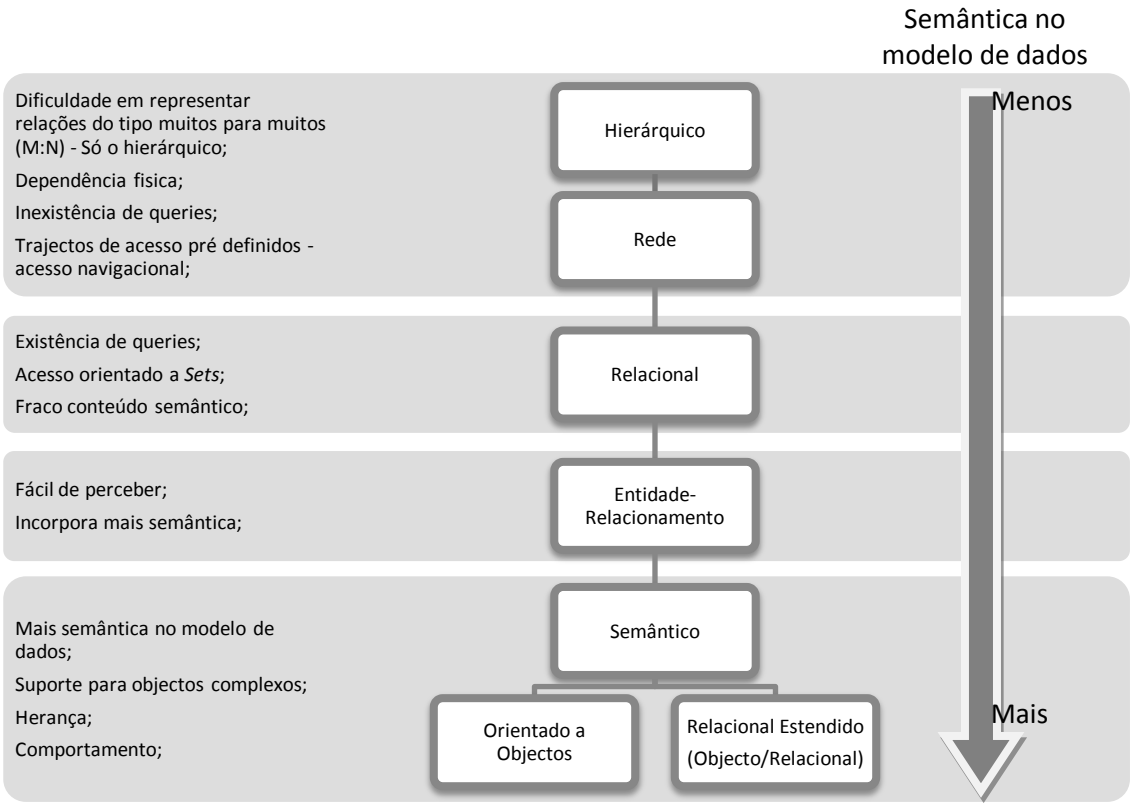

Figura 1 - Desenvolvimento de modelos de dados[26]

Cada novo modelo de dados tornou-se importante devido ao desenvolvimento de modelos anteriores. O modelo de rede substituiu o modelo hierárquico porque o anterior tornou a representação de relações complexas mais simples. Por sua vez, o modelo relacional ofereceu inúmeras vantagens, em relação aos modelos hierárquico e de rede, devido à sua simplicidade na representação de dados, à sua independência de dados superiores e linguagem de consulta

relativamente fácil de utilizar. Apesar de o debate que decorreu durante alguns anos sobre os méritos dos modelos hierárquico, de rede e relacional, este último emergiu como o modelo de dados dominante para as aplicações de negócios. O modelo OO e ERDM têm um nível elevado de conteúdo semântico, fornecem suporte para objectos complexos e de grande dimensão, e inclui herança e comportamento nos seus objectos. Apesar do modelo OO e do ERDM ganharem uma base substancial no domínio de mercado, as suas tentativas para desalojar o modelo relacional do domínio do mercado não foram bem sucedidas [2][26].

Na evolução dos modelos de dados, pode-se determinar algumas características comuns que os modelos de dados têm que ter para serem aceites:

- Um modelo de dados deve mostrar um certo grau de simplicidade conceptual sem comprometer a integridade semântica da base de dados. Não faz sentido ter um modelo de dados que seja mais difícil de perceber do que o mundo real.
- Um modelo de dados deve representar o mais possível o mundo real. Este objectivo é facilmente realizável ao adicionar mais semântica à representação do modelo.
- A representação das transformações do mundo real tem que se encontrar em conformidade com a consistência e integridade das características do modelo de dados.

Com o aumento da utilização da Internet como meio de negócio e comunicação entre organizações, o aparecimento da XML forneceu novas formas de resolver novos desafios de armazenamento em base de dados [8].

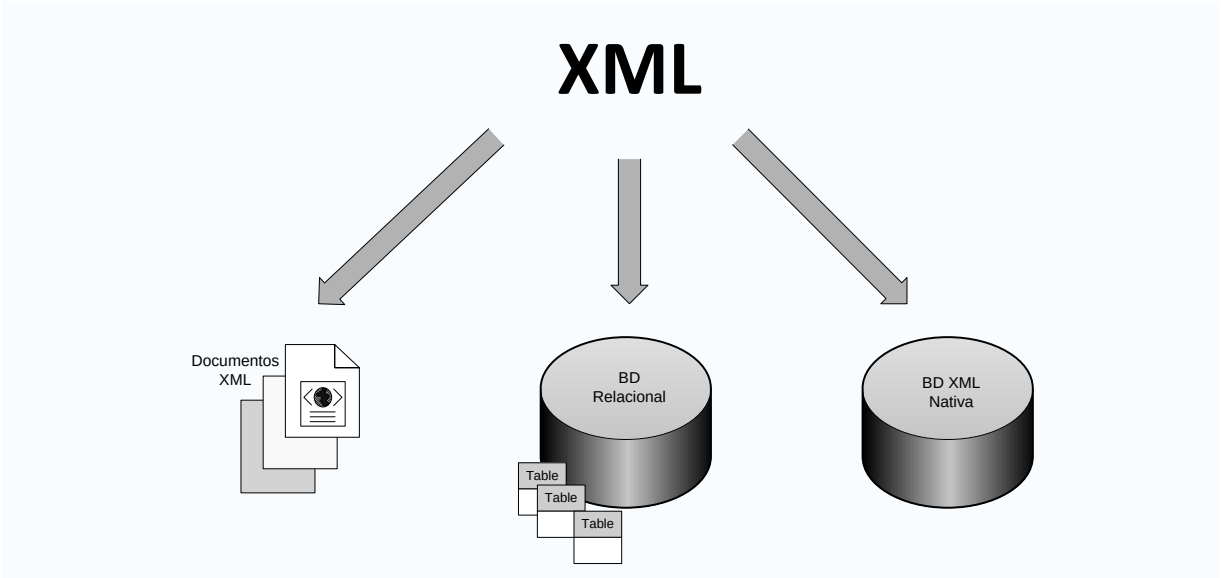

Figura 2 - Armazenamento de XML[8]

Com o objectivo de armazenar dados XML sem perda de informação, da estrutura do documento, nem de desempenho no armazenamento e recuperação de informações, foram desenvolvidas novas soluções de bases de dados que seriam baseadas em ficheiros XML.

De seguida será apresentada uma breve descrição do que é a XML, que vantagens nos traz como solução de armazenamento de dados, bem como os novos tipos de base de dados que apareceram associados a este formato.

2.2. Motivação para linguagens de marcação

Com o aumento dos sistemas de informação nas academias, nas empresas e nos ambientes domésticos, começou a ser necessário um suporte neutro para toda a informação existente numa empresa que fosse independente tanto do software como do hardware utilizado. A maioria da informação encontrava-se em suporte digital, *MsWord*, *Access*, *Excel*, *Acrobat*, etc. A transferência e manipulação dos documentos digitais, entre plataformas, tornava-se dispendiosa e a sua manutenção difícil o que implicava uma baixa longevidade.

Em 1986, é lançado a SGML como norma ISO, ideal para armazenamento e intercâmbio de dados. A *Standard Generalized Markup Language* (*SGML*) [13] é uma metalinguagem através da qual se podem definir linguagens de marcação para documentos. É descendente da *Generalized Markup Language* (*GML*) da IBM, desenvolvida na década de 60 por *Charles Goldfarg*, *Edward Mosher* e *Raymond Lorie*. A SGML foi inicialmente concebida para permitir a partilha de documentos que permitissem a leitura por máquina em projectos de grande dimensão governamentais e na indústria aeroespacial, que necessitaram de permanecer legíveis por várias décadas.

HTML e XML são ambas derivadas do SGML. Enquanto a HTML é uma aplicação da SGML, a XML é um perfil (um subconjunto específico da SGML), projectada para ser mais simples de analisar gramaticalmente e de processar do que SGML [13][37].

2.3. XML

Como foi mencionado, a XML (*eXtensible Markup Language*) [38] é uma tentativa de simplificar SGML para aplicações de uso geral, tais como a *Web* Semântica, que como subtipo de SGML é capaz

de descrever diversos tipos de dados e tem como objectivo principal a facilidade de partilha de informações através da Internet.

2.3.1. Objectivos da XML

Estimulado pela insatisfação com os formatos existentes, um grupo de empresas e organizações que se auto-denominou *World Wide Web Consortium* (W3C) começou a trabalhar em meados da década de 1990 numa linguagem de marcação que combinasse a flexibilidade da SGML com a simplicidade da HTML [8].

O princípio do projecto era criar uma linguagem que pudesse ser lida por software, e que se integrasse com as demais linguagens. A sua filosofia seria incorporada por vários princípios importantes:

- Separação do conteúdo da formatação;
- Simplicidade e Legibilidade, tanto para humanos como para computadores;
- Possibilidade de criação de *tags* sem limitação;
- Criação de arquivos para validação de estrutura (chamados *DTDs*);
- Interligação de bases de dados distintas;
- Concentração na estrutura da informação, e não na sua aparência;

2.3.2. Vantagens da XML

A XML é considerada um bom formato para a criação de documentos com dados organizados de forma hierárquica, como se vê frequentemente em documentos de texto formatados, imagens vectoriais ou bases de dados. Pela sua portabilidade, uma base de dados pode através de uma aplicação escrever num arquivo XML, e outra BD diferente pode então ler esses mesmos dados. Como se trata de um formato texto “puro”, XML pode ser criado e editado em qualquer editor de texto moderno, suportando ainda a maioria das codificações de caracteres.

2.3.3. XML como formato de base de dados

Um ficheiro XML trata-se de uma colecção de dados. Isto não o torna diferente de qualquer outro ficheiro, afinal, todos os ficheiros contêm dados seja de que tipo for.

Como formato de base de dados, a XML tem algumas vantagens, pois é auto descritivo, portável (multiplataforma) e descreve os dados em estrutura em árvore ou grafos, mas também tem algumas desvantagens tal como exigir muito texto e o acesso aos dados ser lento devido ao *parsing* e conversão do texto.

Pelo lado positivo, a XML fornece muitas características que podemos encontrar em base de dados [22]:

- Armazenamento – documentos XML;
- Schemas – DTDs, XML Schemas, RELAX NG, etc;
- Linguagens de Consulta – XQuery, XPath, XQL, XML-QL, QUILT, etc;
- Interface de Programação – SAX, DOM, JDOM;

Pelo lado negativo, falta-lhe muitas das coisas que se podem encontrar numa base de dados relacional:

- Armazenamento eficaz;
- Índices;
- Segurança;
- Transacções;
- Integridade de dados;
- Acesso multi-utilizador;
- Triggers;

Assim, pode ser possível usar um ou vários documentos XML como base de dados em ambientes com quantidades pequenas de dados, poucos utilizadores e requisitos de performance não muito exigentes.

2.3.3.1. Dados vs. Documentos

Para a escolha de uma base de dados XML é necessário que se tenham bem claros os objectivos a serem atingidos, e para isso classificar os documentos XML [29] da seguinte forma:

- **Documentos orientados a dados (*Data-Centric Document*):**
  São documentos que utilizam XML como transporte dos dados. Não é importante para a aplicação, ou para a base de dados, que os dados sejam armazenados num documento XML por algum tempo. Os documentos *Data-Centric* são claramente caracterizados pela sua estrutura regular em que os elementos têm pouca informação.
- **Documentos orientados a documentos (*Document-Centric Document*):**
  São caracterizados pela sua estrutura menos regular ou até mesmo irregular, não correspondendo a nenhum modelo relacional, em que os elementos têm bastantes atributos. Os documentos orientados a documentos são normalmente escritos em XML ou outro formato, tal como RTF, PDF, ou SGML, que depois são convertidos em XML. Ao contrário dos documentos orientados aos dados, não originam na base de dados.

Se necessitarmos de armazenar vários documentos XML, devemos utilizar uma base de dados XML pelas mesmas razões do armazenamento de dados numa base de dados relacional: gestão facilitada, performance de pesquisa melhorada, acesso concorrente, transações seguras, segurança, etc.

Existem duas formas diferentes de armazenar documentos XML numa base de dados:

- A primeira é mapear a estrutura dos documentos para a estrutura da base de dados e transferir os dados de acordo com esse mapeamento.
- A segunda é usar um conjunto fixo de estruturas que podem armazenar qualquer documento XML (Elements, Attributes, Text, etc).

As bases de dados que suportam o primeiro método são denominadas de BDs *XML-Enabled* e as que suportam o segundo método são denominadas de BDs XML Nativas. As BDs *XML-Enabled* têm o seu próprio modelo de dados (relacional, hierárquico, orientado a objectos) e mapeiam instâncias do modelo de dados XML para instâncias com o seu modelo próprio. As BDs XML nativas usam directamente o modelo de dados XML [29].

As BDs *XML-Enabled* são úteis quando os dados são publicados como XML ou quando são importados de um documento XML para uma base de dados (relacional por exemplo) existente. Não são a melhor forma de guardar documentos XML completos porque, apesar de armazenarem os dados e a sua hierarquia, descartam todo o resto tal como a identidade do documento, ordem dos nós, comentários, instruções de processamento, etc. Além disso, por obrigarem a uma estrutura, não podem armazenar documentos que não a respeitem, o que não acontece com as BDs XML nativas

que armazenam documentos completos e podem armazenar qualquer documento, independentemente da sua estrutura [27].

As bases de dados em estudo neste trabalho são as BDs XML nativas, que vão ser apresentadas mais à frente, mas para consulta de alguns exemplos de base de dados *XML-Enabled* ver AnexoB.

2.4. Bases de Dados XML Nativas

O termo base de dados XML nativa salientou-se na campanha de mercado de Tamino [16], uma base de dados XML nativa da Software AG. Devido ao sucesso dessa campanha, o termo tornou-se comum nas empresas que desenvolviam na altura produtos parecidos.

Quando as bases de dados XML nativas apareceram não se sabia muito bem o que fazer com elas. Seriam um substituto às bases de dados relacionais ou um retorno às bases de dados hierárquicas? Elas foram desenhadas para gerir grandes números de documentos XML que não tinham qualquer estrutura, catálogos, e que tinham hierarquias muito alargadas. Mesmo assim não era muito claro para que aplicações reais é que poderiam ser usadas [27].

Os dados relacionais são representados em linhas e colunas bem definidas e organizadas, daí que normalmente se diz que uma base de dados relacional é estruturada, o que não poderia fazer mais sentido [3]. A XML também pode, de certa forma, ser estruturada porque um conjunto de nomes e moradas podem ser igualmente representadas como os dados relacionais. Mesmo assim existem duas diferenças fundamentais: a XML pode representar hierarquias de uma forma que os dados relacionais não conseguem e não precisa de saber o quão longo ou complexo é um dado campo ou registo, enquanto que isso já se torna a essência dos dados relacionais.

Mas afinal o que significa nativa? Essencialmente, significa que os documentos são armazenados, indexados, e devolvidos no seu formato original, com todo o seu conteúdo, tags, atributos, referências a entidades e a com a sua ordem preservada [5].

2.4.1. Objectivos das BDs XML Nativas

As bases de dados XML nativas são destinadas especificamente para armazenar documentos XML [8]. Como outras bases de dados elas têm suporte para transacções, segurança, multiutilizadores, linguagens para consulta, APIs para programação (*DOM – Document Object Model,*

SAX – Simple API for XML), a única diferença é que o seu modelo é baseado em XML enquanto que outras bases de dados possuem outro tipo de modelo, como por exemplo, o modelo relacional [9].

São normalmente usadas para armazenar documentos orientados a documentos (ver 2.3.3.1). A razão principal para isso acontecer é devido ao seu suporte de linguagens de consulta XML. Outra razão é que as base de dados XML preservam a ordem dos documentos, instruções de processamento e comentários enquanto que as base de dados *XML-Enabled* não.

2.4.2. Características das BDs XML Nativas

Podemos caracterizar uma base de dados XML nativa da seguinte maneira [29]:

- Possui um modelo lógico para documentos XML, segundo o qual se deve armazenar e extrair os documentos. No mínimo esse modelo deve incluir elementos, atributos e PCDATA.
- Não é necessário ter uma camada física particular para o armazenamento. A base de dados pode ser construída como relacional, orientada a objectos, em arquivos compactados, etc.
- **Colecção de documentos:** Muitas bases de dados XML nativas suportam a noção de colecção. As colecções podem ser criadas com documentos do mesmo tipo ou então para armazenar todos os documentos juntos. Isto dá a mesma ideia do que uma tabela numa BD relacional ou um directório num sistema de ficheiros;
- **Linguagens de consulta:** Quase todas as bases de dados XML nativas suportam uma ou mais linguagens de consulta. As mais populares são a *XPath* e a *XQuery*;
- ***Update* e *Delete*:** As bases de dados XML nativas têm uma variedade de estratégias para actualizar e apagar documentos. Para tal, existe o *XUpdate* que é uma linguagem baseada em XML. Usa a *XPath* para identificar um conjunto de nós e depois especifica se os pretende apagar ou adicionar, ou inserir novos nós antes ou depois deles.
- **Interface de Programação de Aplicativos(APIs):** Quase todas as base de dados XML nativas oferecem APIs, por exemplo DOM, SAX, etc;
- ***Round-Tripping***: Uma das características mais importantes das bases de dados XML nativas é a capacidade de se poder armazenar um documento XML na base de dados e poder recuperar o documento intacto. Isto é vital, por exemplo, para aplicações médicas e legais nas quais, por lei, é necessário preservar a cópia exacta de cada documento.

- **Dados Remotos:** Podem incluir dados remotos nos documentos armazenados na base de dados.
- **Índices:** Os índices são utilizados para aumentar a velocidade de consulta à base de dados.

2.4.3. Arquitectura das BDs XML Nativas

As arquitecturas das bases de dados XML nativas [9] pode ser de duas categorias:

- **Bases de dados XML nativas baseadas em Texto**
  Nessa arquitectura os documentos XML são armazenados como texto, quer no próprio sistema de arquivos, quer num campo em base de dados relacional ou noutro sistema de textos proprietário. O seu funcionamento é semelhante a uma base de dados hierárquica, e a grande vantagem desta arquitectura é a forma de indexação, pois, assumindo que os dados são armazenados em bytes seguidos, só é necessário uma única pesquisa no índice.
- **Bases de dados XML nativas baseadas em Modelo**
  Nesta arquitectura além de armazenar os documentos como texto é construído um modelo deste documento, que também é armazenado. A forma como esse modelo é armazenado depende da base de dados, que pode ser relacional ou objecto-relacional. Esse tipo de base de dados pode ter problemas de desempenho ao realizar busca e retorno de documentos que não estejam no formato que está armazenado, como por exemplo invertendo a posição dos elementos do documento.

2.5. Estado da Arte

Como um dos objectivos principais deste trabalho foi estudar como as bases de dados XML nativas funcionavam, bem como o seu comportamento com o SInBAD (já mencionado anteriormente), tornou-se importante escolher uma base de dados xml nativa que satisfizesse as nossas exigências. Após analisar as diferenças existentes entre as várias hipótese que nos eram oferecidas, a que se destacou foi o *Sedna XML Native Database System*, desenvolvido pela equipa MODIS ISPRAS (que será analisado com mais detalhe no capitulo 3.2).

Entre outras características que o distinguia de outras bases de dados XML nativas (ver Anexo A), o Sedna era de utilização gratuita, apesar de não ser *open-source*, o que não acontece com a maior parte das BDs XML disponiveis.

Tabela 1 - BDs XML Nativas (gratuitas)

Produto	Desenvolvimento	API .NET	Últ. Actualização
Berkeley DB XML	C/C++	Sim	Dezembro, 2005
eXist	Java	Não	Março, 2004
myXMLDB	Java	Não	Janeiro, 2005
ozone	Java	Não	Março, 2004
Sedna XML DBMS	C/C++	Sim	Junho, 2004
XDBM	C	Não	Novembro, 2000
XDB	C/C++	Não	Novembro, 2001
Xíndice	Java	Não	Junho, 2005

Foi desenvolvido em C/C++ o que o tornava provavelmente mais rápido em termos de performance e tinha uma vasta quantidade de características que o transformavam numa BD segura, estável e de fácil manutenção.

Também pesou bastante o facto que, entre as várias contribuições para o desenvolvimento do Sedna, existisse uma API para desenvolvimento em .NET, que ajudou a implementação do sistema de gestão pretendido.

Algumas interfaces gráficas (GUI) estão disponíveis para gerir a base de dados Sedna, tal como *Sedna Database Aministrator* [12], *SednaAdmin* [11] e *SDBAdmin* [12] mas nenhuma destas ferramentas é *web-based* como era um dos objectivos principais deste trabalho por tornar o sistema válido em qualquer plataforma.

Por outro lado, enquanto que a maior parte dos sistemas fornece a capacidade de indexação, não existe nenhum sistema de gestão de base de dados XML que permite a configuração da indexação dos seus elementos e atributos.

3. ESTUDO DE CASO

Para cumprir da melhor forma o objectivo deste projecto verificou-se mais eficaz a utilização de uma base de dados XML nativa (BDXN) visto que os dados, que se pretendem armazenar, pertencem a documentos XML. Para tal, decidiu-se utilizar a base de dados XML nativa (BDXN) denominada Sedna [21], desenvolvida pela equipa MODIS no ISPRAS.

Por se esperar uma grande quantidade de dados nesta aplicação era aconselhável utilizar uma ferramenta que possibilitasse a indexação e a pesquisa do conteúdo dos documentos XML, para este efeito utilizou-se o Lucene [34], mais especificamente a API Lucene.Net [35], conhecida por DotLucene, por ser o motor de pesquisa *open source* mais rápido e eficaz para .NET.

Mais à frente será apresentado como a base de dados XML nativa Sedna se encontra estruturada bem como os seus dados são armazenados e como o Lucene.Net funciona.

3.1. Análise de Requisitos

Durante a análise dos requisitos, a aplicação que se pretendia desenvolver, deveria ter funcionalidades que permitissem obedecer aos objectivos principais do projecto. No final, a aplicação deveria dar ao utilizador a possibilidade de criar e apagar colecções de documentos XML de forma a agrupá-los por tipos, por exemplo, todos os documentos relativos a Livros na colecção “Livros” e os relativos a Teses ou Dissertações na colecção “Teses”.

As funcionalidades referentes aos próprios documentos, passariam por se poder importar um ou mais documentos, localizados numa pasta do computador local, para a base de dados, dentro ou fora de uma colecção. Estando os documentos incluídos na BD, a aplicação devia fornecer a possibilidade de edição, respeitando sempre a sintaxe da XML, e remoção destes.

A possibilidade de uma forma de pesquisa dos dados na aplicação, implicaria a utilização de uma biblioteca de indexação e pesquisa obrigando à definição dos parâmetros indexados. Para tal, deveria

ser possível criar, editar e remover esses parâmetros em cada colecção para que os dados fossem eficazmente indexados.

Deveria existir uma pesquisa por campos para restringir a pesquisa ao mínimo de resultados possíveis, além da pesquisa geral com base numa palavra-chave.

Finalmente, para que se pudesse livremente actualizar e editar vários documentos ao mesmo tempo, tornou-se indispensável a implementação de um editor de queries que fossem executadas directamente na base de dados.

3.2. Base de Dados XML Nativa – Sedna

O Sedna [21] é um Sistema de base de dados XML nativa que foi desenvolvida pela equipa MODIS no *Institute for System Programmimg of the Russian Academy of Sciences*. Implementa a linguagem *Xquery* e o seu modelo de dados utiliza técnicas desenvolvidas especialmente para esta linguagem.

Encontra-se implementado de raíz, o que permite impedir problemas como o tempo indesejável de interacção com outro sistema de base de dados. A plataforma de implementação foi Windows e encontra-se implementado em *Scheme* e C++, enquanto que a optimização e análise de consultas está em *Scheme*, a gestão das transacções e memória está em C++ [1].

3.2.1. Objectivos do Sedna

O Sedna foi projectado com os principais objectivos [28]:

- Suportar as características de um SGBD tradicional, tal como gestão de memória externa, linguagens de *update* e de consulta, controle de concorrência, optimização de consulta, segurança, etc;
- Suportar de forma eficiente, números ilimitados de documentos XML que poderão ter uma estrutura complexa e irregular;
- Suportar completamente a linguagem *Xquery* do W3C de forma que o sistema possa ser utilizado para resolver problemas de domínios diferentes tal como, consulta de dados XML, transformações de dados e até computação lógica;

3.2.2. Arquitectura do Sedna

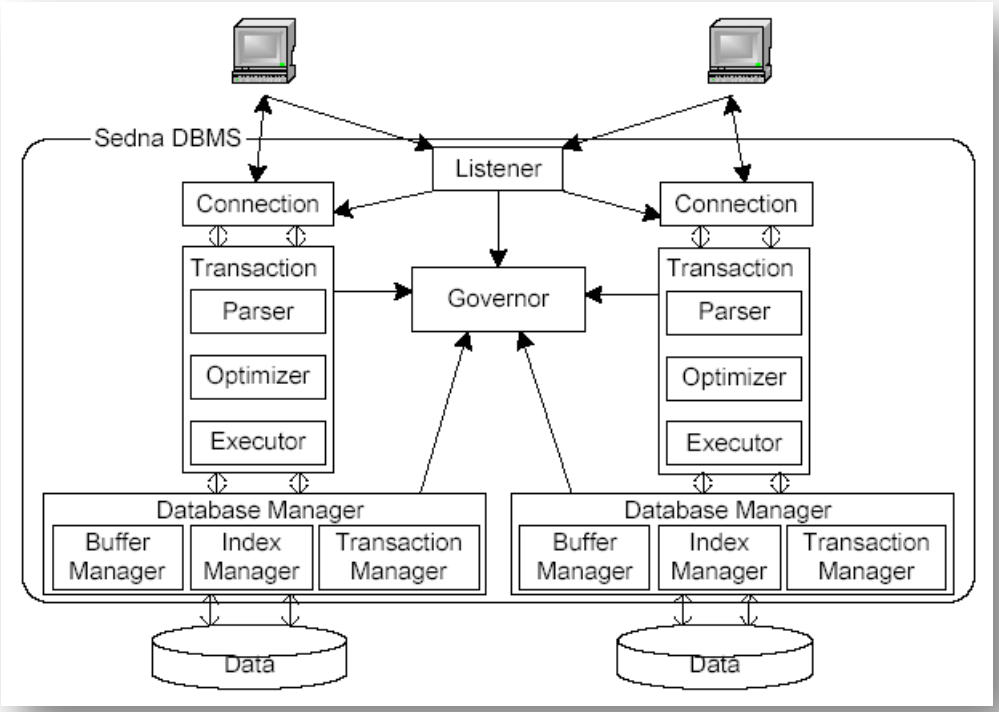

Figura 3 - Arquitectura do Sedna

(Fonte da imagem: [1])

O SGBD Sedna tem os seguintes componentes [1][22]:

- ***Governor:*** Age como centro de controle do sistema. Todos os outros componentes são registados no *governor*. O *governor* tem conhecimento que bases de dados e que transacções estão a decorrer no sistema e quem as controla.
- ***Listener:*** Cria uma instância do componente *Connection* por cada cliente e cria a conexão entre o cliente e o componente *Connection*.
- ***Connection:*** Guarda a sessão do cliente. Cria uma instância do componente *Transaction* por cada pedido "*begin transaction*" do cliente.
- ***Transaction:*** Guarda os seguintes componentes de execução:
    - ***Parser:*** Traduz a *query* na sua representação lógica, ou seja, uma estrutura hierárquica de operações.

- ***Optimizer:*** Guarda a representação lógica de consulta e produz um plano optimizado de execução de consulta que é uma estrutura hierárquica de operações de baixo nível sobre a estrutura de dados físicos.
- ***Executer:*** Interpreta o plano de execução.

- ***Database Manager:*** Cada instância guarda uma única base de dados e consiste nos serviços de gestão dessa BD tal como:
  - ***Index Manager:*** Guarda a localização dos índices construídos na base de dados.
  - ***Buffer Manager:*** É responsável pela interacção entre o disco e a memória principal.
  - ***Transaction Manager:*** Fornece os meios para o controlo da concorrência.

3.2.3. Armazenamento dos Dados

Ao investigar a forma como os dados iam ser organizados, foram tomadas algumas decisões para aumentar a velocidade de processamento [1][22]:

- São utilizados ponteiros para representar as relações entre os nós de um documento XML;
- Foi desenvolvida uma estratégia de armazenamento, baseado num *schema* descritivo, que é gerado dinamicamente, onde os nós de um documento XML estão agrupados de acordo com as suas posições no documento. Cada *path* do documento tem apenas um *path* no *schema* descritivo.

```
<library>
  <book>
    <title>Foundations of Databases</title>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
  </book>
  <book>
    <title>An Introduction to Database
           Systems</title>
    <author>Date</author>
    <issue>
      <publisher>Addison-Wesley</publisher>
      <year>2004</year>
    </issue>
  </book>
  . . .
  <paper>
    <title>A Relational Model for
           Large Shared Data Banks</title>
    <author>Codd</author>
  </paper>
</library>
```

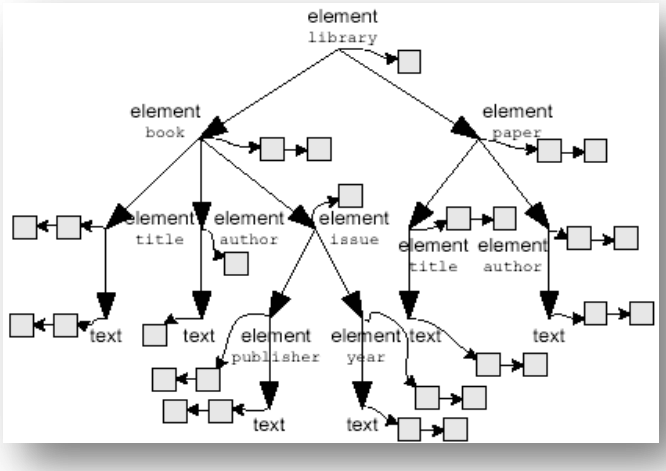

Figura 4 - Organização dos Dados
(Fonte da imagem: [1])

O *schema* descritivo está representado em forma de árvore. Cada nó está representado pelo tipo de nó XML (elemento, atributo, texto, etc, ..) e tem um ponteiro para o bloco de dados onde os nós se encontram armazenados.

No armazenamento do Sedna, a parte estrutural de um elemento é separada da textual [1][22]. Enquanto o texto é armazenado em blocos, a parte estrutural reflecte a sua relação com outros nós segundo uma descrição do nó (*node descriptor*), Figura 5.

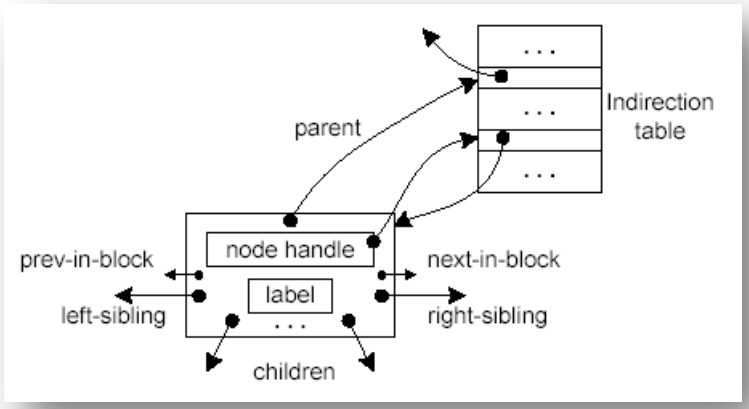

Figura 5 - Estrutura da descrição nodal

(Fonte da imagem: [1])

Componentes*:*

- ***Label:*** Contém a identificação do *numbering scheme*;
- ***Numbering scheme:*** Identifica cada nó acerca da sua posição relativa no documento XML. O objectivo principal é de fornecer mecanismos para determinar, o mais rápido possível, a relação estrutural entre dois nós;
- ***Node Handle:*** É um ponteiro para o registo da *Indirection table* que contém o ponteiro para o nó;
- ***Left-Sibling e Right-Sibling:*** São ponteiros utilizados para suportar a ordenação dos nós "irmãos" no documento;
- ***Next-in-Block e Prev-in-block:*** São ponteiros utilizados para interligar os nós para possibilitar a reordenação do documento;

Armazenar ponteiros para todos os nós filhos pode resultar num tamanho do bloco excessivo. Para evitar que isto aconteça, apenas são armazenados os ponteiros para o primeiro nó filho.

Por exemplo, pela Figura 4, o elemento *library* tem dois elementos *book* e um *paper* como elementos filho. No *schema* descritivo o nó do elemento *library* só tem dois nós filhos, ou seja, são ponteiros para o primeiro elemento *book* e para o primeiro elemento *paper*. Para alcançar todos os elementos *book* do elemento *library*, utiliza-se o ponteiro para o primeiro elemento *book* e depois segue o ponteiro *next-in-block*.

Ter apenas ponteiros para os primeiros nós filho faz com que haja mais espaço de armazenamento livre e torna a descrição dos nós de tamanho fixo [1][22].

3.2.4. Backup de dados

Como qualquer base de dados, por uma questão de segurança, deve ser feito um backup regularmente à base de dados em estudo [18].

No ínicio deste trabalho [30] a BD XML Nativa Sedna não oferecia um método eficaz e seguro para realizar um backup de todo o repositório mas, mais tarde, apareceram novas soluções para assegurar o bom funcionamento da base de dados. A base de dados XML Nativa Sedna ficou assim com várias formas de realizar um *backup*:

- ***Export/Import***
  O comando *se_exp* tem como finalidade exportar e importar dados. Este método gera um conjunto de ficheiros XML e scripts XQuery para repor a base de dados nas mesmas condições quando foi feita a exportação.
- ***File system level backup***
  Uma estratégia alternativa para fazer o backup da base de dados é copiar directamente as directorias que o Sedna usa para guardar os dados da base de dados.
- ***Hot Backup***
  Outra estratégia é fazer o backup enquanto a base de dados está a correr. O objectivo é criar uma cópia de backup consistente enquanto os utilizadores estão a utilizar a BD. Esta cópia pode ainda ser restaurada copiando a directoria do backup correspondente para a pasta onde o Sedna guarda as bases de dados. A principal diferença entre o File system level backup e o Hot backup é que a base de dados em questão não precisa de ser parada. Em contrapartida, a recuperação a partir do Hot backup pode se tornar mais lenta.

3.2.5. Limitações do Sedna

A consistência e integridade dos dados armazenados é de extrema importância e qualquer programador deve ter o cuidado de, sempre que possível, efectuar uma validação dos dados antes de os guardar ou submeter. No entanto, mesmo com todos os cuidados que se possam ter antes de submeter os dados há sempre uma pequena margem de erro, um pormenor que é esquecido de ser validado. Por esta mesma razão, a existência de redundância na validação de dados é desejável.

Infelizmente, mesmo a aplicação sendo desenhada e implementada com o cuidado de evitar a inserção de inconsistência nos dados, o Sedna não tem qualquer mecanismo de controlo para garantir que qualquer falha que possa surgir da aplicação seja corrigida, ou pelo menos, reportada ao utilizador.

Um dos mecanismos que poderia ser utilizado pelo Sedna, seria a possibilidade de definir restrições por elemento para os ficheiros introduzidos, por exemplo, garantir que um documento que contivesse o elemento "data" não pudesse ser introduzido na base de dados se o seu valor fosse negativo. Este é apenas um exemplo de uma solução que certamente iria melhorar o Sedna.

3.3. Lucene

O Lucene [34] é um software *open source,* da *Apache Software Foundation*, de pesquisa e indexação de documentos, totalmente escrito em Java. Tem algumas características que o distingue:

- Escalável;
- Preciso;
- Faz um *ranking* de resultados;
- Ordenação;

Para o Lucene não importa a origem dos dados, o seu formato, ou até mesmo a linguagem em que foram escritos, desde que esses dados possam ser convertidos para texto. Por isto, o Lucene pode ser utilizado para indexar e pesquisar dados armazenados em arquivos, páginas *web* em servidores remotos, documentos num sistema de arquivos local, arquivos de texto, documentos Microsoft Word, documentos HTML, arquivos PDF, ou qualquer outro formato do qual possa ser extraído informação textual [23][34].

O Lucene oferece suporte para outras linguagens além do Java:

- Lucene4C – C;
- CLucene – C++;
- MUTIS – Delphi;
- Lucene.Net – C# .Net;
- Zend Framework – PHP;
- Ferret – Ruby.

Para a aplicação desenvolvida foi utilizada a API Lucene.Net, normalmente denominada de DotLucene, é o motor de pesquisa *open source* mais rápido e eficaz para .NET.

3.3.1. Funcionamento geral do Lucene

A indexação passa por um processo de análise do documento que, automaticamente, o converte para um texto simples. A extracção do texto é feita a partir de um *Analyser*, classe que contém as regras para a realização desse trabalho. O segundo passo após a extracção do texto é organizar o índice, que pode ser acedido por pesquisa no futuro [23][34].

Na Figura 6 está representado o processo de indexação no Lucene:

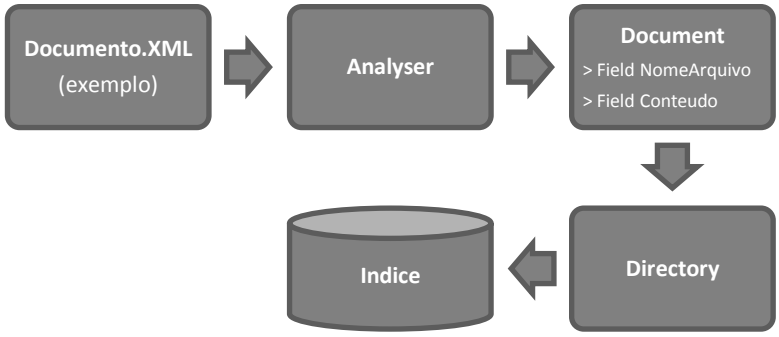

Figura 6 - Processo de Indexação no Lucene

A classe *Document* é uma unidade de indexação e pesquisa que permite armazenar campos (*Fields*). Pode dizer-se que um *field* só pode ser armazenado num *Document*, pois possui um nome e um valor. Não é possível armazenar dois *Fields* com o mesmo nome no mesmo documento. Mas um documento pode conter um ou mais *Fields*.

```
void addDocument(IndexWriter writer, string path)
{
  Document doc = new Document();
  StreamReader sr = new StreamReader(path, System.Text.Encoding.Default);
  doc.Add(Field.Text("text", sr));
  doc.Add(Field.Keyword("filename", path));
  writer.AddDocument(doc);
  sr.Close();
}
```

A classe *Directory* é responsável por endereçar o índice. Os *Documents* são armazenados na *Directory*. A classe *Directory* mostra a localização de onde serão armazenados os arquivos do Lucene.

O *IndexWriter* é o responsável pela criação do índice, ao qual, através desta classe, os *Documents* podem ser adicionados. Recebe como parâmetro o *Directory* e o *Analyser* para efectuar a gravação do índice e o parâmetro "true" informa que um novo índice será criado, ou destruído caso já exista algum.

```
IndexWriter writer = new IndexWriter(@"C:\your\index\directory", new StandardAnalyzer(), true);
```

O *IndexSearcher* tem o papel de executar a pesquisa no índice. Os critérios de pesquisa são passados para a função de pesquisa através do objecto *Query*. Em seguida, o objecto *Query* é construído através da *QueryParser*, para o qual passamos como parâmetro o nome do campo a ser procurado e o valor que este pode ter.

```
IndexSearcher searcher = new IndexSearcher(@"C:\your\index\directory");

string q = "dotlucene";
Query query = QueryParser.Parse(q, "text", new StandardAnalyzer());


Hits hits = searcher.Search(query);
Console.WriteLine("Found " + hits.Length() + " document(s) that matched query '" + q + "':\r\n");
for (int i = 0; i < hits.Length(); i++)
{
  Document doc = hits.Doc(i);
  Console.WriteLine(doc.Get("filename") + "\r\n");
}
```

No capítulo seguinte será abordada a preparação das ferramentas utilizadas, as limitações (da BD e da API) que se encontraram durante a implementação e de que forma foram contornadas. Também será explicada a arquitectura da aplicação e que classes foram implementadas bem como as funcionalidades finais da aplicação de acordo com os requisitos iniciais do sistema.

4. APLICAÇÃO DESENVOLVIDA

Para o desenvolvimento da aplicação/sistema foi utilizada como tecnologia de desenvolvimento a plataforma .NET 2.0 (C#). A interface deste sistema foi construída em ASP.NET e suporte em AJAX.

A aplicação utiliza como base de dados uma base de dados XML nativa (BDXN) denominada Sedna (versão 3.1), desenvolvida pela equipa MODIS no ISPRAS e para a pesquisa e indexação dos documentos XML é utilizado o Lucene (versão 2.4), o qual está explicado mais à frente.

Neste capítulo serão abordados temas como a preparação da ferramenta, as limitações (da BD e da API) que se encontraram durante a implementação e de que forma foram contornadas, arquitectura da aplicação e que classes foram implementadas, funcionalidades finais da aplicação bem como a demonstração do resultado final.

4.1. Preparar base de dados

Antes de correr a aplicação é necessário instalar o Sedna. Os comandos de configuração do Sedna [21] terão que ser introduzidos pela linha de comandos na directoria onde o sedna foi instalado, ver AnexoC.

Para a aplicação funcionar correctamente é necessário iniciar o Sedna Server, e que exista uma base de dados (utilizar-se-á o nome da base de dados “tempdb” que é a que está a ser usada por defeito) e iniciar a base de dados.

Para mais pormenores sobre os comandos do Sedna Server bem como a sequência a utilizar para a iniciar a BD, consultar o AnexoC, onde se pode encontrar as instruções que podem ser executadas, bem como as mensagens que devem aparecer em casos de sucesso.

4.2. Limitações encontradas

Durante a implementação da aplicação encontraram-se algumas limitações tanto na base de dados XML nativa como na API .NET. Algumas dessas limitações foram ultrapassadas com sucesso e outras apenas foram contornadas com soluções alternativas. Essas limitações seguem descritas na secção seguinte.

4.2.1. Limitações da BD XML Nativa Sedna

Com a BD XML nativa Sedna tiveram que se ter em atenção alguns pormenores durante a implementação:

- Ao incluir documentos sem colecção, o servidor do Sedna não suportava a inclusão de mais de 20 documentos seguidos, se por acaso se tentar importar 21 documentos XML seguidos, o servidor bloqueia na 21ª tentativa, sendo necessário reiniciá-lo.
- O facto de não conseguirmos criar colecções cujos nomes contenham acentos e cedilhas também levou a que fosse necessária a criação de uma função auxiliar que substituisse os caracteres que tivessem acentos ao seu correspondente sem acento, por exemplo, “Dissertação” ficaria “Dissertacao”.
- O *Sedna Server* vai abaixo quando se tenta consultar alguns documentos internos, tal como o *$documents.xml* que permitiria listar todos os documentos XML existentes na base de dados, com ou sem colecção;

4.2.2. Limitações da API Sedna.Net

Uma das grandes limitações encontradas na API Sedna.Net foi a falta de métodos que nos permitissem criar e remover colecções, bem como adicionar-lhes documentos. Para colmatar essa falha foi criada a classe *Sedna_Comunic*, mencionada de seguida no capítulo 4.3.1, que contém funções que nos permitem facilmente gerir as colecções de documentos XML.

Outra limitação foi encontrada quando se tentava obter o conteúdo dos documentos XML da base de dados e o texto não estava a ser correctamente codificado. Após testar directamente na linha de comandos, sem intermédio da API .Net, a inclusão de um documento na BD e a seguir reaver o seu conteúdo, verificou-se que o problema se encontrava na API. Acontecia que a comunicação era realizada por *sockets* e os dados recebidos no *socket* eram interpretados como sendo ASCII e deveriam ser lidos como UTF8, então de forma a resolver o problema onde existia

"System.Text.Encoding.ASCII" trocou-se para *"System.Text.Encoding.UTF8"*, o que resolveu de imediato o problema de *encoding*.

4.3. Estrutura da Aplicação

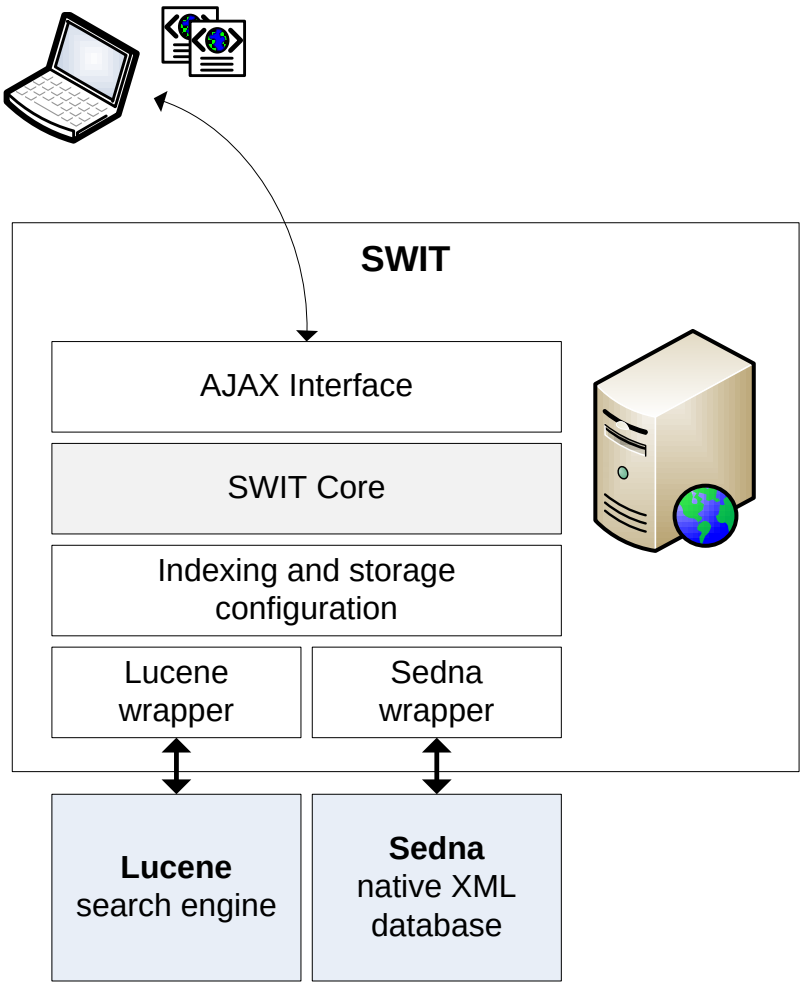

Figura 7 - Estrutura da Aplicação

4.3.1. Classe Sedna_Comunic

Como a aplicação interage com a API .NET do Sedna, foi implementada uma classe, de nome, **Sedna_Comunic**, com o objectivo de integrar todas as funções que necessitavam de comunicar com a API, organizando assim o código da implementação e tornando possível a reutilização de código.

A classe criada também serviu para armazenar métodos que oferecessem suporte para a existência de colecções, pois a API não tinha qualquer método que permitisse ao programador criar ou remover uma colecção, nem importar documentos para colecções existentes na base de dados.

O construtor da classe cria uma sessão na base de dados para que se possa aceder aos dados pelos métodos implementados na classe **Sedna_Comunic**.

```
Sedna_Comunic.SednaComunication();

private SednaSession session = SednaSession.CreateSession("localhost", "tempdb", "SYSTEM", "MANAGER");
```

4.3.1.1. Métodos da classe Sedna_Comunic

De seguida encontram-se apresentados de forma muito sumária, os métodos pertencentes à classe *Sedna_Comunic*:

public void CloseSession()

Função implementada para fechar a sessão do Sedna.

public ListItemCollection ListCollections()

Esta função devolve um lista de todos as colecções existentes na base de dados.

public ListItemCollection ListDocuments()

Esta função devolve uma lista de todos os documentos que não se encontram inseridos numa colecção, ou seja, que são *standalone documents*.

public ListItemCollection ListDocuments(string colName)

Esta função tem a mesma função que a anterior, só que lista todos os documentos pertencentes à colecção, sendo o nome do documento passado através do parâmetro *colName*.

private string ChangeColName(string colName)

Devido à impossibilidade de as colecções poderem ter nomes com acentos nem cedilhas, tornou-se necessário implementar uma função que modificasse o nome da colecção, removendo da palavra tudo o que pudesse originar erro, por exemplo, a função com o parâmetro de entrada “Dissertação”, retornaria a string “Dissertacao”.

public bool ColExists(string colName)

Para um controle de erros, foi criada esta função que nos possibilita, antes de criar uma nova colecção, verificar se a colecção, sendo o nome do documento passado através do parâmetro *colName,* existe na base de dados. Se existir retorna *"true"*, caso contrário *"false"*.

public bool DocExists(string docName)

Esta função foi criada com o objectivo de se poder verificar se um documento, sendo o nome do documento passado através do parâmetro *docName*, existe na base de dados como documento standalone.

public bool DocExists(string docName, string colName)

Com o mesmo objectivo que a anterior, esta função verifica se um documento existe numa colecção, sendo o nome do documento e da colecção passado através dos parâmetros *docName* e *colName*, respectivamente.

public bool IsColEmpty(string colName)

Esta função verifica se a colecção, com o nome passado através do parâmetro *colName,* contém algum documento. Se a colecção se encontrar vazia, sem nenhum documento, a função devolve *"true"*, caso contrário *"false"*.

public void CreateCollection(string colName)

Esta função permite criar uma nova colecção na base de dados, sendo o nome da colecção passado através do parâmetro *colName*.

public void DeleteCollection(string colName)

Esta função permite apagar uma colecção da base de dados, sendo o nome da colecção passado através do parâmetro *colName*.

public string RunQuery(string query)

Para existir a possibilidade de executar queries criadas pelo utilizador, esta função utiliza a *string* passada através do parâmetro *query* e executa-a utilizando o método *Execute* da API .NET do Sedna. A string de retorno contém o resultado da execução da query.

public void UploadDocument(string docName, string path)

Esta função permite inserir um documento XML, sem colecção definida, na base de dados, sendo o seu nome passado através do parâmetro *docName*, localizado no endereço passado através do parâmetro *path*.

***public void** UploadDocument(**string** docName, **string** colName, **string** path)*

Esta função permite inserir um documento XML, localizado no endereço passado através do parâmetro *path*, numa colecção da base de dados, sendo o seu nome e colecção passado através dos parâmetros *docName* e *colName*, respectivamente.

***public void** DeleteDocument(**string** docName)*

Esta função permite apagar um documento, sem colecção, da base de dados, sendo o nome do documento passado através do parâmetro *docName*.

***public void** DeleteDocument(**string** docName, **string** colName)*

Esta função permite remover um documento XML, localizado no endereço passado através do parâmetro *path*, de uma colecção da base de dados, sendo o seu nome e colecção passado através dos parâmetros *docName* e *colName*, respectivamente.

***public string** OpenDocument(**string** docName)*

Para consultar o conteúdo integral de um documento, sem colecção, foi criada esta função que devolve numa *string* todo o conteúdo XML existente no documento, com o nome passado através do parâmetro *docName*.

***public string** OpenDocument(**string** docName, **string** colName)*

Para consultar o conteúdo integral de um documento de uma colecção, sendo o seu nome e colecção passado através dos parâmetros *docName* e *colName*, respectivamente, foi criada esta função que devolve numa *string* todo o conteúdo XML existente no documento.

4.3.2. Classe Lucene_Comunic

Foi implementada uma classe, **Lucene_Comunic**, com o objectivo de integrar todas as funções de indexação e pesquisa que necessitavam de comunicar com a API Lucene.Net, organizando assim o código da implementação, tornando possível a reutilização de código, bem como tornar menos pesada a implementação da *interface*.

4.3.2.1. Métodos da classe Lucene_Comunic

De seguida encontram-se apresentados de forma muito sumária, os métodos pertencentes à classe *Lucene_Comunic*:

private void CreateInfoFile(string IndexFolder)

Esta função vai gerar um ficheiro *.xml denominado de *"standAlone_fields.xml"* que conterá, posteriormente, toda a informação respectiva aos parâmetros da indexação dos documentos sem colecção armazenados na base de dados.

No final de execução desta função, o documento XML que vai conter um elemento raiz denominado de "fields", e será armazenado numa pasta, reservada à indexação, localizada no servidor da aplicação, sendo o endereço da aplicação passado através do parâmetro *IndexFolder*.

private void CreateInfoFile(string colName, string IndexFolder)

Esta função vai gerar um ficheiro *.xml denominado de "[nome_colecção]_fields.xml", sendo o nome da colecção passado através do parâmetro *colName*, que posteriormente, conterá toda a informação respectiva à indexação dos documentos da colecção colName.

No final de execução desta função, o documento XML que vai conter um elemento raiz denominado de "fields", e será armazenado numa pasta, reservada à indexação, localizada no servidor da aplicação, sendo o endereço da aplicação passado através do parâmetro *IndexFolder*.

public bool AddFieldToInfoFile(string fieldName, string path, string IndexFolder)

Esta função edita o ficheiro denominado *"standAlone_fields.xml"* situado na pasta, reservada à indexação, localizada no servidor da aplicação, sendo o endereço da aplicação passado através do parâmetro *IndexFolder*.

Se o no ficheiro em questão ainda não existir nenhuma informação igual à fornecida pelos parâmetros de entrada *fieldName* e *path*, será adicionado um novo nó ao documento sob a forma <field name= fieldName>path<\field> e retornará "true". De outra forma, retornará "false".

public bool AddFieldToInfoFile(string colName, string fieldName, string path, string IndexFolder)

A funcionalidade desta função é semelhante à anterior mas o documento que é editado identifica-se pelo nome "[nome_colecção]_fields.xml", sendo o nome da colecção passado através do parâmetro *colName*.

private IndexWriter CreateIndex(string IndexFolder)

Esta função permite indexar os documentos sem colecção existentes na base de dados. A classe *IndexWriter* da API Lucene.Net cria uma pasta, denominada *"standAlone"*, inserida na pasta "*Index"* (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do

parâmetro *IndexFolder*. Na pasta *"standAlone"* serão armazenados todos os ficheiros dedicados à indexação, criados pela API.

private IndexWriter CreateIndex(string colName, string IndexFolder)

Esta função permite indexar os documentos da colecção, cujo nome é passado através do parâmetro *colName*. A classe *IndexWriter* da API *Lucene.Net* cria uma pasta, denominada com o nome da colecção, inserida na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*. Na pasta criada pelo *IndexWriter* serão armazenados todos os ficheiros dedicados à indexação, criados pela API.

public void DeleteIndex(string IndexFolder)

Esta função consiste em apagar todos os dados referentes à indexação de documentos sem colecção. O documento *"standAlone_fields.xml"* e a pasta com o nome *"standAlone"*, existentes na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*, serão removidos deixando de existir então qualquer informação de indexação para aqueles documentos.

public void DeleteIndex(string colName, string IndexFolder)

Esta função consiste em apagar todos os dados referentes à indexação de documentos da colecção, cujo nome é passado através do parâmetro *colName*. O documento "[nome_colecção]_fields.xml" e a pasta com o nome *"*[nome_colecção]*"*, existentes na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*, serão removidos deixando de existir então qualquer informação de indexação para os documentos da colecção.

public void AddDocToIndex(string IndexFolder)

Esta função recria a indexação dos documentos sem colecção, ou seja, quando é adicionado, editado ou removido, um documento XML à base de dados ou os parâmetros de indexação são modificados torna-se necessário actualizar os dados indexados anteriormente.

Desta forma esta função cria o índice do zero, e extrai a informação dos documentos da base de dados sem colecção tendo em consideração os parâmetros de indexação inseridos no documento *"standAlone_fields.xml"*, existente na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*.

public void AddDocToIndex(string colName, string IndexFolder)

Esta função recria a indexação dos documentos da colecção, cujo nome é passado através do parâmetro *colName*, pois quando é adicionado, editado ou removido, um documento XML à base de dados ou os parâmetros de indexação são modificados torna-se necessário actualizar os dados indexados anteriormente.

Desta forma esta função cria o índice do zero, e extrai a informação dos documentos da base de dados sem colecção tendo em consideração os parâmetros de indexação inseridos no documento "[nome_colecção]_fields.xml", existente na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*.

public ListItemCollection Search(ListItemCollection list, string IndexFolder)

Esta função permite efectuar uma pesquisa tendo em consideração uma lista, que é fornecida pelo parâmetro de entrada *list*, que contém os nomes dos campos e conteúdos que foram preenchidos pelo utilizador.

Vai retornar uma lista contendo todos os nomes dos documentos sem colecção, que correspondem à pesquisa efectuada cruzando os dados fornecidos pelo parâmetro de entrada *list*.

Utiliza a informação correspondente aos documentos sem colecção, existente na pasta *"standAlone"*, existente na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*.

public ListItemCollection Search(string colName, ListItemCollection list, string IndexFolder)

A funcionalidade desta função é semelhante à anterior mas a pesquisa é efectuada a documentos pertencentes a uma colecção, cujo nome é passado através do parâmetro *colName*.

Utiliza a informação correspondente aos documentos à colecção, existente na pasta *"*[nome_colecção]*"*, existente na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*.

public ListItemCollection SearchGlobal(string searchKey, string IndexFolder)

Esta função permite efectuar uma pesquisa tendo em consideração uma palavra chave, que é fornecida pelo parâmetro de entrada *searchKey*, preenchida pelo utilizador.

Vai retornar uma lista contendo todos os nomes dos documentos sem colecção, em que no conteúdo XML aparece a palavra chave.

Utiliza a informação correspondente aos documentos sem colecção, existente na pasta *"standAlone"*, e o documento *"standAlone_fields.xml"* existentes na pasta "*Index*" (reservada à

indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*.

public ListItemCollection SearchGlobal(string colName, string searchKey, string IndexFolder)

A funcionalidade desta função é semelhante à anterior mas a pesquisa é efectuada a documentos pertencentes a uma colecção, cujo nome é passado através do parâmetro *colName*.

Utiliza a informação correspondente aos documentos à colecção, existente na pasta "[nome_colecção]", e o documento "[nome_colecção].xml" existentes na pasta "*Index*" (reservada à indexação) localizada no servidor da aplicação, cujo endereço é passado através do parâmetro *IndexFolder*.

4.4. Funcionalidades da aplicação

De forma a responder aos requisitos iniciais da aplicação foram implementadas as seguintes funcionalidades:

- Criar colecções;
- Apagar colecções;
- Upload de um documento xml;
- Upload de vários documentos xml em simultâneo;
- Editar um documento xml;
- Apagar um documento xml;
- Criação dos parâmetros dos índices;
- Edição e remoção dos parâmetros dos índices;
- Pesquisar por uma palavra-chave;
- Pesquisar avançada por campos;
- Realizar consultas manuais à BD;

Por uma questão de organização, ao implementar a interface da aplicação teve-se o cuidado de criar uma página para cada funcionalidade já que a finalidade de umas é independente das outras. O código investido na implementação de cada funcionalidade acabou por ser pouco mas eficiente devido à utilização das duas classes, *Sedna_Comunic* e *Lucene_Comunic*, já mencionadas anteriormente.

Em cada página existe uma zona onde a aplicação dá um pequeno *feedback* ao utilizador sobre se a operação foi bem sucedida ou se ocorreu algum erro, mostrando ao utilizador que campos foram mal introduzidos.

Durante a implementação tentou-se ao máximo diminuir possíveis situações de erro tornando disponíveis, ou não, certas opções da interface e verificando sempre que possível os dados de entrada. Com isto tenta-se evitar que sejam feitas chamadas ao servidor com dados errados ou incompletos.

De seguida será explicado resumidamente como se pode utilizar a interface.

4.4.1. Criar colecções

Para criar uma nova colecção na base de dados basta introduzir um nome para a colecção como mostra a Figura 8.

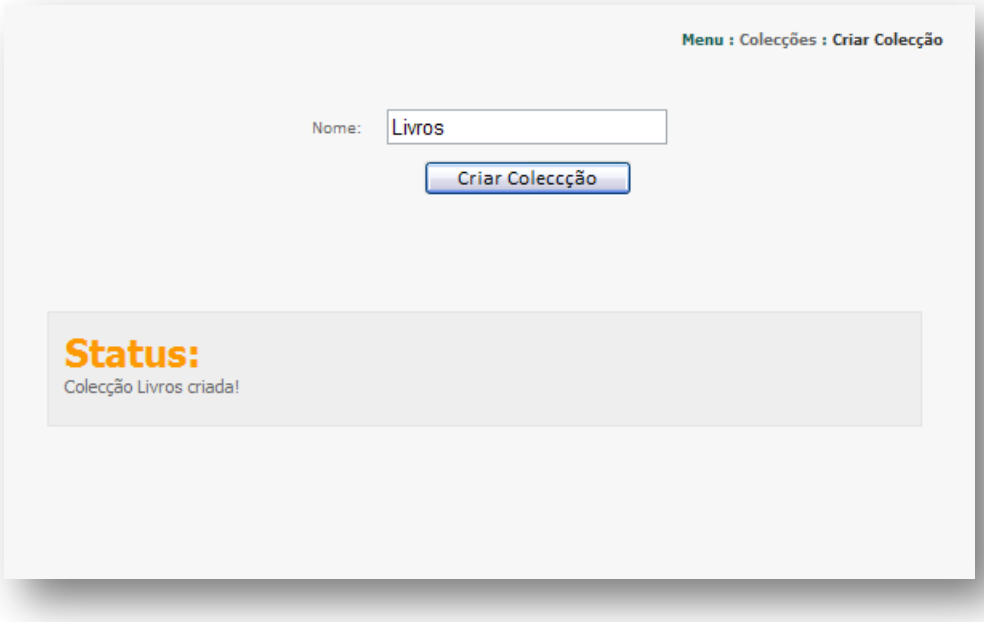

Figura 8 - Interface - Criar Colecção

4.4.2. Remover colecção

Para remover uma colecção, é apresentada uma lista das colecções existentes, onde o utilizador terá que seleccionar a que pretende apagar. A colecção só é removida se estiver vazia, ou seja, se não tiver armazenado nenhum documento. Caso o utilizador pretenda remover a colecção e todo o

seu conteúdo deverá seleccionar a *CheckBox "Apagar Recursivamente"*, como se pode verificar pela Figura 9. Uma mensagem de confirmação aparecerá, à qual o utilizador terá que confirmar que realmente pretende remover.

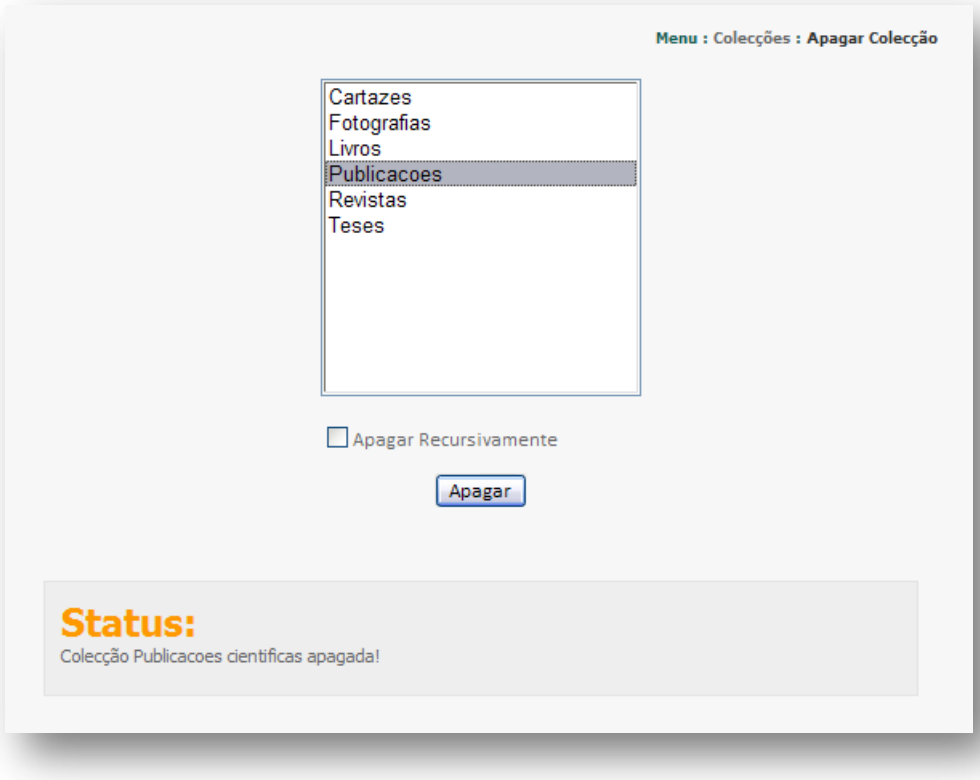

Figura 9 - Interface - Remover Colecção

4.4.3. Importar documentos XML

Pela Figura 10, podemos observar como se pode importar um documento XML para a base de dados. No campo **Ficheiro** é introduzido a localização do ficheiro, depois é escolhida uma colecção da lista, e no campo **Nome** é colocado o nome que se pretende dar ao documento, se o campo estiver vazio é dado por defeito o nome documento XML.

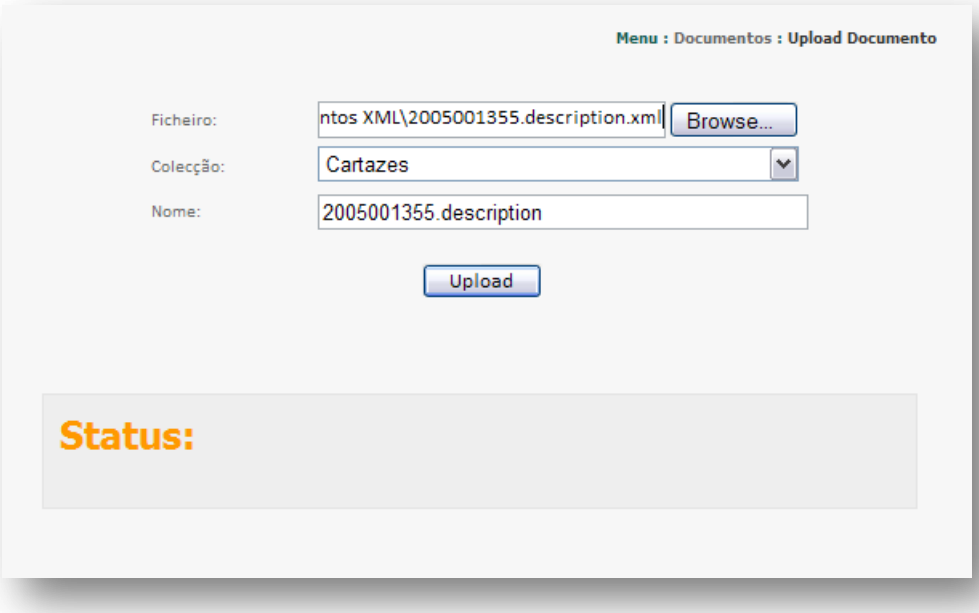

Figura 10 - Interface - Upload de um ficheiro .xml

4.4.4. Importar vários documentos XML em simultâneo

Caso se pretenda importar mais que um ficheiro de cada vez, a aplicação tem a possibilidade de importar um arquivo .zip, que contenha ficheiros XML. A forma como os documentos são colocados nas colecções deixa-se ao critério do utilizador:

- pode seleccionar uma colecção da lista e todos os documentos serão colocados nessa colecção;
- se seleccionar a *CheckBox "***Organizar automaticamente***"* a aplicação coloca os documentos na colecção a que pertencem, a partir do URL do campo //dc:identifier (específico do SInBAD).

A colecção (com o nome do tipo de documento) é criada, se a aplicação verificar, pela função colExists(nome_colecção) (capítulo 4.3.1.1) da classe **Sedna_Comunic**, que a colecção não existe. Figura 11.

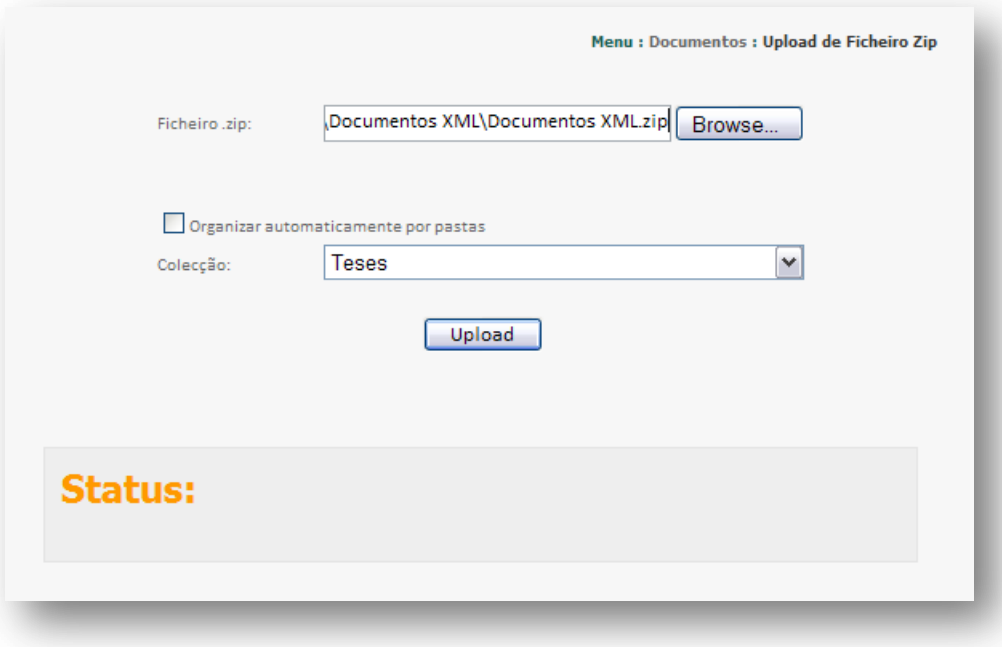

Figura 11 - Interface - Upload de um arquivo .zip de ficheiros .xml

4.4.5. Editar documento

Também existe a possibilidade de edição de um documento que se encontra na base de dados. Como se pode ver pela Figura 12, seleccionando uma colecção, de seguida um documento que lhe pertença e clicando em **Abrir**, o conteúdo é exposto ao utilizador para poder editar o que pretender.

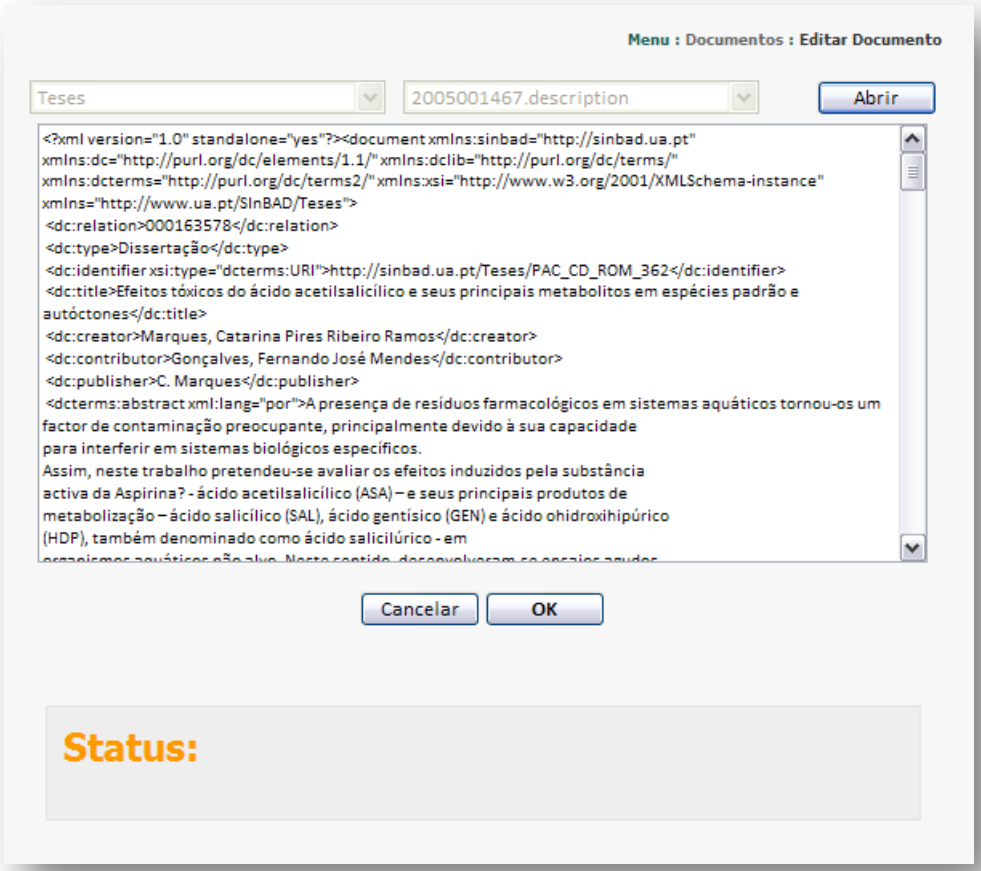

Figura 12 - Interface - Editar um documento

4.4.6. Remover documento XML

Para o caso do utilizador pretender remover algum documento basta que este escolha a colecção, de seguida o documento que pretende remover e clicar em **Remover**. Uma mensagem de confirmação aparecerá, à qual o utilizador terá que confirmar que realmente pretende remover.

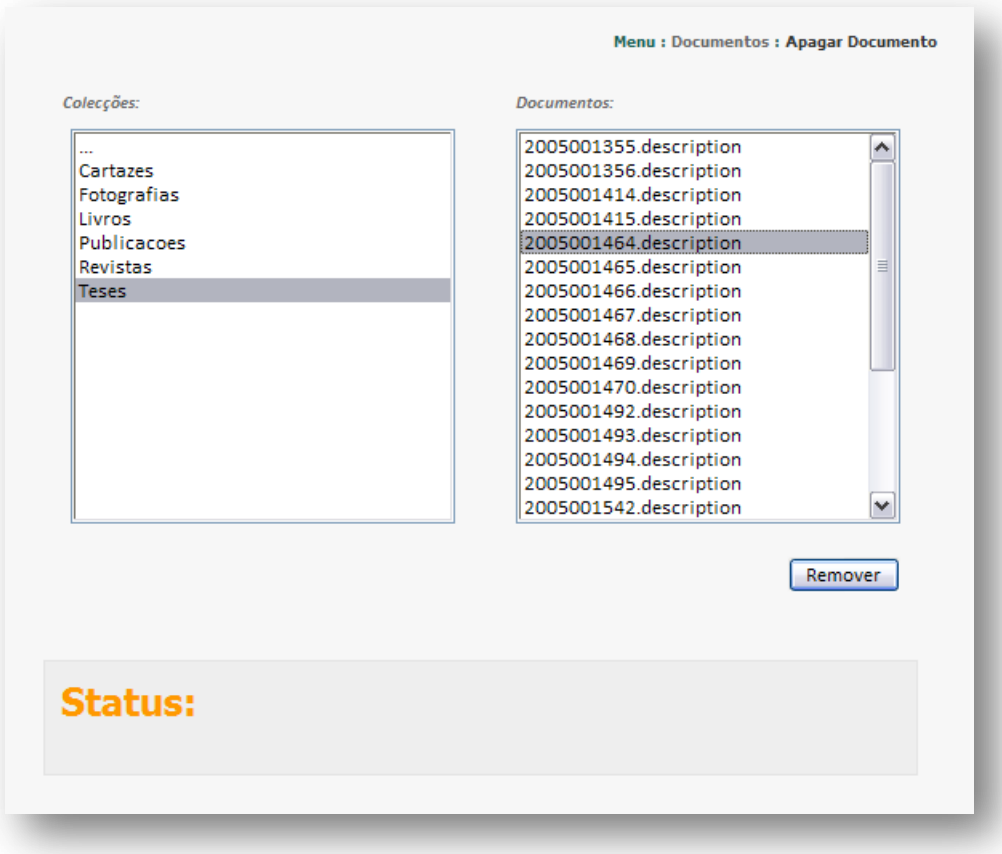

Figura 13 - Interface - Remover um documento

4.4.7. Indexar dados da BD

Para que a indexação dos dados seja efectuada da melhor forma, podemos adicionar parâmetros de indexação como indica a Figura 14. Basta escolher a colecção que queremos que seja afectada, de seguida introduzir o *XPath* do documento XML e o nome do campo que lhe queremos atribuir e clicar em **Criar Índice**.

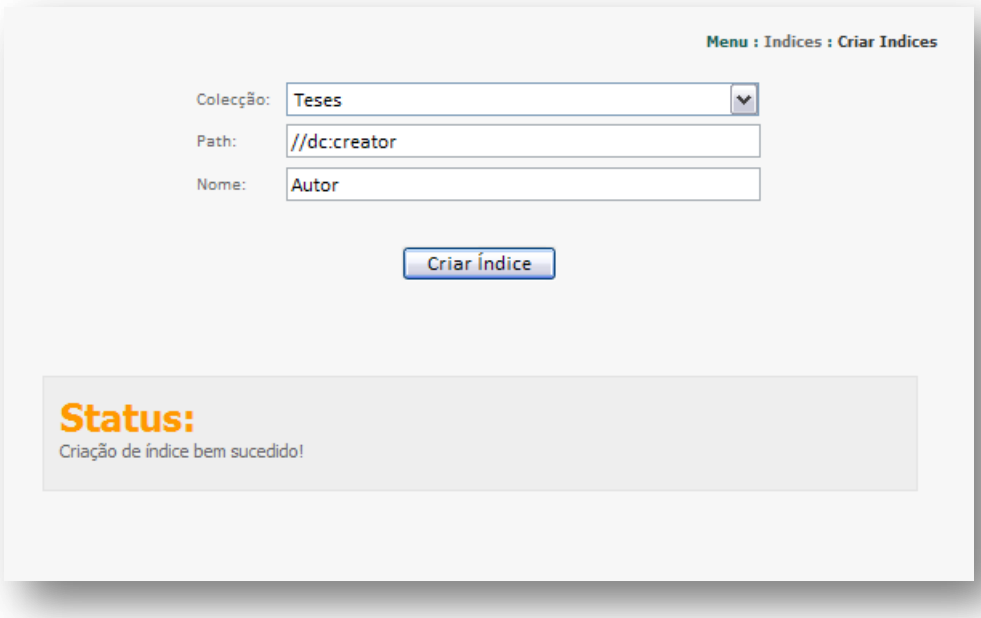

Figura 14 - Interface - Criação de parâmetros para indexação

4.4.8. Editar parâmetros de indexação

Os parâmetros da indexação podem ser consultados, editados e removidos pela aplicação. Ao escolher a colecção da lista, são listados os parâmetros para poderem ser editados, Figura 15.

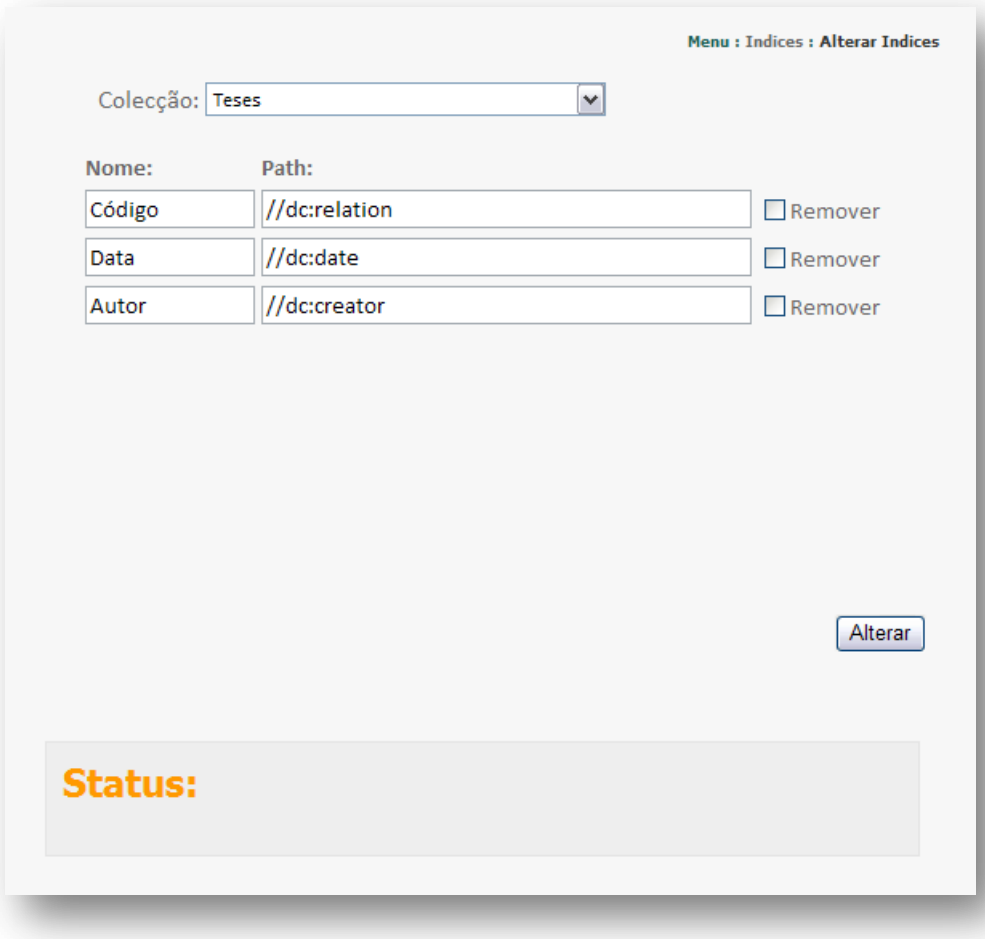

Figura 15 - Interface - Edição dos parâmetros da indexação

4.4.9. Pesquisa simples

O mecanismo de pesquisa simples obriga a que seja escolhida uma colecção para que os dados dos documentos dessa colecção possam ser consultados. Introduzindo uma palavra-chave, a aplicação vai listar o nome dos documentos que tenham a palavra chave nos campos indexados, Figura 16.

Após a listagem dos resultados da pesquisa o utilizador poderá escolher um resultado e abrir para ver os dados desse ficheiro, como se pode ver pela Figura 17.

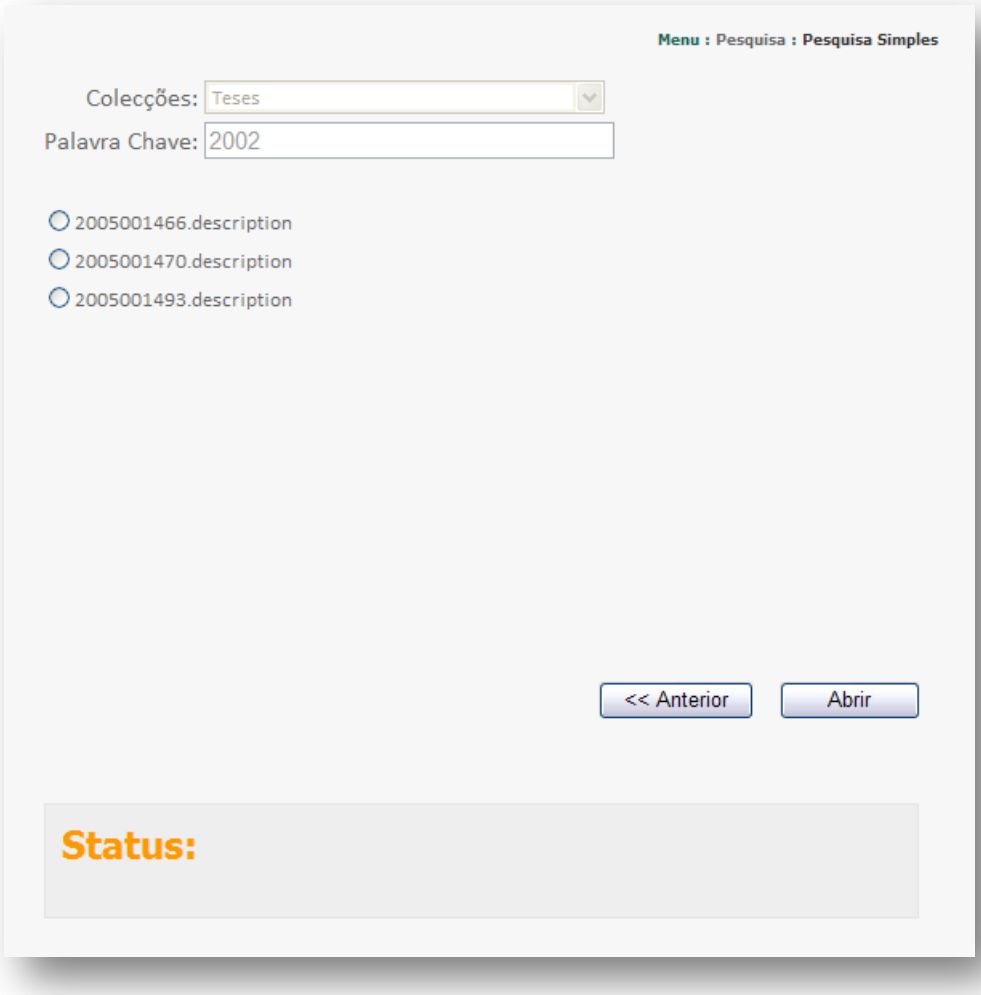

Figura 16 - Interface - Pesquisa Simples v1

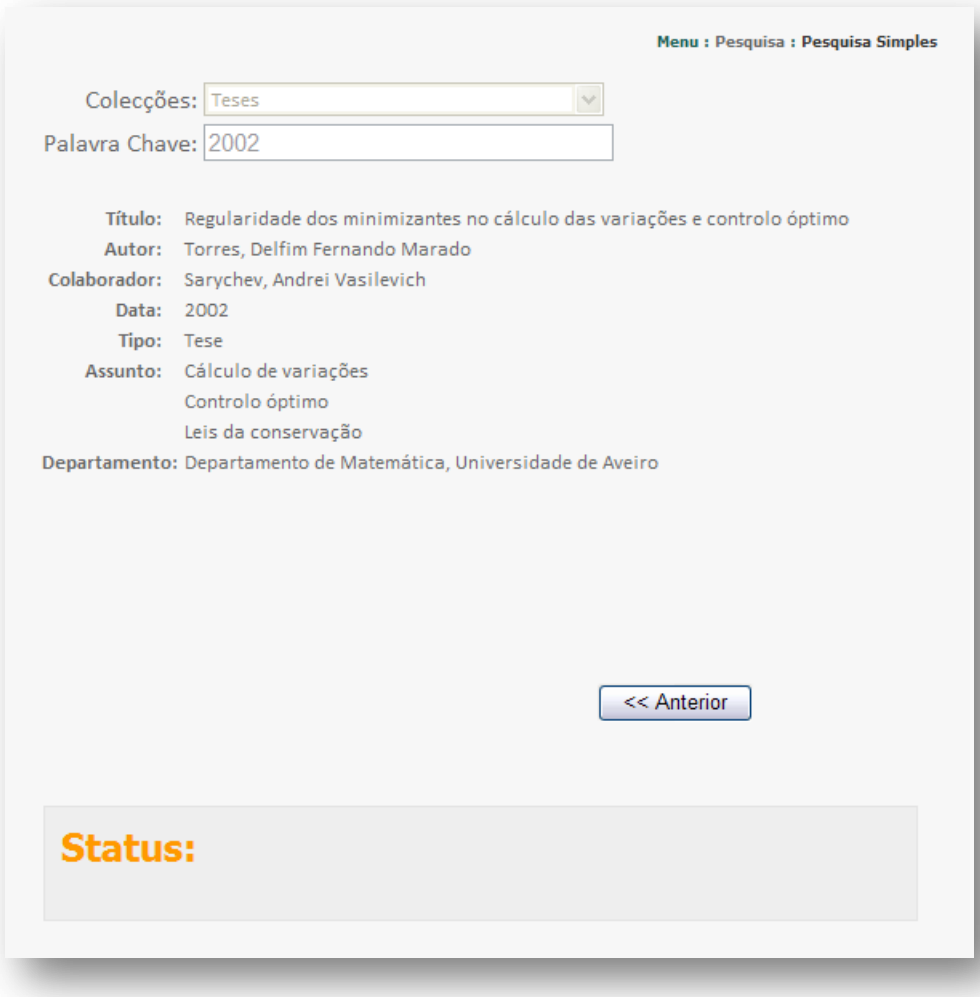

Figura 17 - Interface - Pesquisa Simples v2

4.4.10. Pesquisa avançada

O outro tipo de pesquisa, requer que o utilizador preencha os campos pelos quais pretende efectuar a pesquisa dos dados da colecção que escolher da lista, Figura 18. A sequência de visualização do resultado da pesquisa é semelhante ao da pesquisa simples.

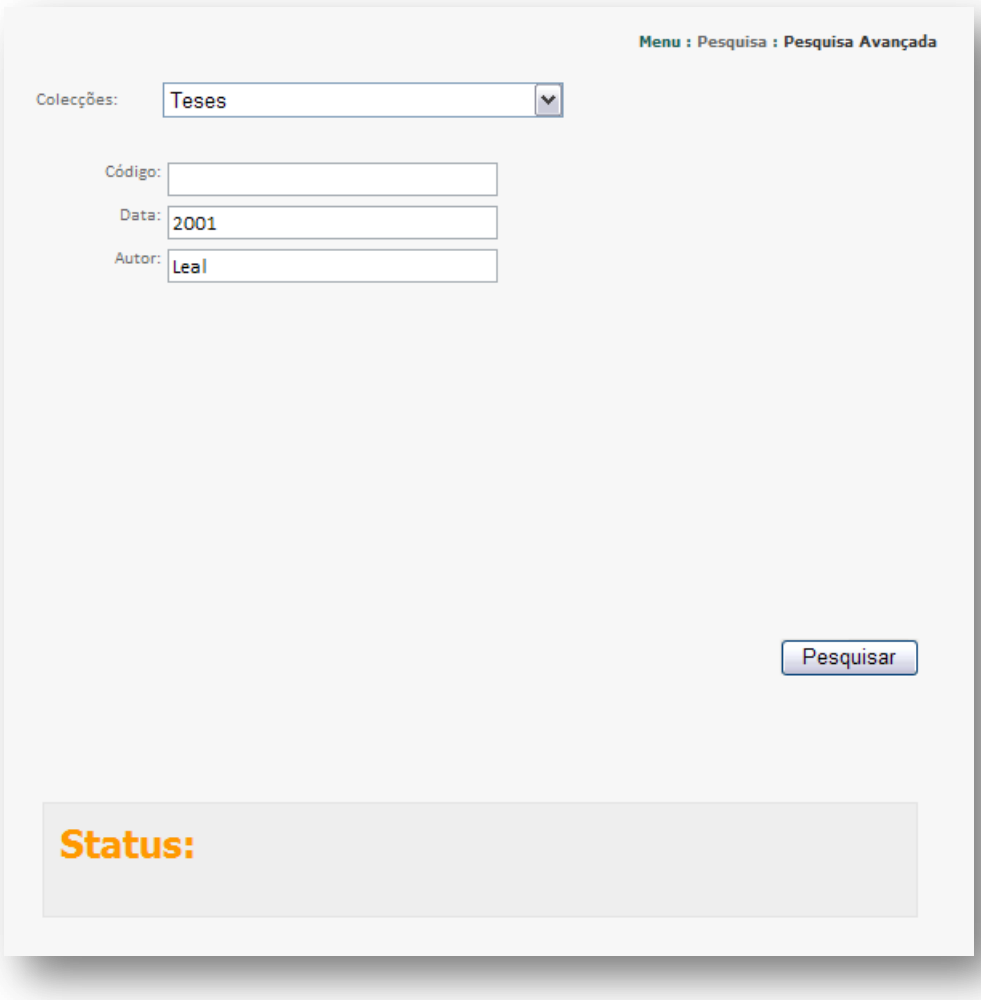

Figura 18 - Interface - Pesquisa Avançada

4.4.11. Consultar manualmente a BD

Nesta opção da interface, o utilizador pode construir a query na caixa de texto superior, podendo ainda seleccionar um exemplo da lista. Tal como se pode observar na Figura 19, clicando no botão **Correr Query**, a query escrita pelo utilizador será executada na aplicação, e o resultado dessa execução aparecerá na caixa de texto inferior.

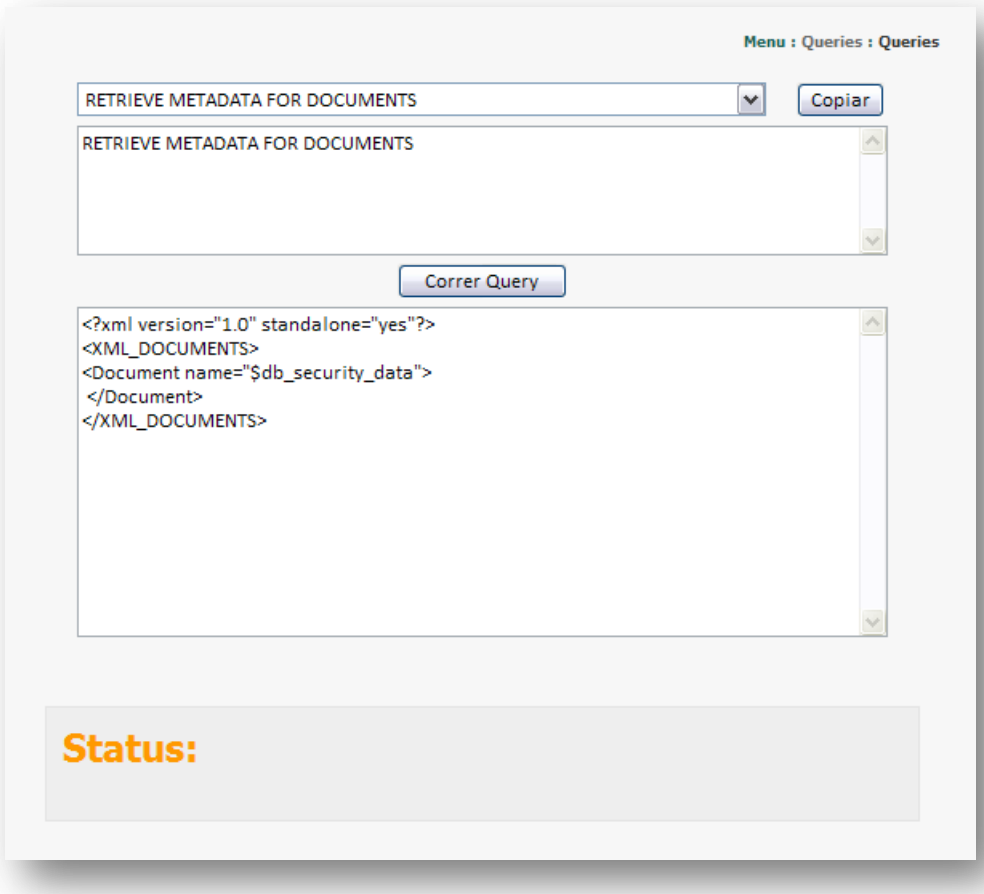

Figura 19 - Interface – Querying

5. AVALIAÇÃO E RESULTADOS

Para testar a eficácia da pesquisa implementada na aplicação, procederam-se a testes que consistiram em contabilizar o tempo que cada pesquisa demorava. Estes testes foram efectuados usando um Intel Pentium M, 1.59GHz, 1.25GB RAM, Windows XP Professional, Sedna 3.1.10 e Lucene 2.0.

O interesse desta análise permite verificar se a utilização do Lucene como biblioteca de indexação melhora a rapidez da pesquisa.

Testaram-se as pesquisas:

- Sem indexação;
- Indexada com o Sedna;
- Indexadas com o Lucene:
    - simples (apenas por uma palavra chave)
    - pesquisa avançada (por campos)

Os testes foram efectuados a 4.742 documentos incluidos numa colecção, "Cartazes", onde estavam localizados os documentos do tipo Cartazes. Realizaram-se 15 testes a cada tipo de pesquisa efectuada ao campo *"date"*, os resultados seriam todos os documentos que tivessem a data "2005".

A pesquisa sem indexação, realizada na interface da aplicação, "*Querying"*, necessitava da construção de uma query, ficando:

```
declare namespace dc="http://purl.org/dc/elements/1.1/";
for $x in collection('Cartazes') where $x//dc:date/text() = '2005' return document-uri($x)
```

Podemos verificar o resultado médio (em milisegundos) do tempo de execução da pesquisa na Tabela 2.

Tabela 2 - Resultados de pesquisa não indexada

	Pesquisa XQuery (ms)
Média:	597,526

Também a pesquisa indexada com o Sedna foi realizada a partir da opção da interface da aplicação, *"Querying".* Criaram-se os índices a partir da query:

```
declare namespace dc="http://purl.org/dc/elements/1.1/";
    CREATE INDEX 'idx_date'
    ON collection('Cartazes') BY .//dc:date
    AS xs:string
```

De seguida foi efectuada a pesquisa nos índices criados. O tempo médio (em milisegundos) da execução desta pesquisa encontra-se na

Tabela 3.

```
    declare namespace dc="http://purl.org/dc/elements/1.1/";
    for $x in index-scan("idx_date", "2005", "EQ")//dc:date return document-uri($x)
```

Tabela 3 - Resultados de pesquisa indexada com o Sedna

	Pesquisa XQuery indexada (ms)
Média:	478,688

Quantos aos resultados das pesquisas indexadas pelo Lucene, podemos observar os tempos médios (em milisegundos) de pesquisa pela Tabela 4

Tabela 4 - Resultados de pesquisas indexadas com o Lucene

	Pesquisa Simples (ms)	Pesquisa Avançada (ms)
Média:	10,014	6,009

Comparando os resultados das tabelas anteriores, podemos observar que os tipos de pesquisa que utilizam o Lucene são bastante mais rápidas por os seus tempos médios serem bastante mais baixos, o que prova que o Lucene é uma ferramenta eficaz e que melhora a performance da aplicação.

6. CONCLUSÕES E TRABALHO FUTURO

Ao terminar este trabalho, pode-se afirmar que os objectivos propostos foram concretizados com sucesso e que constituiu um importante momento de aprendizagem por permitir aprender novas tecnologias e ferramentas de trabalho.

O primeiro objectivo deste projecto era estudar os diferentes tipos de bases de dados disponíveis e compará-las com as base de dados XML nativas (que se tornou um dos objectos de estudo deste trabalho).

Como BD XML nativa utilizou-se o Sedna, que mostrou ser uma boa escolha de base de dados para a aplicação desenvolvida apesar das suas limitações como foi mencionado no capítulo APLICAÇÃO DESENVOLVIDA. Para manipular a BD, foi utilizada a API .Net do Sedna, que mesmo com a falta de suporte para gerir as colecções, foi uma mais valia para a implementação da aplicação, dado que simplificou bastante a comunicação com o servidor do Sedna. A maior dificuldade sentida ao trabalhar com o Sedna, bem como com a API .Net, foi a falta de documentação de apoio. O material de apoio disponível era muito geral e foi aqui que a *mailing list* do grupo Sedna se tornou numa boa fonte de informação.

Também foi necessário tomar conhecimento de como indexar os documentos XML utilizando a biblioteca Lucene que provou ser uma ferramenta fundamental para aumentar a velocidade e eficácia na pesquisa de informação. A informação disponível *online* não era muita mas era o suficientemente eficaz para realizar o trabalho pretendido.

A utilização do Visual Studio como ambiente de desenvolvimento e o uso das funções disponíveis pela .Net *framework* facilitou muito a aprendizagem das linguagens usadas, C# e ASP.NET, assim como o desenvolvimento da aplicação *Web*.

A nível pessoal, foi um trabalho gratificante e que deu imenso gosto de realizar devido ao bom ambiente e ajuda. Possibilitou trabalhar num ambiente diferente, desenvolver actividades que nunca tinha executado e concluir que estamos sempre a aprender, a melhorar e que um trabalho é uma mudança constante, pois a aplicação foi sempre sofrendo algumas evoluções e sendo refinada.

Apesar dos problemas inusitados, que nem sempre se conseguem controlar, o resultado final parece ser positivo e aplicável na vida real. No entanto poderia ter algumas melhorias relativamente ao tratamento de erros, oferta de mais *feedback* ao utilizador e implementação de mais algumas funcionalidades.

Esta nova metodologia poderá ainda constituir uma importante inovação nos sistemas de informação das bibliotecas digitais, constituindo provavelmente o inicio de uma segunda geração de bibliotecas digitais baseadas em repositórios de ficheiros XML.

6.1. Trabalho Futuro

Apesar dos objectivos principais do projecto terem sido atingidos, como trabalho futuro propõe-se que o tratamento de erros e excepções seja realizada de uma forma mais eficaz para prevenir que o servidor do Sedna vá abaixo ou que a aplicação falhe.

Algumas funcionalidades adicionais poderiam ainda ser implementadas, tais como:

- A possibilidade de mudar um documento de colecção, por exemplo, passar o documento X da colecção1 para a colecção2 sem que o utilizador tenha que remover o documento X da colecção1 e depois ter que o importar novamente do seu computador;
- A pesquisa ser realizada em todo o repositório e não apenas aos dados de uma colecção;

7. BIBLIOGRAFIA

[1] A. Fomichev, M. Grinev, S. Kuznetsov, "*Sedna: a native XML database*", In Proceedings of SOFSEM 2006: 32nd Conference on Theory and Practice of Computer Science (Merin, Czech Republic, January 2006), 272-281, 2006.

[2] A. Silberschatz, H. Korth, S. Sudarshan, "*Database Systems Concepts*", 5th Edition, McGraw-Hill, 2005;

[3] B. Trippe, "*XML hits the big time: major database player get into XML*", 2002;

[4] C. J. Date, "*An Introduction to Database Systems*", 8th Edition, Addison-Wesley, 2004;

[5] D. Carr, "*XML-Native Databases*", 2001;

[6] D. Esposito, "*Applied XML Programming for Microsoft .NET*", Microsoft Press, 1st Edition, 2002;

[7] F. Grime, "*Microsoft .NET for Programmers*", Manning Publications; 1st Edition, 2002;

[8] F. Sousa, H. Carneiro Filho, P. Andrade, J. Machado, "RepliX: Um Mecanismo para a Replicação de Dados XML", In: XXII Simpósio Brasileiro de Banco de Dados, Brasil, 2007

[9] G. Mendes, N. Silva, P. Henriques, "Utilizando uma base de dados XML nativa aplicada ao tratamento de erros num Sistema de logs", 2005;

[10] G. Pellegrino, H. Leite, H. Junior, R. Tenório, "XML e Banco de Dados para Web: Panorama, Estado atual e Perspectivas", Instituto de Matemática – Universidade Federal da Bahia, Brasil, 2007

[11] GREaT – Grupo de Redes de Computadores, Engenharia de Softwares e Sistemas, "*SednaAdmin*", Acedido em Março 2008 em: http://www.great.ufc.br/~sednaadmin

[12] Institute for System Programming of Russia Academy of Sciences, "*Sedna XML database*", Acedido em Outubro de 2008 em: http://modis.ispras.ru/sedna

[13] J. Ramalho, "Linguagens de Anotação", Grupo de Especificação e Processamento de Linguagens, Universidade do Minho, 2001

[14] K. Staken, "Introduction to Native XML Database", 2001; http://www.xml.com/pub/a/2001/10/31/nativexmldb.html

[15] L. Dykes, E. Tittel, "XML For Dummies", For Dummies, 4th Edition, 2005;

[16] Linha de Código. *Tamino - Servidor XML Nativo*. Acedido em Outubro de 2008, em: http://www.linhadecodigo.com.br/artigos.asp?id_ac=55

[17] M. MacDonald, J. Templeman, "Beginning ASP.NET 2.0 in C# 2005: From Novice to Professional", Apress, 2006;

[18] MODIS ISPRAS (2008). *Sedna XML Database – Sedna Administration Guide*. Acedido em Outubro de 2008, em http://modis.ispras.ru/sedna/adminguide/AdminGuide.html

[19] MODIS ISPRAS (2008). *Sedna XML Database – Sedna Client/Server Protocol*. Acedido em Outubro de 2008, em http://modis.ispras.ru/sedna/serverprotocol/ClientServerProtocol.html

[20] MODIS ISPRAS (2008). *Sedna XML Database* – Sedna Programmer's Guide. Acedido em Outubro de 2008, em http://modis.ispras.ru/sedna/progguide/ProgGuide.html

[21] MODIS ISPRAS (2008). *Sedna XML Database – Sedna Quick Start*. Acedido em Outubro de 2008, em http://modis.ispras.ru/sedna/quick-start.html

[22] N. Cuong, M. Valenta, "*XML Native Database Systems: Review of Sedna, Ozone, NeoCoreXMS*", Czech Technical University in Prague, 2006

[23] O. Gospodnetic, E. Hatcher, "Lucene in Action: A guide to the Java search engine", Manning Publications, 2005;

[24] P. Almeida, M. Fernandes, A. Alho, J. A. Martins, J. S. Pinto, "SInBAD - A digital library to aggregate multimedia documents", Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, 173-179, 2006;

[25] P. Davies, "Database Systems", 3rd Edition, Palgrave Macmillan, 2003;

[26] P. Rob, C. Coronel, "Database Systems: Design, Implementation and Management", 6th Edition, Course Technology, 2004;

[27] R. Bourret, "Going Native: Making the Case for XML Databases", 2005; http://www.xml.com/pub/a/2005/03/30/native.html

[28] R. Bourret, "Native XML Databases". Acedido em Outubro de 2008, em: http://www.rpbourret.com/

[29] R. Bourret, "*XML and databases*". Acedido em Outubro de 2008, em: http://www.rpbourret.com/

[30] S. Silva, M. Fernandes, P. Almeida, J. A. Martins, J. S. Pinto, "*SWIT : An open-source web-based database management system with indexing engine integration*", EATIS 2008 : Euro American Conference on Telematics and information Systems, Aracaju, BR, 2008

[31] S. Davis, C. Sphar, "*C# 2005 For Dummies*", For Dummies, 2005;

[32] S. Pal, V. Parikh, V. Zolotov, L. Giakoumakis, M. Rys, "*XML Best Practices for Microsoft SQL Server 2005*", Microsoft Corporation, 2004, Revised 2007 http://msdn2.microsoft.com/en-us/library/ms345115.aspx

[33] SourceForge - Open Source Software. *Exist - Open Source Native XML Database*. Acedido em Outubro de 2008, em: http://exist.sourceforge.net/

[34] The Apache Software Foundation (2008). *Apache Lucene project*. Acedido em Outubro de 2008, em: http://lucene.apache.org

[35] The Apache Software Foundation (2008). *Apache Lucene.net project*. Acedido em Outubro de 2008, em: http://incubator.apache.org/lucene.net

[36] The Apache Software Foundation (2008). *Apache Xindice Project*. Acedido em Outubro de 2008, em: http://xml.apache.org/xindice/

[37] V. Barcaroli, R. Mello, "Análise Comparativa de Linguagens de Consulta para Banco de Dados XML Nativos.", Universidade Federal de Santa Catarina**,** Brasil, 2005

[38] W3Schools Online Web Tutorials. *XML Tutorial*. Acedido em Outubro de 2008, em: http://www.w3schools.com/

Anexo A

Produto	Colaborador	Licença	Últ. Actualização	Url
Berkeley DB XML	Oracle	Open Source	Dezembro, 2005	http://www.oracle.com/database/berkeley-db/xml/index.html
Birdstep RDM XML	Birdstep	Comercial	Outubro, 2002	http://www.birdstep.com/database_technology/rdm_xml.php3
Centor Interaction Server	Centor Software Corp.	Comercial	Setembro, 2001	http://www.centor.com/solutions/technology.shtml
Dieselpoint	Dieselpoint, Inc.	Comercial	Janeiro, 2007	http://www.dieselpoint.com/xmlsearch.html
DOMSafeXML	Ellipsis	Comercial	Junho, 2004	http://www.ellipsis.nl/content/products.htm
eXist	Wolfgang Meier	Open Source	Março, 2004	http://exist.sourceforge.net
eXtc	M/Gateway Dev. Ltd.	Comercial	Março, 2004	http://www.mgateway.tzo.com/php/mgw/extc.php
Extraway	3D Informatica	Comercial	Agosto, 2005	http://www.3di.it/h3/h3/aSito_3DI/finizio
GoXML DB	XML Global	Comercial	Maio, 2003	http://www.xmlglobal.com/prod/db/index.jsp
Infonyte DB	Infonyte	Comercial	Fevereiro, 2002	http://www.infonyte.com/prod_db.html
Ipedo	Ipedo	Comercial	Agosto, 2003	http://www.ipedo.com/html/products.html
Lore	Stanford University	Research	Novembro, 2000	http://www-db.stanford.edu/lore/home/index.html
MarkLogic Server	Mark Logic Corp.	Comercial	Junho, 2004	http://www.marklogic.com/products/index.html
myXMLDB	Mladen Adamovic	Open Source	Janeiro, 2005	http://sourceforge.net/projects/myxmldb/
Natix	data ex machina	Comercial	Junho, 2004	http://www.dataexmachina.de/natix.html
NaX Base	Naxoft	Comercial	Maio, 2004	http://www.naxoft.com/produit-presentation.html
Neocore XMS	Xpriori	Comercial	Verão, 2001	http://www.xpriori.com/index.html
ozone	ozone-db.org	Open Source	Março, 2004	http://ozone-db.org/frames/home/what.html
Sedna XML DBMS	MODIS, ISPRAS	Free	Junho, 2004	http://modis.ispras.ru/sedna/index.htm
Sekaiju	Media Fusion	Comercial	Fevereiro, 2002	http://www.mediafusion.co.jp/usa/seihin/sekaiju/index.html
SQL/XML-IMDB	QuiLogic	Comercial	Fevereiro, 2003	http://www.quilogic.cc/
Sonic XML Server	Sonic Software	Comercial	Abril, 2003	http://www.sonicsoftware.com/products/sonic_xml_server/index.ssp
Tamino	Software AG	Comercial	Novembro, 2002	http://www.softwareag.com/Corporate/products/tamino/default.asp

TeraText DBS	TeraText Solutions	Comercial	Agosto, 2002	http://www.saic.com/products/software/teratext/
TEXTML Server	IXIASOFT, Inc.	Comercial	Junho, 2005	http://www.ixiasoft.com/default.asp?xml=/xmldocs/webpages/textml-server.xml
TigerLogic XDMS	Raining Data	Comercial	Janeiro, 2003	http://www.rainingdata.com/products/tl/abouttl.html
Timber	University of Michigan	Open Source	Outubro, 2005	http://www.eecs.umich.edu/db/timber
TOTAL XML	Cincom	Comercial	Julho, 2003	http://tiger.cincom.com/pages/aboutTotalXML.html
Virtuoso	OpenLink Software	Comercial	Novembro, 2000	http://www.openlinksw.com/virtuoso/
XDBM	M. Parry, P. Sokolovsky	Open Source	Novembro, 2000	http://sourceforge.net/projects/xdbm/
XDB	ZVON.org	Open Source	Novembro, 2001	http://zvon.org/index.php?nav_id=61
XediX TeraSolution	AM2 Systems	Comercial	Junho, 2005	http://www.am2systems.com/technologies-EN.html
X-Hive/DB	X-Hive Corporation	Comercial	Maio, 2005	http://www.x-hive.com/products/db/index.html
Xíndice	Apache Sw Foundation	Open Source	Junho, 2005	http://xml.apache.org/xíndice/
XML Transactional DOM	Ontonet	Comercial	Agosto, 2003	http://ontonet.com/XML_Product.html
XpSQL	Makoto Yui	Open Source	Março, 2004	http://gborg.postgresql.org/project/xpsql/projdisplay.php
XQuantum XMLDS	Cognetic Systems	Comercial	Junho, 2006	http://www.cogneticsystems.com/server.html
XStreamDB Native XML DB	Bluestream DB Sw Corp.	Comercial	Maio, 2003	http://www.bluestream.com/products/xstreamdb32
Xyleme Zone Server	Xyleme SA	Comercial	Julho, 2002	http://www.xyleme.com/page/xml_storage/

Anexo B

Produto	Colaborador	Licença	Últ. Actualização	Url
Access 2002	Microsoft	Comercial	Fevereiro, 2002	http://msdn.microsoft.com/library/en-us/dnacc2k2/html/odc_acxmllnk.asp
Cache	InterSystems Corp.	Comercial	Março, 2004	http://www.intersystems.com/cache/technology/components/xml/index.html
DB2	IBM	Comercial	Outubro, 2005	http://www-306.ibm.com/software/data/db2/udb/viper/
eXtremeDB	McObject LLC	Comercial	Junho, 2005	http://www.mcobject.com/standardedition.shtml
FileMaker	FileMaker	Comercial	Fevereiro, 2002	http://www.filemaker.com/xml/overview.html
FoxPro	Microsoft	Comercial	Abril, 2002	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fox7help/html/lngrfxmlfunctions.asp
Informix	Informix	Comercial	Novembro, 2000	http://www.informix.com/idn-secure/webtools/ot/
Matisse	Matisse Software	Comercial	Agosto, 2003	http://www.matisse.com/product_information/
Objectivity/DB	Objectivity	Comercial	Fevereiro, 2002	http://www.objectivity.com/DevCentral/Products/TechDocs/Datasheets/ObjyXML.html
OpenInsight	Revelation Software	Comercial	Janeiro, 2003	http://www.revelation.com/website/vipweb.nsf/e4a3262250066019852563190009114 9/178609577b262b2a852563f500770fc5?OpenDocument
Oracle	Oracle	Comercial	Novembro, 2005	http://www.oracle.com/technology/tech/xml/xmldb/index.html
Access 2002	Microsoft	Comercial	Fevereiro, 2002	http://msdn.microsoft.com/library/en-us/dnacc2k2/html/odc_acxmllnk.asp
Cache	InterSystems Corp.	Comercial	Março, 2004	http://www.intersystems.com/cache/technology/components/xml/index.html
DB2	IBM	Comercial	Outubro, 2005	http://www-306.ibm.com/software/data/db2/udb/viper/
eXtremeDB	McObject LLC	Comercial	Junho, 2005	http://www.mcobject.com/standardedition.shtml
FileMaker	FileMaker	Comercial	Fevereiro, 2002	http://www.filemaker.com/xml/overview.html
FoxPro	Microsoft	Comercial	Abril, 2002	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fox7help/html/lngrfxmlfunctions.asp
Informix	Informix	Comercial	Novembro, 2000	http://www.informix.com/idn-secure/webtools/ot/
Matisse	Matisse Software	Comercial	Agosto, 2003	http://www.matisse.com/product_information/
Objectivity/DB	Objectivity	Comercial	Fevereiro, 2002	http://www.objectivity.com/DevCentral/Products/TechDocs/Datasheets/ObjyXML.html
OpenInsight	Revelation Software	Comercial	Janeiro, 2003	http://www.revelation.com/website/vipweb.nsf/e4a3262250066019852563190009114 9/178609577b262b2a852563f500770fc5?OpenDocument
Oracle	Oracle	Comercial	Novembro, 2005	http://www.oracle.com/technology/tech/xml/xmldb/index.html

PostgreSQL	PostgreSQL Global Dev Group	Open Source	Dezembro, 2006	http://Colaborador.postgresql.org/cvsweb.cgi/pgsql/contrib/xml2/README.xml2?rev=1.7
SQL Server	Microsoft	Comercial	Novembro, 2000	http://www.microsoft.com/sql/default.mspx
Sybase ASE 12.5	Sybase	Comercial	Julho, 2002	http://my.sybase.com/content/1013051/3929XMLwpaperv9.pdf
UniData	IBM	Comercial	Abril, 2004	http://www-306.ibm.com/software/data/u2/unidata/
UniVerse	IBM	Comercial	Abril, 2004	http://www-306.ibm.com/software/data/u2/universe/
Versant enJin	Versant Corp.	Comercial	Novembro, 2000	http://www.versant.com/products/enjin/datasheet.html
View500	eB2Bcom	Comercial	Janeiro, 2006	http://www.view500.com/view500.php

AnexoC

Preparar base de dados

Antes de correr a aplicação é necessário instalar o Sedna. Para tal basta descompactar o arquivo para uma directoria qualquer, por exemplo, “c:\sedna”.

Os comandos de configuração do Sedna [21] terão que ser introduzidos pela linha de comandos na directoria onde o sedna foi instalado, seguindo o exemplo anterior, “c:\sedna\bin”.

Para a aplicação funcionar correctamente é necessário iniciar o Sedna Server, e que exista uma base de dados (utilizar-se-á o nome da base de dados “tempdb” que é a que está a ser usada por defeito) e iniciar a base de dados, tal como:

- se_gov – iniciar o Sedna Server;
- se_cdb tempdb – criar a a base de dados;
- se_sm tempdb – iniciar a base de dados;

Se a base de dados *tempdb* foi anteriormente criada basta:

- se_gov – iniciar o Sedna Server;
- se_sm tempdb – iniciar a base de dados;

Se pretendermos terminar a base de dados e o servidor:

- se_smsd tempdb – parar a base de dados;
- se_stop – parar o Sedna Server;

De seguida seguem alguns comandos do Sedna Server, onde se pode encontrar também as instruções que podem ser executadas, bem como as mensagens que devem aparecer em casos de sucesso.

Iniciar o Sedna:

- Correr o comando: ***se_gov***
- Se o Sedna for iniciado com sucesso irá aparecer uma mensagem:

“GOVERNOR has been started in the background mode”

Criar uma nova base de dados:

- Correr o comando: ***se_cdb tempdb***
- Se a base de dados for iniciada com sucesso irá aparecer uma mensagem:

"The database 'tempdb' has been created successfully"

Iniciar a base de dados:

- Correr o comando ***se_sm tempdb***
- Se a base de dados for iniciada com sucesso aparecerá uma mensagem:

"SM has been started in the background mode"

Parar a base de dados:

- Correr o comando ***se_smsd tempdb***
- Se a base de dados for parada com sucesso aparecerá uma mensagem:

"The database 'tempdb' has been shut down successfully"

Apagar a base de dados:

- Correr o comando ***se_ddb tempdb***
- Se a base de dados for removida com sucesso aparecerá uma mensagem:

"The database 'tempdb' has been dropped"

Parar o Sedna

- Correr o comando **se_stop**
- Se o sedna for finalizado com sucesso aparecerá uma mensagem:

"Sedna Server has been shut down successfully"