UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

CENTRO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENG. ELÉTRICA E DE COMPUTAÇÃO

FÁBIO AUGUSTO PROCÓPIO DE PAIVA

ESPECIFICAÇÃO E IMPLEMENTAÇÃO DE UM PROTÓTIPO DE SERVIÇO *WEB* PARA BUSCAS BASEADAS EM CONTEXTOS COMPARTILHADOS DEFINIDOS A PARTIR DE SINTAGMAS E RELACIONAMENTOS

Natal – RN
Agosto de 2007

FÁBIO AUGUSTO PROCÓPIO DE PAIVA

Especificação e implementação de um protótipo de serviço *web* para buscas baseadas em contextos compartilhados definidos a partir de sintagmas e relacionamentos

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica e de Computação da Universidade Federal do Rio Grande do Norte como exigência parcial para obtenção do título de Mestre em Ciências.

Área de concentração: Telemática

Orientador: Prof. Dr. Cláudio R. Muniz da Silva

Natal – RN
Agosto de 2007

FÁBIO AUGUSTO PROCÓPIO DE PAIVA

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica e de Computação do Centro de Tecnologia da Universidade Federal do Rio Grande do Norte, como requisito parcial para obtenção do grau de Mestre em Ciências.

Aprovada em _____/_____/2007.

BANCA EXAMINADORA

_____

Prof. Dr. Cláudio Rodrigues Muniz da Silva (Orientador)
Departamento de Engenharia Elétrica da UFRN

_____

Prof. Dr. Oscar Gabriel Filho (Examinador Externo)
Departamento de Engenharia de Computação da UnP

_____

Prof. Dr. José Alfredo Ferreira Costa (Examinador Interno)
Departamento de Engenharia Elétrica da UFRN

_____

Prof. Dr. Laércio Martins de Mendonça (Examinador Interno)
Departamento de Engenharia Elétrica da UFRN

DEDICATÓRIA

Às mulheres da minha vida:
minha mãe, minha avó e Raissa.

*Ao meu avô **Zé** (in memoriam).*

Tudo é do Pai
Toda honra e toda glória
É dele a vitória
Alcançada em minha vida
Tudo é do Pai
Se sou fraco ou pecador
Bem mais forte é o Meu Senhor
Que me cura por amor

Frederico Cruz

AGRADECIMENTOS

No decorrer deste trabalho, tive a certeza de que muitas pessoas são, realmente, muito importantes para mim. Direta ou indiretamente, elas participaram desta dissertação. Apesar de não ser possível citar o nome de todas, gostaria de realçar algumas que merecem um destaque especial.

Prof. Dr. Cláudio Muniz, *que foi muito mais que um professor-orientador: um companheiro, um conselheiro e um amigo.*

Prof. Dr. José Alfredo*,* ***Prof. Dr. Laércio Martins*** *e* ***Prof. Dr. Sandro Gonçalves*** *que participaram do exame de qualificação desse estudo e contribuíram por meio de críticas e de sugestões.*

Prof. Dr. Oscar Gabriel Filho *por ter me indicado ao Programa de Pós-graduação em Engenharia Elétrica e de Computação da UFRN e pela participação na banca examinadora.*

Minha mãe *por seu amor incondicional e por ser meu parâmetro de personalidade.*

Raissa*, meu amor, por todos os anos de companhia, de dedicação e de carinho.*

Minha avó *pelo cuidado, pela maternidade e pelo incentivo.*

Meu pai*, que, mesmo distante, sempre esteve presente me estimulando.*

Meus colegas de trabalho*, que sempre estiveram dispostos a me substituir nos momentos em que me ausentei e por se disporem a me ajudar em tantas outras coisas.*

Neste momento, em minha mente, também estão todos os ***meus amigos e familiares****, que me incentivaram a alcançar esta etapa. Contudo, é impossível citar tantos nomes, então só me resta dizer que sou muito grato a todos eles.*

E, por fim, ***Deus****, indubitavelmente, o maior responsável por esse trabalho.*

RESUMO

A popularização da *Internet* estimulou o surgimento das Máquinas de Busca, que têm como objetivo auxiliar os usuários no processo de pesquisa da informação na *Web*. Porém, é comum usuários formularem consultas e receberem resultados que não satisfazem a sua intenção inicial de pesquisa. A técnica *Information Retrieval in Context* (IRiX) permite que informações relacionadas a um determinado tema possam ser atribuídas à consulta formulada originalmente pelo usuário, possibilitando, dessa forma, melhores resultados. Este trabalho apresenta um protótipo de Máquina de Busca baseada em contextos construídos a partir de um conjunto de sintagmas e de relacionamentos definido pelo usuário. As informações de contextos podem ser compartilhadas com *softwares* e com outros usuários da ferramenta a fim de promover a socialização de contextos.

Palavras-chave: *Máquina de Busca, Contexto, Sintagma, Relacionamento, Web Service***.**

ABSTRACT

The popularization of the Internet has stimulated the appearance of Search Engines that have as their objective aid the users in the Web information research process. However, it's common for users to make queries and receive results which do not satisfy their initial needs. The Information Retrieval in Context (IRiX) technique allows for the information related to a specific theme to be related to the initial user query, enabling, in this way, better results. This study presents a prototype of a search engine based on contexts built from linguistic gatherings and on relationships defined by the user. The context information can be shared with softwares and other tool users with the objective of promoting a socialization of contexts.

Keywords: *Search Engine, Context, Linguistic gatherings, Relationship, Web Service.*

LISTA DE FIGURAS

2.1 Representação genérica de uma Máquina de Busca 26
2.2 Arquitetura de um *Web Service* 32
3.1 Cenários de uso 36
4.1 Diagrama de Casos de Uso 43
4.2 Diagrama de Classes 52
4.3 Diagrama de seqüência representando o caso de uso *Criar conta* 54
4.4 Diagrama de seqüência representando o caso de uso *Efetuar login* 54
4.5 Diagrama de seqüência representando o caso de uso *Solicitar nova senha* 54
4.6 Diagrama de seqüência representando o caso de uso *Alterar senha* 55
4.7 Diagrama de seqüência representando o caso de uso *Atualizar cadastro* 55
4.8 Diagrama de seqüência representando o caso de uso *Configurar perfil* 55
4.9 Diagrama de seqüência representando o caso de uso *Configurar parâmetros* 56
4.10 Diagrama de seqüência representando o caso de uso *Cadastrar fontes de recursos* 56
4.11 Diagrama de seqüência representando o caso de uso *Atualizar administrador* 56
4.12 Diagrama de seqüência representando o caso de uso *Alterar nível de acesso* 56
4.13 Diagrama de seqüência representando o caso de uso *Realizar pesquisa* 57
4.14 Diagrama de seqüência representando o caso de uso *Integrar contextos* 57
4.15 Relação entre o protótipo e os elementos utilizados na implementação 58
4.16 Esquema de indexação de recursos 63
4.17 Esquema de pesquisa de recursos 65
4.18 Fluxograma da rotina de pesquisa 67
4.19 Página inicial do protótipo 68
4.20 Página de pesquisa com a utilização de contextos 69
4.21 Configuração do perfil de pesquisa 70
4.22 Integração de contextos 71
4.23 Aplicação cliente VB requisitando o serviço *web* listagem de contextos 74
4.24 Aplicação cliente *Java* requisitando o serviço *web* Pesquisa 74
4.25 Esquema das visões e dos seus cenários 76

LISTA DE SIGLAS, ABREVIATURAS E ACRÔNIMOS

API	Application Programming Interface
CDD	Classificação Decimal de Dewey
CDU	Classificação Decimal Universal
CERN	Conseil Européen pour la Recherche Nucléaire
FAB	Força Aérea Brasileira
HTML	HyperText Markup Language
HTTP	Hipertext Transfer Protocol
IRiX	Information Retrieval in Context
JSP	Java Server Pages
MOF	Meta-Object Facility
NCSA	National Center Supercomputer Applications
SGML	Standard Generalized Markup Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
UFRN	Universidade Federal do Rio Grande do Norte
UML	Unified Modeling Language
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Services Descriptions Language
WWW	World Wide Web
XML	eXtensible Markup Language

LISTA DE TABELAS

4.1 Documentação do caso de uso *Criar conta* 45
4.2 Documentação do caso de uso *Efetuar login* 45
4.3 Documentação do caso de uso *Solicitar nova senha* 46
4.4 Documentação do caso de uso *Alterar senha* 46
4.5 Documentação do caso de uso *Atualizar usuário* 46
4.6 Documentação do caso de uso *Criar contexto* 47
4.7 Documentação dos casos de uso *Configurar perfil*, *Atualizar relacionamento*, *Atualizar sintagma* e *Atualizar fonte* 47
4.8 Documentação do caso de uso *Configurar parâmetros* 48
4.9 Documentação do caso de uso *Cadastrar fontes de recursos* 49
4.10 Documentação do caso de uso *Cadastrar administrador* 49
4.11 Documentação do caso de uso *Alterar nível de acesso* 50
4.12 Documentação do caso de uso *Realizar pesquisa* 50
4.13 Documentação do caso de uso *Integrar contextos* 51
4.14 Documentação do caso de uso *Executar pesquisa* 51
4.15 Documentação do caso de uso *Importar contextos* 51

LISTAGENS

2.1 Documentação de uma estrofe musical utilizando XML 31
4.1 Trecho da rotina de descoberta de recursos na *Web* 59
4.2 Trecho da rotina de indexação de recursos 60
4.3 Trecho do método que retorna resultados encontrados na pesquisa 61
4.4 Trechos que criam e relacionam informações aos contextos 62
4.5 Trecho da rotina que identifica os sintagmas associados ao(s) contexto(s) 65
4.6 Trecho da rotina que formata consultas 66
4.7 Consulta formatada 69
4.8 Resposta do *web service* à aplicação cliente 72
4.9 WSDL que descreve o serviço de listagem de contextos 73

SUMÁRIO

1 INTRODUÇÃO 16
1.1 Motivações 18
1.2 Objetivos 19
1.3 Contribuição 19
1.4 Organização do trabalho 20

2 FUNDAMENTAÇÃO TEÓRICA 22
2.1 Recuperação da Informação 22
2.2 Ferramentas de busca *web* 23
2.2.1 Diretórios por Assunto 23
2.2.2 Máquinas de Busca 25
2.2.2.1 Descoberta 26
2.2.2.2 Indexação 27
2.2.2.3 Pesquisa 27
2.2.2.4 Evolução 28
2.2.3 Meta-máquinas de Busca 28
2.3 Contextos 29
2.4 Sintagmas e Relacionamentos 29
2.5 XML 31
2.6 *Web Service* 32
2.7 Resumo 33

3 ANÁLISE DE REQUISITOS 35
3.1 Definição dos cenários 35
3.2 Definição dos requisitos 36
3.3 Resumo 40

4 ESPECIFICAÇÃO, IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS 42
4.1 Especificação 42
4.1.1 Diagrama de Casos de Uso 42
4.1.1.1 Documentação 45
4.1.2 Diagrama de Classes 52
4.1.3 Diagrama de Seqüência 53
4.2 Implementação 57
4.2.1 Protótipo 57
4.2.2 Rotina de indexação 62
4.2.3 Rotina de pesquisa 64
4.2.4 Operacionalização 68
4.2.5 Biblioteca de indexação e de busca 75
4.2.6 Linguagem de programação 75
4.2.7 Disponibilidade de serviços para usuários 75
4.2.8 Disponibilidade de serviços para aplicações 75

4.2.9 Banco de dados 75
4.2.10 *Container* 76
4.3 Validação dos cenários 76
4.4 Resumo 78

5 CONCLUSÕES 80
5.1 Conclusões gerais 80
5.2 Conclusões específicas 80
5.3 Trabalhos futuros 82

REFERÊNCIAS 84

GLOSSÁRIO 89

ANEXOS 90

“Se você quer ser bem sucedido, precisa ter dedicação total, buscar seu último limite e dar o melhor de si mesmo.”

Ayrton Senna

CAPÍTULO 1
INTRODUÇÃO

1 INTRODUÇÃO

No início dos anos 90, a *Internet* era o foco da atenção apenas de pesquisadores de universidades, governos e indústrias. No entanto, em agosto de 1991, Tim Berners-Lee, físico do *Conseil Européen pour la Recherche Nucléaire* (CERN), desenvolveu uma aplicação chamada *World Wide Web* (WWW), que, mais tarde, juntamente com o navegador *Mosaic*, da *National Center Supercomputer Applications* (NCSA), revolucionou a *Internet* e deu início a geração 1.0 da *Web*, atraindo milhares de novos usuários [1-2] por meio dos serviços de texto, imagens, vídeo, som e hipertexto.

Em outubro de 2004, foi realizada uma conferência entre a editora *O'Reilly Media* e a promotora de eventos *MediaLive International*, na qual foram iniciadas as discussões sobre a nova geração da *Web* [3-5]. Naquele encontro, Tim O'Reilly, presidente da empresa que tem o seu nome, divulgou o termo *Web* 2.0.

A *Web* 2.0 consiste na utilização de serviços que são mantidos através da colaboração, do compartilhamento e da participação de diversos usuários. Exemplos disso são os *sites* de rede sociais, *wikis*, ferramentas de comunicação e *folksnomia*. As *wikis* e os *blogs* são os principais representantes dessa geração [6].

São características da segunda geração o fato de o usuário poder participar da criação e da organização das informações; de as versões serem perpetuamente beta (o que facilita a atualização e a correção de *bugs*); e de a *Internet* ser vista como uma plataforma [7-9] em que é dado suporte à execução dos programas (*Web Operating System*) [10].

Em dezembro de 2006, O'Reilly disse que

> *Web* 2.0 é a mudança para uma *Internet* como plataforma e um entendimento das regras para obter sucesso nesta nova plataforma. Entre outras, a regra mais importante é desenvolver aplicativos que aproveitem os efeitos de rede para se tornarem melhores quanto mais são usados pelas pessoas [11-12].

Ainda não existe uma definição formalizada para *Web* 2.0, talvez por essa razão haja tantos admiradores [13-15] e críticos [16-18]. Estes dizem que as tecnologias usadas por essa geração já existiam antes mesmo da sua divulgação e ainda que a única diferença entre ela e a primeira é que a quantidade de usuários que acessam a rede foi aumentada.

Em maio de 2001, anos antes de a *Web* 2.0 ter sido divulgada, Berners-Lee, James Hendler e Ora Lassila publicaram um artigo na revista *Scientific American* com o título de "*The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*". Nesse artigo, por meio de um exemplo, eles explicaram a forma como a *Web* Semântica é capaz de operar [19].

Derivada da WWW, a *Web* Semântica associa semântica à informação, permitindo que o conteúdo *web* seja interpretado e compreendido não apenas por seres humanos, mas também por programas de computador. Assim, a *Web* Semântica é apresentada como uma alternativa para que as máquinas realizem o processamento da informação [20-21].

Essa tecnologia objetiva inserir semântica às informações de modo que elas sejam compreendidas pelas pessoas e pelos *softwares*. Pretende criar estruturas e oferecer significado semântico ao conteúdo das páginas *web*, possibilitando que as máquinas obtenham melhores resultados de busca e troquem informações entre si [22].

A *Web* Semântica instigou o surgimento do termo *Web* 3.0, usado pela primeira vez pelo jornalista John Markoff em um artigo publicado no *New York Times*. A terceira geração da *Web* visa a transformar a *Web* em uma grande base de dados [23]. Nesse estágio, resultados de pesquisas são mais relevantes porque a informação está associada à semântica e porque o *site* conhece as necessidades de seus usuários. Assim, é possível que a tarefa de localizar documentos na rede deixe de ser enfadonha [24].

Em entrevista ao jornal Folha de São Paulo, em fevereiro de 2007, o empresário Spivack, incentivador da *Web* 3.0 e o maior especialista em semântica da rede [25], disse que "Na *Web* 3.0, o foco é mais no bastidor que no usuário. [...] Vai se parecer bastante com a *Web* 2.0, mas será mais inteligente.". Complementa afirmando que "[...] estamos passando da *World Wide Web* (rede mundial) para *World Wide Database* (base de dados mundial).".

Vemos que o conteúdo da informação disponível na *Internet* está armazenado em inúmeros bancos de dados sob diversos formatos (áudio, imagens, texto e outros), os quais abrangem distintas áreas do conhecimento, contemplando usuários com variados perfis de interesse. Contudo, o desafio é tornar o acesso a essas informações rápido e, sobretudo, torná-las relevantes para os usuários que as procuram.

Com a popularização da *Internet*, as Máquinas de Busca surgiram no intuito de auxiliar os usuários na pesquisa de informação na *Web* de modo rápido e eficiente, mostrando os resultados de uma forma organizada [26]. Porém, é comum usuários formularem consultas e receberem resultados que não satisfazem a sua intenção inicial de pesquisa. Por exemplo, um estudante de Sistemas de Comunicações Móveis formulou uma consulta com a palavra

"célula". No entanto, os resultados obtidos pela Máquina de Busca abordaram as áreas de Biologia e de Telecomunicações, apesar de seu interesse estar relacionado unicamente à sua área de estudo. Ou ainda, suponhamos que um pesquisador de Ciência da Computação formule uma consulta com a sigla MOF a fim de obter informações relacionadas à *Meta-Object Facility*, todavia, além dos esperados, foram retornados resultados referentes ao *Microsoft Operations Framework*.

Tendo em vista as dificuldades encontradas no processo de recuperação da informação, diversos congressos e *workshops* vêm sendo realizados [27-29] com o objetivo de estudar e discutir formas para otimizar os resultados obtidos pelos algoritmos de busca. A técnica *Information Retrieval in Context* (IRiX), bastante discutida em eventos internacionais de Recuperação da Informação, permite que informações relacionadas a um determinado tema – por exemplo, palavras-chave, títulos, frases ou parágrafos extraídos a partir de um texto – possam ser atribuídas à consulta formulada originalmente pelo usuário, possibilitando, dessa forma, melhores resultados.

Neste trabalho, será apresentado um protótipo de uma ferramenta de recuperação da informação baseada em contextos que são construídos a partir de um conjunto de sintagmas e de relacionamentos definido pelo usuário. Esse conjunto de informações, apesar de estar inicialmente associado ao seu usuário criador, pode ser compartilhado com agentes de *software* e com outros usuários da ferramenta a fim de promover a socialização dos contextos. O protótipo também deve ser capaz de permitir que, além dos usuários humanos, as aplicações clientes solicitem pesquisas na base de documentos.

1.1 MOTIVAÇÕES

No período de 28 de setembro de 2004 e 12 de janeiro de 2005, uma empresa especializada na área de *marketing* em *sites* de busca realizou uma pesquisa *on-line* e obteve 719 opiniões de usuários de Máquinas de Busca. A pesquisa teve como intuito avaliar diversos aspectos, dentre eles a satisfação dos usuários com as ferramentas de busca. Nesse aspecto, a pesquisa revelou que apenas 12,80% dos usuários participantes mostraram-se plenamente satisfeitos com os resultados que as ferramentas de busca retornam [30], um percentual consideravelmente baixo.

Apesar de o objetivo das Máquinas de Busca ser auxiliar eficientemente a pesquisa de informação, a literatura atual descreve a dificuldade dos usuários em utilizá-las pelo fato de as

interfaces não serem totalmente amigáveis ou de os resultados apresentados ainda estarem distantes das reais intenções das consultas [31].

Assim, é indispensável que os mecanismos de recuperação da informação sejam mais eficazes no que diz respeito à recuperação de documentos. É fundamental que os resultados localizados sejam coerentes com as expectativas iniciais e estimulem os usuários a acessarem os documentos listados.

1.2 OBJETIVOS

As Máquinas de Busca, assim como a *Web*, vêm evoluindo no decorrer dos últimos anos através de gerações. Um dos motivos dessa evolução é a tentativa de otimizar os algoritmos de recuperação da informação [40]. Estes devem estar preparados para buscar e localizar resultados que sejam realmente relevantes para seus usuários, que se apresentam mais exigentes e com perfis de interesse e de conhecimento bastante diversificados.

Abaixo, são listados os objetivos deste trabalho:

- estudar *web service* com a finalidade de compreender os propósitos e as características desta tecnologia;
- pesquisar as Máquinas de Busca para compreender as suas características e o seu comportamento;
- especificar e implementar um protótipo capaz de recuperar documentos através de contextos construídos a partir de informações associadas a determinados assuntos;
- analisar, por meio da validação dos cenários, realizada no capítulo 4 (Especificação, Implementação e Análise de Resultados), a coerência e a precisão dos resultados obtidos pelo protótipo.

1.3 CONTRIBUIÇÃO

Como dito na seção anterior, os usuários possuem vários perfis de interesse e de conhecimento e esperam que as Máquinas de Busca atendam satisfatoriamente as suas necessidades.

Em muitas Máquinas de Busca atuais, para efetuar uma pesquisa, o usuário formula uma consulta informando frases e/ou palavras-chave. No entanto, se, em um outro momento,

ele desejar efetuar a mesma pesquisa, deverá informar novamente as mesmas frases e palavras-chave.

O protótipo a ser desenvolvido agilizará a formulação de consultas. Se, por ventura, o usuário desejar efetuar uma busca já realizada com o uso das informações dos seus contextos de pesquisa, será preciso apenas que ele escolha o contexto que contém as informações necessárias para a sua busca e, automaticamente, o protótipo adicionará essas informações de sintagmas e de relacionamentos à consulta formulada.

Outro ponto a se considerar é a forma como os contextos estarão disponíveis. No momento da criação, eles poderão ser configurados para que estejam disponíveis para outros usuários, que podem ser seres humanos ou *softwares*. Quando o usuário em questão for um *software*, a comunicação entre o protótipo e a aplicação cliente será feita por meio de *web services* a partir da troca de mensagens *eXtensible Markup Language* (XML). A funcionalidade de permitir que a informação seja legível tanto para pessoas como para máquinas faz com que o protótipo possua uma das características da *Web* Semântica, enquanto o compartilhamento das informações de contextos entre usuários e aplicações e a socialização (ou inteligência coletiva) são características da *Web* 2.0.

1.4 ORGANIZAÇÃO DO TRABALHO

Esta dissertação está organizada da seguinte forma: no capítulo 1 (Introdução), é apresentada uma breve descrição do trabalho, os aspectos que motivaram o seu desenvolvimento, os seus objetivos e as suas contribuições. No capítulo 2 (Fundamentação Teórica), são descritos os conceitos e as tecnologias envolvidas durante as fases da pesquisa. No capítulo 3 (Análise de Requisitos), são expostos os cenários e os requisitos necessários para a especificação e para a implementação do protótipo da Máquina de Busca. No capítulo 4 (Especificação, Implementação e Análise de Resultados), são apresentadas a especificação e a implementação do protótipo e, finalizando-o, é feita a validação dos cenários descritos anteriormente. No último capítulo (Conclusões), são apresentadas as conclusões obtidas pela pesquisa e feitas sugestões para trabalhos futuros.

“A alegria está na luta, na tentativa, no sofrimento envolvido. Não na vitória propriamente dita.”

Mahatma Gandhi

CAPÍTULO 2
FUNDAMENTAÇÃO TEÓRICA

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os principais conceitos e as tecnologias utilizadas para o desenvolvimento do protótipo de uma Máquina de Busca baseada em contextos, proposto nesta dissertação.

Na seção 2.1, é exposto o conceito de recuperação da informação, suas etapas e seus modelos. Na seção 2.2, são descritos os tipos e a funcionalidade das ferramentas de busca *Web*, como também a evolução das Máquinas de Busca através de gerações. Na seção 2.3, é mostrado o conceito de contextos e como tem sido empregado. Na seção 2.4, são apresentados os conceitos de sintagmas e de relacionamentos. Na 2.5, é exposto o padrão XML, abordando sua importância para simplificar a comunicação entre aplicações. Na seção 2.6, é abordado o conceito de *Web Service*, os seus componentes e a sua aplicabilidade; e, por fim, na seção 2.7, o resumo deste capítulo.

2.1 RECUPERAÇÃO DA INFORMAÇÃO

A Recuperação da Informação, termo criado por Calvin Mooers em 1950, é baseada em diversas áreas, tornando-se a interseção de diversos campos já estabelecidos, como a Arquitetura da Informação, a Ciência da Informação, a Ciência da Computação, a Biblioteconomia, a Estatística e outros [33].

O processo de recuperação da informação consiste em analisar os documentos armazenados no sistema e selecionar os que atendem satisfatoriamente as necessidades dos seus usuários. Estes solicitam e esperam que o sistema recupere informações sobre um determinado assunto, e não dados sobre a sua expressão de busca [34]. O processo de recuperação da informação é divido em três etapas:

- indexação de documentos – consiste na criação das estruturas de dados relacionadas aos textos disponíveis, etapa em que a lista de arquivos invertidos é uma estrutura comumente utilizada;
- formulação da consulta – tem como propósito recuperar um número de documentos que atendam às necessidades do usuário por meio de uma linguagem artificial ou especificada em linguagem natural;

- recuperação de resultados – é uma listagem de documentos recuperados gerada a partir da consulta formulada pelo usuário e ordenada decrescentemente em função da relevância dos documentos para a consulta.

Para realizar o cálculo da relevância de um documento, o sistema utiliza um modelo de representação dos documentos e da consulta. Existem vários modelos, no entanto apenas três são considerados clássicos [35]:

- modelo vetorial – representa documentos através de vetores, que descrevem a relevância do termo indexado no documento para determinadas consultas;
- modelo *booleano* – recupera os documentos que atendem a uma expressão lógica da consulta, uma vez que esta pode ser formada por elementos lógicos (*AND*, *OR* e *NOT*);
- modelo probabilístico – representa os documentos através de pesos binários que determinam a presença ou a ausência de termos.

Várias técnicas de recuperação da informação vêm sendo propostas, contudo a lista de documentos recuperados por esses sistemas ainda não satisfaz plenamente as necessidades de seus usuários, estimulando, assim, pesquisas sobre novas formas de recuperar informação.

2.2 FERRAMENTAS DE BUSCA WEB

Com o crescimento do número de páginas *web*, sentiu-se a necessidade de criação de algum mecanismo que auxiliasse os usuários na tarefa de pesquisar informação nos conteúdos disponíveis na *Internet*. Para atender a essa demanda, foram desenvolvidas diversas ferramentas com o propósito de tornar menos complexo o processo de busca. Elas se classificam em 3 tipos [36]: Diretórios por Assunto, Máquinas de Busca e Meta-Máquinas de Busca.

2.2.1 Diretórios por Assunto

Os Diretórios por Assunto, considerados os precursores da busca de informação na *Internet* [37], são organizados hierarquicamente por meio de categorias e de subcategorias,

permitindo aos usuários efetuarem pesquisas em sua base de dados de forma manual. A fim de atender a vários grupos de usuários, eles armazenam *sites* relacionados a diversos assuntos, como esporte, política, lazer, educação, cultura, tecnologia, finanças, saúde, compras, dentre outros.

Antes de serem categorizados, os *sites* são coletados por seres humanos (ou editores), por agentes de *softwares* que percorrem a *Web* em busca de novos recursos ou a partir de sugestões de seus próprios usuários. Depois da coleta, os *sites*, cujo conteúdo é avaliado pelos editores e submetido à aprovação deles, têm seu endereço publicado. Por terem sido avaliados por pessoas, é possível, mas não garantido, que os Diretórios possuam *sites* de boa qualidade.

Os Diretórios ainda podem diferenciar-se um dos outros em função de algumas características relacionadas aos recursos no que concerne à forma de organização, à descrição e aos assuntos abordados.

Em relação à forma de organização dos recursos, normalmente os Diretórios utilizam a forma de hierarquização por assunto. No entanto, existem alguns que utilizam outras formas de classificação, geralmente adotadas por bibliotecários, como a Classificação Decimal de Dewey e a Classificação Decimal Universal (CDU).

Quanto à descrição, os Diretórios podem distinguir-se em função da quantidade de palavras utilizadas para descrever um *site.* Uma outra forma de descrever *sites* é atribuir a eles detalhes mais específicos sobre o assunto abordado, como, por exemplo, críticas e comentários.

No que diz respeito aos assuntos abordados pelos recursos, nem todos abrangem o público como um todo, mas só parte dele. Alguns Diretórios são especializados apenas em áreas específicas como Informática, Política, Medicina, Telecomunicações, entre outras.

Depois que o CERN lançou o primeiro Diretório, o *World Wide Web Virtual Library*, vários outros apareceram no mercado com peculiaridades distintas. Alguns exemplos são:

- *Directorio web*, disponível em: http://www.directorioenlinea.com.ar;
- *Open Directory Project*, disponível em: http://www.dmoz.org;
- *Spravka Web Directory*, disponível em: http://www.spravka.co.uk;
- *WebzDirectory*, disponível em: http://www.webzdirectory.com;
- *Yahoo*! Cadê?, disponível em: http://br.cade.yahoo.com;
- Buscando TUDO, disponível em: http://www.buscandotudo.com;
- Achei Aqui, disponível em: http://www.acheiaqui.com.br.

2.2.2 Máquinas de Busca

Surgiram quando a quantidade de páginas disponíveis na *Internet* começou a dificultar a seleção manual e a navegacional dos recursos [37], como é feita nos Diretórios por Assunto. As páginas armazenadas em sua base de dados são localizadas e analisadas por um programa chamado aranha. Porém, a análise dos recursos selecionados corre o risco de não ser tão bem realizada como seria se fosse feita manualmente.

As Máquinas de Busca apresentam características particulares em relação à forma como os documentos são pesquisados, armazenados e manipulados [38]. Apesar disso, existem tarefas que são comuns para a maioria das Máquinas de Busca:

- descoberta – consiste em percorrer a *Web* em busca de recursos para armazená-los em sua base de dados;
- indexação – compreende a criação de índices de palavras que foram localizadas pelas aranhas com seu respectivo caminho de acesso, além de outras informações;
- pesquisa – permite pesquisas de recursos armazenados em sua base de dados a partir de consultas de palavras ou de combinações.

Outro ponto em comum a se considerar são os elementos que compõem uma Máquina de Busca: aranha, indexador, motor de busca e *interface*.

A Fig. 2.1 representa, de uma forma genérica, uma Máquina de Busca. O controlador é um programa responsável por inicializar os serviços executados pelas aranhas e pelo indexador. Quando o controlador inicializa as aranhas, elas iniciam uma busca na *Web* à procura de recursos que comporão a base de dados. Quanto ao indexador, quando inicializado, ele efetua a indexação dos recursos identificados pelas aranhas e os armazena na base de dados. Quando o usuário deseja realizar uma pesquisa, ele formula uma consulta e a envia para a *interface*, que, por sua vez, formata esta consulta enviando-a para o motor de busca. Este faz a pesquisa na base de dados de recursos que atendam aquela consulta. Depois de localizados, o motor de busca faz a ordenação dos recursos e os entrega à *interface*, que monta uma listagem e apresenta ao usuário.

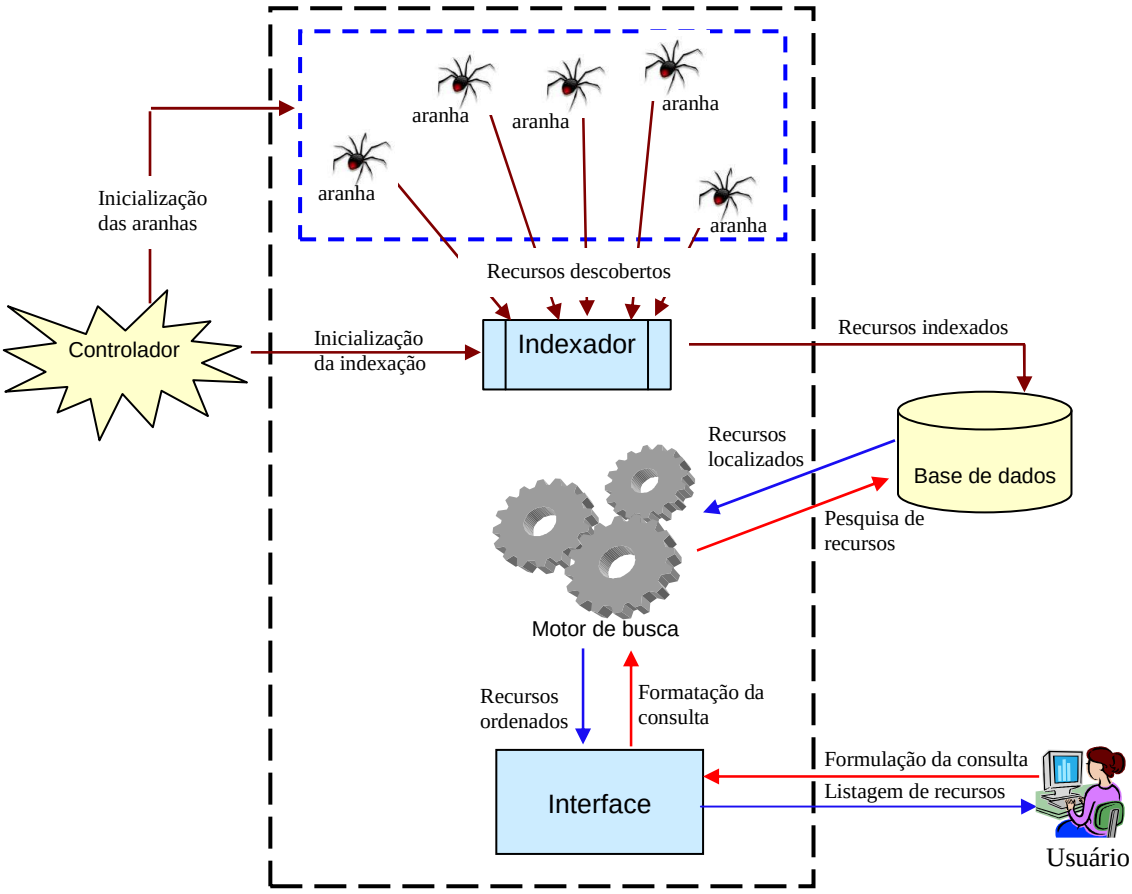

Fig. 2.1 – Representação genérica de uma Máquina de Busca

2.2.2.1 Descoberta

Para que uma Máquina de Busca seja capaz de informar aos seus usuários os locais onde eles podem encontrar recursos, é necessário, inicialmente, que ela mesma conheça esses locais.

A descoberta de recursos é uma tarefa realizada por um programa chamado aranha (ou *crawler*), que pode ser executado individualmente ou em paralelo. O processo consiste em percorrer a *Web* à procura de informações sobre o maior número possível de recursos com o propósito de armazenamento.

Os *crawlers* possuem algoritmos próprios que determinam a forma de navegação entre os documentos de um *site*. Normalmente, a procura é iniciada a partir de *sites* bastante acessados que contêm vários *links*. Para cada documento visitado é montada uma lista com informações que lhe são peculiares [39].

Algumas informações, além do próprio conteúdo, podem ser adicionadas a um *site* sem serem interpretadas pelos navegadores. O *HyperText Markup Language* (HTML) permite que sejam utilizadas *metatags* simbolizadas por <META>, que podem ser usadas por meio do par descrição-valor representando atributos do documento como, por exemplo, autor, data de expiração, ferramenta utilizada para criação da página, palavras-chave, descrição, entre outros.

Concluída essa tarefa, o próximo passo será indexar as informações descobertas e armazená-las no banco de dados.

2.2.2.2 Indexação

Depois do trabalho de descoberta de informações, uma nova tarefa deve ser realizada para garantir que as informações estejam disponíveis aos usuários da Máquina de Busca. Esse processo é chamado de indexação ou construção de índices.

A indexação é realizada por um programa chamado de indexador. Ele é responsável por extrair o conteúdo dos *links* descobertos e armazenar na base de dados informações como título, endereço e palavras-chave relacionadas àqueles recursos.

2.2.2.3 Pesquisa

Depois de os recursos serem descobertos e, posteriormente, indexados, a Máquina de Busca já pode oferecer aos seus usuários consultas em sua base de dados.

A pesquisa de recursos é iniciada quando o usuário formula, numa *interface* (*web*, *desktop* ou linha de comando), uma consulta e esta é encaminhada para um programa chamado motor de busca, elemento que realiza efetivamente a pesquisa na base de dados e se encarrega de determinar a ordem dos resultados a serem listados. A ordenação destes é definida com base na relevância que têm naquela consulta, partindo do mais para o menos relevante.

Alguns exemplos de Máquinas de Busca são:

- *Google*, disponível em: http://www.google.com.br;
- MSN, disponível em: http://www.msn.com.br;
- Altavista, disponível em: http://br.altavista.com;
- *Lycos*, disponível em: http://www.lycos.com;

2.2.2.4 Evolução

A primeira geração se caracterizou por determinar a relevância das páginas com o emprego de *metatags* e de palavras-chave, principalmente se estas fossem encontradas na *Uniform Resource Locator* (URL) [40].

A segunda geração faz uso de rastreamento de cliques, reputação de páginas, popularidade de *link*, rastreamento temporal e qualidade de *link*.

A terceira geração se distingue por efetuar extração automática de palavras-chave, permitindo a categorização de páginas. Faz uso de mapas *web*, ferramentas úteis para filtrar sites duplicados, e de páginas autônomas que direcionam a busca para determinados lugares.

Por fim, a quarta geração é chamada de Máquinas de Tema, que, assim como as máquinas das outras gerações, também possuem um programa de indexação que realiza a extração das principais palavras-chave e determina o tema ao qual aquele site (ou página) faz referência. Algumas Máquinas de Busca são classificadas como baseadas “em contexto”, ou “em *rank* e reputação”, ou ainda “em tópico”. Porém, essas são várias definições para as ferramentas que fazem parte do grupo de Máquinas de Tema [40].

2.2.3 Meta-máquinas de Busca

Por possuírem características diferentes de indexação e de pesquisa, as Máquinas de Busca normalmente localizam resultados diferentes para uma mesma consulta. Em função disso, para se obter uma pesquisa mais apurada, é de boa prática utilizar mais de uma Máquina de Busca.

Para agilizar essa forma de pesquisa, foram criadas as Meta-máquinas, que realizam consultas em Máquinas de Busca e em Diretórios. Em outras palavras, as Meta-máquinas de Busca possuem uma *interface* que recebe consultas formuladas por usuários e as envia para outras ferramentas de busca *web*. Depois de receber o processamento, os resultados são centralizados em uma única listagem. Em geral, essa ferramenta não possui uma base de dados própria, o que não é, todavia, uma regra.

Exemplos de Meta-Máquinas de Busca são:

- *Mamma*, disponível em: http://www.mamma.com;
- *Metasearch Search Engine*, disponível em: http://www.search.com.

2.3 CONTEXTOS

Atualmente, o termo contexto tem sido bastante empregado na área de Máquinas de Busca. Em razão disso, vê-se que existem várias definições para a mesma palavra, apesar da existência de pequenas diferenças entre cada uma delas [41].

Neste estudo, contexto é tomado como um conjunto de termos que possui uma relação com um determinado tema. Essas informações – que podem ser palavras-chave, títulos, frases ou parágrafos extraídos a partir de um texto – são atribuídas à consulta original do usuário.

Com o uso da técnica de recuperação da informação baseada em contextos, surgiu o termo “pesquisa contextualizada”, que consiste em realizar pesquisas utilizando informações obtidas a partir de um conjunto de termos. A pesquisa contextualizada é a convergência das três principais áreas de recuperação da informação moderna: inteligência, flexibilidade e performance [42].

2.4 SINTAGMAS E RELACIONAMENTOS

A maioria dos sistemas de recuperação da informação utiliza como forma de acesso à informação a palavra. Contudo, esta apresenta uma característica denominada polissemia, que consiste na existência de vários significados para uma mesma forma, permitindo que o resultado gerado pelos sistemas que usam essa forma de acesso à informação seja comprometido. Abaixo, são listados alguns exemplos de polissemia com o uso da palavra *aloha*:

- um dos primeiros métodos de acesso para rede de computadores sem fio, desenvolvido na Universidade do Havaí;
- companhia de transportes aéreos;
- partido político americano;
- saudação havaiana.

Percebe-se, então, que a palavra pode não ter um sentido preciso se for analisada fora do seu contexto original. Assim, para Kuramoto, analisada isoladamente, ela não representa a melhor forma de acessar a informação. Por isso, como proposta para recuperar recursos com maior precisão, ele sugeriu a indexação e a recuperação da informação por meio de sintagmas nominais [31].

Formados por um ou mais vocábulos, os sintagmas desempenham uma função determinada na frase. Nesta, as palavras não são posicionadas aleatoriamente, mas sim formando um conjunto (o sintagma) em torno de um núcleo. Dependendo deste, o sintagma pode ser verbal ou nominal [43-44].

Com o objetivo de mostrar o potencial de estruturação dos sintagmas nominais por meio de relações de encadeamento, a partir da frase “As características do meio ambiente do mundo dos negócios”, são extraídos quatro sintagmas nominais [61]:

→ sintagma de nível 1: os negócios
→ sintagma de nível 2: o mundo dos negócios
→ sintagma de nível 3: o meio ambiente do mundo dos negócios
→ sintagma de nível 4: as características do meio ambiente do mundo dos negócios

A cada extração, os sintagmas nominais são classificados em níveis. Assim, a partir do 4, é extraído o 3, o qual origina o 2 e, assim por diante. No exemplo, percebe-se que o nível 1 é o mais simples e o 4 é o original.

Outro conceito que será bastante utilizado neste trabalho é o de relacionamento. Os relacionamentos representam uma relação entre dois termos que podem conter mais de uma palavra. Eles são formados por um conjunto de siglas, abreviaturas, acrônimos e símbolos relacionados aos seus respectivos significados.

Uma sigla é formada pelas letras iniciais dos vocábulos fundamentais de uma denominação ou título, apesar de algumas vezes ser interpolada por alguma vogal para facilitar a pronúncia. Exemplo: FAB, de Força Aérea Brasileira.

Uma abreviatura é a redução de uma palavra seguida de um ponto, mas pode acontecer de as letras não seguirem uma seqüência. Exemplos: “ass.”, de assinatura, e “pte.”, de presidente.

Já um acrônimo é formado pela primeira letra (ou mais de uma) de cada uma das partes sucessivas de uma locução ou pela maioria dessas partes [45]. Exemplo: “radar”, de *radio detecting and raging*.

E, por fim, símbolo é um sinal que representa um nome de uma coisa ou de uma ação [46]. Exemplo: “cm”, de centímetro.

2.5 XML

XML, sigla de *eXtensible Markup Language* e derivada da *Standard Generalized Markup Language* (SGML), é uma linguagem de marcação recomendada pela *World Wide Web Consortium* (W3C) capaz de representar diversos tipos de dados e que tem como objetivo facilitar o compartilhamento de informações via *Internet*. Por utilizar caracteres *Unicode*, o XML viabiliza o armazenamento de qualquer tipo de caractere ou símbolo, o que permite a representação das informações em qualquer idioma [47].

O XML é um padrão bastante útil para efetivar a comunicação entre aplicações que utilizam plataforma e/ou ambientes distintos. Com XML, essa comunicação torna-se menos complexa.

```
<músicas>
  <música>
    <nome>O caderno</nome>
    <compositor>Toquinho</compositor>
    <estrofe>
      <linha>Sou eu que vou seguir você</linha>
      <linha>Do primeiro rabisco até o be-a-bá</linha>
      <linha>Em todos os desenhos coloridos vou estar:</linha>
      <linha>A casa, a montanha, duas nuvens no céu</linha>
      <linha>E um sol a sorrir no papel</linha>
    </estrofe>
  <música>
</músicas>
```

Listagem 2.1 – Documentação de uma estrofe musical utilizando XML

As seguintes características são atribuídas ao padrão XML:

- facilidade de interpretação tanto para pessoas como para computadores;
- comunicação entre bancos de dados distintos;
- criação ilimitada de *tags*;
- separação entre os dados e a apresentação.

A Listagem 2.1 é um exemplo de como documentar estrofes musicais em uma estrutura XML. Para representar um trecho da música “O caderno”, do compositor Toquinho, foram utilizadas as *tags* música, nome, compositor, estrofe e linha.

2.6 WEB SERVICE

Web Service é uma tecnologia utilizada para realizar a integração de sistemas e permitir a comunicação entre diversas aplicações, mesmo que elas sejam desenvolvidas em plataformas diferentes. Normalmente, são *Application Programming Interfaces* (APIs) que podem ser acessadas via rede e executadas remotamente em um cliente requisitante de um serviço [48-49].

A arquitetura de um *web service*, representada na Fig. 2.2, é formada por três componentes:

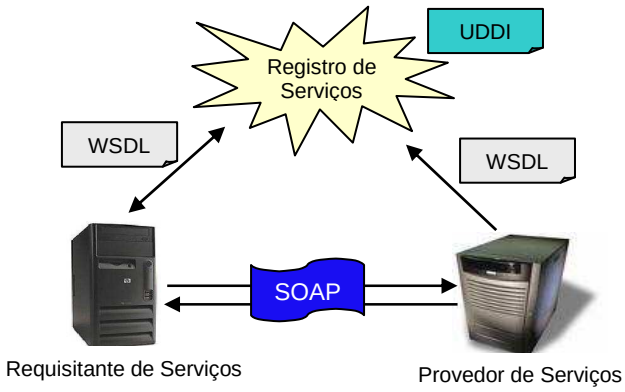

Fig. 2.2 – Arquitetura de um *Web Service*

- provedor de serviços – responsável por disponibilizar os serviços para que estes possam ser utilizados pelos clientes;
- registro de serviços – local onde o provedor de serviços pode disponibilizar os seus serviços;
- requisitante de serviços – cliente que utiliza qualquer serviço disponibilizado pelo provedor de serviços.

Os *web services* são descritos por *Web Service Description Language* (WSDL), registrados via *Universal Description, Discovery and Integration* (UDDI) e acessados por meio do *Simple Object Access Protocol* (SOAP) [48]. Abaixo, uma breve explicação dessas tecnologias:

- SOAP – protocolo baseado em XML que permite a troca de mensagens entre aplicações e é utilizado para efetivar a comunicação entre *web services*;
- WSDL – linguagem de descrição dos *web services* que possibilita a descrição de serviços e a troca de mensagens;

- UDDI – especificação que permite aos provedores publicar os seus serviços e aos requisitantes identificá-los.

São várias as vantagens em se utilizar *web services*. A seguir são destacadas algumas [50]:

- processamento distribuído – a aplicação pode ser dividida em partes e estar sendo executada em vários servidores simultaneamente;
- ambiente distribuído – os servidores que hospedam a aplicação podem estar em lugares distintos;
- processamento do lado do servidor – os clientes requisitantes não precisam ter alto poder de processamento pelo fato de este ser realizado no servidor;
- padronização no desenvolvimento – o código da aplicação que é executado na rede local é o mesmo da *Internet*;
- custo – a obtenção de licenças para uso não é necessária.

2.7 RESUMO

Neste capítulo, foram apresentadas as etapas e os modelos dos sistemas de recuperação da informação, além dos tipos de ferramentas de busca *web* – Diretórios por Assunto, Máquinas e Meta-máquinas de Busca – e suas características. Foram descritos os conceitos de contextos, sintagmas e relacionamentos. Foi também abordado o padrão XML, bem como sua importância na comunicação entre sistemas distintos. E, por fim, foram expostos os *web services*, sua arquitetura e tecnologias utilizadas.

"Qualquer ato de amor, por menor que seja, é um trabalho pela paz."

Madre Tereza de Calcutá

CAPÍTULO 3

ANÁLISE DE REQUISITOS

3 ANÁLISE DE REQUISITOS

Este capítulo descreve os requisitos indispensáveis ao desenvolvimento do protótipo de uma Máquina de Busca sendo dividido da seguinte forma: na seção 3.1, são apresentados os cenários necessários para o desenvolvimento do protótipo; na seção 3.2, em detalhes, são descritos os requisitos fundamentais que atendem aos cenários apresentados; e, na seção 3.3, é feito um resumo do que foi discutido no capítulo.

3.1 DEFINIÇÃO DOS CENÁRIOS

A definição de cenários é uma tarefa indispensável para se descobrirem os principais requisitos de um sistema. Quando bem definidos, os cenários auxiliam o projetista a compreender os objetivos do projeto, bem como identificar as visões individuais de cada ator que interage com o sistema.

Assim, foram descobertas e analisadas três visões que representam a interação entre os atores e o sistema: a visão de um usuário, a visão de um administrador e a visão de uma aplicação.

1. Visão de um usuário – disponibiliza para o usuário as principais funcionalidades do protótipo. É nela que podem ser criados e configurados os perfis de usuários, além da execução de suas pesquisas.

2. Visão de um administrador – permite que o administrador configure as funções do protótipo. A configuração consiste na parametrização da execução de algumas funções, como também no cadastro de fontes de recursos e na criação de contas de administradores.

3. Visão de uma aplicação – possibilita que as aplicações requisitem o serviço oferecido de busca baseada em contexto e que possam obter contextos dos usuários cadastrados no protótipo.

A Fig 3.1 apresenta, graficamente, os cenários que compõem as visões identificadas e os seus respectivos cenários:

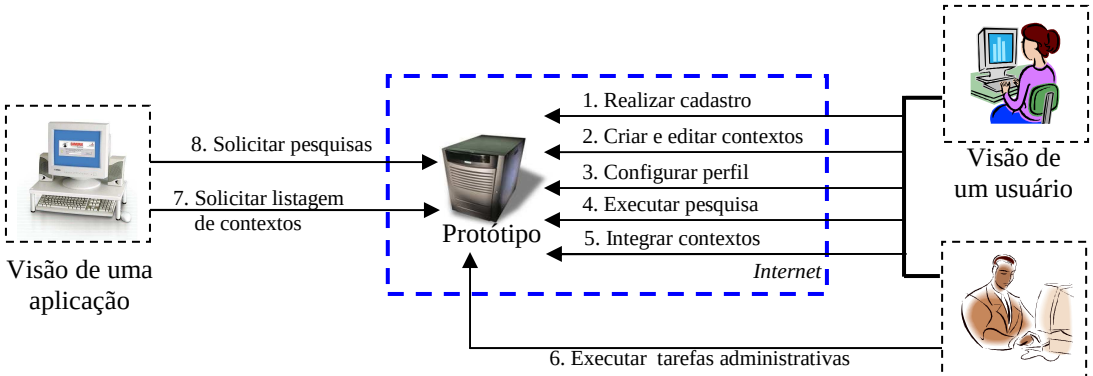

Fig 3.1 – Cenários de uso

A visão de um usuário é composta pelos cenários 1, 2, 3, 4 e 5. No cenário 1, o usuário informa alguns dados necessários para a realização do seu cadastro; no 2, fornece informações utilizadas para criar e editar contextos de pesquisa; no 3, configura seus perfis de pesquisa por meio da atribuição de sintagmas, relacionamentos e fontes de recursos associados aos contextos criados no cenário 2; no 4, efetivamente, as suas pesquisas; e no 5, integra os contextos cadastrados por outros usuários ao seu perfil.

A visão de um administrador abrange todos os cenários de que o usuário participa, além do cenário 6. Neste, o administrador controla algumas operações utilizando parâmetros de configuração – os quais determinam o comportamento do protótipo – ou ainda cria novos usuários com permissões de administrador.

A visão de uma aplicação é composta pelos cenários 7 e 8. No primeiro, o protótipo disponibiliza a listagem dos contextos cadastrados em sua base de dados às aplicações clientes; no segundo, o serviço de busca por ele oferecido.

3.2 DEFINIÇÃO DOS REQUISITOS

Depois da definição dos cenários que compõem o escopo do protótipo, faz-se necessário analisá-los mais detalhadamente com o intuito de descobrir os requisitos fundamentais para especificá-los e implementá-los. A seguir, são descritos seus requisitos.

Visão de um usuário

- Cenário 1 – permite que o usuário realize o seu cadastro e utilize as principais funções do protótipo. Para isso, é necessário que o requisito 1 seja satisfeito.

  - **Requisito 1** – o usuário informa dados básicos – como nome, *login*, senha, data de nascimento e outros – e os envia para o protótipo, que deve fazer o armazenamento em sua base de dados para futuros controles, como, por exemplo, a identificação no sistema e o perfil de pesquisa. Atendido esse requisito, as principais funções do protótipo ficam disponíveis para o usuário.

- Cenário 2 – possibilita que o usuário crie e edite contextos associados ao seu perfil de pesquisa. Para tanto, é necessário que o requisito 2 seja satisfeito.

  - **Requisito 2** – o protótipo deve receber os dados informados pelo usuário – como nome do contexto, compartilhamento disponível (sim ou não) e *status* (ativo ou inativo) – relacionando-os à identificação desse usuário. Atendido esse requisito, o processo de configuração de perfil torna-se disponível para o usuário.

- Cenário 3 – torna possível que o usuário configure os perfis utilizados em suas pesquisas. Para isso, é necessário que os requisitos 3, 4, 5 e 6 sejam satisfeitos.

  - **Requisito 3** – o usuário seleciona, a partir de uma lista, os contextos definidos no cenário 2, e o protótipo deve fazer a associação entre o nome do contexto selecionado e o perfil que está sendo configurado.
  - **Requisito 4** – o usuário pode definir relacionamentos associados ao contexto selecionado no requisito 3. É possível a criação ou edição de relacionamentos através do par código/descrição, em que o primeiro consiste num acrônimo, uma abreviatura, uma sigla ou um símbolo; e o segundo, num significado para o código.
  - **Requisito 5** – o usuário pode definir sintagmas relacionados ao contexto do requisito 3. O processo se dá através da criação ou da edição de palavras e/ou frases que são relacionadas ao contexto.

- **Requisito 6** – o protótipo também pode acrescentar ao contexto selecionado no requisito 3 sugestões de usuários informando URLs, as quais possuem informações que possivelmente lhes serão úteis.

Com a realização dos quatro requisitos descritos anteriormente, o processo de configuração de perfil de pesquisa torna-se disponível para o usuário.

- Cenário 4 – realiza a pesquisa dos usuários. Para tanto, é necessário que os requisitos 7, 8, 9 e 10 sejam atendidos.

  - **Requisito 7** – os sintagmas e os relacioanamentos associados ao contexto escolhido para a pesquisa devem ser identificados pelo protótipo, o qual, por sua vez, deve acrescentar as informações à consulta originalmente formulada pelo usuário, caso este selecione a opção de pesquisa contextualizada.
  - **Requisito 8** – os contextos criados pelo usuário de pesquisa são listados pelo protótipo, o qual, por sua vez, deve mostrar outros cadastrados e compartilhados pelos usuários a fim de disponibilizar uma lista maior de contextos de pesquisa, promovendo a socialização.
  - **Requisito 9** – o protótipo deve ler a consulta e formatá-la de acordo com a sintaxe utilizada pelo motor de busca. A formatação é um processo obrigatório, independente da consulta ter sido acrescida ou não de informações adquiridas a partir dos contextos utilizados.
  - **Requisito 10** – os documentos que atendem satisfatoriamente as consultas dos usuários devem ser pesquisados pelo protótipo em sua base de dados. Encontrada essa lista de documentos, o protótipo deve ordená-la conforme a relevância da consulta.

  Com a satisfação dos requisitos 7, 8, 9 e 10, descritos anteriormente, o protótipo é capaz de gerar uma lista de documentos que, possivelmente, atenderá à consulta formulada pelo usuário.

- Cenário 5 – é nesse cenário que o usuário integra ao seu perfil os contextos que foram definidos por outros usuários. Para isso, é necessário que o requisito 11 seja satisfeito.

- **Requisito 11** – a integração de contextos requer a existência de alguns deles criados e compartilhados por outros usuários. O contexto é definido como compartilhado no momento da sua criação. Satisfeito este requisito, o cenário 5 é atendido.

Visão de um administrador

Como mostrado em Fig. 3.1, o ator Administrador, assim como o ator Usuário, também participa dos cenários 1, 2, 3, 4 e 5. Porém, para evitar a repetição dos mesmos cenários e dos seus respectivos requisitos, nesta visão, só será apresentado o 6, uma vez que os demais e seus requisitos já foram discutidos.

- Cenário 6 – permite que o administrador realize as operações de administração do protótipo. Para tanto, é necessário que o requisito 12 seja satisfeito.

  - **Requisito 12** – o protótipo deve armazenar e utilizar parâmetros de configuração definidos pelos administradores. Além disso, o administrador pode informar fontes de recursos a serem utilizadas pelos *crawlers* e criar novos usuários com privilégios iguais aos seus. A realização desse requisito atende, assim, ao cenário 6.

Visão de uma aplicação

- Cenário 7 – possibilita que o protótipo disponibilize o cadastro dos contextos de pesquisa, exigindo que o requisito 13 seja atendido.

  - **Requisito 13** – os contextos cadastrados devem ser pesquisados, em sua base de dados, pelo protótipo. Depois disso, este deve ser capaz de associar todas as informações de sintagmas, relacionamentos e fontes de recursos (que estão relacionadas a esses contextos) e de disponibilizá-las para que aplicações clientes possam utilizá-las. Por meio desse requisito, o cenário 7 se cumpre satisfatoriamente.

- Cenário 8 – oferece, por meio do protótipo, um serviço de busca que atende às requisições das aplicações. Faz-se necessário, no entanto, que o requisito 14 seja seguido.

- **Requisito 14** – o protótipo deve ser capaz de responder às requisições de outras aplicações que solicitam o serviço de busca. Para isso, ele deve interpretar as requisições enviadas, processá-las, obter os resultados que atendam a elas e disponibilizar estes para a aplicação requisitante. Dessa forma, a disponibilidade do serviço de busca para aplicações é atendida.

3.3 RESUMO

Neste capítulo, foram apresentadas as visões que contemplam o escopo do protótipo da Máquina de Busca proposta. Foram identificados os cenários que compõem as visões de um usuário, de um administrador e de uma aplicação, bem como os requisitos necessários para que os cenários dessas visões sejam seguidos.

"No meio de toda dificuldade, sempre existe uma oportunidade."

Albert Eistein

CAPÍTULO 4

ESPECIFICAÇÃO, IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS

4 ESPECIFICAÇÃO, IMPLEMENTAÇÃO E ANÁLISE DE RESULTADOS

Este capítulo descreve a especificação e a implementação do protótipo da Máquina de Busca além da avaliação dos resultados obtidos.

O capítulo está organizado da seguinte forma: na seção 4.1, é mostrada a especificação do protótipo, que é dividida na apresentação do diagrama de casos de uso, do diagrama de classes e do diagrama de seqüências; na seção 4.2, é apresentada, efetivamente, a implementação do protótipo por meio de trechos de código, esquemas de representação e elementos utilizados no desenvolvimento; na seção 4.3 é feita a análise dos resultados por meio da validação dos cenários; e, por fim, na seção 4.4, é exposto um resumo do capítulo.

4.1 ESPECIFICAÇÃO

Os cenários analisados no capítulo anterior geraram requisitos que se tornaram fundamentais para a especificação do protótipo de uma Máquina de Busca baseada em contextos. A configuração de perfis, a execução de pesquisas e os serviços *web* são exemplos de alguns dos cenários discutidos no capítulo 3.

Nas subseções relacionadas à especificação do protótipo, serão apresentados diagramas *Unified Modeling Language* (UML), utilizados com o intuito de representar as principais características identificadas na análise de requisitos. São vários os diagramas padronizados pela UML, no entanto, neste trabalho, só serão utilizados três: o diagrama de casos de uso, o diagrama de classes e o diagrama de seqüência.

4.1.1 Diagrama de casos de uso

O diagrama de caso de uso tem como objetivo apresentar, de uma forma bastante simples, o comportamento de um sistema. A idéia de simplificar é fazer com que o sistema seja facilmente compreendido por qualquer pessoa que tenha participação direta ou indireta neste sistema.

Os diagramas de casos de uso se concentram, basicamente, nos seguintes elementos:

- atores – representam os usuários que, possivelmente, utilizarão os serviços e funções do sistema. Pode ser uma pessoa, um processo ou um outro sistema. No diagrama, são representados por *bonecos magros*;

- casos de uso – descrevem as funcionalidades e os serviços oferecidos pelo sistema. Permitem a documentação e a representação dos comportamentos previstos para as funções do sistema. No diagrama, os casos de uso são representados por elipses que possuem um texto indicando a sua funcionalidade;
- Associações – são responsáveis por indicar a interação existente entre os atores do sistema e os casos de uso ou ainda a interação entre estes com outros casos de uso.

Fig. 4.1 – Diagrama de Casos de Uso

A Fig. 4.1 apresenta os casos de uso envolvidos nas operações e nos serviços do protótipo. O caso de uso *Criar conta* permite que os usuários criem contas de acesso. Já o *Efetuar login*, como o nome sugere, possibilita que aqueles que possuem conta de acesso conectem-se ao protótipo. O *Solicitar nova senha* cria uma nova senha de acesso para um usuário requisitante, e o *Alterar senha* torna possível a troca de senha. O caso de uso *Atualizar cadastro* permite a criação ou alteração de dados, e o C*riar contexto* permite que informações de contexto possam ser manipuladas. O *Realizar pesquisa* efetua buscas na base de documentos à procura de recursos que atendam às necessidades dos usuários requisitantes.

Ainda na Fig. 4.1, em *Configurar perfil*, os usuários definem o seu perfil de pesquisa através da associação entre contexto e outras informações. Esse caso de uso é complementado por outros três: *Atualizar relacionamento*, *Atualizar sintagma* e *Atualizar fonte*. *Atualizar relacionamento* permite que o usuário faça uma relação entre um código – que é representado por uma sigla, por um acrônimo, por uma abreviatura ou por um símbolo – e o seu respectivo significado. Em *Atualizar sintagma*, o usuário define palavras e/ou frases associadas ao contexto selecionado. E, por fim, em *Atualizar fonte*, o usuário informa endereços que contêm informações que lhe são úteis.

Além de todos os casos de uso citados anteriormente, o ator Administrador também se relaciona com os seguintes: *Configurar parâmetros*, *Cadastrar fontes de recursos*, *Cadastrar administrador* e *Alterar nível de acesso*. O *Configurar parâmetros* permite que o Administrador determine parâmetros que configuram a execução de algumas funções do protótipo. *Cadastrar fontes de recursos* registra endereços que serão visitados pelo programa *crawler*. *Cadastrar administrador* possibilita que usuários com o nível de acesso administrativo tenham seus dados cadastrados e alterados. E o caso de uso *Alterar nível de acesso* permite que o Administrador mude o nível de acesso de usuário para administrador ou o contrário.

A figura também apresenta os casos de uso *Executar pesquisa* e *Importar contextos*, que estão disponíveis apenas para aplicações clientes que acessam o serviço *web* do protótipo. O primeiro permite que sua base de documentos possa ser consultada por uma aplicação que deseja utilizar esse serviço. Já o segundo possibilita que os contextos cadastrados por seus usuários possam ser disponibilizados para aplicações. E, por fim, o *Integrar contextos* torna possível que os usuários possam utilizar os contextos cadastrados por outros usuários do protótipo.

4.1.1.1 Documentação

Com o objetivo de descrever mais detalhadamente o funcionamento de cada caso de uso apresentado na Fig. 4.1, a seguir, é apresentada uma documentação que contém informações relacionadas aos casos de uso. As informações descritas consistem no nome do caso de uso, nos atores envolvidos, na finalidade, num breve resumo e no tipo. Além disso, é mostrada a seqüência de eventos envolvidos nos casos de uso.

Tabela 4.1 – Documentação do caso de uso Criar conta	
Nome:	**Criar conta**
Atores:	Usuário, Administrador
Finalidade:	Criar uma conta que permita o acesso ao protótipo.
Resumo:	O ator acessa a página de criação de conta de acesso, informa os dados necessários para o cadastro e uma conta de acesso é criada.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator acessa o sistema para criar uma nova conta de acesso.	
2. O ator informa os dados necessários para criação da conta.	3. Validar informações: a. se as informações necessárias estiverem corretas, mostrar a mensagem “Operação realizada com sucesso.”. b. em caso contrário, mostrar uma mensagem descrevendo o erro identificado.

Tabela 4.2 – Documentação do caso de uso *Efetuar login*	
Nome:	**Efetuar *login***
Atores:	Usuário, Administrador
Finalidade:	Acessar o protótipo.
Resumo:	O ator informa *login* e senha. O protótipo valida as informações e, se estiverem correras, libera o acesso.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator informa o *login* e a senha.	2. Validar login e senha: a. se o usuário está cadastrado, verificar a senha. a_1. se a senha estiver correta, permitir o acesso e só deixar disponíveis as operações associadas ao nível de acesso do usuário(que pode ser Usuário ou Administrador); a_2. se a senha estiver errada, mostrar a mensagem “Usuário e senha não conferem”. b. se o usuário não estiver cadastrado no sistema, mostrar a mensagem “Login inválido.”

Tabela 4.3 – Documentação do caso de uso *Solicitar nova senha*	
Nome:	**Solicitar nova senha**
Atores:	Usuário, Administrador
Finalidade:	Gerar uma nova senha de acesso.
Resumo:	O ator solicita uma nova senha. É gerada e enviada uma nova senha de acesso.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator informa o *e-mail* para o qual será enviada sua nova senha.	2. Verificar *e-mail* informado: a. se o *e-mail* estiver associado a algum usuário cadastrado, gerar um nova senha, enviá-la para o *e-mail* informado e mostrar a mensagem "A senha foi enviada para o seu *e-mail*." b. caso contrário, mostrar a mensagem "Usuário não localizado".

Tabela 4.4 – Documentação do caso de uso *Alterar senha*	
Nome:	**Alterar senha**
Atores:	Usuário, Administrador
Finalidade:	Substituir a senha atual por uma nova senha.
Resumo:	O ator informa seu *login*, senha atual de acesso e a nova senha. A antiga senha é substituída.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando o ator informa o *login*, a senha atual e a nova senha.	2. Verificar usuário e senha: a. se usuário e senha conferem: a_1. se a nova senha é diferente da senha atual: alterar a senha para a senha sugerida e mostrar a mensagem "Senha alterada com sucesso". a_2. se a senha atual é igual a nova senha, mostrar a mensagem "A nova senha não pode ser igual a atual." b. se usuário e senha não conferem, mostrar mensagem "Usuário e senha não conferem".

Tabela 4.5 – Documentação do caso de uso *Atualizar usuário*	
Nome:	**Atualizar usuário**
Atores:	Usuário, Administrador
Finalidade:	Inserir ou alterar dados do cadastro.
Resumo:	Permite que o ator altere ou insira os seus dados.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando o ator acessa a página de atualização de usuário.	2. Verificar o usuário conectado: a. se o nível de acesso for de Usuário, mostrar todos os dados relacionados àquele

	usuário b. se o nível de acesso for de Administrador, mostrar os dados relacionados ao *login* informado na consulta.
3. Informar os dados necessários para inclusão ou alteração dos dados.	4. Validar dados informados: a. se os dados necessários estiverem corretos, atualizar cadastro e mostrar mensagem "Operação realizada com sucesso." b. se pelo menos um dos dados necessários não for validado com sucesso, mostrar mensagem informando o motivo da invalidação.

Tabela 4.6 – Documentação do caso de uso *Criar contexto*	
Nome:	**Criar contexto**
Atores:	Usuário, Administrador
Finalidade:	Criar e alterar contextos de pesquisa
Resumo:	Construir contextos de pesquisa que, mais tarde, serão relacionados ao perfil do usuário.
Tipo:	Primário e essencial
Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator cria um contexto de pesquisa.	2. Verifica se já existe um contexto cadastrado com o nome informado para o usuário em questão: a. se já existir, mostrar a mensagem "Contexto já cadastrado." b. se não existir, atualizar cadastro do contexto e mostrar a mensagem "Operação realizada com sucesso."

Tabela 4.7 – Documentação dos casos de uso *Configurar perfil*, *Atualizar relacionamento*, *Atualizar sintagma* e *Atualizar fonte*.	
Nome:	**Configurar perfil**
Atores:	Usuário, Administrador
Finalidade:	Construir um perfil de pesquisa através do relacionamento de contextos com outras informações.
Resumo:	O ator configura um perfil de pesquisa baseado em informações de sintagmas, relacionamentos e fontes de recursos que contenham informações pertinentes ao contexto configurado.
Tipo:	Primário e essencial
Seção:	Principal
Seqüência típica de eventos	
Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator acessa a página de configuração de perfil.	
2. O ator seleciona um contexto.	
3. Caso deseje, o ator pode associar algumas	

informações relacionadas a este contexto: a. Relacionamentos, ver seção *Atualizar relacionamento* b. Sintagmas, ver seção *Atualizar sintagma* c. Fontes de recursos, ver seção *Atualizar Fontes de recursos*	
	4. Relaciona as informações com o contexto e mostra a mensagem "Operação realizada com sucesso."
Seção: Atualizar relacionamento	
Seqüência típica de eventos	
Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator informa um símbolo, sigla, acrônimo ou abreviatura associando à sua respectiva descrição.	2. Validar informações: a. se já existirem um código e uma descrição iguais aos digitados, mostrar a mensagem "Relacionamento já cadastrado." b. se as duas informações forem informadas, atualizar o cadastro de relacionamentos e mostrar a mensagem "Operação realizada com sucesso." c. se pelo menos uma destas duas informações for omitida, mostrar uma mensagem solicitando o preenchimento da informação omitida.
Seção: Atualizar sintagma	
Seqüência típica de eventos	
Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator informa um sintagma referente ao contexto selecionado na página de configuração de perfil.	2. Validar sintagma digitado: a. se existir um sintagma com o mesmo nome para o digitado, mostrar a mensagem "Sintagma já cadastrado." b. se não existir, atualizar o cadastro de sintagmas e mostrar a mensagem "Operação realizada com sucesso".
Seção: Atualizar fontes de recursos	
Seqüência típica de eventos	
Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o ator informa uma fonte de recurso na qual se pode obter informações relacionadas ao contexto selecionado.	2. Validar fonte de recurso digitada: a. se existir uma fonte de recurso cadastrada, mostrar a mensagem "Fonte de recurso já cadastrada." b. se não existir, atualizar o cadastro de fontes de recurso e mostrar a mensagem "Operação realizada com sucesso".

Tabela 4.8 – Documentação do caso de uso *Configurar parâmetros*

Nome:	**Configurar parâmetros**
Atores:	Administrador
Finalidade:	Configurar os parâmetros do sistema

Resumo:	Informar ao protótipo os parâmetros que devem ser utilizados durante a sua operacionalização.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o Administrador informa os parâmetros de configuração.	2. Verificar parâmetros informados: a. se algum parâmetro não tiver sido informado, mostrar uma mensagem informando que o parâmetro omitido deve ser informado.

Tabela 4.9 – Documentação do caso de uso *Cadastrar fontes de recursos*

Nome:	**Cadastrar fontes de recurso**
Atores:	Administrador
Finalidade:	Informar endereços onde podem ser localizados recursos.
Resumo:	O Administrador informa endereços onde podem ser localizadas informações relevantes.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. Este caso de uso começa quando o Administrador informa endereços onde podem ser localizados recursos.	2. Verificar fonte de recurso informada: a. se existir uma fonte de recurso cadastrada com o mesmo nome, mostrar a mensagem “Fonte de recurso já cadastrada.” b. se não existir, atualizar o cadastro e mostrar a mensagem “Operação realizada com sucesso”.

Tabela 4.10 – Documentação do caso de uso *Cadastrar administrador*

Nome:	**Cadastrar administrador**
Atores:	Administrador
Finalidade:	Criar um usuário com nível de acesso de administrador.
Resumo:	O Administrador informa dados necessários para criar um usuário com nível de acesso de administrador.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando o Administrador acessa a página de cadastro de administrador.	
2. Informar os dados necessários para inclusão ou alteração do cadastro.	3. Validar dados informados: a. se os dados necessários estiverem corretos, atualizar cadastro e mostrar mensagem “Operação realizada com sucesso.” b. se pelo menos um dos dados necessários não for validado com sucesso, mostrar mensagem informando o motivo da invalidação.

Tabela 4.11 – Documentação do caso de uso *Alterar nível de acesso*	
Nome:	**Alterar nível de acesso**
Atores:	Administrador
Finalidade:	Mudar o tipo de acesso do usuário.
Resumo:	O Administrador informa o *login* do usuário que terá seu nível de acesso alterado.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando o Administrador acessa a página de alteração de nível de acesso.	
2. Informar o *login* do usuário que terá seu acesso alterado.	3. Validar *login* informado: a. se o *login* existir, ler os dados relacionados àquele *login* e mostrar na página. b. em caso contrário, mostrar a mensagem “Login inválido”.
4. Se *login* válido, o Administrador informa o novo tipo de acesso do usuário selecionado.	5. Atualizar o novo nível de acesso do usuário e mostrar mensagem “Operação realizada com sucesso.”

Seqüência alternativa:
Linha 4: Administrador informa um novo *login*.

Tabela 4.12 – Documentação do caso de uso *Realizar pesquisa*	
Nome:	**Realizar pesquisa**
Atores:	Usuário, Administrador
Finalidade:	Pesquisar documentos.
Resumo:	O ator formula uma consulta, envia a consulta para o motor de busca. Os resultados são localizados e finalmente retornados para o usuário.
Tipo:	Primário e essencial

Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando o ator formula uma consulta e a envia para o protótipo.	2. Verificar o tipo da pesquisa: se for selecionado algum contexto: a. identificar relacionamentos existentes; b. identificar sintagmas existentes. c. adicionar informações à consulta original.
	3. Formata a consulta e envia para o mecanismo de busca.
	4. Efetua buscas para localizar os documentos que têm maior relevância para a consulta.
	5. Ordena os documentos localizados por ordem decrescente de relevância.
6. Analisa os resultados obtidos: a. se desejar abrir um documento listado na página atual, clica no *link* e analisa com	

mais detalhe o documento; b. se na página atual nenhum documento for relevante, clicar no botão “próximo” e analisar os outros resultados. Persistindo a irrelevância dos novos resultados, clicar no botão “próximo” até encontrar resultados relevantes ou refazer a consulta.	

Tabela 4.13 – Documentação do caso de uso *Integrar contextos*	
Nome:	**Integrar contextos**
Atores:	Usuário, Administrador
Finalidade:	Solicitar a listagem dos perfis definidos por usuários
Resumo:	A Aplicação requisita ao protótipo a listagem de todos os perfis cadastrados por seus usuários. O protótipo gera esta listagem e envia para a Aplicação.
Tipo:	Primário e essencial
Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando o ator escolhe o contexto que deseja adquirir.	2. Mostrar as informações relacionadas ao contexto selecionado.
3. Confirma o processo de integração.	4. Executar a integração e mostrar a mensagem “Operação realizada com sucesso”.

Tabela 4.14 – Documentação do caso de uso *Executar pesquisa*	
Nome:	**Executar pesquisa**
Ator:	Aplicação
Finalidade:	Disponibilizar o serviço de pesquisa para aplicações clientes através de um *web service*.
Resumo:	A Aplicação se conecta ao *web service* e envia os parâmetros da consulta. O *web service* executa a consulta e retorna o resultado da pesquisa no formato de um arquivo XML.
Tipo:	Primário e essencial
Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando a Aplicação envia os parâmetros da consulta para o *web service*.	2. Efetua buscas para localizar os documentos que têm maior relevância para a consulta.
	3. Ordena os documentos localizados por ordem decrescente de relevância.
4. Processa o retorno recebido.	

Tabela 4.15 – Documentação do caso de uso *Importar contextos*	
Nome:	**Importar contextos**
Ator:	Aplicação
Finalidade:	Disponibilizar os contextos de usuários cadastrados através de um *web service* para as aplicações clientes.
Resumo:	A Aplicação se conecta ao *web service*, informando-lhe o seu nome e recebe o retorno no formato de um arquivo XML.
Tipo:	Primário e essencial
Ação do ator	**Resposta do sistema**
1. O caso de uso começa quando a	2. Retorna uma lista de informações

Aplicação informa o seu nome ao *web service*.	relacionadas ao contexto que podem ser: nome do contexto, relacionamentos, sintagmas ou fontes de recurso.
3 Processa o retorno recebido.	

4.1.2 Diagrama de classes

O diagrama de classes é mais um dos diagramas utilizados pelo padrão UML. Tem como objetivo apresentar as classes que serão manipuladas pelo sistema, seus atributos e métodos, a forma como elas se relacionam, as associações existentes entre elas e as mensagens trocadas entre si.

A seguir, na Fig 4.2, com o objetivo de apresentar os métodos e atributos necessários para a implementação do protótipo, são apresentadas as principais classes envolvidas.

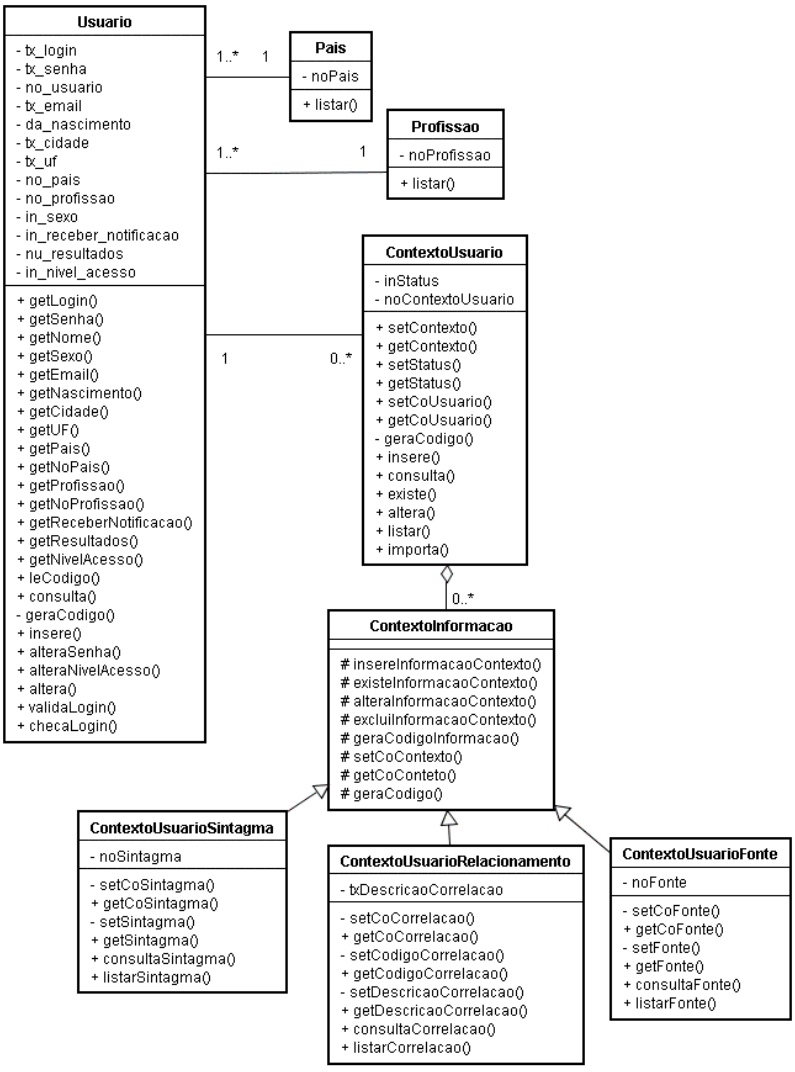

Fig. 4.2 – Diagrama de Classes

A classe *Pais* controla informações relacionadas aos países cadastrados no protótipo. Analisando a figura, percebe-se que uma instância dessa classe pode estar relacionada a uma ou a inúmeras instâncias da classe *Usuario*; o mesmo acontece com a *Profissao* – controladora das informações relacionadas às profissões cadastradas – que pode ter uma instância associada a uma ou a várias da classe *Usuario*.

Uma instância da classe *Usuario*, por sua vez, pode ter nenhuma, uma ou várias da *ContextoUsuario* relacionadas a ela (multiplicidade 0..*), porém uma instância de *UsuarioContexto* está associada única e exclusivamente a uma de *Usuario*.

Ainda em relação à Fig. 4.2, uma instância de *ContextoUsuario* pode estar relacionada a nenhuma, a uma ou a várias de *ContextoInformacao*. A *ContextoInformacao* foi criada para que as classes relacionadas aos contextos de pesquisa do usuário (*ContextoUsuariorRelacionamento, ContextoUsuarioSintagma* e *ContextoUsuarioFonte*) possam herdar seus métodos.

Uma instância da classe *ContextoUsuarioRelacionamento* está associada a uma única de *ContextoInformacao*. A *ContextoUsuarioRelacionamento* controla as informações de relacionamento definidas pelo usuário para um contexto.

Uma classe *ContextoUsuarioSintagma* está associada a uma única instância de *ContextoInformacao*, enquanto esta pode instanciar vários objetos de *ContextoUsuarioSintagma*. Esta controla os sintagmas que estão relacionados ao contexto.

Uma classe *ContextoUsuarioFonte* está associada a uma única instância de *ContextoInformacao*, enquanto *esta* pode instanciar vários objetos daquela, a qual controla as fontes de recursos definidas pelo usuário relacionadas ao contexto.

A associação existente entre as classes *ContextoUsuario*/*ContextoInformacao* é do tipo agregação, a qual indica que as informações de um objeto-todo (classe localizada ao lado do losango) precisam ser complementadas por objetos-parte de uma ou mais classes.

4.1.3 Diagrama de seqüência

Este diagrama procura apresentar as condições que devem ser satisfeitas e os métodos que devem ser disparados entre os objetos envolvidos, além da ordem de execução desses eventos. Em outras palavras, ele mostra a interação entre os objetos ao longo do tempo e apresenta aqueles que participam da interação e a seqüência de mensagens trocadas

A seguir, são apresentados os diagramas de seqüência associados aos casos de uso analisados na seção 4.1.1:

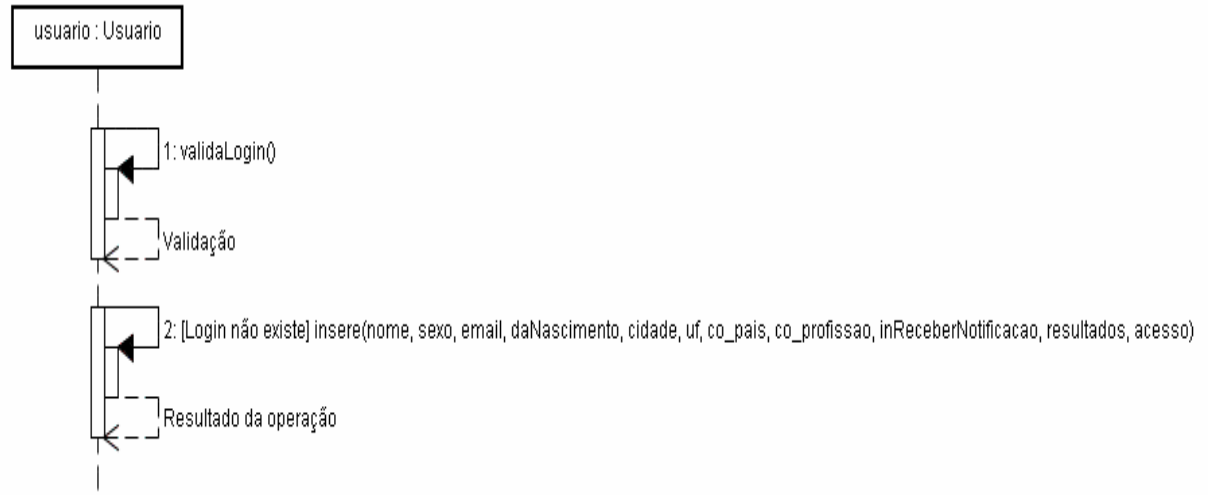

Fig. 4.3 – Diagrama de seqüência representando o caso de uso *Criar conta*

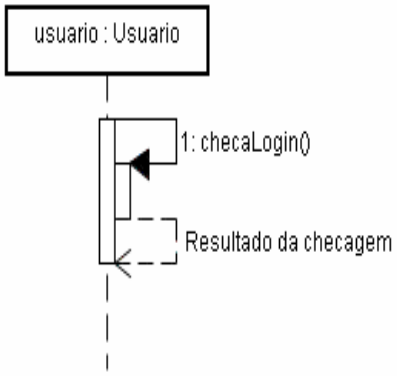

Fig. 4.4 – Diagrama de seqüência representando o caso de uso *Efetuar login*

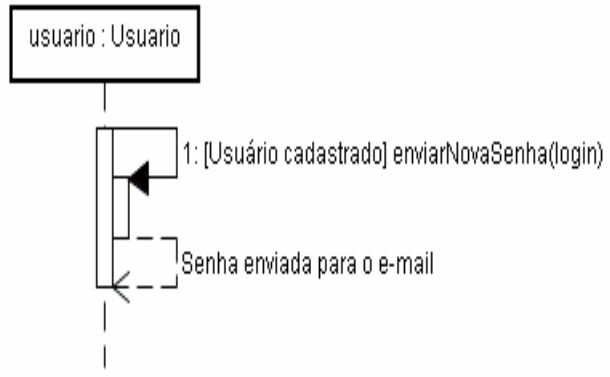

Fig. 4.5 – Diagrama de seqüência representando o caso de uso *Solicitar nova senha*

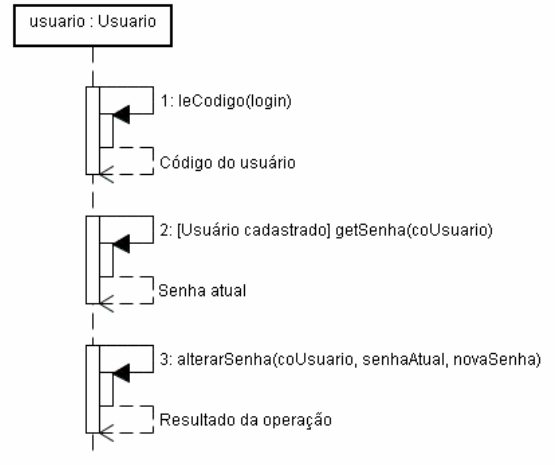

Fig. 4.6 – Diagrama de seqüência representando o caso de uso *Alterar senha*

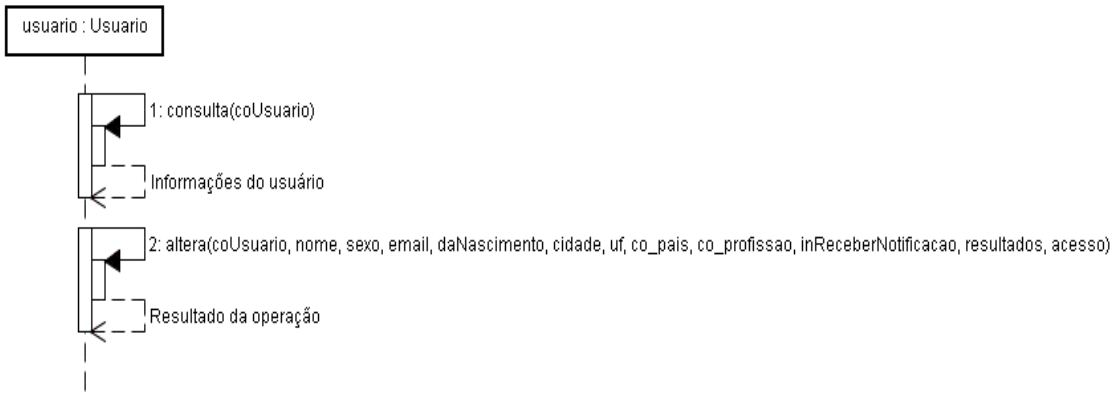

Fig. 4.7 – Diagrama de seqüência representando o caso de uso *Atualizar cadastro*

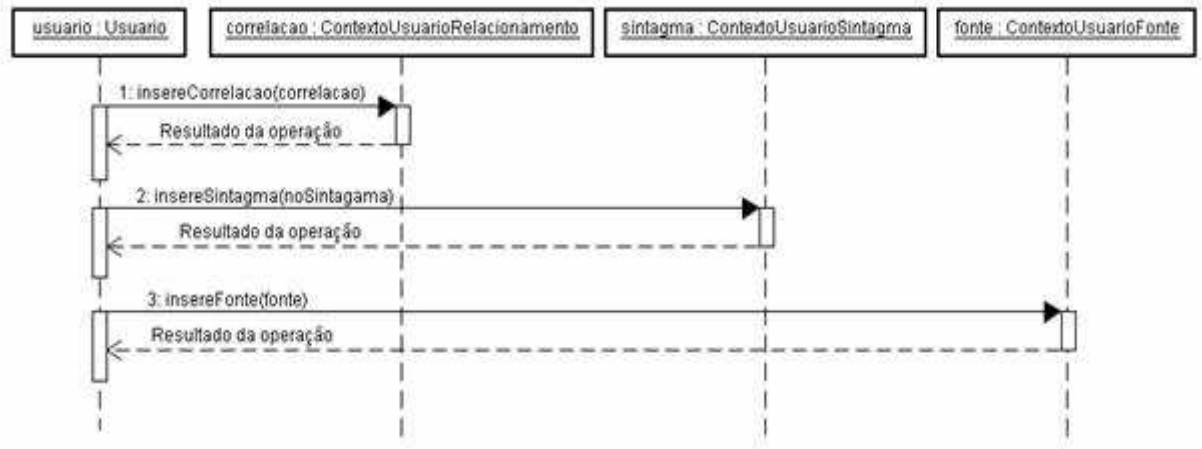

Fig. 4.8 – Diagrama de seqüência representando o caso de uso *Configurar perfil*

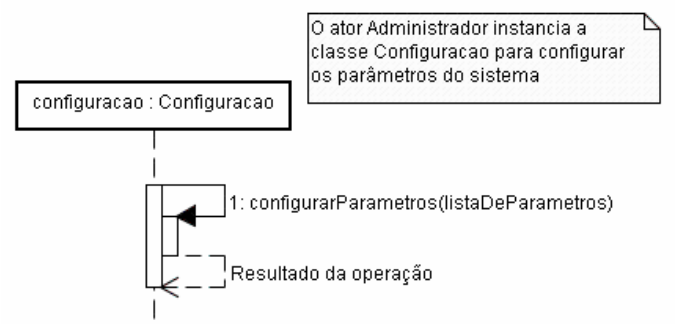

Fig. 4.9 – Diagrama de seqüência representando o caso de uso *Configurar parâmetros*

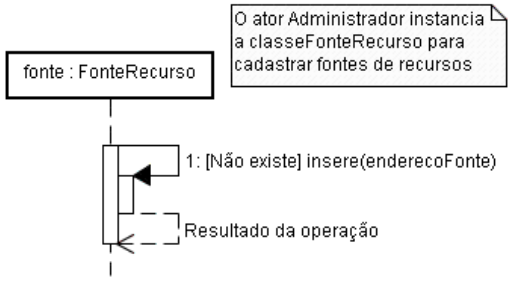

Fig. 4.10 – Diagrama de seqüência representando o caso de uso *Cadastrar fontes de recurso*

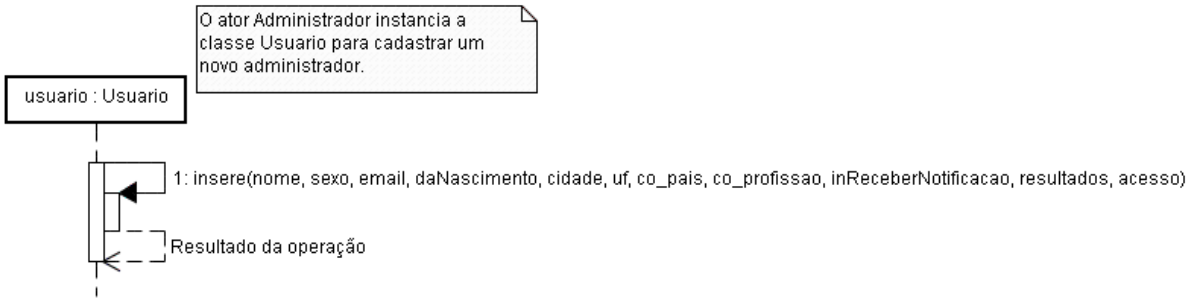

Fig. 4.11 – Diagrama de seqüência representando o caso de uso *Cadastrar administrador*

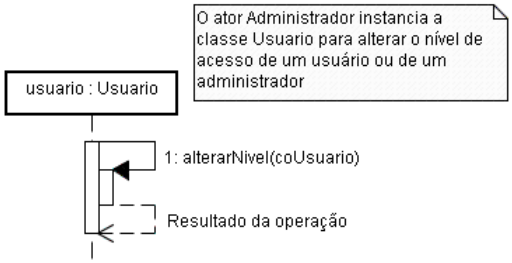

Fig. 4.12 – Diagrama de seqüência representando o caso de uso *Alterar nível de acesso*

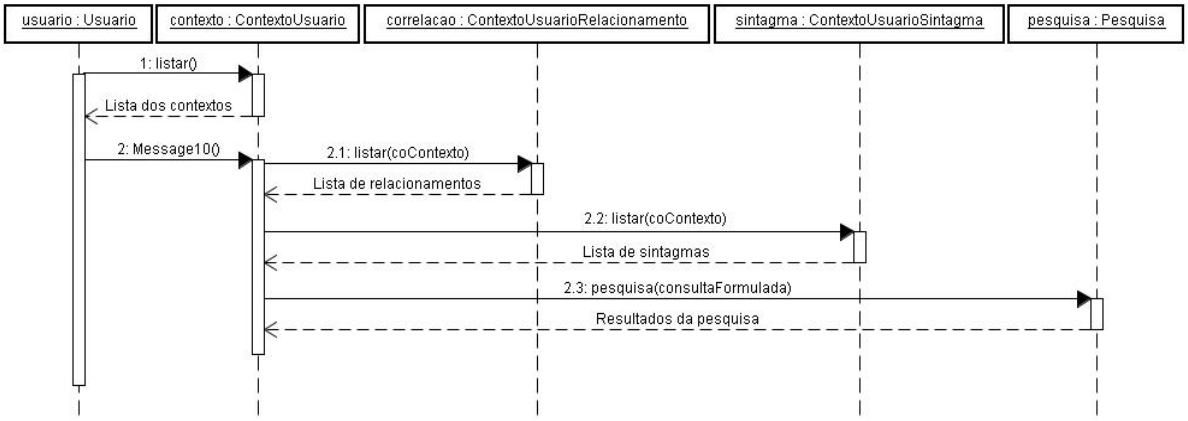

Fig. 4.13 – Diagrama de seqüência representando o caso de uso *Realizar pesquisa*

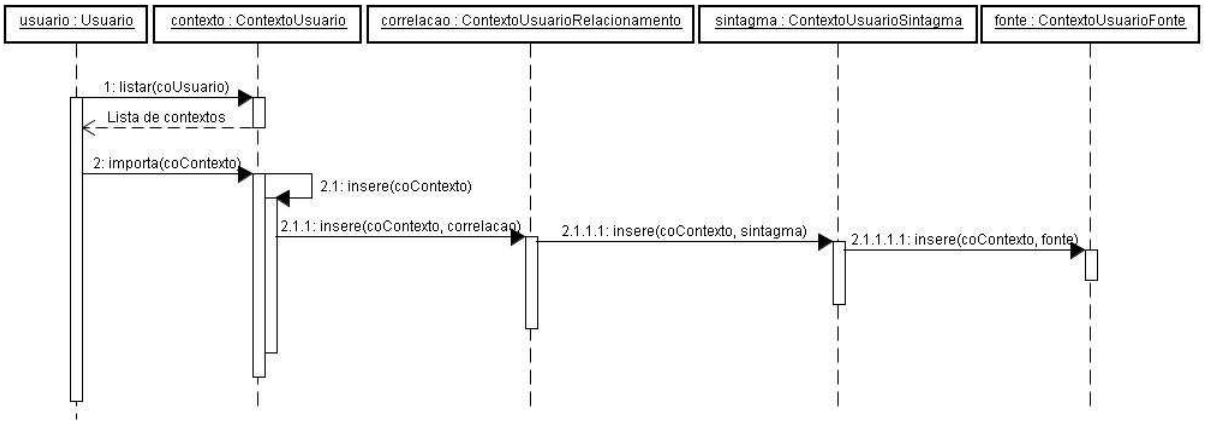

Fig. 4.14 – Diagrama de seqüência representando o caso de uso *Integrar contextos*

4.2 IMPLEMENTAÇÃO

Depois da análise de requisitos, realizada no capítulo 3, e da especificação do protótipo, apresentada na seção anterior, serão mostrados, nas próximas subseções, os aspectos utilizados na implementação.

4.2.1 Protótipo

Na implementação do protótipo, foram utilizados alguns elementos que contribuíram com a operacionalização de algumas funcionalidades, conforme mostrado na Fig. 4.15:

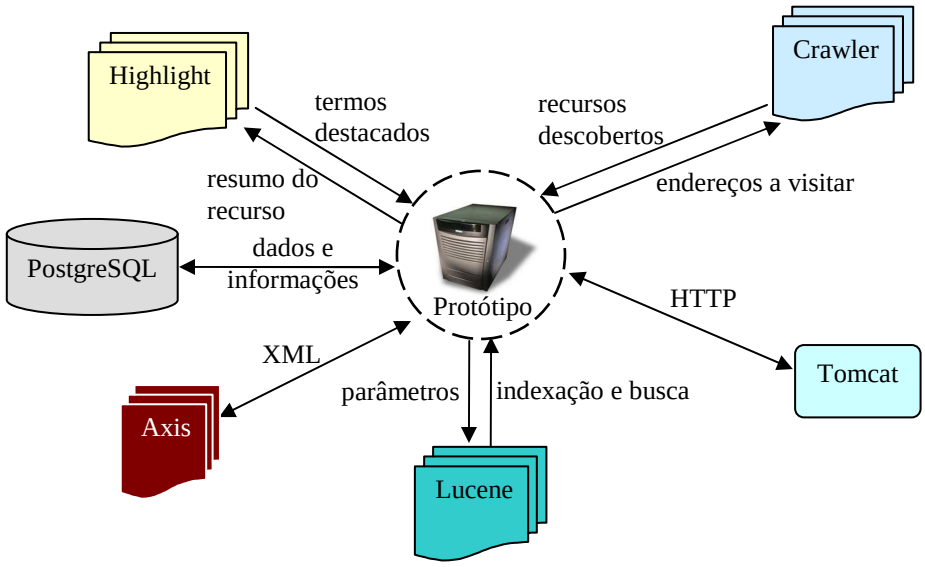

Fig. 4.15 – Relação entre o protótipo e os elementos utilizados na implementação

A biblioteca *Crawler* [52], desenvolvida por Lars Torunski, realiza a tarefa de descobrir recursos, descrita no capítulo 2 (Fundamentação Teórica). A partir dos endereços informados pelo protótipo, a biblioteca percorre a *Web* em busca de novos endereços e, assim que novos endereços são descobertos, estes são enviados para a rotina de indexação do protótipo.

Lucene [53] é uma biblioteca que, baseada nos parâmetros informados para execução, permite a realização das tarefas de indexação e de busca de recursos. As tarefas indexação e busca de recursos foram discutidas no capítulo 2 nas seções Indexação e Pesquisa, respectivamente.

A biblioteca *Highlight* [54] foi desenvolvida para dar suporte à *Lucene*. Ela, normalmente, é utilizada para extrair e destacar os trechos considerados mais relevantes para uma pesquisa requisitada.

O *Axis*, um *framework* de código aberto desenvolvido na linguagem *Java* e baseado no padrão XML, é utilizado para a construção de *web service* no padrão SOAP.

Para o armazenamento dos dados, é utilizado o *PostgreSQL* [55], e, para o gerenciamento do tráfego de mensagens HTTP, é utilizado o *container web* do projeto *Apache*, *Tomcat* [56].

A Listagem 4.1 apresenta trechos da classe *Crawler*, baseada na biblioteca de mesmo nome, que é responsável por realizar a tarefa de descoberta de recursos:

```
package br.com.crawler;
...
import com.torunski.crawler.MultiThreadedCrawler;
import com.torunski.crawler.filter.ServerFilter;
import com.torunski.crawler.model.MaxIterationsModel;
import com.torunski.crawler.parser.httpclient.SimpleHttpClientParser;
...
public class Crawler
{
 ...
 public void efetuaVarredura(String url)
 {
  //Configuração do crawler
  MultiThreadedCrawler crawler = new MultiThreadedCrawler(8, 1);
  crawler.setLinkFilter(new ServerFilter(url));
  crawler.setModel(new MaxIterationsModel(30));
  crawler.start(url, "/");
  ...
  //Configura o proxy
  SimpleHttpClientParser s = new SimpleHttpClientParser();
  s.setProxy(nomeProxy, portaProxy);

  //Carrega os links descobertos e visitados pelo crawler
  Collection visitedLinks = crawler.getModel().getVisitedURIs();
  Iterator list = visitedLinks.iterator();
  while (list.hasNext())
  {
   //Armazena os endereços descobertos na base de dados
   PaginaIndexada pag = new PaginaIndexada();
   String erro = pag.insere(list.next().toString());
  }

  //Carrega os links descobertos, mas não visitados pelo crawler
  Collection notVisitedLinks = crawler.getModel().getToVisitURIs();
  Iterator listNot = notVisitedLinks.iterator();
  while (listNot.hasNext())
  {
   //Adiciona links encontrados na tabela tb_pagina_indexada
   PaginaIndexada p = new PaginaIndexada();
   String erro = p.insere(listNot.next().toString());
  }
 }
...
}
```

Listagem 4.1 – Trecho da rotina de descoberta de recursos na Web

O método *efetuaVarredura* executa, de fato, a tarefa de descoberta de recursos. Nele são atribuídos os parâmetros de configuração do *crawler* e é feita a recuperação dos recursos que serão indexados.

Na Listagem 4.2, é apresentado um trecho da classe *Indexacao*. Esta utiliza a biblioteca *Lucene* e é responsável por armazenar, na base de dados e na base de índices, os recursos que foram descobertos pelo método *efetuaVarredura* da classe *Crawler*:

```
package br.com.indexacao;
...
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.Term;
import org.apache.lucene.index.TermDocs;
import org.apache.lucene.document.Field.Store;
import org.apache.lucene.index.IndexModifier;
...
public class Indexacao
{
 ...
 try
 {
  int indiceApagados;
  Integer coPagina = new Integer(co_pagina);

  //Insere conteúdo da página na base de dados
  PaginaIndexada p = new PaginaIndexada();
  p.insereConteudo(co_pagina, titulo, conteudo, html);

  //Deve efetuar a reindeixação => apagar índice existente
  if (reindexar)
   indiceApagados = index.deleteDocuments(new Term("co_pagina",
                                                   coPagina.toString()));

  //Efetua a indexação das páginas
  Document doc = new Document();
  doc.add(new Field("co_pagina", coPagina.toString(),
                    Field.Store.YES, Field.Index.UN_TOKENIZED));
  doc.add(new Field("endereco", url, Field.Store.YES,
                    Field.Index.TOKENIZED));
  doc.add(new Field("titulo", titulo, Field.Store.YES,
                    Field.Index.TOKENIZED));
  doc.add(new Field("conteudo", conteudo, Field.Store.YES,
                    Field.Index.TOKENIZED));
  index.addDocument(doc);
 }
 catch(IOException e)
 {
  System.err.println("Erro: " + e.getMessage());
 }
...
}
```

Listagem 4.2 – Trecho da rotina de indexação de recursos

A Listagem 4.3 apresenta trechos do método que realiza a pesquisa de recursos:

```
...
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.search.Hits;
import org.apache.lucene.search.highlight.Highlighter;
...
//Formata string para executar a consulta
String stringQuery = ("(conteudo:").concat(consultaFormatada);
stringQuery = stringQuery.trim();
stringQuery = stringQuery.concat(") OR ");
stringQuery = stringQuery.concat("(titulo:");
stringQuery = stringQuery.concat(consultaFormatada);
stringQuery = stringQuery.trim();
stringQuery = stringQuery.concat(")");

QueryParser queryParser = new QueryParser("conteudo", analyzer);
//procurar pelo valor armazenado em conteudo e titulo
Query query = queryParser.parse(stringQuery);
Hits hits = searcher.search(query);
...
//Lista os recursos que estão entre resultadoInicial e resultadoFinal
for (int i = resultadoInicial; i < resultadoFinal; i++)
{
 if (i < hits.length())
 {
  //Recupera o conteúdo da busca
  Highlighter highlighter = new Highlighter(new QueryScorer(query));

  String conteudo = hits.doc(i).get("conteudo");
  TokenStream tokenStream = analyzer.tokenStream("conteudo", new
                                                 StringReader(conteudo));

  //Pega os 3 fragmentos mais relevantes para a consulta
  String conteudoHighlighter = highlighter.getBestFragments(tokenStream,
                               conteudo, 3, "...");

  String titulo = hits.doc(i).get("titulo");
  tokenStream=analyzer.tokenStream("titulo", new StringReader(titulo));
  String tituloHighlighter =  highlighter.getBestFragment(tokenStream, titulo);
  if (tituloHighlighter == null)
   tituloHighlighter = titulo;

  Pesquisa pesq = new Pesquisa();
  pesq.setTitulo(tituloHighlighter);
  pesq.setEndereco(hits.doc(i).get("endereco"));
  pesq.setConteudo(conteudoHighlighter);
  pesq.setNuResultados(hits.length());

  result.add(pesq);
  return result;
 }
...
```

Listagem 4.3 – Trecho do método que retorna resultados encontrados na pesquisa

No trecho anterior, foi mostrado como a consulta é iniciada e realizada, como o conteúdo dos recursos é recuperado e como os termos mais relevantes desses conteúdos são destacados através do uso da classe *Highlight*.

Na Listagem 4.4, são mostrados trechos relacionados à criação do perfil de pesquisa de um usuário. Os trechos fazem referência à inserção das informações de contexto, dos sintagmas e dos relacionamentos em suas respectivas tabelas na base de dados.

```
//Trecho que insere dados na tabela de contextos
...
coContexto = geraCodigo();

cmd = "INSERT INTO tb_contexto_usuario(co_contexto_usuario,
                   no_contexto_usuario, in_status, co_usuario)" +
      "VALUES (" + coContexto + ", '" + noContexto + "', " + inStatus +
                "," + coUsuario + ")";

s.executeUpdate(cmd);
...


//Trecho que insere dados na tabela de correlações
...
coCorrelacao = geraCodigo(coContexto);

cmd = "INSERT INTO tb_contexto_usuario_correlacao(co_contexto_usuario,
                   co_correlacao, tx_codigo_correlacao,
                   tx_descricao_correlacao)" +
      "VALUES (" + coContexto + ", " + coCorrelacao + ", '" +
                txCodigoCorrelacao + "', '" + txDescricaoCorrelacao +
                "')";

s.executeUpdate(cmd);
...


//Trecho que insere dados na tabela de sintagmas
...
coSintagma = geraCodigo(coContexto);
cmd = "INSERT INTO tb_contexto_usuario_sintagma(co_contexto_usuario,
                   co_sintagma, no_sintagma) " +
"VALUES (" + coContexto + ", " + coSintagma + ", '" + noSintagma + "')";

s.executeUpdate(cmd);
...
```

Listagem 4.4 – Trechos que criam e relacionam informações aos contextos

4.2.2 Rotina de indexação

Para agilizar a procura de informações textuais contidas em diversos recursos, é necessário, a princípio, que os textos estejam indexados e convertidos em um formato que

possibilite a localização rápida, evitando, assim, o processo de pesquisa seqüencial no conteúdo do arquivo.

A Fig. 4.16 é um esquema que representa o funcionamento da rotina de indexação executada no protótipo:

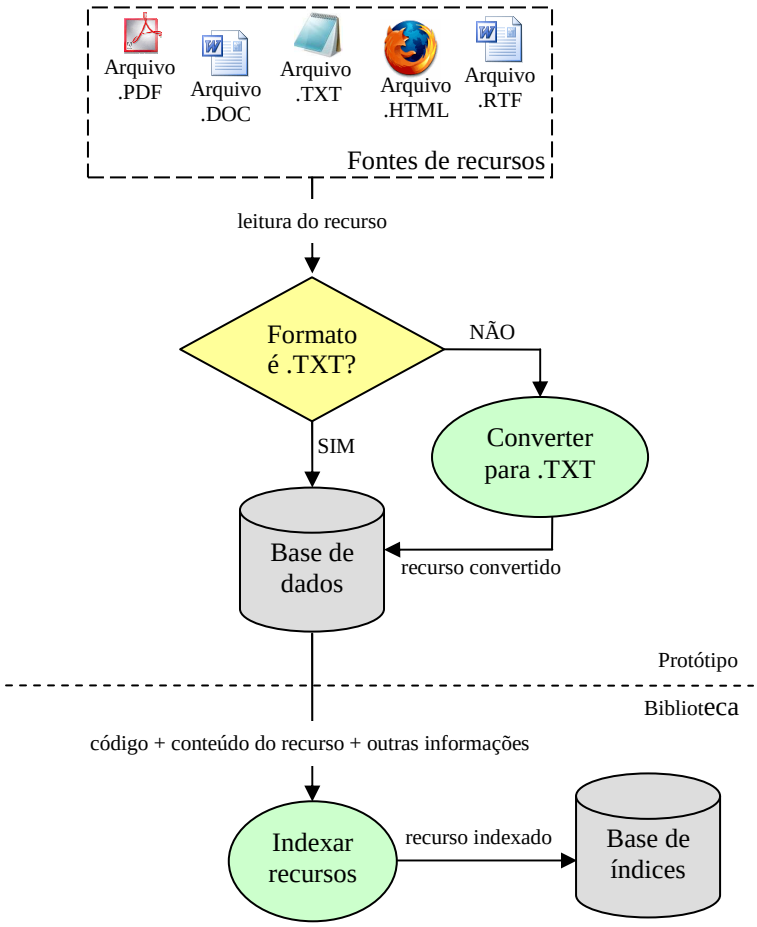

Fig. 4.16 – Esquema de indexação de recursos

A figura mostra que a rotina é dividida em duas camadas: a do protótipo e a da biblioteca. A primeira contempla os processos executados pelo protótipo, enquanto a segunda é gerenciada pela biblioteca *Lucene*.

O processo é iniciado quando o protótipo obtém os diversos recursos disponíveis para indexação. A figura mostra que estes se encontram no formato DOC, PDF, TXT, RTF e HTML.

Depois de obter documentos da fonte de dados, a rotina verifica o formato do recurso a ser indexado. Se o formato não for TXT, o recurso é convertido para esse formato. No momento em que seu formato torna-se o esperado, o recurso tem o seu conteúdo armazenado na base de dados.

Quando o conteúdo do documento é armazenado no banco de dados, gera-se um código identificador. Este é enviado em conjunto com o conteúdo do recurso, além de outras informações relacionadas a ele.

O conteúdo do arquivo recebido é, então, associado ao código identificador e às informações relacionadas. Feito isso, ele é analisado e, finalmente, indexado e armazenado em uma base de índices.

4.2.3 Rotina de pesquisa

Pesquisa é o processo que consiste em procurar palavras em um índice a fim de encontrar documentos nos quais elas apareçam. Normalmente, a qualidade de uma pesquisa é descrita através de dois índices de avaliação [57]: precisão e revocação. A precisão é a razão entre os documentos relevantes e os documentos recuperados; a revocação, entre os documentos relevantes recuperados e os documentos relevantes [35].

A Fig. 4.17 é um esquema que representa o funcionamento da rotina de uma pesquisa realizada por um usuário. O processo é iniciado quando o usuário formula a sua consulta e indica a forma de pesquisa que será realizada na busca (a pesquisa pode ser baseada em contextos ou apenas em palavras-chave e/ou frases). Se o usuário não configurar a consulta para utilizar contextos, ela será simplesmente formatada de acordo com a sintaxe processada pela biblioteca de consulta. Contudo, se ele resolver usar contextos para pesquisar recursos, será necessário que sua consulta original tenha um tratamento diferente. O protótipo deverá:

a) identificar os contextos que foram selecionados no momento da formulação da consulta;
b) selecionar (caso existam) os sintagmas, as siglas, os acrônimos e os símbolos associados aos contextos utilizados, com base nos contextos identificados no passo anterior;
c) concatenar a consulta original, depois de obtidas as informações relacionadas aos contextos selecionados, com essas informações e, finalmente, formatá-la de acordo com a sintaxe processada pela biblioteca *Lucene*.

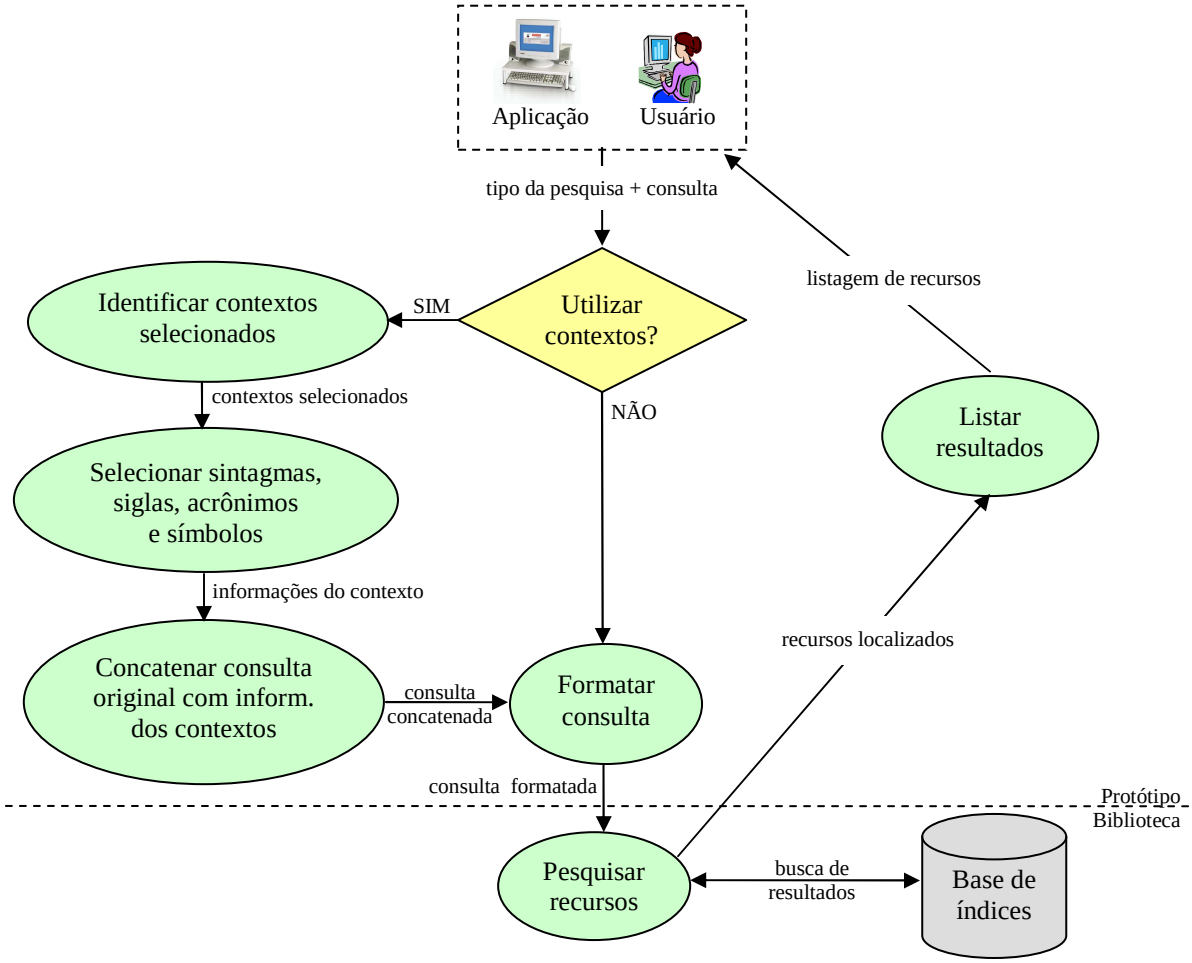

Fig. 4.17 – Esquema de pesquisa de recursos

Depois que a consulta é formatada, ela é enviada para a rotina, que executa a busca na base de índices à procura de recursos que atendam às suas condições. Por fim, caso existam resultados que atendam aos requisitos da consulta, é gerada e enviada uma listagem contendo os recursos localizados.

```
StringTokenizer token = new StringTokenizer(hdnContexto, "#");
while (token.hasMoreTokens())
{
 coContexto = token.nextToken();
 ContextoUsuarioSintagma s = new ContextoUsuarioSintagma();
 ArrayList lista = s.listar(Integer.parseInt(coContexto));
 for (int i = 0; i < lista.size(); i++)
 {
  ContextoUsuarioSintagma sint = (ContextoUsuarioSintagma)lista.get(i);
  sintagma = sintagma.concat(sint.getSintagma());
  sintagma = sintagma.concat(" ");
 }
 sintagma = (" ").concat(sintagma);
}
```

Listagem 4.5 – Trecho da rotina que identifica os sintagmas associados ao(s) contexto(s)

A Listagem 4.5, mostrada anteriormente, é um pequeno trecho da rotina que faz a concatenação dos sintagmas associados aos contextos selecionados com a pesquisa original formulada pelo usuário.

A Listagem 4.6 apresenta um trecho de uma rotina do protótipo que formata as consultas recebidas e as converte na sintaxe requerida para execução:

```
consultaUsuario = consultaUsuario.toLowerCase();
String consultaFormatada = "";
int nuAspas = 0;

/*Desmembra a consulta digitada pelo usuário*/
StringTokenizer token = new StringTokenizer(consultaUsuario, " ");
String palavra = null;
while (token.hasMoreTokens())
{
  ...
  if (!v.contains(palavra)) //Se a palavra encontrada não for STOPWORD
  {
    if (nuAspas == 0)
    {
      consultaFormatada = consultaFormatada.concat(palavra);
      consultaFormatada = consultaFormatada.concat(" AND ");
    }
    else
    {
      consultaFormatada = consultaFormatada.concat(palavra);
      consultaFormatada = consultaFormatada.concat(" ");
    }
    if (nuAspas == 2)
    {
      consultaFormatada = consultaFormatada.trim();
      consultaFormatada = consultaFormatada.concat(" AND ");
      nuAspas = 0;
    }
  }
}
consultaFormatada = consultaFormatada.substring(0, consultaFormatada.lastIndexOf("AND"));
...
return consultaFormatada;
```

Listagem 4.6 – Trecho da rotina que formata consultas

A Fig 4.18 é um fluxograma que complementa a ilustração da Fig 4.17. A primeira apresenta o processo de pesquisa de uma forma mais detalhada, abordando passos não apresentados na Fig. 4.17.

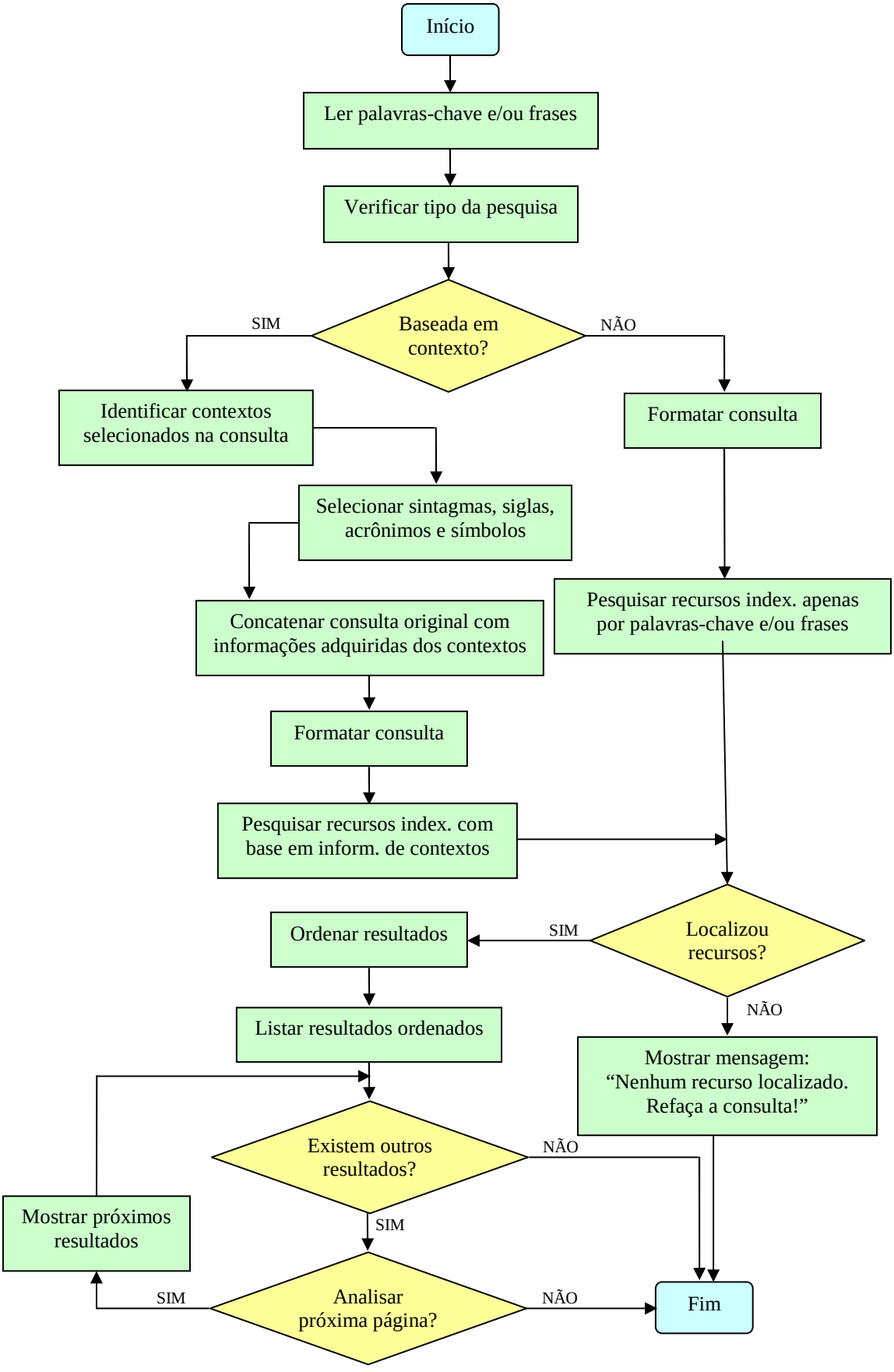

Fig. 4.18 – Fluxograma da rotina de pesquisa

Na Fig 4.18, é mostrada a forma como é realizado o controle da listagem dos resultados obtidos. Quando ainda existem resultados a serem listados, o protótipo disponibiliza um botão que apresenta os próximos resultados. Se o usuário desejar analisá-los, bastará pressionar o botão, e uma nova lista de resultados aparecerá.

4.2.4 Operacionalização

A Fig 4.19 representa a página inicial do protótipo chamado *OlheAki!*. A partir dela, é possível que um usuário não cadastrado no sistema possa criar uma conta de acesso ou ainda realizar pesquisas, mas sem a utilização de contextos. Para um usuário já cadastrado, é permitida a alteração de sua senha atual.

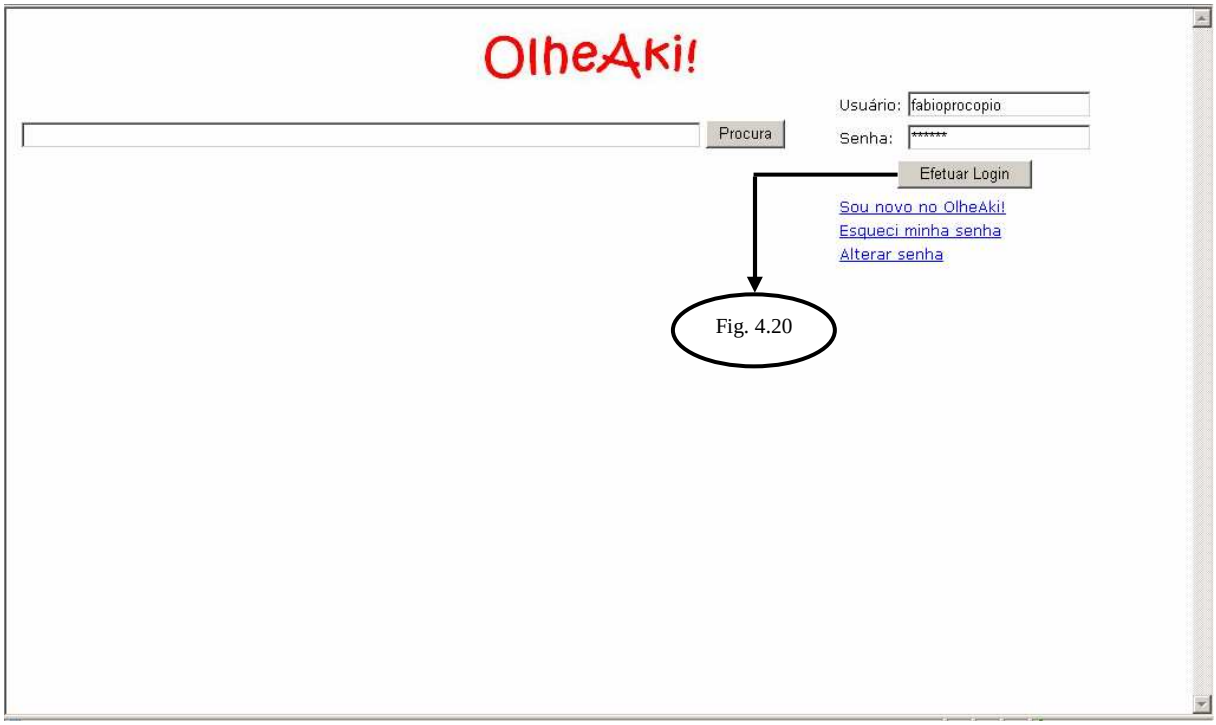

Fig. 4.19 – Página inicial do protótipo

Através dessa página, um usuário cadastrado pode acessar o sistema e utilizar as funções oferecidas. Uma vez informados *login* e senha, o usuário terá acesso à página mostrada na Fig. 4.20.

A Fig 4.20 ilustra a página que permite a utilização de pesquisas baseadas nos contextos definidos no perfil do usuário que está conectado.

Em (A), o usuário informa as palavras-chave e/ou frases que serão utilizadas na pesquisa. No exemplo, é informada a sigla *SQL*. Em (C), é informado o tipo de pesquisa que será realizado: baseando-se apenas em palavras-chave ou utilizando contextos.

Em (D), são listados todos os contextos configurados para o perfil do usuário, exceto "*SQL Server*", uma vez que este foi o selecionado para a consulta. Na figura, os contextos relacionados ao usuário conectado são "Banco de Dados", "Djavan", "*Java*", "Redes de Computadores", "UML" e "*Wireless*".

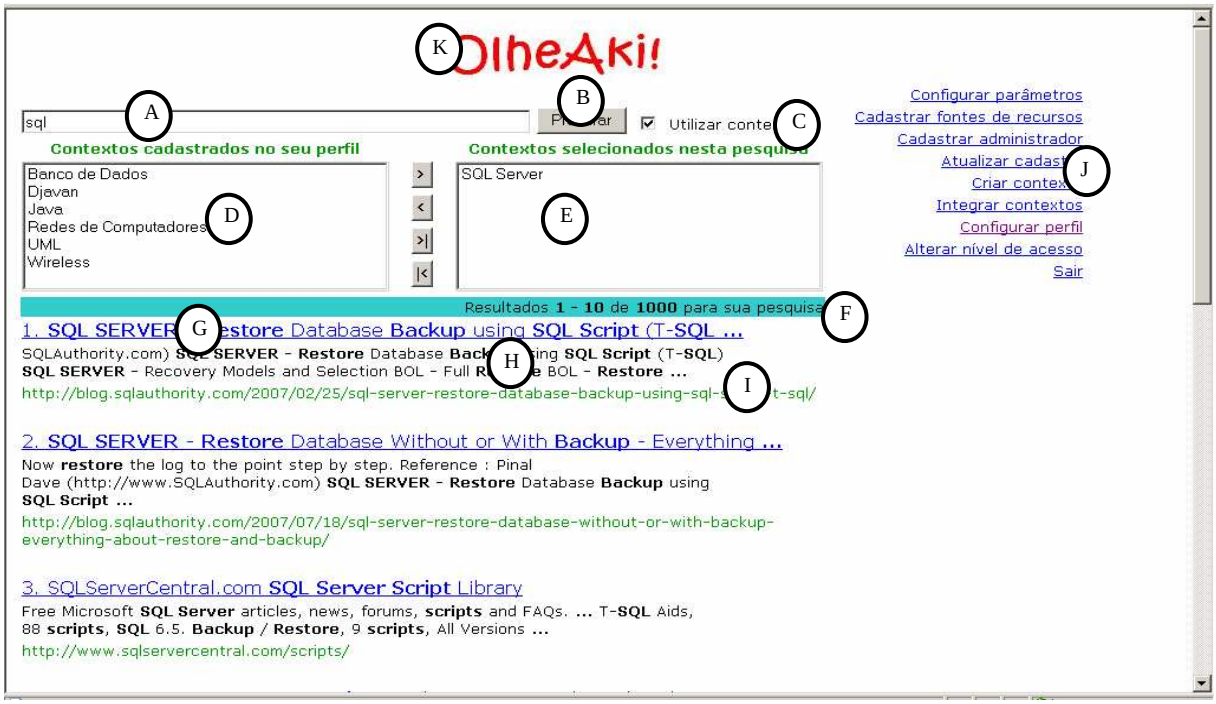

Fig. 4.20 – Página de pesquisa com a utilização de contextos

Já em (E), são listados os contextos selecionados para a pesquisa. No exemplo, apenas o contexto "*SQL Server*" foi selecionado. Informados esses dados, depois que o usuário pressionar o botão Procurar (B), o protótipo iniciará o processo de pesquisa.

Em (F), o protótipo apresenta um resumo da quantidade de documentos localizados na pesquisa. (G) representa o título do recurso localizado. No exemplo da figura anterior, o título localizado foi "**SQL Server** – **Restore** Database **Backup** using **SQL Script** (T-**SQL**...". Em (H), é mostrado um pequeno resumo do recurso ilustrado em (G). (I) exibe o endereço onde o recurso está disponível. (J) exibe os *links* que podem ser acessados pelo perfil do usuário, e (K) é o logotipo do protótipo.

```
(conteudo: "SQL" AND "backup" AND "restore" AND "script" AND "SQL Server")
 OR (titulo "SQL" AND "backup" AND "restore" AND "script" AND "SQL Server")
```

Listagem 4.7 – Consulta formatada

No exemplo ilustrado na Fig. 4.20, a consulta enviada para o mecanismo de busca é representada na Listagem 4.7. Esta mostra que, antes de a consulta ser enviada para o motor de busca, os sintagmas atribuídos ao contexto "*SQL Server*" foram concatenados com a consulta original, configurando-se, assim, a contextualização da pesquisa. Vê-se que na base de índices, existem dois campos utilizados para pesquisa: *conteudo* e *titulo*.

A Fig. 4.21 representa a página na qual é realizada a configuração do perfil do usuário. (A) permite invocar uma página em que são relacionados todos os contextos cadastrados pelo usuário. Quando se seleciona um desses contextos, ele é enviado para a página requisitante da listagem, conforme (B) (na figura, o contexto selecionado é o *SQL Server*). (C) lista os relacionamentos associados ao contexto selecionado e suas respectivas descrições (no exemplo, nenhum relacionamento foi definido).

As figuras (D), (E) e (F) são utilizadas para atualizar os relacionamentos associados ao contexto selecionado. Elas invocam páginas que efetuam a inserção, a alteração e a remoção de relacionamentos, respectivamente.

Em (G) são listados todos os sintagmas relacionados ao contexto selecionado. Na figura, os sintagmas definidos são "*backup*", "*restore*" e "*script*".

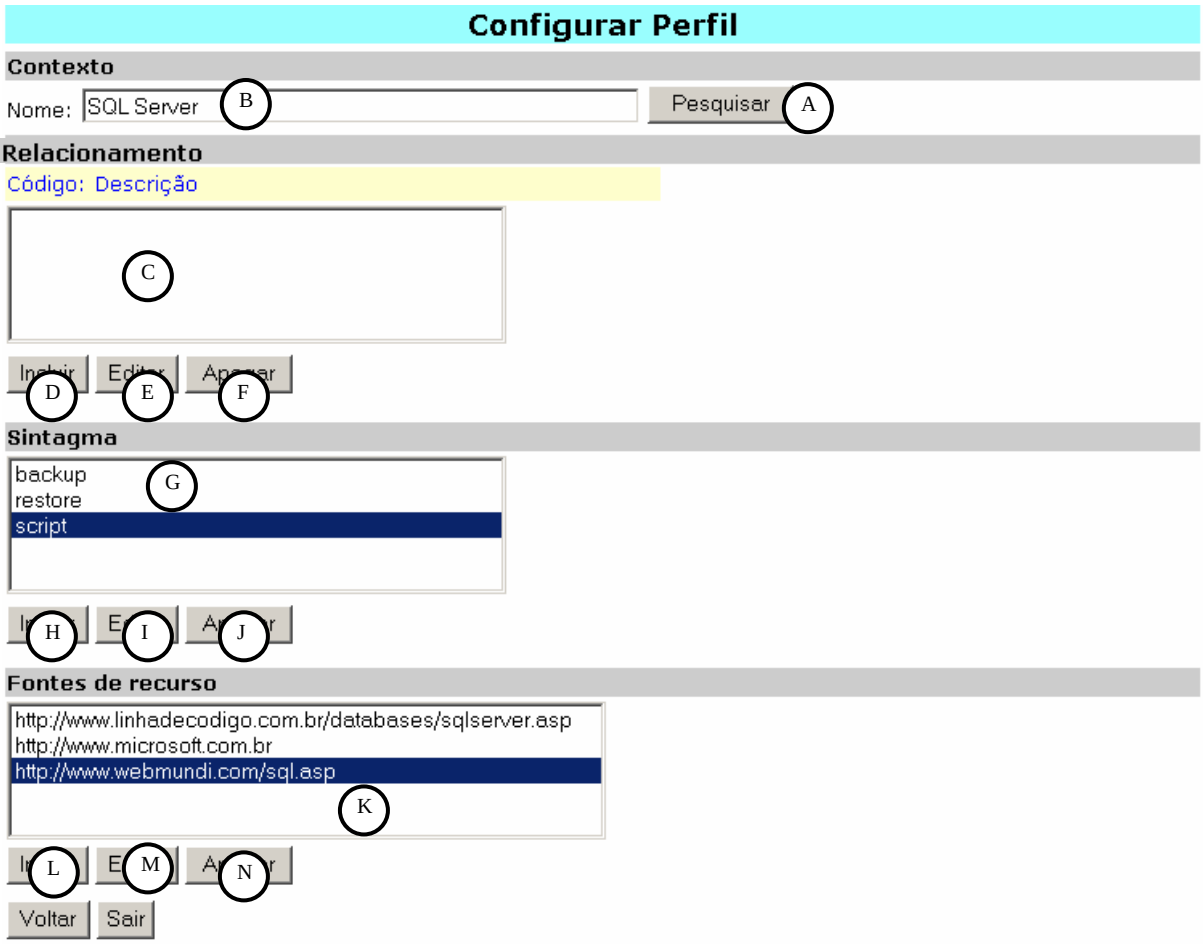

Fig 4.21 – Configuração de um perfil de pesquisa

(H), (I) e (J) são utilizadas para atualizar os sintagmas relacionados ao contexto selecionado. Elas invocam páginas que efetuam a inserção, a alteração e a remoção de sintagmas, respectivamente.

Em (K) são listados todos os endereços de fontes de recurso relacionados ao contexto selecionado.

(L), (M) e (N) são usadas para atualizar as fontes de recursos relacionadas ao contexto selecionado. Invocam páginas que realizam, respectivamente, a inserção, a alteração e a remoção das fontes de recurso.

Depois de informados esses dados, o contexto é configurado para ser utilizado nas pesquisas do usuário.

A Fig. 4.22 representa o procedimento de integração de contextos compartilhados. Esse processo consiste em adicionar contextos cadastrados por outros usuários ao perfil do usuário conectado ao protótipo. No exemplo, o contexto a ser importado está relacionado à área médica.

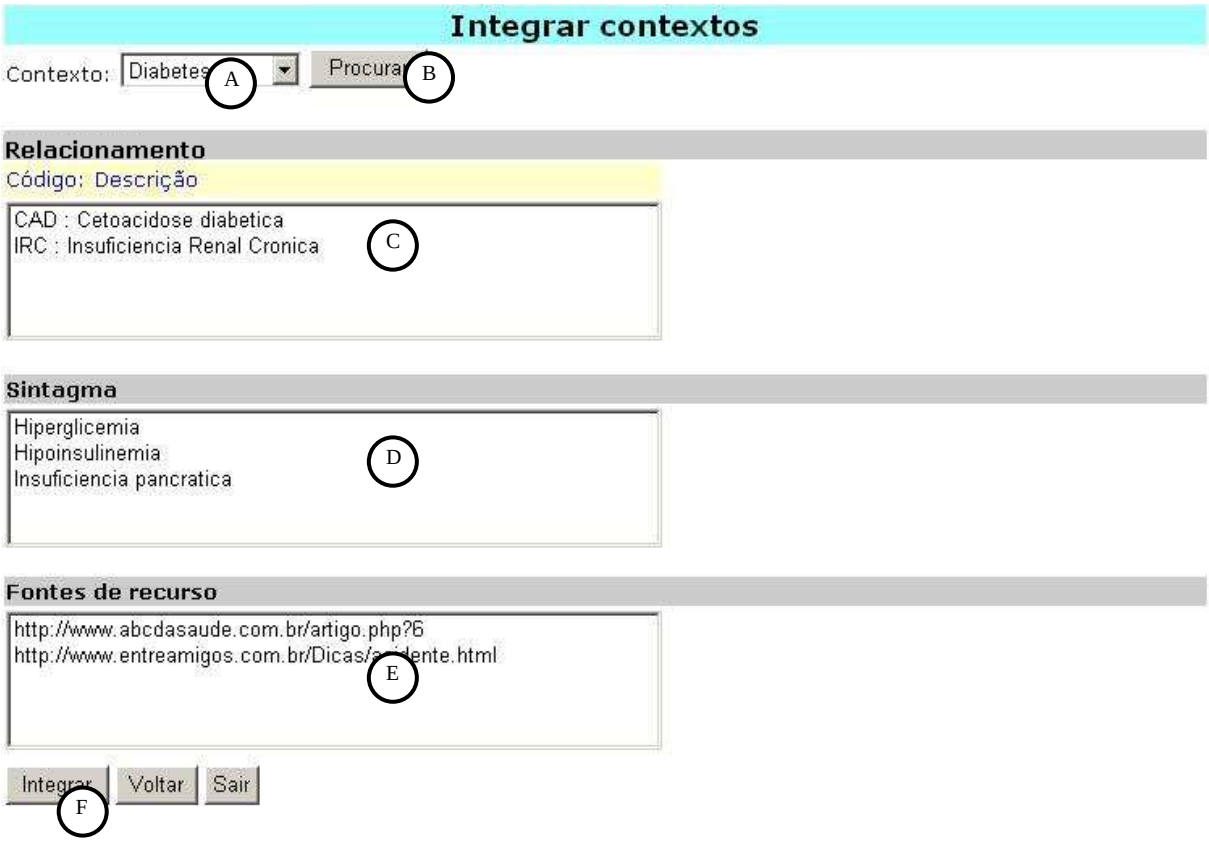

Fig 4.22 – Integração de contextos

Em (A) são listados os contextos disponíveis para compartilhamento que foram cadastrados por outros usuários. Ao pressionar o botão Procurar (B), o protótipo apresentará as informações de relacionamentos, de sintagmas e de fontes de recurso definidas inicialmente pelo usuário criador. (C), (D) e (E) apresentam, respectivamente, essas informações. Ao pressionar o botão Integrar (F), o protótipo realizará, efetivamente, a integração do novo contexto ao perfil do usuário corrente.

Como mostrado na Fig 4.1, na seção Diagrama de Casos de Uso (4.1.1), existem dois casos de uso implementados no protótipo que oferecem serviços *web*: “Executar pesquisa” e “Importar contextos”. As classes que realizam essas tarefas são publicadas sob a gerência do *framework Axis* e executadas a partir da solicitação de aplicações clientes.

A Listagem 4.8 representa uma parte de uma resposta do *web service* às requisições de aplicações clientes que solicitam uma lista contendo informações de contextos disponíveis. A resposta enviada está no formato do padrão XML, o que permite a facilidade na interpretação e no processamento por parte das aplicações.

```
<contextos>
  <contexto>
    <nome>Aterosclerose</nome>
       <sintagma>
          <nome>Hipóxia</nome>
          <nome>Isquemia</nome>
          <nome>Necrose tecidual</nome>
          <nome>Vasoclusão</nome>
      </sintagma>
      <correlacao>
        <relacao>
          <codigo>AVC</codigo>
          <descricao>Acidente Vascular Cerebral</descricao>
        </relacao>
       <relacao>
          <codigo>IAM</codigo>
          <descricao>Infarto Agudo do Miocárdio</descricao>
        </relacao>
      </correlacao>
      <fonte>
        <endereco>http://www.abcdasaude.com.br/artigo.php?6</endereco>
        <endereco>http://www.entreamigos.com.br/Dicas/acidente.html</endereco>
      </fonte>
  </contexto>
</contextos>
```

Listagem 4.8 – Resposta do *web service* à aplicação cliente

A Listagem 4.9 é um arquivo WSDL que descreve o serviço de listagem de contextos implementada no *web service*:

```
<wsdl:definitions targetNamespace="http://localhost:8080/axis/WSContexto.jws">
<!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
-->
 <wsdl:message name="getContextoXMLResponse">
  <wsdl:part name="getContextoXMLReturn" type="xsd:string"/>
 </wsdl:message>
 <wsdl:message name="getContextoXMLRequest">
  <wsdl:part name="nomeAplicacao" type="xsd:string"/>
 </wsdl:message>
 <wsdl:portType name="WSContexto">
  <wsdl:operation name="getContextoXML">
    <wsdl:input message="impl:getContextoXMLRequest" name="getContextoXMLRequest"/>
    <wsdl:output message="impl:getContextoXMLResponse" name="getContextoXMLResponse"/>
   </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="WSContextoSoapBinding" type="impl:WSContexto">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="getContextoXML">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="getContextoXMLRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="http://DefaultNamespace" use="encoded"/>
   </wsdl:input>
   <wsdl:output name="getContextoXMLResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="http://localhost:8080/axis/WSContexto.jws" use="encoded"/>
   </wsdl:output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="WSContextoService">
  <wsdl:port binding="impl:WSContextoSoapBinding" name="WSContexto">
   <wsdlsoap:address location="http://localhost:8080/axis/WSContexto.jws"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Listagem 4.9 – WSDL que descreve o serviço de listagem de contextos

A Fig. 4.23 mostra uma aplicação, desenvolvida na linguagem de programação *Visual Basic*, que invoca o serviço da classe *Java web service WSContexto.jws*. O serviço *web* processa a requisição e envia a resposta para a aplicação, conforme a figura:

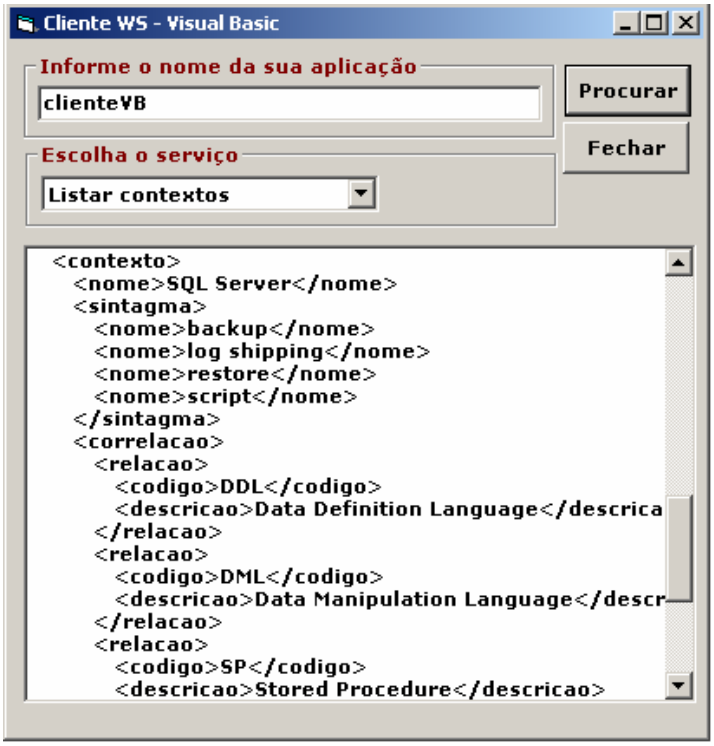

Fig. 4.23 – Aplicação cliente VB requisitando o serviço *web* listagem de contextos

Por fim, a Fig. 4.24 é o exemplo do protótipo de uma aplicação *Java* que invoca o serviço de pesquisa implementado pelo *web service*. A aplicação envia uma consulta e o *web service* executa a requisição e localiza recursos que satisfaçam à consulta.

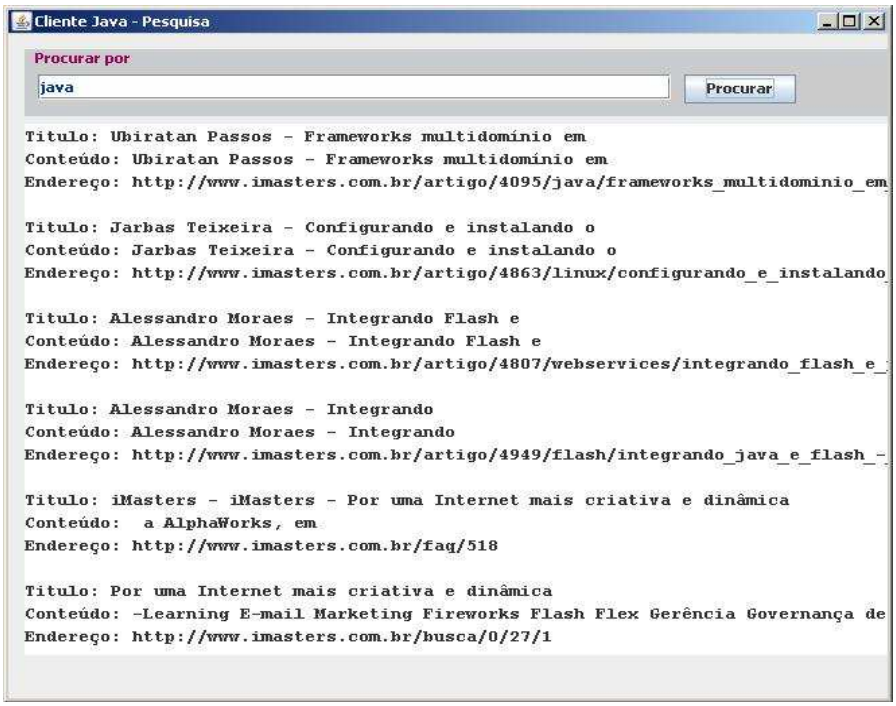

Fig. 4.24 – Aplicação cliente *Java* requisitando o serviço *web* pesquisa

4.2.5 Biblioteca de indexação e de busca

Lucene [53] é uma das tecnologias do projeto *Apache* [55]. *Open-source* e escrita totalmente em *Java*, essa biblioteca implementa um mecanismo de busca de alta performance e é ideal para aplicações que requerem pesquisas *full-text*. Por essa razão, *Lucene* foi escolhida para executar as tarefas de busca e indexação.

4.2.6 Linguagem de programação

Para o desenvolvimento do protótipo, a linguagem de programação *Java* [56] foi a escolhida por apresentar algumas características consideradas relevantes: portabilidade, emprego da orientação a objetos, conjunto vasto de bibliotecas disponíveis e licença pública de uso.

4.2.7 Disponibilidade de serviços para usuários

Java Server Pages (JSP) é uma tecnologia que se baseia na linguagem de programação *Java* para criar páginas *web* [57]. O JSP foi escolhido para construir as interfaces do protótipo pelo fato de permitir que as páginas *web* possam ser executadas em vários servidores *web* e em várias plataformas.

4.2.8 Disponibilidade de serviços para aplicações

Foi desenvolvido um *web service* pelo fato de este permitir que outras aplicações possam executar funcionalidades do protótipo. Destas as disponíveis no serviço *web* são: pesquisa de recursos e listagem de contextos.

4.2.9 Banco de dados

O *PostgreSQL* [58] é um sistema de gerenciamento de banco de dados que possui algumas características consideradas importantes: multiplataforma, *open-source* e suporte de um grande volume de dados. Em razão disso, optou-se por sua utilização.

4.2.10 *Container*

O *Tomcat* é um servidor de aplicações *Java* para a *Web* que possui algumas características consideradas importantes: distribuição livre, robustez [59] e suporte a páginas JSP. Por esse motivo, escolheu-se esse *container*.

4.3 VALIDAÇÃO DOS CENÁRIOS

A análise dos resultados obtidos será realizada por meio da validação dos cenários. Esta é efetuada a partir de testes realizados no protótipo com o objetivo de comprovar a funcionalidade desses cenários. Antes de iniciar, é importante destacar novamente os cenários definidos no capítulo anterior e as suas respectivas visões: de um usuário, de um administrador e de uma aplicação. A Fig. 4.25 representa essas visões:

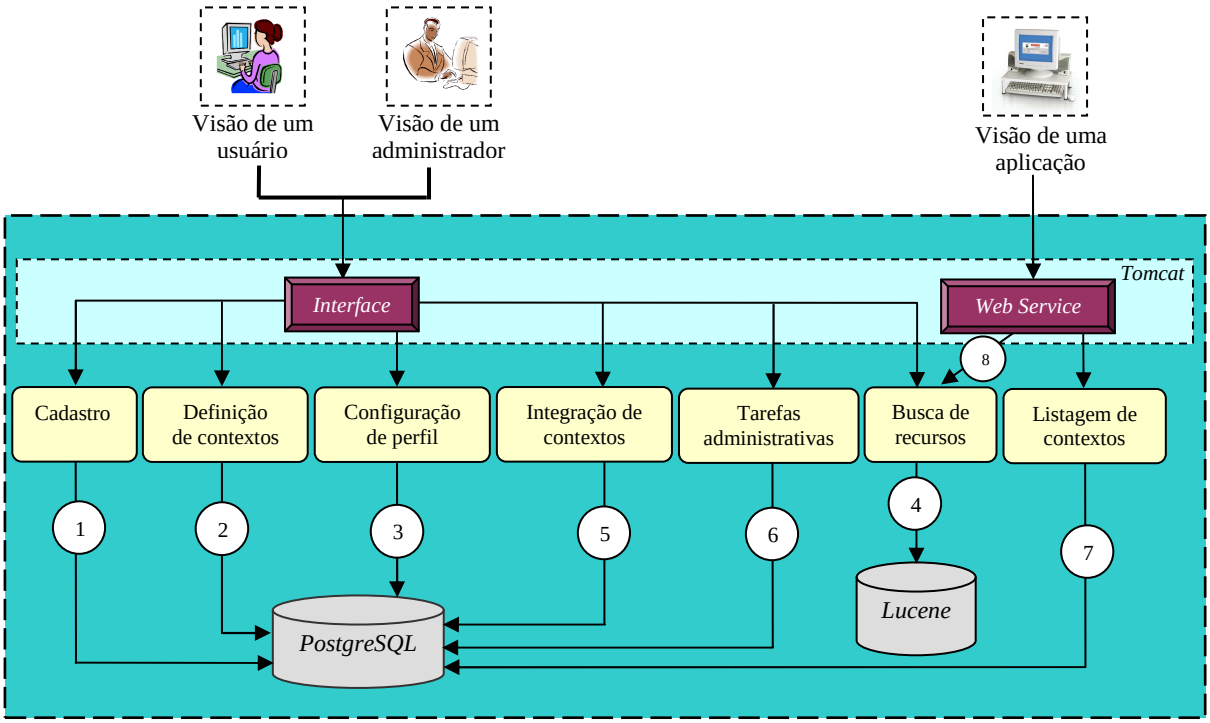

Fig 4.25 – Esquema das visões e dos seus cenários

Visão de um usuário

Representa as operações que podem ser realizadas por um usuário.

- ✓ O cenário 1, na Fig. 4.25, relacionou o cadastro de usuários com a finalidade de permitir o acesso ao protótipo e a utilização das funcionalidades dele. Assim, conforme proposto, o cenário 1 foi atendido.
- ✓ O cenário 2, também na Fig. 4.25, mostrou que é permitido ao usuário criar, editar, compartilhar e desabilitar contextos de pesquisa. Dessa forma, o cenário 2 também foi satisfeito.
- ✓ O cenário 3 possibilitou que o usuário configurasse o seu perfil por meio da relação entre contextos cadastrados e informações de sintagmas, relacionamentos e fontes de recursos. Portanto, os requisitos e, conseqüentemente, o próprio cenário foram atendidos.
- ✓ O cenário 4 representou a formulação e a execução de uma consulta. Nele, o protótipo obteve os parâmetros informados na consulta e as informações relacionadas aos contextos selecionados (caso existam). Em seguida, a consulta foi enviada para o mecanismo de busca que, finalmente, efetuou a pesquisa. Assim, o cenário 4 também se realizou.
- ✓ O cenário 5 tornou possível o compartilhamento das informações de contextos entre os usuários. Conforme proposto, este cenário foi satisfeito.

Visão de um administrador

Permite que o administrador configure o comportamento de algumas funções do protótipo.

- ✓ A Fig. 4.25 mostra que o Administrador participou dos mesmos cenários em que o Usuário interagiu, no entanto o 6 foi exclusivo apenas do Administrador. Nesse cenário, o Administrador informou os parâmetros responsáveis pela definição do comportamento de algumas funcionalidades. Dessa forma, ele se cumpriu.

Visão de uma aplicação

Possibilita que as aplicações obtenham informações dos contextos que foram definidos pelos usuários do protótipo e possam requisitar pesquisas de recursos.

- ✓ O cenário 7 disponibilizou uma lista contendo o conjunto de contextos criados pelos usuários do protótipo. Tal lista se tornou acessível a partir de um serviço *web*. Portanto, o objetivo do cenário foi atingido.
- ✓ O cenário 8 permitiu que aplicações clientes pudessem requisitar um serviço de busca a partir do uso de um *web service* através da *Internet*. Atendido o requisito, o cenário se realizou.

4.4 RESUMO

Este capítulo apresentou efetivamente a construção do protótipo da Máquina de Busca proposta. Foram mostradas a especificação e a implementação do protótipo com utilização de diagramas UML, de trechos de rotinas implementados na codificação e de esquemas de execução; e com operacionalização. Por fim, foi realizada a análise de resultados realizada através da validação dos cenários.

"Não coloque o objetivo longe demais de suas mãos, abrace os que estão ao seu alcance hoje."

Aristóteles

CAPÍTULO 5
CONCLUSÕES

5 CONCLUSÕES

Neste último capítulo, são apresentadas as considerações finais do trabalho. Na seção 5.1, mostram-se as considerações gerais abordando os conhecimentos adquiridos com o desenvolvimento do trabalho; na 5.2, as considerações específicas relacionadas à pesquisa, analisando as dificuldades encontradas na elaboração e as suas limitações; e, finalmente, na seção 5.3, sugerem-se propostas de trabalhos futuros na tentativa de otimizar a versão inicial deste protótipo.

5.1 CONCLUSÕES GERAIS

Por meio do estudo e da utilização de algumas tecnologias empregadas no desenvolvimento deste trabalho, foi adquirido um embasamento em algumas tecnologias, tais como: linguagem de programação *Java*, *Web* Semântica e funcionamento e construção de uma Máquina de Busca. Abaixo, são citados os conhecimentos adquiridos:

- aprendizado em programação de *web service* com *Java* – a construção de um serviço *web* proporcionou a aquisição de conhecimentos relacionados à tecnologia SOA e à própria linguagem de programação utilizada;
- aprendizado em *Web* Semântica – a *Web* Semântica tem como objetivo principal tornar a informação disponível e legível para os computadores de modo que o relacionamento homem-máquina seja mais efetivo, relacionando-se, pois, com o conhecimento adquirido em *web service*.
- aprendizado em Máquinas de Busca – conhecimento adquirido durante o estudo dos fundamentos teóricos que embasaram o trabalho, principalmente no decorrer da fase de implementação do protótipo, na qual foram construídos algoritmos com a finalidade de realizar as tarefas básicas de uma Máquina de Busca: descoberta, indexação e pesquisa.

5.2 CONCLUSÕES ESPECÍFICAS

Nesta seção, são apresentadas as considerações relacionadas ao desenvolvimento do trabalho e feitas algumas sugestões de melhorias, limitações e dificuldades encontradas durante a sua elaboração.

Nos próximos itens, são listadas algumas considerações sobre o trabalho. O objetivo é descrever algumas melhorias que podem ser avaliadas e, se possível, implementadas na tentativa de aumentar o escopo do protótipo:

- o cenário 1, apesar de ter sido satisfeito, conforme a validação realizada no capítulo anterior, pode ser melhorado no sentido de informar ao usuário, de alguma forma (*e-mail*, por exemplo), que os endereços para visita sugeridos ao *crawler* foram indexados;
- a criação de outros cenários relacionados a serviços *web*, como, por exemplo, uma lista (completa ou parcial) com os endereços que foram descobertos pelo *crawler*, possibilitaria que outras aplicações tenham essas fontes de recursos disponíveis, uma vez que há apenas dois cenários desse tipo.

Existem algumas limitações identificadas no protótipo. A primeira delas diz respeito à descoberta de recursos, apesar de o objetivo principal do protótipo ser a tarefa de pesquisa. O processo de descoberta de recursos ainda é bastante limitado porque são poucos os endereços-raiz disponíveis para visitas. Um número maior desses endereços aumentaria consideravelmente a quantidade de novas descobertas.

A indexação também é mais uma das limitações deste trabalho porque ela só é realizada sobre recursos que têm o formato de texto ou que podem ter seu conteúdo convertido para tal. Isso implica dizer que arquivos no formato de imagem, áudio ou vídeo, por exemplo, não podem ser indexados e, conseqüentemente, localizados pelo protótipo.

Uma outra limitação diz respeito à filtragem no processo de pesquisa. Na formulação da consulta, o usuário não tem como definir filtros de restrição de pesquisa como, por exemplo: idioma, país, formato do recurso ou *sites* adultos. Isso possibilitaria que os resultados fossem ainda mais restritos.

E a última limitação está relacionada aos endereços não localizados. Eventualmente, alguns endereços de acesso podem estar indisponíveis. Caso isso aconteça, o protótipo não possui um mecanismo capaz de disponibilizar a localização desses recursos, caracterizando, assim, um recurso descoberto, porém indisponível.

Além das limitações citadas anteriormente, foram encontradas algumas dificuldades no decorrer do trabalho, cujo fator positivo foi, em função da necessidade de construção do protótipo, o fomento de várias pesquisas em *sites*, fóruns e bate-papos especializados no assunto. A seguir, são mostradas as principais dificuldades:

- a tarefa de realizar a descoberta de recursos é uma das funcionalidades mais complexas implementada no protótipo. Por isso, para abstrair essa complexidade, foi utilizada a biblioteca *Crawler*, pela qual, a partir de endereços *web* fornecidos, novos endereços relacionados aos que foram informados como parâmetro são descobertos;
- a linguagem *Java*, por ser bastante robusta e por possuir diversos recursos que auxiliam o programador no desenvolvimento de aplicações, exigiu, em paralelo ao desenvolvimento desta dissertação, a dedicação ao seu estudo a fim de viabilizar o desenvolvimento do protótipo.

5.3 TRABALHOS FUTUROS

O desenvolvimento de novos trabalhos é de extrema importância por possibilitar a otimização de funções deficientes encontradas nesta versão inicial e viabilizar a construção de funcionalidades ainda não implementadas. A seguir, são sugeridas algumas idéias que podem contribuir com a melhoria deste trabalho:

- indexação – esta versão indexa recursos que estão no formato texto ou que podem ser convertidos para esse formato. A sugestão é criar um mecanismo capaz de indexar recursos em outros formatos (imagens, áudio e vídeo), o que possibilitaria um maior número de recursos disponíveis para pesquisa;
- localização de endereços indisponíveis – alguns recursos podem não estar com o seu endereço de localização ativo. Nesse caso, sugere-se o desenvolvimento de um mecanismo capaz de recuperar o conteúdo desses recursos. Uma idéia seria o armazenamento de todo o conteúdo dos recursos na base de dados e, se por ventura o endereço estivesse inativo no momento da pesquisa, o protótipo construiria uma página HTML a partir do conteúdo armazenado;
- filtro de pesquisa – os recursos localizados são filtrados por meio de seu formato, local de hospedagem, idioma utilizado e acessibilidade de conteúdo adulto;
- relevância de termos no contexto – os sintagmas e os relacionamentos possuem um peso em relação ao contexto utilizado durante uma pesquisa. A relevância desses termos (variando de 0 a 1) poderia ser definida de uma forma automática tendo como base a quantidade de vezes que eles aparecem em uma consulta;

- sugestão de contextos – os contextos utilizados mais freqüentemente por um usuário seriam o ponto de partida para que o protótipo sugerisse contextos semelhantes aos usados por outros usuários. Essa funcionalidade disponibilizaria uma quantidade maior de contextos e, conseqüentemente, de termos relacionados às áreas de interesse daquele usuário.

REFERÊNCIAS

[1] TANEMBAUM, Andrew S. **Redes de Computadores**. Tradução por Insight Serviços de Informática. 3. ed. Rio de Janeiro: Campus, 1997.

[2] CERN, ***Welcome to info.cern.ch:*** *the website of the world's first-ever web server*. Disponível em: <http://info.cern.ch/>. Acesso em: 15 mar. 2007.

[3] O'REILLY, Tim. ***What is Web 2.0:*** *Design Patterns and Business Models for the Next Generation of Software*. San Francisco: 2005. Disponível em: <http://www.oreillynet.com/lpt/a/6228>. Acesso em: 15 mar. 2007.

[4] WEB 2.0 SUMMIT. ***About Web 2.0 Summit:*** *The Fourth Annual Web 2.0 Summit*. Disponível em: <http://www.web2summit.com/>. Acesso em: 15 mar. 2007.

[5] Eric's Blog. **Web 2.0 History**.
Disponível em: <http://learningremix.net/f2006integ/enapier/2006/09/web_20.shtml>. Acesso em: 15 mar. 2007.

[6] WIKIPEDIA, **Web 2.0**. Disponível em: <http://en.wikipedia.org/wiki/Web_2>. Acesso em: 15 mar. 2007.

[7] O'REILLY RADAR. ***Web 2.0 Compact Definition****: Trying Again*. Disponível em: <http://radar.oreilly.com/archives/2006/12/web_20_compact.html>. Acesso em: 15 mar. 2007.

[8] O'REILLY, Tim. ***People Inside & Web 2.0***. *Open Business*, 25 abr. 2006. Disponível em: http://www.openbusiness.cc/2006/04/25/people-inside-web-20-an-interview-with-tim-o-reilly/. Acesso em: 15 mar. 2007.

[9] GLASER, Mark. ***What is Web 2.0, and Should You Care?*** Disponível em: <http://www.pbs.org/mediashift/2006/04/jargon_watchwhat_is_web_20_and.html>. Acesso em: 18 mar. 2007.

[10] O'REILLY, Tim. ***What kind of internet operating system do we want?*** Disponível em: <http://www.oreillynet.com/pub/wlg/1262>. Acesso em: 18 mar. 2007.

[11] JR, Gilberto. ***Tim O'Reilly define a Web 2.0***. Disponível em: <http://w2br.com/2006/11/30/tim-oreilly-define-a-web-20/>. Acesso em: 18 mar. 2007.

[12] MENEZES, José de. **Definições da web dois ponto zero**. Disponível em: <http://www.plugmasters.com.br/sys/materias/622/1/Defini%E7%F5es-da-web-dois-ponto-zero>. Acesso em: 18 mar. 2007.

[13] SCHWARTZ, Erik. ***Semantic Search*****:** *Offering Meaningful Results for the Web 2.0 Era*. Disponível em: <http://www.semantic-conference.com/2007/sessions/l1.html>. Acesso em: 20 mar. 2007.

[14] HINCHCLIFFE, Dion. ***Visualizing Web 2.0***. Disponível em: <http://web2.wsj2.com/visualizingweb20.htm >. Acesso em: 20 mar. 2007.

[15] PARTER, Josua; MACMANUS, Richard. ***Web 2.0 Design:*** *Bootstrapping the Social Web*. Disponível em: <http://www.digital-web.com/articles/web_2_for_designers/>. Acesso em: 20 mar. 2007.

[16] Steve's Blog. ***Is Tim just Mizundastood?*** Disponível em: <http://geocontext.org/node/7>**.** Acesso em: 25 mar. 2007.

[17] INTERAKT, ***Upgrading the Internet – Web 2.0***. Disponível em: <http://www.interaktonline.com/Support/Articles/Details/Upgrading+the+Internet+%96+Web+2.0.html?id_art=39>. Acesso em: 25. mar. 2007.

[18] TORRES, Bruno. ***Web 0.1α***. Disponível em: <http://brunotorres.net/2005/10/14/web-01>. Acesso em: 25. mar. 2007.

[19] BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. ***The Semantic Web:*** *A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. Disponível em: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&pageNumber=1&catID=2>. Acesso em: 26 mar. 2007.

[20] PALMER, Sean B. ***The Semantic Web:*** *An Introduction.* Disponível em: <http://infomesh.net/2001/swintro/>. Acesso em: 26 mar. 2007.

[21] WIKIPEDIA, ***Semantic Web***.
Disponível em: <http://en.wikipedia.org/wiki/Semantic_web>.
Acesso em: 19 mar. 2007.

[22] DZIEKANIAK, Gisele Vasconcelos; KIRINUS, Josiane Boeira. **Web Semântica**. Ciência da Informação**,** p 20-19, Florianópolis, n.18, 2004.

[23] WIKIPEDIA, **Web 3.0**. Disponível em: <http://pt.wikipedia.org/wiki/Web_3.0>. Acesso em: 20 mar. 2007.

[24] TOLDA, Stelleo. **O e-commerce e a web 3.0**. Disponível em: <http://wnews.uol.com.br/site/colunas/materia.php?id_secao=9&id_conteudo=358>.
Acesso em: 20 mar. 2007.

[25] SPIVACK, Nova. Pioneiro vê enorme potencial para negócios de buscas. Washington: **Folha de São Paulo**, São Paulo, 21 fev. 2007. Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u21657.shtml>. Acesso em: 25 mar. 2007.

[26] WIKIPEDIA, Motor de busca.
Disponível em: <http://pt.wikipedia.org/wiki/M%C3%A1quinas_de_busca>.
Acesso em: 24 mar. 2007.

[27] JONES, Gareth J. F., BROWN, Peter J. ***The Role of Context in Information Retrieval***. In: Annual International ACM SIGIR Conference. 27., 2004, Sheffield, South Yorkshire. **Proceedings…** Sheffield, South Yorkshire: 2004. P. 20-22

[28] **Conference on Research & Development on Information Retrieval** – SIGIR 2006, Seattle, United States.

[29] ESSIR 2007 – **European Summer School in Information Retrieval**, Glasgow, Scotland

[30] BACCHIN, Thiago Rhicter(coord). O uso dos sites de busca no Brasil. **Cadastra.** Disponível em: <http://www.cadastra.com.br/Pesquisa_Cadastra_O_Uso_dos_sites_de_busca_no_Brasil.pdf> Acesso em: 13 fev. 2007.

[31] KURAMOTO, Hélio. Sintagmas Nominais: uma Nova Proposta para a Recuperação da Informação. **Ciência da Informação**, Brasília, v. 3, n. 1, fev. 2002.

[32] GOSPODNETIC, Otis. HATCHER, Erik. ***Lucene in Action***. Greenwich: Manning, 2005.

[33] WIKIPEDIA, **Recuperação de informação**. Disponível em: <http://pt.wikipedia.org/wiki/Recupera%C3%A7%C3%A3o_de_Informa%C3%A7%C3%B5es>. Acesso em: 04/04/2007.

[34] FERNEDA, Edberto. **Recuperação de Informação:** Análise sobre a contribuição da Ciência da Computação para a Ciência da Informação. 2003. 147f. Tese(Doutorado em Ciência da Informação e Documentação) – Escola de Comunicação e Artes, Universidade de São Paulo, São Paulo, 2003.

[35] CARDOSO, Olinda Nogueira Paes. **Recuperação de Informação**. Lavras, [s.d].

[36] GUIMARÃES, Francisco José Zamith. **Utilização de ontologias no domínio B2C**. 2002. 195f. Dissertação(Mestrado em Informática) - Departamento de Informática, Pontifícia Universidade Católica, Rio de Janeiro, 2002.

[37] WIKIPEDIA, **Diretórios de sites**. Disponível em: <http://pt.wikipedia.org/wiki/Diret%C3%B3rios_de_sites> Acesso em: 01 abr. 2007.

[38] CENDÓN, Beatriz Valadares. Ferramentas de busca na Web. **Ciência da Informação**, Brasília, v. 30, n. 1, p. 39-49, jan.-abr.2001.

[39] BEZERRA, Allan José de Souza. **Detecção de sites replicados em base de dados de Máquinas de Busca**. Manaus: UFAM, 2004. 72f. Monografia(Bacharelado em Ciência da Computação) - Departamento de Ciência da Computação, Universidade Federal do Amazonas, 2004.

[40] NOBLES, Robin. ***The future of search engine optimizing:*** *theme engines*. In: *Search Engine Workshops*, 2003.

[41] FINKELSTEIN, Lev; GABRILOVICH Evgeniy; MATIAS, Yossi et al. ***Placing Search in Context:*** *The Concept Revisited*. In: *Proceedings International WWW Conference*(10), Hong-Kong, 2001.

[42] OLSTAD, Bejørn; SERES, Silvija. *What is Contextual Search?* ***KM World***, p S10-S11, nov.-dez. 2005.

[43] PEAD(Português – Ensino a Distância), **A Classe dos Sintagmas**. Disponível em: <http://acd.ufrj.br/~pead/tema16/aclassedossintagmas.html>. Acesso em: 05 fev. 2007.

[44] PERINI, Mário A. **Gramática descritiva do português**. 2 ed. São Paulo: Ática, 1995. 380p.

[45] FERREIRA, Aurélio Buarque de Holanda. **Novo Dicionário Aurélio - Século XXI**. Rio de Janeiro: Nova Fronteira, 1999.

[46] FURASTÉ, Pedro Augusto. **Normas Técnicas para o Trabalho Científico:** Explicitação em Normas da ABNT. 12 ed. Porto Alegre: [s.n.], 2003.

[47] FARIA, Rogério Amorim de. **Treinamento avançado em XML**. São Paulo: Digerati Books, 2005.

[48] GIRARDI, Reubem Alexandre D'Almeida. **Framework para coordenação e mediação de Web Services modelados como Learning Objects para ambientes de aprendizado na Web**. 2004. 111f. Dissertação(Mestrado em Informática) - Departamento de Informática, Pontifícia Universidade Católica, Rio de Janeiro, 2004.

[49] WEBSERVICES.ORG. Disponível em: <http://www.webservices.org>. Acesso em: 09 abr. 2007.

[50] IMASTERS. ***WEB Services*** **com Visual FoxPro 9.0**. Disponível em: <http://www.imasters.com.br/artigo/4025/webservices/web_services_com_visual_foxpro_90> Acesso em: 09 abr. 2007.

[51] GUEDES, Gilleanes T. A.. **UML:** uma abordagem prática. 2 ed. São Paulo: Novatec, 2006.

[52] JAVA.NET. ***Smart and Simple Web Crawler***. Disponível em: <https://crawler.dev.java.net/> . Acesso em: 13 jan. 2007

[53] LUCENE. Disponível em: <http://lucene.apache.org>. Acesso em: 03 fev. 2006.

[54] Disponível em: <http://lucene.apache.org/java/docs/api/org/apache/lucene/search/highlight/package-summary.html>. Acesso em: 21/04/2007

[55] ***POSTGRESQL***. Disponível em: <http://www.postgresql.org>. Acesso em: 17/04/2007.

[56] ***APACHE TOMCAT***. Disponível em: <http://tomcat.apache.org>. Acesso em: 05 dez. 2005.

[57] ***THE APACHE SOFTWARE FOUNDATION***. Disponível em: <http://www.apache.org>. Acesso em: 05 dez. 2005.

[58] APACHE LUCENE, ***Who We Are***. Disponível em: <http://lucene.apache.org/java/docs/whoweare.html#contrib>. Acesso em: 10 dez. 2005.

[59] ***JAVA TECHNOLOGY***. Disponível em: <http://java.sun.com>. Acesso em: 17 nov. 2005.

[60] JGURU. ***JavaServer Pages Fundamentals, Short Course Contents***. Disponível em: <http://java.sun.com/developer/onlineTraining/JSPIntro/contents.html#Introduction>. Acesso em: 18 abr. 2007.

[61] KURAMOTO, Hélio. Uma abordagem alternativa para o tratamento e a recuperação de informação textual: os sintagmas nominais. **Ciência da Informação**, Brasília, v. 25, n. 2, 1995.

[62] GLOVER, Eric J., LAWRENCE, Steve, Birmingham, William P. et al. **Architecture of a Metasearch Engine that Supports User Information Needs**. In: *International Conference on Information and Knowledge Management*. 8., 1999. **Proceedings…**CIKM 99, Kansas City, Missouri, p. 210-216.

[63] LAWRENCE, Steve. *Context in Web Search.* ***IEEE Data Engineering Bulletin***, v. 23, n. 3, p. 25-32, set. 2000.

[64] FINKELSTEIN, Lev; GABRILOVICH Evgeniy; MATIAS, Yossi et al. ***Placing Search in Context:*** *The Concept Revisited.* In: *Proceedings International WWW Conference*(10), Hong-Kong, 2001.

[65] KRAFT, Reiner; MAGHOUL, Farzin; CHANG, Chi Chao et al. **Searching with Context**. In*: International World Wide Web Conference*, IW3C2 2006, Edinburgh, Scotland, 2006.

[66] SANTOS, Rafael. **Introdução à programação orientada a objetos usando Java**. Rio de Janeiro: Campus, 2003.

[67] SIKORA, Zbigniew M. ***Java:*** guia prático para programadores. Tradução por Altair Caldas de Moraes. Rio de Janeiro: Campus, 2003.

[68] SOUZA, Renato Rocha. Uma proposta de metodologia para indexação automática utilizando sintagmas nominais. **Ciência da Informação**, Florianópolis, n. esp., 2006.

GLOSSÁRIO

Base de dados – armazena as informações e permite que os usuários busquem e atualizem essas informações quando necessário.

Caso de uso – descreve o comportamento que o *software* deverá apresentar quando estiver pronto.

Cenário – de uma forma geral, pode-se dizer que é uma instância de um caso de uso.

Classe – representação de um conjunto de objetos que possuem características comuns.

Contexto – conjunto de termos que possui uma relação com um determinado tema.

Relacionamento – formada por um conjunto de siglas, abreviaturas, acrônimos e símbolos que representam uma relação entre dois termos que podem conter mais de uma palavra.

Container – objeto que contém outros objetos.

Crawler – programa que percorre a Web à procura de endereços com a finalidade de adicioná-los ao banco de dados de uma Máquina de Busca.

Máquina de Busca – programa especializado em localizar e listar páginas da *Internet* a partir de termos indicados pelo usuário.

Objeto – representa uma entidade que pode ser de software, física ou conceitual e que é instanciado a partir de uma classe.

Protótipo – produto ainda não comercializado, mas que está na fase de construção ou de teste.

Recuperação da Informação – área da Ciência da Computação que tem como finalidade promover o armazenamento e a recuperação automática da informação.

Sintagma – representado por uma ou várias palavras que possuem uma determinada função em uma frase.

Web service – tecnologia utilizada para simplificar a integração entre aplicações distintas.

XML – padrão que facilita a comunicação entre aplicações que utilizam plataforma e/ou ambientes distintos.

ANEXOS

```
/*
 * WSContexto.jws
 *
 * Created on 11 de Maio de 2007, 00:27
 *
 * Web service responsável por oferecer o serviço de pesquisa
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

import java.io.IOException;
import java.io.StringReader;
import java.io.StringWriter;
import java.io.File;

import java.lang.String;

import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;
import java.util.Vector;

import java.sql.Connection;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.SQLException;

public class WSContexto
{
   private String titulo, endereco, conteudo, tx_stop_word, noContexto, noSintagma,
           codigoCorrelacao, descricaoCorrelacao, txFonte;
   private int nuResultados, coContexto;

   private String nomeServidor = null;
   private String nomeBD = null;
   private String nomeUsuario = null;
   private String senhaBD = null;

   private Connection db = null;
   private Connection conexao = null;
   private Statement s;

   /*Métodos de manipulação do banco de dados*/
   private void configuraConexao(String nomeServidor, String nomeBD,
                        String nomeUsuario, String senhaBD)
   {
      this.nomeServidor = nomeServidor;
      this.nomeBD = nomeBD;
      this.nomeUsuario = nomeUsuario;
      this.senhaBD = senhaBD;
   }

   private Connection abreConexao()
   {
     String url = "jdbc:postgresql://" + nomeServidor + "/" + nomeBD;
```

```
        try
        {
          Class.forName("org.postgresql.Driver");
        }
        catch (java.lang.ClassNotFoundException e)
        {
          System.err.print("ClassNotFoundException: " );
          System.err.println(e.getMessage () );
        }

 try
        {
          db = DriverManager.getConnection(url, nomeUsuario, senhaBD);
        }
        catch (SQLException ex)
        {
          System.err.println( "SQLException: " + ex.getMessage() );
        }
 return db;
}

private void fechaConexao(Connection db)
{
  try
  {
    db.close();
  }
  catch (SQLException ex)
  {
    System.err.println( "SQLException: " + ex.getMessage() );
  }
}
/*Fim dos métodos de manipulação do banco de dados*/

private void setTitulo(String titulo)
{
   this.titulo = titulo;
}

private String getTitulo()
{
   return titulo;
}

private void setEndereco(String endereco)
{
   this.endereco = endereco;
}

private String getEndereco()
{
   return endereco;
}

private void setConteudo(String conteudo)
{
   this.conteudo = conteudo;
}

private String getConteudo()
```

```
{
   return conteudo;
}

private void setNuResultados(int nuResultados)
{
   this.nuResultados = nuResultados;
}

private int getNuResultados()
{
   return nuResultados;
}

private void setStopWord(String tx_stop_word)
{
  this.tx_stop_word = tx_stop_word;
}

private String getStopWord()
{
  return tx_stop_word;
}

private void setCoContexto(int coContexto)
{
  this.coContexto = coContexto;
}

private int getCoContexto()
{
  return coContexto;
}

private void setContexto(String noContexto)
{
  this.noContexto = noContexto;
}

private String getContexto()
{
  return noContexto;
}

private void setSintagma(String noSintagma)
{
  this.noSintagma = noSintagma;
}

private String getSintagma()
{
  return noSintagma;
}

private void setCodigoCorrelacao(String codigoCorrelacao)
{
  this.codigoCorrelacao = codigoCorrelacao;
}

private String getCodigoCorrelacao()
```

```
{
  return codigoCorrelacao;
}

private void setDescricaoCorrelacao(String descricaoCorrelacao)
{
  this.descricaoCorrelacao = descricaoCorrelacao;
}

private String getDescricaoCorrelacao()
{
  return descricaoCorrelacao;
}

private void setFonte(String txFonte)
{
  this.txFonte = txFonte;
}

private String getFonte()
{
  return txFonte;
}

public String getContextoXML()
{
  String respostaXML = null;

  respostaXML = "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>";
  respostaXML = respostaXML.concat("\n");
  respostaXML = respostaXML.concat("<contextos>");

  WSContexto contexto = new WSContexto();
  ArrayList listaContexto = contexto.listarContexto();
  for (int i = 0; i < listaContexto.size(); i++)
  {
    WSContexto c = (WSContexto)listaContexto.get(i);
    respostaXML = respostaXML.concat("\n");
    respostaXML = respostaXML.concat("   <contexto>\n");
    respostaXML = respostaXML.concat("      <nome>");
    respostaXML = respostaXML.concat(c.getContexto());
    respostaXML = respostaXML.concat("</nome>\n");

    coContexto = c.getCoContexto();
    WSContexto sintagma = new WSContexto();
    ArrayList listaSintagma = sintagma.listarSintagma(coContexto);
    respostaXML = respostaXML.concat("      <sintagma>\n");
    for (int j = 0; j < listaSintagma.size(); j++)
    {
      WSContexto s = (WSContexto)listaSintagma.get(j);

      respostaXML = respostaXML.concat("         <nome>");
      respostaXML = respostaXML.concat(s.getSintagma());
      respostaXML = respostaXML.concat("</nome>\n");
    }
    respostaXML = respostaXML.concat("      </sintagma>\n");

    WSContexto correlacao = new WSContexto();
    ArrayList listaCorrelacao = correlacao.listarCorrelacao(coContexto);
    respostaXML = respostaXML.concat("      <correlacao>\n");
```

```
      for (int k = 0; k < listaCorrelacao.size(); k++)
      {
        WSContexto cor = (WSContexto)listaCorrelacao.get(k);

        respostaXML = respostaXML.concat("         <relacao>\n");
        respostaXML = respostaXML.concat("           <codigo>");
        respostaXML = respostaXML.concat(cor.getCodigoCorrelacao());
        respostaXML = respostaXML.concat("</codigo>\n");
        respostaXML = respostaXML.concat("           <descricao>");
        respostaXML = respostaXML.concat(cor.getDescricaoCorrelacao());
        respostaXML = respostaXML.concat("</descricao>\n");
        respostaXML = respostaXML.concat("         </relacao>\n");
      }
      respostaXML = respostaXML.concat("      </correlacao>\n");

      WSContexto fonte = new WSContexto();
      ArrayList listaFonte = fonte.listarFonte(coContexto);
      respostaXML = respostaXML.concat("      <fonte>\n");
      for (int l = 0; l < listaFonte.size(); l++)
      {
        WSContexto f = (WSContexto)listaFonte.get(l);

        respostaXML = respostaXML.concat("         <endereco>");
        respostaXML = respostaXML.concat(f.getFonte());
        respostaXML = respostaXML.concat("</endereco>\n");
      }
      respostaXML = respostaXML.concat("      </fonte>\n");
      respostaXML = respostaXML.concat("   </contexto>\n");
    }
    respostaXML = respostaXML.concat("</contextos>");

    return respostaXML;
  }

  private ArrayList listarContexto()
  {
    String cmd = null;

    configuraConexao("localhost", "olheaki", "postgres", "123456");
    conexao = abreConexao();

    java.sql.ResultSet rs = null;

    String sql = "SELECT distinct co_contexto_usuario, no_contexto_usuario" +
                 "  FROM tb_contexto_usuario" +
                 " WHERE in_compartilhado = 1" +
                 " ORDER BY no_contexto_usuario";

    try
    {
      if(s == null) s = conexao.createStatement();
      rs = s.executeQuery(sql);
    }
    catch (java.sql.SQLException e)
    {
      System.out.println ("Erro ao executar: " + sql);
    }

    ArrayList result = new ArrayList();
    try
```

```
  {
    while (rs.next())
    {
      WSContexto c =  new WSContexto();

      c.setCoContexto(rs.getInt("co_contexto_usuario"));
      c.setContexto(rs.getString("no_contexto_usuario"));

      result.add(c);
    }
    fechaConexao(conexao);
    return result;
  }
  catch (java.sql.SQLException e)
  {
    System.out.println ("Erro na execucao do metodo WSContexto.listarContexto(). " + e.getMessage());
    fechaConexao(conexao);
    return result;
  }
}

private ArrayList listarSintagma(int coContexto)
{
  String cmd = null;

  configuraConexao("localhost", "olheaki", "postgres", "123456");
  conexao = abreConexao();

  java.sql.ResultSet rs = null;

  String sql = "SELECT no_sintagma" +
               "  FROM tb_contexto_usuario_sintagma" +
               " WHERE co_contexto_usuario = " + coContexto +
               " ORDER BY no_sintagma";

  try
  {
    if(s == null) s = conexao.createStatement();
    rs = s.executeQuery(sql);
  }
  catch (java.sql.SQLException e)
  {
    System.out.println ("Erro ao executar: " + sql);
  }

  ArrayList result = new ArrayList();
  try
  {
    while (rs.next())
    {
      WSContexto c =  new WSContexto();

      c.setSintagma(rs.getString("no_sintagma"));

      result.add(c);
    }
    fechaConexao(conexao);
    return result;
  }
  catch (java.sql.SQLException e)
```

```
    {
      System.out.println ("Erro na execucao do metodo WSContexto.listarSintagma(). " + e.getMessage());
      fechaConexao(conexao);
      return result;
    }
  }

  private ArrayList listarCorrelacao(int coContexto)
   {
    String cmd = null;

    configuraConexao("localhost", "olheaki", "postgres", "123456");
    conexao = abreConexao();

    java.sql.ResultSet rs = null;

    String sql = "SELECT tx_codigo_correlacao, tx_descricao_correlacao" +
                 "  FROM tb_contexto_usuario_correlacao" +
                 " WHERE co_contexto_usuario = " + coContexto +
                 " ORDER BY tx_codigo_correlacao";

    try
    {
      if(s == null) s = conexao.createStatement();
      rs = s.executeQuery(sql);
    }
    catch (java.sql.SQLException e)
    {
      System.out.println ("Erro ao executar: " + sql);
    }

    ArrayList result = new ArrayList();
    try
    {
      while (rs.next())
      {
        WSContexto c =  new WSContexto();

        c.setCodigoCorrelacao(rs.getString("tx_codigo_correlacao"));
        c.setDescricaoCorrelacao(rs.getString("tx_descricao_correlacao"));

        result.add(c);
      }
      fechaConexao(conexao);
      return result;
    }
    catch (java.sql.SQLException e)
    {
      System.out.println ("Erro na execucao do metodo WSContexto.listarCorrelacao(). " + e.getMessage());
      fechaConexao(conexao);
      return result;
    }
  }

  private ArrayList listarFonte(int coContexto)
   {
    String cmd = null;

    configuraConexao("localhost", "olheaki", "postgres", "123456");
    conexao = abreConexao();
```

```
    java.sql.ResultSet rs = null;

    String sql = "SELECT tx_fonte" +
                 "  FROM tb_contexto_usuario_fonte" +
                 " WHERE co_contexto_usuario = " + coContexto +
                 " ORDER BY tx_fonte";

    try
    {
      if(s == null) s = conexao.createStatement();
      rs = s.executeQuery(sql);
    }
    catch (java.sql.SQLException e)
    {
      System.out.println ("Erro ao executar: " + sql);
    }

    ArrayList result = new ArrayList();
    try
    {
      while (rs.next())
      {
        WSContexto c =  new WSContexto();
        c.setFonte(rs.getString("tx_fonte"));
        result.add(c);
      }
      fechaConexao(conexao);
      return result;
    }
    catch (java.sql.SQLException e)
    {
      System.out.println ("Erro na execucao do metodo WSContexto.listarFonte(). " + e.getMessage());
      fechaConexao(conexao);
      return result;
    }
  }
}


/*
 * Indexacao.java
 *
 * Created on 26 de Dezembro de 2006, 15:59
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package br.com.indexacao;

import br.com.bd.Configuracao;
import br.com.bd.PaginaIndexada;
import br.com.pesquisa.Pesquisa;

import br.com.geral.HtmlParser;

import java.io.File;
import java.io.IOException;
import java.io.FileReader;
```

```
import java.util.ArrayList;
import java.util.StringTokenizer;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.commons.httpclient.DefaultHttpMethodRetryHandler;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpException;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.params.HttpMethodParams;

/*Classes Lucene*/
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.Hits;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.Term;
import org.apache.lucene.index.TermDocs;
import org.apache.lucene.document.Field.Store;
import org.apache.lucene.index.IndexModifier;
import org.apache.lucene.queryParser.ParseException;

import org.htmlparser.beans.StringBean;
import org.htmlparser.util.ParserException;

/**
 *
 * @author fabiop
 */
public class Indexacao extends Thread
{
  /*tipos de recursos indexados*/
  private static final String TIPO_URL    = "1";
  private static final String ARQUIVO_TXT = "2";
  private static final String ARQUIVO_DOC = "3";
  private static final String ARQUIVO_PDF = "4";
  private static final String ARQUIVO_RTF = "5";

  public void iniciaIndexacao() throws IOException, ParseException, ParserException
  {
    boolean reindexar = false;
    IndexModifier index = manipulaArquivoIndice();
    index.setMaxFieldLength(1000000);

    PaginaIndexada pag = new PaginaIndexada();
    ArrayList lista = pag.listar();
    for (int j = 0; j < lista.size(); j++)
    {
      PaginaIndexada p = (PaginaIndexada)lista.get(j);

      Indexacao i = new Indexacao();

      int coPagina    = p.getCodigo();
```

```
    String endereco = p.getEndereco();
    String titulo   = i.getTitulo(endereco);
    titulo          = titulo.replace("'", "''");
    String html     = "";//getCodigoHTML(endereco);
    //html          = html.replace("'", "''");

    /*
    //Trecho retirado da internet: http://javaboutique.internet.com/tutorials/HTMLParser/index-2.html
    StringExtractor se = new StringExtractor(endereco);
    String conteudo = se.extractStrings(true);
    conteudo = conteudo.replace("'", "''");

    Pattern pattern = Pattern.compile("<[^>]*>");
    Matcher m = pattern.matcher(conteudo);

    //Substitui os links por espaço em branco
    conteudo = m.replaceAll(" ");
    */

    //Trecho retirado do documento da API do HTMLParser
    StringBean sb = new StringBean();
    sb.setLinks(false);
    sb.setReplaceNonBreakingSpaces(true);
    sb.setCollapse(true);
    sb.setURL (endereco); // the HTTP is performed here
    String conteudo = sb.getStrings();
    conteudo = conteudo.replace("'", "''");

    if (titulo != null && conteudo != null)
    {
      if (existeIndiceURL(coPagina))
         reindexar = true;
      i.indexaURL(coPagina, endereco, titulo, conteudo, html,
                index, reindexar);
    }
  }
  index.optimize();
  index.close();
}

public static String[] setStopWords(Vector v)
{
  //Insere STOPWORDS num array
  String stopWords[] = new String[v.size()];
  for (int i = 0; i < v.size(); i++)
  {
     stopWords[i] = v.get(i).toString();
  }
  return stopWords;
}

public static Vector getStopWords()
{
  Vector v = new Vector();

  //Carrega as STOPWORDS
  Configuracao conf = new Configuracao();
  ArrayList lista = conf.listar();
  for (int i = 0; i < lista.size(); i++)
  {
```

```
      Configuracao c = (Configuracao)lista.get(i);
      v.add(c.getStopWord());
    }

    return v;
  }

  private static String getCodigoHTML(String url)
  {
    String html = null;
    // Create an instance of HttpClient.
    HttpClient client = new HttpClient();

    // Create a method instance.
    GetMethod method = new GetMethod(url);

    //Provide custom retry handler is necessary
    method.getParams().setParameter(HttpMethodParams.RETRY_HANDLER, new
DefaultHttpMethodRetryHandler(3, false));

    try
    {
      // Execute the method.
      int statusCode = client.executeMethod(method);

      if (statusCode != HttpStatus.SC_OK)
        return html;

      //html = method.getResponseBodyAsStream().toString();
      byte[] responseBody = method.getResponseBody();
      html = new String(responseBody);

      return html;
    }
    catch (HttpException e)
    {
      //System.err.println("Fatal protocol violation: " + e.getMessage());
      //e.printStackTrace();
      return html;
    }
    catch (IOException e)
    {
      //System.err.println("Fatal transport violation: " + e.getMessage());
      //e.printStackTrace();
      return html;
    }
    finally
    {
      // Release the connection.
      method.releaseConnection();
    }
  }

  private static boolean existeIndiceURL(int coPagina) throws ParseException, IOException
  {
     boolean existeIndice = false;

     Pesquisa p = new Pesquisa();
     existeIndice = p.existeIndice(coPagina);
```

```
    return existeIndice;
}

private static IndexModifier manipulaArquivoIndice() throws IOException
{
    boolean criaIndice = true;

    //Recupera o caminho onde os índices serão gerados
    Configuracao conf = new Configuracao();
    conf.consulta();
    File caminho = new File(conf.getTxCaminhoIndice());

    if (!caminho.exists() || !caminho.isDirectory())
    {
        System.out.println("Caminho inexistente.");
        criaIndice = false;
    }

    //Cria o objeto do índice
    Vector v = getStopWords(); //Lê da base de dados as stop words
    String[] stopWords = setStopWords(v); //Define as stop words
    StandardAnalyzer analyzer = new StandardAnalyzer(stopWords);

    //Verifica se o arquivo .cfs(arquivo de índices) existe
    if (existeArquivoIndice(caminho))
      criaIndice = false;
    else
      criaIndice = true;

    IndexModifier index = new IndexModifier(caminho, analyzer, criaIndice);
    return index;
}

private static boolean existeArquivoIndice(File caminho) throws IOException
{
   boolean existe = false;
   File[] files = caminho.listFiles();

   for (int i=0; i < files.length; i++)
   {
        File f = files[i];

        if (f.getName().endsWith(".cfs"))
        {
           existe = true;
        }
   }
   return existe;
}

private String getTitulo(String url)
{
   //Recupera o título da URL
   HtmlParser t = new HtmlParser();
   String titulo = t.getTitulo(url);

   return titulo;
}

private void indexaURL(int co_pagina, String url, String titulo, String conteudo, String html,
```

```
                    IndexModifier index, boolean reindexar) throws IOException
  {
    try
    {
      int indiceApagados;
      Integer coPagina = new Integer(co_pagina);

      //Insere conteúdo da página na base de dados
      PaginaIndexada p = new PaginaIndexada();
      p.insereConteudo(co_pagina, titulo, conteudo, html);

      if (reindexar) //Deve efetuar a reindeixação => apagar índice existente
        indiceApagados = index.deleteDocuments(new Term("co_pagina", coPagina.toString()));

      //Efetua a indexação das páginas
      Document doc = new Document();
      doc.add(new Field("co_pagina", coPagina.toString(), Field.Store.YES, Field.Index.UN_TOKENIZED));
      doc.add(new Field("endereco", url, Field.Store.YES, Field.Index.TOKENIZED));
      doc.add(new Field("titulo", titulo, Field.Store.YES, Field.Index.TOKENIZED));
      doc.add(new Field("conteudo", conteudo, Field.Store.YES, Field.Index.TOKENIZED));

      index.addDocument(doc);
    }
    catch(IOException e)
    {
      System.err.println("Erro: " + e.getMessage());
    }
  }
}
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo