

INTEGRANDO MOTORES DE INDEXAÇÃO DE DADOS PARA A CONSTRUÇÃO DE SISTEMAS DE RECUPERAÇÃO DE INFORMAÇÃO EM AMBIENTES HETEROGÊNEOS

*INTEGRATING DATA INDEX ENGINES IN ORDER TO BUILD
INFORMATION RETRIEVAL SYSTEMS IN HETEROGENEOUS
ENVIRONMENTS*

André Mano Amazonas

Universidade Salvador, Bahia, Brasil

Daniel Souza Santos Ribeiro

Universidade Salvador, Bahia, Brasil

Diego Souza de Barros

Universidade Salvador, Bahia, Brasil

Manoel Carvalho Marques Neto

Universidade Salvador, Bahia, Brasil

Celso Alberto Saibel Santos

Universidade Salvador, Bahia, Brasil

Recebido em/*Manuscript first received*: 24/01/2008 Aprovado em/*Manuscript accepted*: 18/07/2008
Endereço para correspondência/*Address for correspondence*

André Mano Amazonas, Bacharel em Ciência da Computação (Universidade Federal da Bahia, Brasil) e pesquisador da Universidade Salvador, Bahia, Brasil, CPERC, Rua Ponciano De Oliveira, 126, Rio Vermelho CEP 41950-275, Salvador - BA, Tel: +55(71)3330-4655, Fax:+55(71)3330-4630, E-mail: andreamazonas@gmail.com

Daniel Souza Santos Ribeiro, Bacharel em Ciência da Computação (Universidade Salvador, Bahia, Brasil) e atualmente programador da Portugal Telecom Inovação SA, Tel: +55(71)8193-0213, E-mail: danielssribeiro@gmail.com

Diego Souza de Barros, Bacharelando em Ciência da Computação pela Faculdade Ruy Barbosa, Salvador, Bahia, Brasil, CPERC, Rua Ponciano De Oliveira, 126, Rio Vermelho CEP 41950-275, Salvador - BA, Tel: +55(71)3330-4655, Fax:+55(71)3330-4630 E-mail: diegosb87@gmail.com

Manoel Carvalho Marques Neto, Mestre em Redes de Computadores (Universidade Salvador, Bahia, Brasil) doutorando em Ciência da Computação pela UFBA – UNIFACS – UEFS e pesquisador da Universidade Salvador, Bahia, Brasil, CPERC, Rua Ponciano De Oliveira, 126, Rio Vermelho CEP 41950-275, Salvador - BA, Tel: +55(71)3330-4655, Fax:+55(71)3330-4630, E-mail: manoel.netom@unifacs.br

Celso Alberto Saibel Santos, Docteur em Informática (Université Paul Sabatier de Toulouse) e Prof. Titular II, Universidade Salvador, Bahia, Brasil, CPERC, Rua Ponciano De Oliveira, 126, Rio Vermelho CEP 41950-275, Salvador - BA, Tel: +55(71)3330-4640, Fax:+55(71)3330-4630, E-mail: saibel@unifacs.br

ISSN online: 1807-1775

Publicado por/*Published by*: TECSI FEA USP – 2008

ABSTRACT

The different proposals for information recovery put forward a set of limitations about the environments which they recover information from, they usually restrict the information recovery to a very specific data source or knowledge area. In this context, this paper proposes a model and a framework in order to integrate different index engines to enable the development of Information Retrieval Systems capable of retrieving information in heterogeneous data sources, and, therefore, showing up as a solution to fulfill the demand to access information in corporate environments.

Keywords: information retrieval, index engine, IRS, heterogeneous data sources, integration.

RESUMO

As diversas propostas existentes para a recuperação de informação apresentam limitações referentes ao ambiente em que atuam, restringindo a recuperação de informações a fontes de dados ou a áreas do conhecimento específicas. Neste contexto, este artigo propõe um modelo e um arcabouço para a integração de motores de indexação a fim de possibilitar o desenvolvimento de Sistemas de Recuperação de Informação (SRIs) capacitados a recuperar informações em ambientes de fontes de dados heterogêneas, apresentando-se assim, como uma abordagem para atender a demanda de acesso às informações em ambientes corporativos.

Palavras-chave: recuperação de informação, motor de indexação, SRI, fontes de dados heterogêneas, integração.

1. INTRODUÇÃO

A chamada “Era Digital” se caracteriza como uma época na qual a informação (conhecimento) em uma forma eletronicamente acessível (i.e. digitalizada) pode ser acessada, compartilhada e utilizada de forma fácil, imediata e ampla em atividades econômicas. Na “Era Digital”, informação e conhecimento assumem um papel estratégico, alavancando novas possibilidades de crescimento em termos de produtos e serviços em empresas, governos e demais instituições (LAU, 2003). Os produtos e serviços oferecidos por estas organizações tornam, assim, dependentes do capital intelectual dos indivíduos que nelas trabalham. Como consequência, a estruturação, o processamento e a utilização dos dados de uma organização para a geração de conhecimento é uma forma de obter benefícios próprios (MESQUITA MOTA, 2003). Segundo Drucker (1987), o funcionamento das companhias é dependente do processamento de informação. Neste contexto, Mesquita Mota (2003) observa que a gestão da informação é um requisito fundamental para o processo de geração de conhecimento.

Outro fato importante é que a utilização da informação como recurso básico para as atividades econômicas em conjunto com a popularização de tecnologias de comunicação têm como consequência direta o aumento da quantidade de informações digitais geradas, seja por usuários comuns, seja por organizações. Segundo Gantz (2007), se fosse possível imprimir em papel todos os *exabytes*¹ de informação digital

¹ 1 Exabyte (EB) = 1024³ Gigabyte (GB) = 1024⁶ bytes;

gerada, capturada e replicada em 2006, seria possível embrulhar a Terra quatro vezes.

Apesar de parecer extraordinário, em 2009, o tamanho desse universo será de 988 *exabytes*, com uma taxa de crescimento anual de 57%, sendo possível embrulhar a Terra vinte e cinco vezes.

As informações geradas por uma organização podem ser provenientes de diferentes fontes, tais como documentos de texto, planilhas, arquivos de imagem, áudio e vídeo gerados pela própria organização; documentos, e-mails e páginas pessoais dos membros; e bases de dados de aplicações corporativas em bancos de dados relacionais ou orientados a objeto. A crescente expansão do universo digital, aliada à heterogeneidade das fontes de dados e à falta de controle e organização dos dados gerados, torna a utilização desses dados e sua transformação em informação útil cada vez mais complexa. Este fato se apresenta como um importante problema a ser solucionado em grande parte das organizações que utilizam tais informações como insumo. Sem uma solução adequada, muitas das informações geradas por uma organização, que poderiam ser de grande valia para o seu processo de inovação, acabam sendo aproveitadas de modo superficial e insuficiente, ou até mesmo, nem são utilizadas (GANTZ, 2007).

Algumas propostas buscam solucionar o problema de acesso a essas informações, sem, contudo, tratar a questão da heterogeneidade do ambiente informacional como problema central. A consequência disso é que o universo de documentos (e, portanto, de informações) a serem recuperados tende a ser limitado. Exemplos de propostas neste sentido são os Sistemas de Recuperação de Informação (SRIs) apresentados por Amorim (2007), Nascimento (2004) e Nunes (2007). Estas soluções direcionam o foco para áreas específicas do conhecimento, o que limita ainda mais a sua abrangência. A solução de Amorim (2007) é direcionada para as áreas de Arquitetura, Engenharia e Construção. A solução apresentada por Nascimento (2004) delimita a recuperação de informações em documentos relacionados à Construção Civil. Por outro lado, a solução de Nunes (2007) é direcionada para a resolução de problemas na área da Jurisprudência.

Outros trabalhos propõem uma arquitetura específica para a realização de recuperação de informações, porém fazem uso de apenas um motor de indexação², limitando a abrangência da solução às fontes de dados suportadas por este motor, como no modelo apresentado por Beppler (2005), por exemplo. Outros ainda utilizam a integração de diferentes motores de busca, como a solução de Ma (2005), mas não possibilitam a realização de indexação e busca de informações em diferentes fontes de dados, pois se limitam apenas ao ambiente *Web*.

Nenhuma das propostas apresentadas soluciona o problema da heterogeneidade do ambiente, uma vez que todas elas se limitam à indexação e à busca de informações em fontes de dados específicas. Há ainda algumas propostas comerciais para solução do problema da recuperação de informações. Entre elas estão softwares baseados em

² Um motor de indexação é um software capaz de organizar os dados de forma estruturada para facilitar o processamento destes.

indexação para ambientes *desktop* como: o *Windows Desktop Search* (MICROSOFT, 2007a) e o *Google Desktop Search* (GOOGLE, 2007a). Estes softwares oferecem uma interface amigável para o usuário, mas também não solucionam o problema de recuperação de informações em fontes de dados heterogêneas. O *Google Desktop Search* oferece possibilidade de extensão limitada, não sendo possível a indexação de informações em fontes que não façam parte de uma das categorias pré-definidas por ele. O *Windows Desktop Search*, por sua vez, oferece extensão para a indexação e busca de informações em qualquer fonte de dados. No entanto, exige esforço de programação para possibilitar a indexação e busca de informações em algumas fontes, como em bancos de dados relacionais, por exemplo.

Diante do panorama apresentado, este artigo tem como objetivo propor uma solução para o problema de recuperação de informações em ambientes de fontes de dados heterogêneas. A solução está baseada na construção de um arcabouço (*framework*) capaz de integrar tecnologias de indexação e busca de informações de forma transparente. Este arcabouço possibilita a construção de SRIs capazes de recuperar informações em ambientes de fontes de dados heterogêneas. O desenvolvimento deste arcabouço foi baseado em técnicas já consolidadas de sistemas orientados a objeto, bem como padrões de projeto (GAMMA, 1994 e FREEMAN, 2004). E, por isso, conta com os benefícios que são oferecidos por eles. A solução proposta possui dois pontos fortes: 1) Permite maior abrangência na indexação e busca em um ambiente informacional, o que possibilita a indexação de dados provenientes de diversas fontes, inclusive de bases em bancos de dados relacionais, objeto-relacionais ou orientados a objeto; 2) dá maior flexibilidade na escolha dos motores de indexação a serem utilizados, fornecendo uma maneira simples de substituição dos motores de indexação utilizados, caso necessário.

Este artigo está organizado em seis seções, além da Introdução. A seção 2 apresenta de forma sucinta a metodologia utilizada no desenvolvimento trabalho. Na seção seguinte, alguns trabalhos correlatos à recuperação de informação são analisados. Na seqüência do texto, a seção 4 descreve em detalhes o desenvolvimento da proposta, apresentando a visão geral da arquitetura proposta e os principais componentes do arcabouço para integração de tecnologias de indexação e busca de informações. As seções 5 e 6 apresentam, respectivamente, as conclusões e trabalhos futuros. Por fim, a seção 7 apresenta as referências utilizadas no artigo.

2. METODOLOGIA

O estudo exploratório apresentado neste artigo teve início com uma pesquisa bibliográfica sobre os aspectos e os conceitos relacionados ao tema central, com o intuito de construir um embasamento teórico que serviria de suporte às etapas posteriores. A segunda fase do estudo consistiu na concepção e desenvolvimento do arcabouço de integração. Por fim, foi desenvolvida uma aplicação exemplo, que teve como principal objetivo a validação do arcabouço de integração proposto.

3. TRABALHOS CORRELATOS

A integração de tecnologias em ambientes heterogêneos e a recuperação de informações têm sido estudadas por diversos autores. Alguns deles propõem sistemas que têm como foco a recuperação de informação. Outros propõem modelos de integração de tecnologias para solucionar o problema da heterogeneidade das fontes de dados em diversos contextos. Existem ainda alguns trabalhos que possuem, ambas características, a recuperação de informação e a integração de tecnologias.

Os sistemas de recuperação de informação de Amorim (2007), Nascimento (2004) e Nunes (2007) fazem parte do primeiro grupo. Amorim (2007) apresenta um SRI que tem como objetivo melhorar a precisão e revocação³ na busca de informações relacionadas à área de Arquitetura, Engenharia e Construção (AEC). Para isso, ele utiliza ontologias, que dão significado ao conteúdo dos documentos indexados. Nascimento (2004) também propõe um SRI ligado à área de AEC. No entanto, Nascimento (2004) utiliza algoritmos específicos que têm como base o conhecimento do domínio da construção civil a fim de obter maior eficácia na busca dos dados. Nunes (2007) apresenta um SRI que utiliza uma abordagem bem parecida com a de Amorim (2007), já que faz uso de ontologias e anotações semânticas para auxiliar na pesquisa por documentos, porém atua no campo da Jurisprudência.

Uchôa (1999) apresenta um arcabouço (*framework*) para integração de sistemas de bancos de dados heterogêneos. Este arcabouço orientado a objetos foi desenvolvido na linguagem C++ e permite que sejam criados sistemas para gerenciar bancos de dados heterogêneos. O trabalho vê a heterogeneidade dos dados como um problema, e busca solucioná-lo dentro do escopo dos sistemas gerenciadores de bancos de dados. Para tanto, o autor utiliza integração de tecnologias.

Ma (2005) também apresenta um modelo de integração de tecnologias. Em seu modelo, o autor apresenta uma maneira de construir um sistema de buscas na *Web* fazendo uso de outros sistemas de busca integrados, e, além disso, faz uso de conceitos de inteligência artificial para aprimorar a precisão nas consultas realizadas. Para a criação do seu modelo, MA (2005) utilizou como base a estrutura de um meta-motor de busca que pode ser observado na Figura 1. A estrutura apresentada pela Figura 1 define que o *Servidor Web* recebe a consulta do usuário e verifica se os mesmos resultados foram pesquisados recentemente. Caso tenham sido, o retorno é feito a partir da base de resultados. Caso contrário, a consulta é enviada para a *Interface de Processamento Web* que a executa paralelamente nos vários motores de busca que estão sendo utilizados. Os resultados destas consultas são estruturados e exibidos para o usuário, e também salvos na base de resultados para consultas posteriores.

³ *Precisão* - Relação entre o número de documentos relevantes recuperados e o número total de documentos recuperados. *Revocação* - Razão do número de documentos relevantes recuperados sobre o total de documentos relevantes disponíveis na base de dados (BAEZA-YATES, 1999).

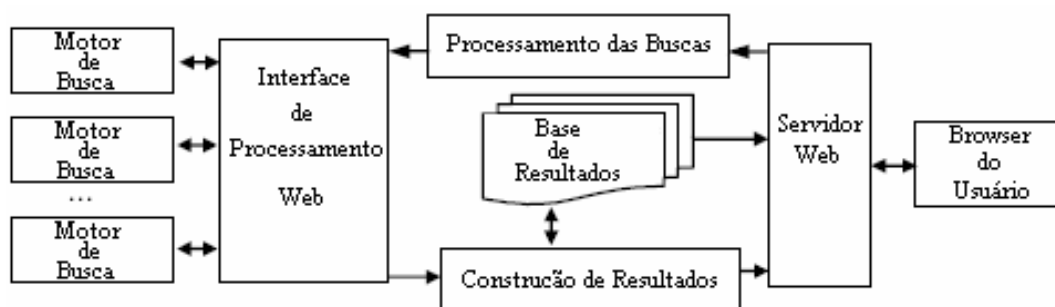


Figura 1 - Estrutura de um meta-motor de busca

4. DESENVOLVIMENTO DO TRABALHO

4.1. SISTEMAS DE RECUPERAÇÃO DE INFORMAÇÕES

4.1.1. TÉCNICAS DE RECUPERAÇÃO DE INFORMAÇÕES

Uma alternativa para o acesso às informações é a navegação pelos documentos, sem a necessidade de indexação prévia (SOUZA, 2006). No entanto, segundo Zobel (2006), esta alternativa somente é válida em sistemas onde o número de consultas para busca de informações é considerado pequeno, ou ainda, cujos dados são bastante voláteis. Em ambientes onde muitas requisições de busca são necessárias ou existe uma grande quantidade de dados, a utilização de técnicas de recuperação de informações baseadas em indexação se torna necessária, visto a economia de tempo em buscas posteriores.

Segundo Baeza-Yates (1999), a recuperação de informação lida com as tarefas de representação, armazenamento, organização e acesso aos itens que contém a informação. Neste conceito, Souza (2006) define os Sistemas de Recuperação de Informações como sistemas que organizam e viabilizam o acesso aos itens de informação ao desempenhar as tarefas citadas acima.

Os principais modelos de recuperação de informação são o booleano, o vetorial e o probabilístico. No modelo de recuperação booleano, a cada consulta realizada, são retornados todos os documentos que possuem os dados solicitados pelo usuário. Com o intuito de aperfeiçoar a busca, é possível que sejam utilizados os operadores *OR*, *AND* e *NOT* para correlacionar as palavras da pesquisa. Este modelo é baseado na teoria dos conjuntos e os documentos são qualificados em apenas duas categorias, se possuem ou não o termo pesquisado. Desta forma, não é possível medir grau de relevância dos documentos retornados através deste modelo de recuperação.

O modelo vetorial permite que seja determinado grau de relevância para cada documento, o que possibilita a criação de um *ranking*, e estabelece um critério mínimo de inclusão dos documentos no resultado das pesquisas. Isto é possível devido à representação de cada documento como vetores no espaço de dimensão n , onde n é o número de palavras contidas em todos os documentos que possuem representatividade no índice. A determinação do grau de relevância pode ser feita com base na distância vetorial entre os documentos e as consultas mapeadas no espaço n -dimensional. A fim de exemplificar, considere um índice com apenas três palavras indexadas (universidade, comunidade, digital) pelas quais os documentos presentes no mesmo podem ser

pesquisados. Neste caso, o espaço criado possuiria três dimensões. Considere também que existam três documentos neste índice (D1, D2 e D3) e que eles contenham, respectivamente, as seguintes palavras: universidade, comunidade; comunidade, digital; e universidade, comunidade, digital.

A Figura 2 mostra como esses documentos seriam mapeados no referido espaço, e como seriam calculadas as distâncias destes para uma consulta (com as palavras universidade e digital), também mapeada neste mesmo espaço. Estas distâncias determinam a relevância do documento para a consulta, quanto menor a distância entre o documento e a consulta, mais relevante ele é para esta. A Figura 2 evidencia que o documento D3 possui maior relevância para a consulta mapeada ($d3$ é menor que $d1$ e $d2$), e que D1 e D2 apresentam a mesma relevância para esta consulta ($d1$ é igual a $d2$).

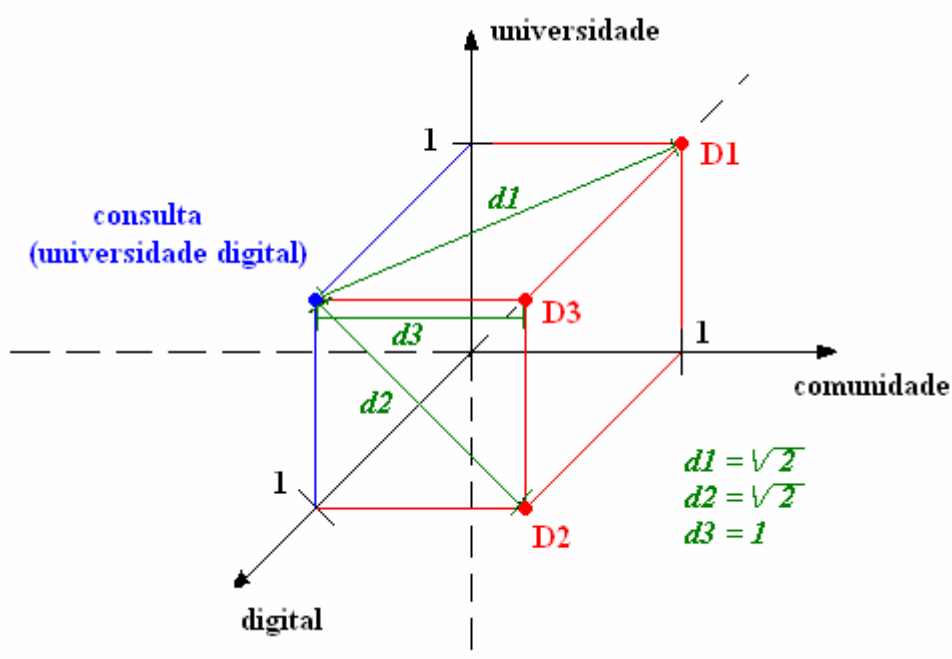


Figura 2 - Exemplo do modelo vetorial de recuperação de informações

No modelo probabilístico de recuperação de informações, supõe-se que exista um conjunto ótimo de documentos para cada pesquisa dos usuários, e que ele seja passível de recuperação. Com o intuito de obtê-lo, este modelo utiliza outro método de recuperação para obter uma lista inicial e, a partir dela, realizando interações sucessivas com os usuários, fazer análises de relevância dos documentos a serem retornados.

Atualmente, grande parte dos SRIs é baseada no modelo vetorial de recuperação de informações. Novas técnicas estão sendo estudadas, com o intuito de complementar os modelos de recuperação existentes, possibilitando a criação de SRIs mais eficientes. São alguns exemplos: 1) a identificação de padrões semânticos existentes nos documentos; 2) a utilização de metadados para facilitar a identificação dos documentos; 3) a criação de novas metáforas visuais, de modo a facilitar a extração de conhecimento das informações recuperadas; 4) a utilização de informações pessoais e comportamentais dos usuários, obtidas através de *websites* de relacionamento e interações, com o intuito de definir relevância específica para cada usuário; 5) a inserção de dados semânticos nos documentos, com o intuito de aumentar a precisão e a

revocação no retorno (SOUZA, 2006).

4.1.2. INDEXAÇÃO

A indexação de dados incorpora diversos conceitos multidisciplinares de lingüística, matemática, psicologia cognitiva, ciência da computação e outros. Além disso, envolve diversos aspectos relacionados a outras atividades de gestão do conhecimento, dentre os quais podem ser destacadas as técnicas de recuperação e de interpretação de informações a serem utilizadas e a estrutura de armazenamento dos dados indexados. Estes conceitos se inter-relacionam com o objetivo de possibilitar a recuperação das informações de forma rápida e precisa, poupando tempo e trabalho necessários à execução de tarefas que as utilizem como recurso (LIMA, 2003).

O processo de indexação pode ser executado sobre dados provenientes de diversas fontes, desde arquivos de texto até arquivos de vídeo, áudio e imagem. Apesar de ser possível ou necessário o registro de diferentes informações na indexação destes dados, o processo de indexação consiste basicamente do mesmo princípio em todas as ocasiões, ou seja, mapear os termos de um documento em uma estrutura de dados específica chamada de índice (ZOBEL, 2006).

Para efetuar a construção de um índice, quaisquer que sejam os dados a serem indexados são necessários os seguintes passos: 1) *Tokenize* (Análise Léxica), 2) *Analysis* (Retirada das palavras sem significado na linguagem natural) e 3) *Stemming* (Radicalização).

O *Tokenize* consiste na separação e armazenamento de cada token⁴. Nesta etapa, também é possível o registro de informações relevantes sobre cada um deles. Ao fim da etapa de *Tokenize*, obtém-se uma lista com todos os *tokens* de um documento. O processo *Analysis* retira os *tokens* pouco relevantes para as pesquisas da lista gerada na etapa anterior, como artigos, preposições, pontuações, espaços em branco, entre outros.

Os *tokens* são considerados relevantes ou não de acordo com a língua utilizada para escrever o documento que os contém (GOSPODNETIC, 2005). Além disso, nesta etapa, podem ser retirados os acentos e substituídas letras em maiúsculo de cada *token*, o que possibilita uma pesquisa mais abrangente nos documentos.

A etapa *Stemming* consiste em reduzir cada *token* remanescente das etapas anteriores em sua palavra base. Para isso, são retirados prefixos, sufixos, plural, bem como flexões verbais, e qualquer outra flexão existente na palavra original. Não se faz necessário que a palavra-base obtida pelo processo seja exatamente a raiz daquela palavra (de acordo com formalidades da língua). É satisfatório que palavras relacionadas ou derivadas possuam a mesma base dentro do índice. Dessa forma, é possível que seja efetuada a recuperação de palavras derivadas da mesma base ao se realizar uma única consulta (PENG, 2007 e GOSPODNETIC, 2005).

Os tipos de índices mais comuns são o Índice Invertido e o Índice Seqüencial. O Índice Invertido consiste no mapeamento de cada *token* para a lista de documentos aos quais ele pertence. Este índice pode se apresentar também como uma árvore binária, o

⁴ Uma seqüência de caracteres que pode ser tratada como uma unidade dentro da gramática de uma linguagem (APPEL, 2002).

que reduz o tempo das pesquisas. No entanto, a utilização de índices dispostos como árvores binárias aumenta a necessidade de alocação de espaço em memória para o armazenamento (CORMEN, 2002). A criação e atualização de um índice invertido exigem que sejam pesquisados todos os *tokens* a serem inseridos ou alterados, o que pode gerar um aumento de tempo considerável (*overhead*) no processo de indexação. Um exemplo de Índice Invertido pode ser observado na Figura 3 que apresenta uma lista de *tokens* à esquerda, sob o título ‘Palavras’, apontando, cada *token*, para a lista de documentos que o possuem (apresentadas de forma horizontal sob o título ‘Documentos’).

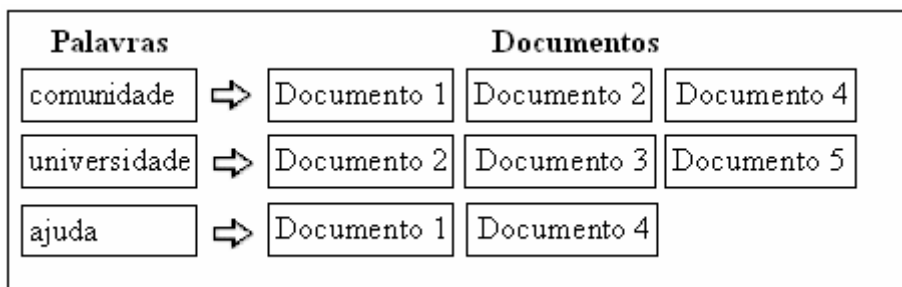


Figura 3 – Exemplo de Índice Invertido

O Índice Sequencial consiste de uma lista de pares (documento, *token*) ou (documento, lista de *tokens*), ordenados pelos documentos. Esta abordagem faz com que seja reduzido o *overhead* na inclusão e atualização existente nos Índices Invertidos. No Índice Sequencial, as inserções sempre ocorrem no final, e as atualizações ocorrem diretamente no par do documento a ser atualizado. Geralmente, um Índice Sequencial é transformado em Índice Invertido em tempo de execução com o intuito de reduzir o tempo de pesquisa.

Um exemplo de Índice Sequencial é apresentado na Figura 4 que apresenta à esquerda uma lista de documentos indexados que aponta, cada um deles, para a lista de *tokens* que possui.

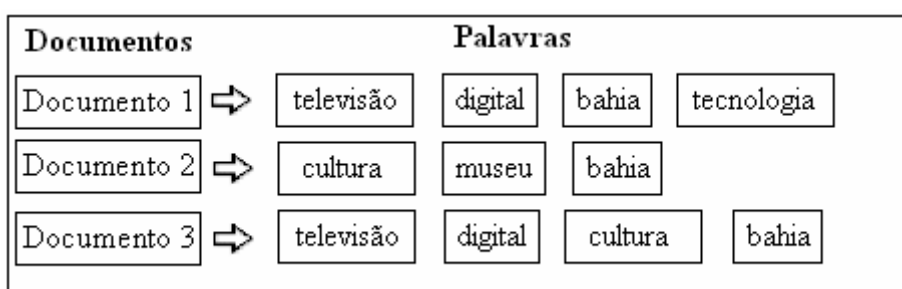


Figura 4 – Exemplo de Índice Sequencial

4.2. MOTORES DE INDEXAÇÃO DE DADOS

Os motores de indexação armazenam documentos em um repositório e mantém um índice desses documentos. As consultas são enviadas para este índice e os documentos que possuem as palavras consultadas são retornados. Estes documentos são recuperados através de diferentes técnicas e podem não conter necessariamente todos os termos consultados. Além disso, estes documentos são classificados através de alguma métrica

com o objetivo de retornar os documentos relevantes para o usuário (ZOBEL, 2006).

Com o objetivo de desenvolver o modelo de integração, os motores de indexação *Windows Search* (MICROSOFT, 2007a), o *Google Desktop Search* (GOOGLE, 2007a) e o *Lucene* (APACHE, 2007) foram estudados de modo a permitir que o projeto desenvolvido desse suporte às características comuns a eles e a outros motores de indexação que seguem o mesmo padrão. Isto possibilitou que o arcabouço de integração criado desse suporte ao desenvolvimento de SRIs utilizando alguns dos principais motores de indexação e busca (COLE, 2005).

O primeiro motor de indexação estudado foi o *Windows Search*, que é o motor de indexação de dados utilizado pelo sistema integrado de busca e organização do sistema operacional *Windows Vista*. Este motor também é utilizado pelo *Windows Desktop Search* (WDS), um software independente que pode ser instalado nos sistemas operacionais *Windows XP* e *Windows 2003 Server* (MICROSOFT, 2007a). O estudo realizado sobre o *Windows Search* foi baseado no *software* WDS.

O objetivo do WDS é fornecer administração dos dados digitais, indexando e buscando mais de 200 tipos de arquivos comumente utilizados. Dentre estes estão e-mails e contatos gerados pelos produtos da *Microsoft* como *Outlook Express* e *Microsoft Office*, documentos de texto, apresentações, metadados de arquivos de imagem e música e arquivos de código fonte. Outra característica desta aplicação é permitir a obtenção de dados presentes em arquivos da máquina local ou de alguma outra máquina da rede (MICROSOFT, 2007a).

O *Windows Search* disponibiliza um *Software Development Kit* (SDK) que pode ser utilizado para adicionar a funcionalidade de busca de informações a outras aplicações a serem desenvolvidas. Através deste SDK, o WDS pode ser estendido para possibilitar a indexação e busca em outras fontes de dados. Com o objetivo de facilitar o acesso aos dados indexados pelo *Windows Search*, o seu SDK torna indiferente tanto o tipo de dado retornado, como a aplicação que faz uso deste, possibilitando o acesso aos dados de maneira única.

Com o intuito de facilitar o desenvolvimento de eventos que exerçam as principais funcionalidades intrínsecas a um sistema de busca de informações, este SDK provê implementações de interfaces que permitem efetuar as operações de indexação, pesquisa, reindexação e remoção (desindexação) dos dados presentes no índice (MICROSOFT, 2007b).

Após o *Windows Search*, foi estudado o motor de indexação do *Google Desktop Search* (GDS). Este software é similar ao WDS e tem como objetivo facilitar o acesso a informações. O GDS é capaz de indexar e pesquisar informações em diversos tipos de arquivos, tais como: e-mails do *Microsoft Outlook E-mail*, *Outlook Express*, *Thunderbird* e *Netscape Mail*; arquivos do *Microsoft Word*, *Excel*, *PowerPoint* e em formato *pdf*; conversas do *Windows Messenger*, *AOL Instant Messenger* e *Google Talk*; histórico do *Internet Explorer*, *Firefox*, *Mozilla* e *Netscape*; metadados de arquivos de música, imagens e vídeo. A ferramenta também permite que sejam indexados arquivos de computadores remotos através de uma rede local (GOOGLE, 2007a).

Existem também algumas possibilidades de extensão das características do GDS com o uso do seu SDK que permite dois tipos de utilização: 1) o desenvolvimento de *plugins* para adicionar a capacidade de indexação de novos tipos de arquivos não

reconhecidos originalmente; e 2) o uso das funcionalidades de indexação e busca em outras aplicações. No entanto, não é possível a recuperação de informações em algumas fontes de dados através do GDS (como em bancos de dados, por exemplo). Apesar de ser extensível, este motor possui extensão limitada a categorias pré-definidas que não englobam todas as fontes existentes (GOOGLE, 2007b).

Por fim, foram estudadas as características do *Lucene*, um projeto de código aberto, licenciado sob a *Apache Software License*. Diferentemente dos *softwares* apresentados nas seções anteriores, o *Lucene* não apresenta uma interface de interação com o usuário. Isto ocorre porque o *Lucene* não é um *software* destinado ao usuário final, e sim uma biblioteca de recuperação de informação, com o objetivo de permitir que o programador adicione à sua aplicação a capacidade de indexar e buscar dados. Esses dados podem ser provenientes de diversas fontes devido ao fato deste motor não assumir qualquer característica do que será indexado. A única premissa é que os dados possam ser convertidos para um formato textual. No entanto, esta biblioteca não fornece suporte nativo a nenhum tipo de arquivo. A indexação de qualquer fonte de dados depende de rotinas de responsabilidade do programador (GOSPODNETIC, 2005).

Apesar de ser originalmente escrito em *Java*, atualmente, o *Lucene* está transcrito em outras linguagens, o que possibilita a sua utilização em um número maior de projetos. Algumas dessas linguagens são *C++*, *Perl*, *Python*, *.NET/C#* e *Ruby*. Isto significa que as funcionalidades da biblioteca *Lucene* podem ser exploradas através de qualquer uma dessas linguagens e, inclusive, por mais de uma ao mesmo tempo, já que os índices gerados são padronizados para a utilização por qualquer um desses projetos (GOSPODNETIC, 2005).

A Tabela 1 apresenta uma visão geral das principais características dos motores de indexação *Windows Search* (WS), *Google Desktop Search* (GDS) e *Lucene*. Foram levados em conta a existência de interface com o usuário, a possibilidade de extensão, a disponibilidade de SDK, as linguagens de programação disponíveis pelo SDK e as fontes de dados que são indexadas por padrão (sem a necessidade de extensão).

	Interface com o usuário	Extensível	Possui SDK	Linguagens de programação	Fontes de dados indexadas por padrão
WS	SIM	SIM	SIM	C++ e .NET(C#)	E-mail (eml, msg), Contatos (vcf), Documentos de texto (doc, dot, htm, html, mht, one, rtf, txt, xml), Planilhas (xls, xlw), Apresentações (pot, pps, ppt, xls, xlw), Pastas e Outros (bat, c, cmd, cpp, cxx, Dif, disco, h, hpp, hxx, idl, inc, inf, inx, js, nws, pl, ppa, pwz, rc, reg, resx, slk, url, vbs, xla, xld, xlt, xlv, xsl)
GDS	SIM	SIM *	SIM	C e Java (não oficial)	E-mail (Gmail, Outlook Email, Outlook Express, Netscape Mail, Thunderbird), Documentos de texto (Word), Planilhas (Excel), Apresentações (PowerPoint), Histórico de navegadores (Internet Explorer, Firefox, Mozilla, Nestcape), Conversas (Google Talk, MSN Instant

					Messenger, AOL Instant Messenger), Outros (arquivos pdf, zip)
Lucene	NÃO	SIM	SIM	Java, C++, Perl, Python, .NET/C# e Ruby	Todas as fontes cujos dados possam ser convertidos em formato textual **
* Extensibilidade limitada às categorias pré-definidas					
** A indexação de qualquer fonte de dados depende de rotinas de responsabilidade do programador					

Tabela 1- Comparação dos motores de indexação estudados

O estudo sobre estes motores de indexação evidenciou características específicas a cada um deles que limitam a indexação de informações em ambientes de fontes de dados heterogêneas utilizando apenas um deles. Algumas fontes de dados comuns em ambientes informacionais corporativos, como bancos de dados relacionais, não são suportadas originalmente para indexação e busca de informações por estes motores. O WDS pode ser estendido através do seu SDK para suportar tal característica ou pode ser desenvolvido um módulo que realize esta operação através do Lucene. O GDS não suporta este tipo de extensão.

Outras fontes de dados não suportadas para a recuperação de informações através destes motores são imagem, áudio e vídeo. Essas fontes são as principais responsáveis pelo crescimento do universo digital existente atualmente (GANTZ, 2007). Os motores de indexação estudados indexam somente os metadados destas fontes, não processando os dados contidos nelas. Assim, não é possível a recuperação de informações existentes nestes formatos através destes motores. Apesar de limitar a recuperação de informações em ambientes de fontes de dados heterogêneas, os motores estudados possuem propriedades fundamentais que facilitam a indexação de informações nestes ambientes, pois são capazes de executá-la sobre conjuntos de dados específicos.

4.3. O MODELO DE INTEGRAÇÃO

O modelo de integração de motores de indexação proposto tem o intuito de proporcionar o desenvolvimento de SRIs que visem solucionar o problema apresentado: a dificuldade de acesso às informações digitais em ambientes de fontes de dados heterogêneas. Este modelo propõe o trabalho conjunto e simultâneo de diferentes motores de indexação, e a independência destes em relação às lógicas de negócio das aplicações desenvolvidas sobre o modelo. Esta seção apresenta o modelo de integração proposto, destacando a arquitetura sobre a qual ele foi projetado e o arcabouço de integração que proporciona o uso conjunto de diferentes tecnologias de indexação e busca de informações.

4.3.1. A ARQUITETURA

Com o objetivo de integrar diferentes tecnologias de indexação e busca, o modelo proposto é dividido em três camadas independentes: 1) um universo heterogêneo de dados; 2) os motores de indexação e busca; e 3) o arcabouço de integração. Com isso, a arquitetura proposta, apresentada na Figura 5, é suficientemente flexível.

Na Figura 5, a primeira camada representa um universo heterogêneo de dados, análogo a de um ambiente corporativo, onde os dados serão indexados e recuperados.

Este universo pode ser composto por dados advindos de diversas fontes, tais como documentos de texto ou planilha, páginas *Web*, arquivos de imagem, áudio e vídeo, bancos de dados relacionais ou orientados a objeto, dentre outras. As diferentes tecnologias utilizadas para o propósito de indexação e busca de dados – os motores de indexação – compõem a segunda camada do modelo. Apesar de todos os motores serem responsáveis por indexar e pesquisar conteúdo, eles exercem funções distintas dentro dessa camada ao indexar fontes de dados específicas. Para exemplificar a distinção de tarefas dos motores de indexação, pode-se imaginar um SRI que faça uso do modelo proposto e utilize dois motores de indexação para indexar diferentes fontes de dados. O primeiro responsável pela indexação de arquivos comuns em um disco. E o segundo por indexar informação contida em um banco de dados relacional.

A terceira camada consiste no arcabouço de integração, sendo a mais importante do modelo proposto. Esta camada possui os objetos que propiciam a atuação conjunta das diferentes tecnologias de indexação utilizadas. Além disso, ela permite que a indexação e a recuperação dos dados sejam realizadas de maneira transparente para os programadores e para os usuários dos SRIs desenvolvidos sobre o modelo de integração. Devido à sua relevância, o arcabouço de integração será tema exclusivo da seção seguinte.

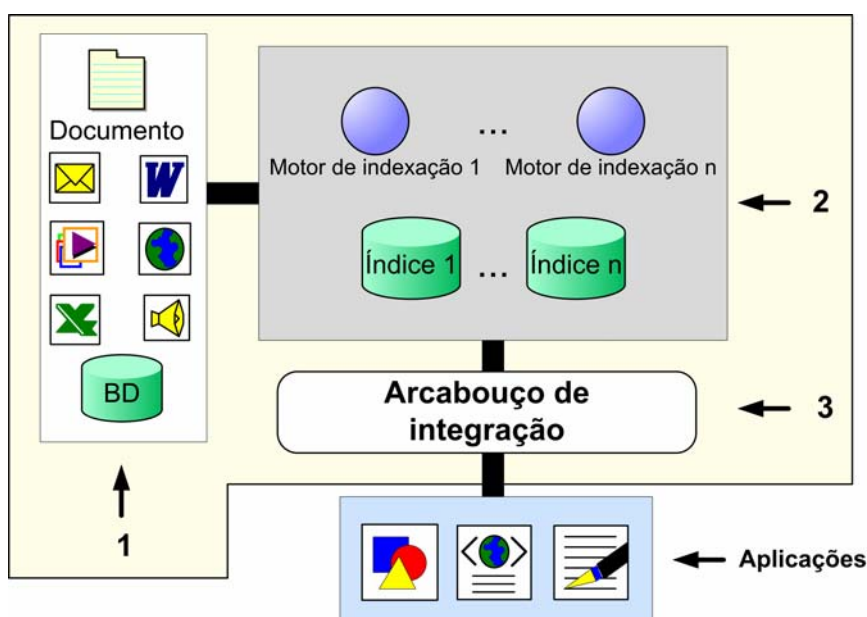


Figura 5 - Arquitetura proposta para um Sistema de Recuperação de Informações (SRI)

Abaixo do arcabouço de integração podem existir diversas aplicações com o objetivo de indexar e recuperar as informações e, possivelmente, extrair conhecimento destas. Essas aplicações formam uma camada de interação com os usuários, que, propositalmente, não faz parte do modelo proposto. Os dados recuperados podem ser apresentados em diferentes metáforas visuais, processados sob algoritmos de Mineração de Dados com o intuito de efetuar a extração de conhecimento, ou ainda serem disponibilizados através de serviços *Web* (*WebServices*). Essas são apenas algumas das possibilidades de tratamento dos dados recuperados. A definição final do processamento destes dados fica a cargo do desenvolvedor e dos requisitos dos usuários do sistema.

Conforme apresentado, o modelo de integração proposto possibilita a recuperação de informações provenientes de diversas fontes de maneira simultânea. Isto é consequência da integração de dois ou mais motores na indexação e busca de dados. Dessa forma, o modelo proposto se apresenta como uma solução para a recuperação de informações em ambientes de fontes de dados heterogêneas. Outra propriedade do modelo é a possibilidade de recuperação dos dados de forma padronizada. Isso ocorre independentemente de quais ou quantos são os motores de indexação utilizados, mesmo que eles forneçam as respostas às consultas em um formato diferente. Com isso, torna-se possível realizar operações sobre os dados sem a necessidade de conhecimento do motor de indexação responsável pela recuperação dos mesmos.

4.3.2. O ARCABOUÇO DE INTEGRAÇÃO

A Figura 6 apresenta o arcabouço de integração que compõe a terceira camada da arquitetura apresentada na seção anterior. O arcabouço é composto por três módulos: Indexação, Busca e *Factory*. Estes módulos foram desenvolvidos sob o paradigma de orientação a objetos, tendo como base conceitos já consolidados da Engenharia de Software, bem como de Padrões de Projeto. Assim, o modelo de integração se aproveita dos benefícios consequentes das melhores práticas de desenvolvimento de software.

O módulo de indexação é composto pelas interfaces *IIndex*, *IDocumentIndex* e *IDBIndex*. As interfaces *IDocumentIndex* e *IDBIndex* são especializadas de *IIndex*. Além dessas interfaces, o módulo de indexação ainda deve contar com a implementação de classes que cumpram o que é especificado nelas, para que a indexação dos dados seja possível. A principal finalidade dessas interfaces é definir o comportamento dos tipos de motores de indexação, e disponibilizá-los de modo que o programador não necessite ter conhecimento sobre qual motor será utilizado para uma indexação específica.

A interface *IIndex* especifica um comportamento geral que todos os motores de indexação possuem. Este comportamento é simbolizado, somente, pelo método *ClearIndex*, que tem como propósito efetuar a limpeza do índice gerado pelo respectivo motor de indexação que o implementar. Em trabalhos futuros, caso seja observado algum outro comportamento-padrão para todos os motores de indexação, deverá ser possível expressá-lo através desta interface.

A interface *IDocumentIndex* representa um padrão de procedimentos que são executados por motores de indexação de documentos comuns de um sistema de arquivos. Ela define o método *Index* que deve indexar o conteúdo de arquivos internos a um diretório-raiz que é recebido como parâmetro. É previsto que essa indexação aconteça, inclusive, em todos os arquivos contidos nos subdiretórios a partir deste diretório-raiz. Além do método de indexação, esta interface define o método *IncrementalIndex* que deve possibilitar a indexação incremental de um diretório recebido por parâmetro. A indexação incremental deve ser realizada sobre os arquivos que sofreram alterações ou que ainda não foram indexados.

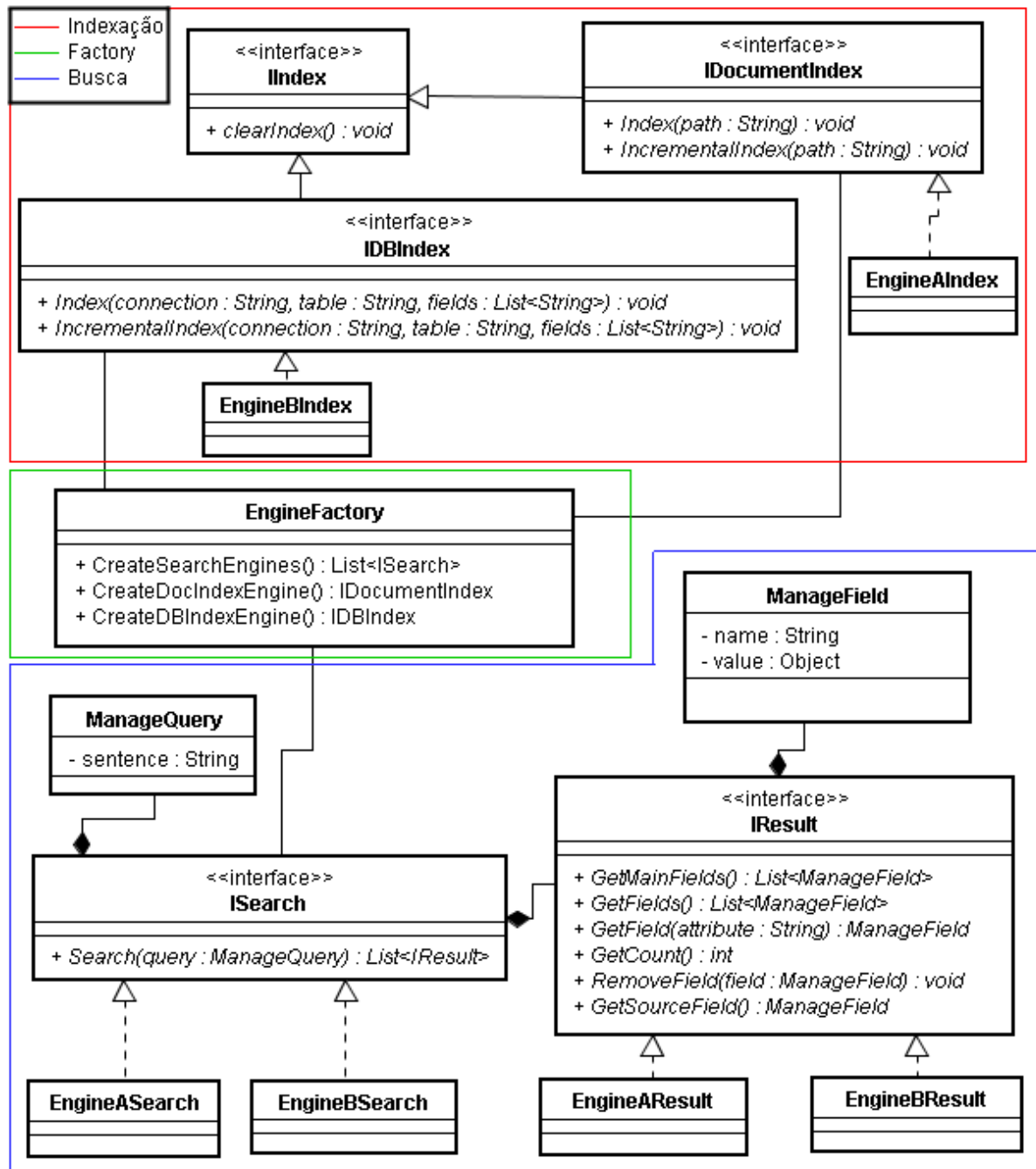


Figura 6 - Diagrama de classes do arcabouço de integração

A terceira interface deste módulo, a *IDBIndex*, define o formato de um motor de indexação de informações presentes em um banco de dados relacional. Assim como a *IDocumentIndex*, os métodos *Index* e *IncrementalIndex*, que recebem outros parâmetros, tem o objetivo de realizar a indexação de dados presentes em um banco de dados relacional.

As implementações das interfaces *IDocumentIndex* e *IDBIndex* são

representadas na Figura 6 por *EngineAIndex* e *EngineBIndex*, respectivamente. Estas implementações cumprem o que foi determinado pelas interfaces e delegam aos motores de indexação a tarefa de indexar os dados.

O módulo de busca é o que contém o maior número de classes. Este fato acontece porque ele tem a obrigação de cobrir uma maior quantidade de responsabilidades. Este módulo é formado pelas classes *ManageQuery* e *ManageField*, além das interfaces *ISearch* e *IResult* e suas respectivas implementações, como pode ser observado na Figura 6.

Estas interfaces, assim como as interfaces do módulo de indexação, representam abstrações e fornecem ao programador as mesmas características oferecidas pelo módulo de indexação. Ao contrário do que acontece no módulo de indexação, entretanto, os objetos definidos pelas interfaces deste módulo são fornecidos em forma de listas: lista de buscadores (*ISearch*) e lista de resultados (*IResult*) associada a cada um dos buscadores.

A interface *ISearch* tem como principais objetivos definir o comportamento dos motores de busca do sistema e possibilitar o acesso aos mesmos na camada seguinte da arquitetura de forma transparente, ou seja, sem que se saiba quais motores estão sendo utilizados para a realização das buscas. Ela também define o método *Search*, que é responsável por solicitar a pesquisa ao motor de busca relativo à implementação e retornar os valores encontrados no padrão definido pela interface *IResult*. O objeto que tem representação nessa interface é composto por uma lista desses resultados e por uma consulta (*ManageQuery*) que serão detalhados a seguir.

A consulta (*ManageQuery*) é um objeto que encapsula os dados a serem requisitados. Essa classe possui apenas o atributo *sentence*, que simboliza o texto que efetivamente será pesquisado pelos motores. Porém, a criação dessa classe tem o propósito de facilitar a extensão do arcabouço de integração para suportar futuras pesquisas possibilitadas pelo uso de outro motor ainda não conhecido ou pela evolução de algum dos motores estudados.

O padrão do resultado que será retornado por todos os buscadores é delineado pela interface *IResult*, como já foi explicitado. Definir essa padronização é o principal intuito desta interface. Dessa forma, todos os resultados advindos de motores de indexação diferentes podem ser submetidos ao mesmo tratamento na camada seguinte. Isto retira a importância do motor real e a transfere para sua abstração, como no princípio da inversão de dependência descrito por Freeman (2004). Um objeto que implementa a interface *IResult* é composto por uma lista de campos e possui os métodos apropriados para permitir operações sobre eles. Cada um desses campos representados pela classe *ManageField* possui um nome e um valor. Esse valor é definido genericamente para suportar os tipos de informações heterogêneas que podem estar contidas nesses campos, tais como: texto comum, números inteiros, números reais, datas, entre outros.

É importante ressaltar que para cada motor de indexação utilizado é indispensável que se tenha uma implementação da interface *ISearch* e outra da *IResult*. A implementação da interface *ISearch* permite a realização de consultas no seu respectivo motor de indexação, enquanto a implementação da interface *IResult* padroniza os resultados de um motor específico para que eles possam ser retornados.

O módulo *Factory* é constituído exclusivamente pela implementação da classe *EngineFactory*. Este módulo foi criado com o objetivo de encapsular a criação dos objetos do arcabouço de integração em uma única classe, permitindo o desenvolvimento voltado para interfaces e não para as implementações das classes concretas do arcabouço, caracterizando o princípio *Open-Closed* – código aberto para extensões, porém fechado para modificações (FREEMAN, 2004 e MEYER, 2000).

A criação deste módulo foi baseada nos padrões *Abstract Factory* e *Factory Method* (GAMMA, 1994; FREEMAN, 2004). Foi utilizada uma adaptação apresentada por Freeman (2004) como *Simple Factory*, que é uma prática muito utilizada para o encapsulamento da criação de objetos.

A classe *EngineFactory* é composta de três métodos, o *CreateSearchEngines*, o *CreateDocIndexEngine* e o *CreateDBIndexEngine*.

O *CreateSearchEngines* retorna uma lista com todas as instâncias dos buscadores dos motores utilizados. Dessa maneira, a busca pode ser realizada em todos os índices criados por esses motores, mantendo o conceito de índice único com o qual o arcabouço trabalha. Isto garante também a transparência na recuperação dos dados pela camada de utilização.

Os métodos *CreateDocIndexEngine* e *CreateDBIndexEngine* são responsáveis por instanciar o motor de indexação de documentos e o de banco de dados, respectivamente, retornando-os para a utilização do programador. Como foi dito anteriormente, estes motores têm o comportamento delineado pelas interfaces *IDocumentIndex* e *IDBIndex*.

Os métodos da classe *EngineFactory* asseguram a transparência dos motores utilizados pelo programador nas aplicações sobre o modelo. Isto permite que a indexação e a busca de informações sejam realizadas sem o conhecimento prévio sobre quais motores estão sendo utilizados em operações específicas.

4.3.3. A EXTENSÃO DO MODELO

A partir das definições dos módulos do arcabouço de integração apresentados na seção anterior é possível desenvolver uma aplicação que faça uso de dois ou mais motores de indexação e busca de maneira integrada. Todavia, podem ser necessárias extensões ou modificações dos motores utilizados nessa aplicação. Exemplos seriam a adição de um motor de indexação a uma aplicação já estruturada com dois ou mais motores e, também, a substituição de um dos motores utilizados.

Há ainda uma terceira possibilidade de alteração que é a introdução da capacidade de indexação de um novo tipo de dados não coberto pelo arcabouço atual. O arcabouço de integração foi projetado para permitir essas alterações de modo a não tornar necessária nenhuma alteração na codificação das classes de negócio da aplicação e garantir, dessa forma, a independência entre as camadas do modelo proposto. A análise de cada uma dessas modalidades de adaptação fornecidas pelo arcabouço é discutida na seqüência do texto.

A incorporação de um motor de indexação ao arcabouço é feita através de algumas implementações de interfaces já pertencentes aos módulos de busca e indexação e de pequenas alterações no módulo *Factory*.

No módulo de busca devem ser criadas implementações das interfaces *ISearch* e *IResult* relativas ao motor que se deseja adicionar. Já no módulo de indexação é preciso que seja implementada apenas uma das interfaces que defina o tipo de dados que serão indexados com o novo motor: *IDocumentIndex* ou *IDBIndex*. Essas alterações são ilustradas na Figura 7.

Para finalizar a inclusão de um novo motor, devem ser feitas alterações na classe *EngineFactory*. Em primeiro lugar, deve ser criado um novo método que seja responsável pela criação do objeto utilizado para realizar a indexação usando este motor. Além disso, o novo motor deve ser adicionado ao método *CreateSearchEngines* da classe *EngineFactory*, que é responsável por retornar os objetos que executam as buscas.

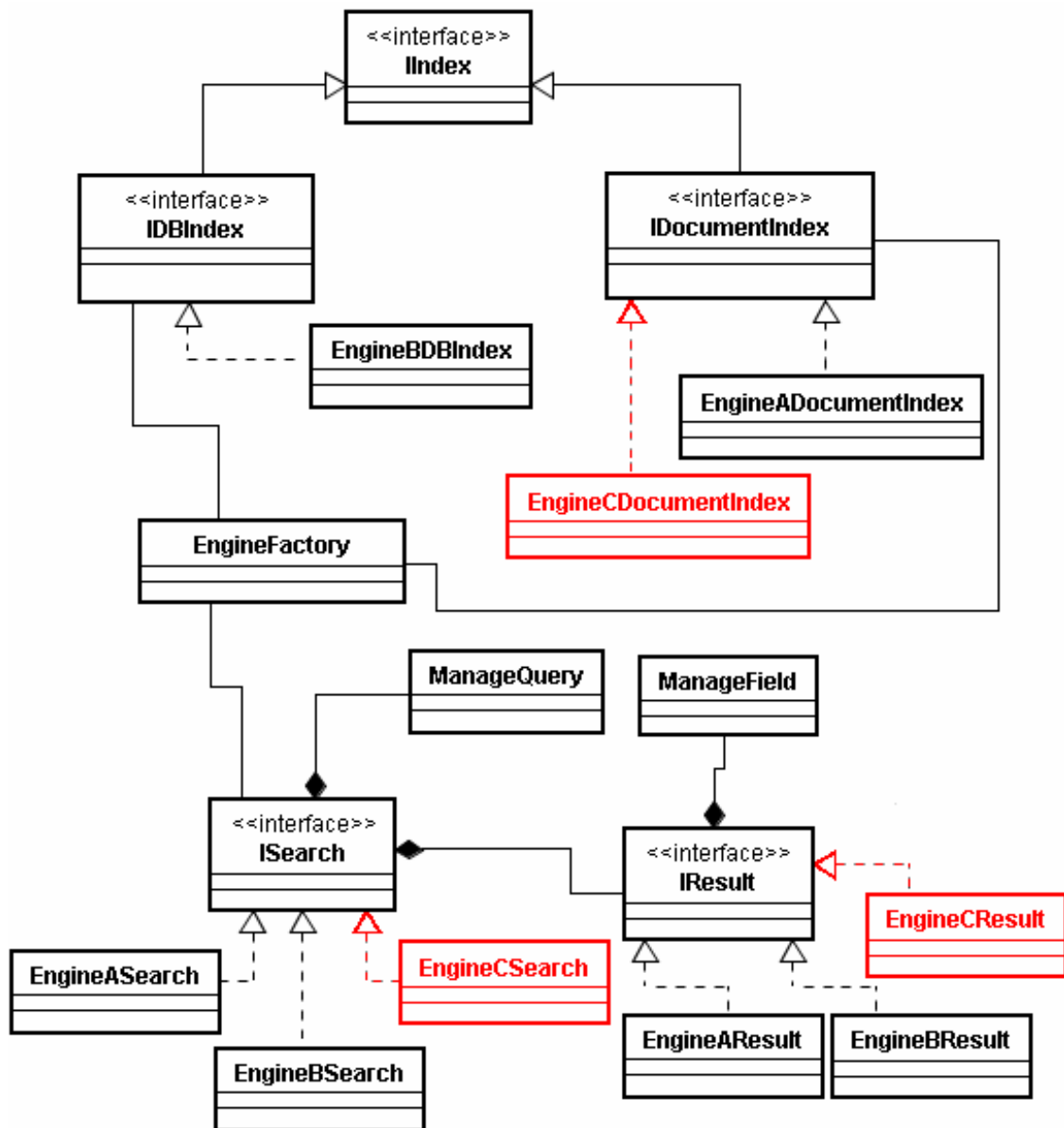


Figura 7 - Adição de um motor de indexação ao arcabouço de integração

A substituição de um dos motores que estão sendo utilizados tem um grau de complexidade baixo para quem possui conhecimento sobre os padrões de

desenvolvimento orientados a objeto. É preciso que sejam codificadas as mesmas implementações dos módulos de busca e indexação necessárias à inclusão de um novo motor. Entretanto, no módulo *Factory* deve-se apenas substituir a criação do motor de indexação que se deseja colocar pelo que dará lugar a ele, bem como adicionar o buscador da mesma forma que é feita na inclusão de um novo motor.

Na Figura 8, são exibidas em vermelho as classes que devem ser adicionadas e, em cinza, as classes que deixarão de ser utilizadas ao se efetuar a substituição do motor *EngineA* pelo *EngineC*.

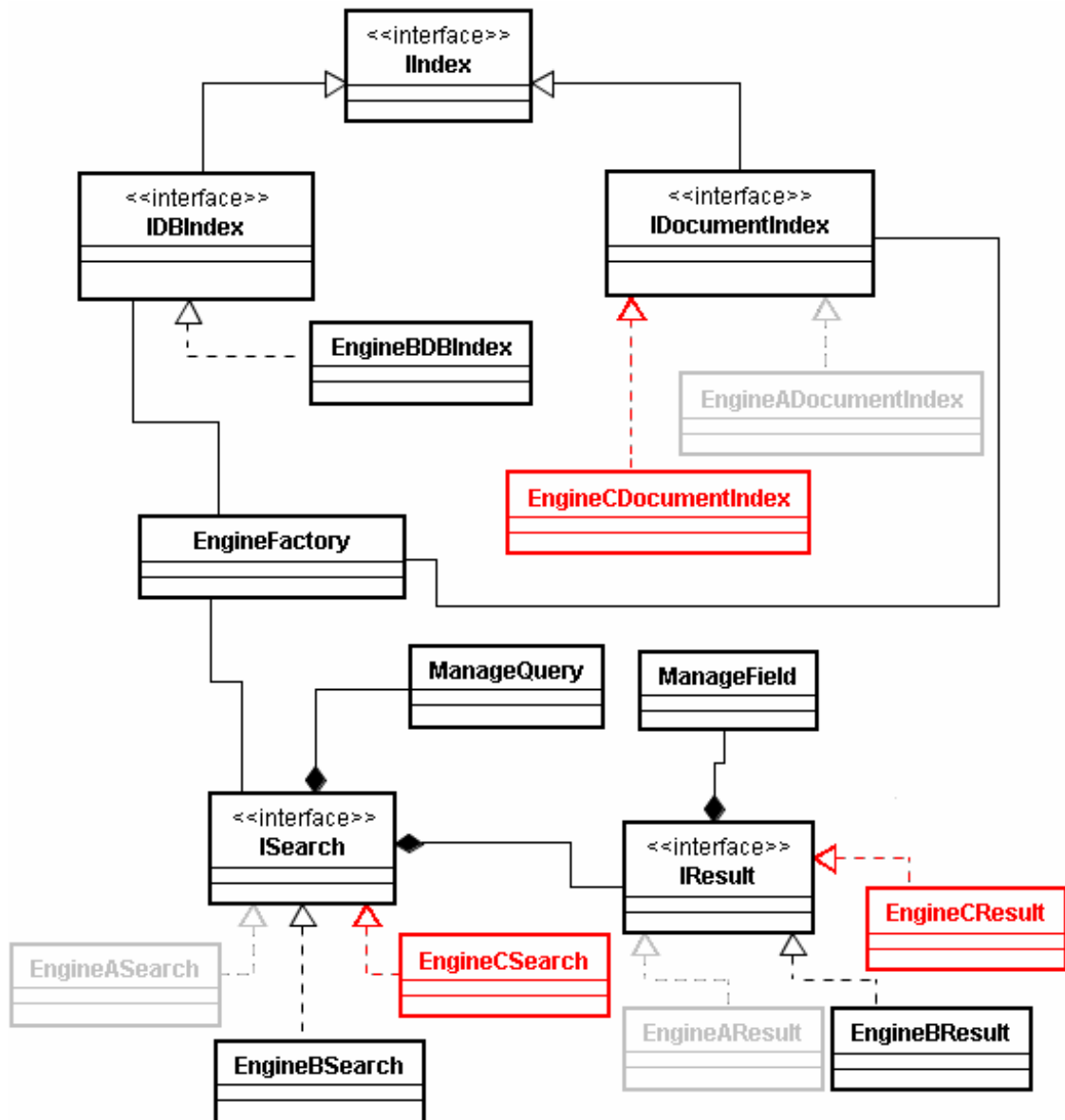


Figura 8 - Substituição de um motor de indexação no arcabouço de integração

A extensão com maior complexidade é a adição da indexação de novas fontes de dados não definidas inicialmente. Para tal, é preciso criar uma interface de indexação que herde as características de *IIndex* e que defina um padrão para a indexação dessa nova fonte de dados. Feito isso, o restante do processo compreenderia praticamente aos

mesmos passos realizados para a incorporação de um novo motor ao arcabouço.

A única diferença é que a implementação do módulo de indexação deverá seguir a nova interface criada. Novamente em vermelho, são apresentadas, na Figura 9, as alterações necessárias para realização da inclusão da capacidade de indexação e busca de informações em uma nova fonte de dados.

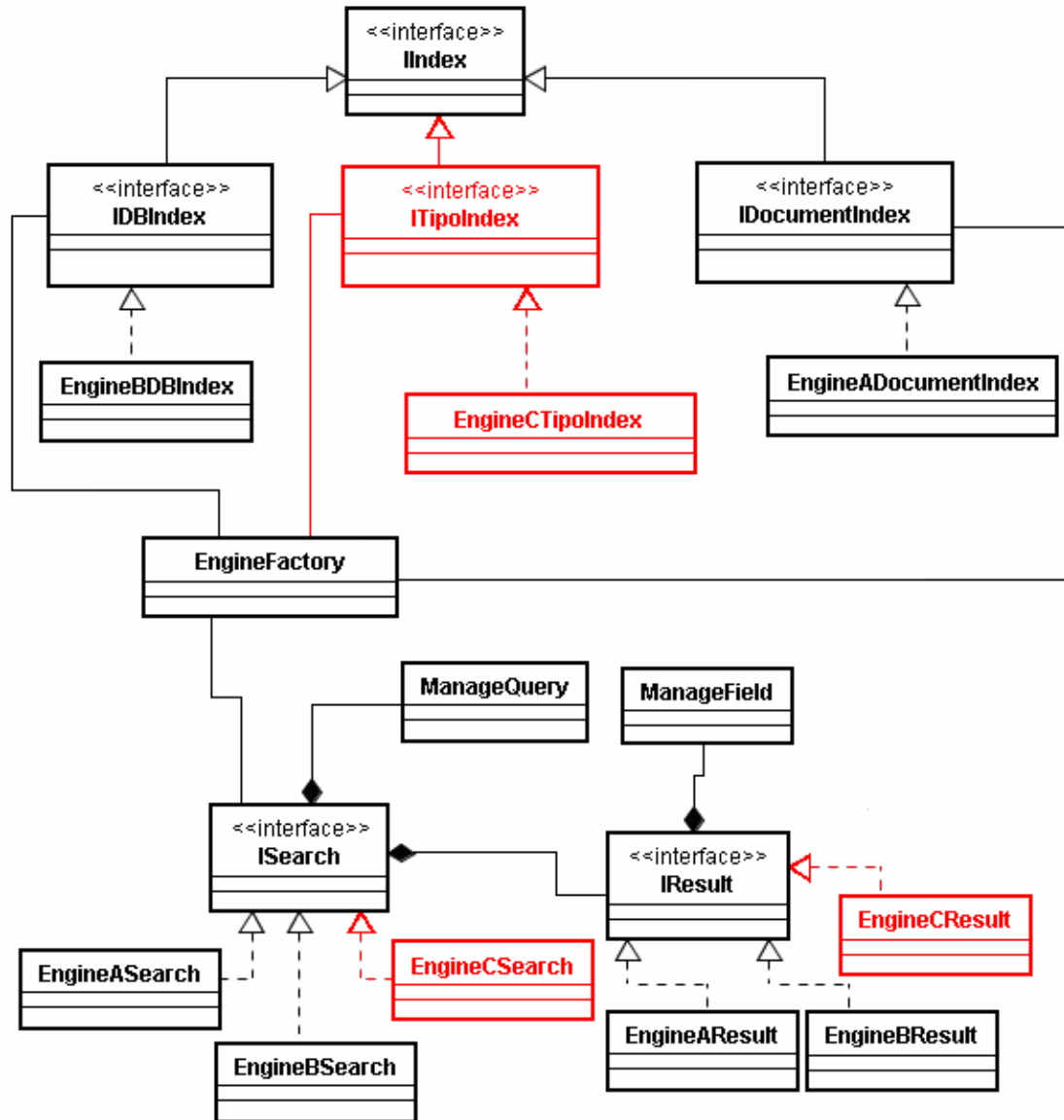


Figura 9 – Adição de um tipo de dados a ser indexado ao arcabouço de integração

4.4. UM EXEMPLO DE APLICAÇÃO

Uma aplicação-exemplo foi desenvolvida com o objetivo de ratificar as características e utilidades do modelo de integração proposto, bem como servir de avaliação prática do que foi apresentado. A aplicação foi montada sobre a plataforma *Web* e é capaz de indexar, pesquisar e exibir dados de diversas fontes, dentre outras funcionalidades, como pode ser observado no diagrama de casos de uso apresentado na Figura 10.

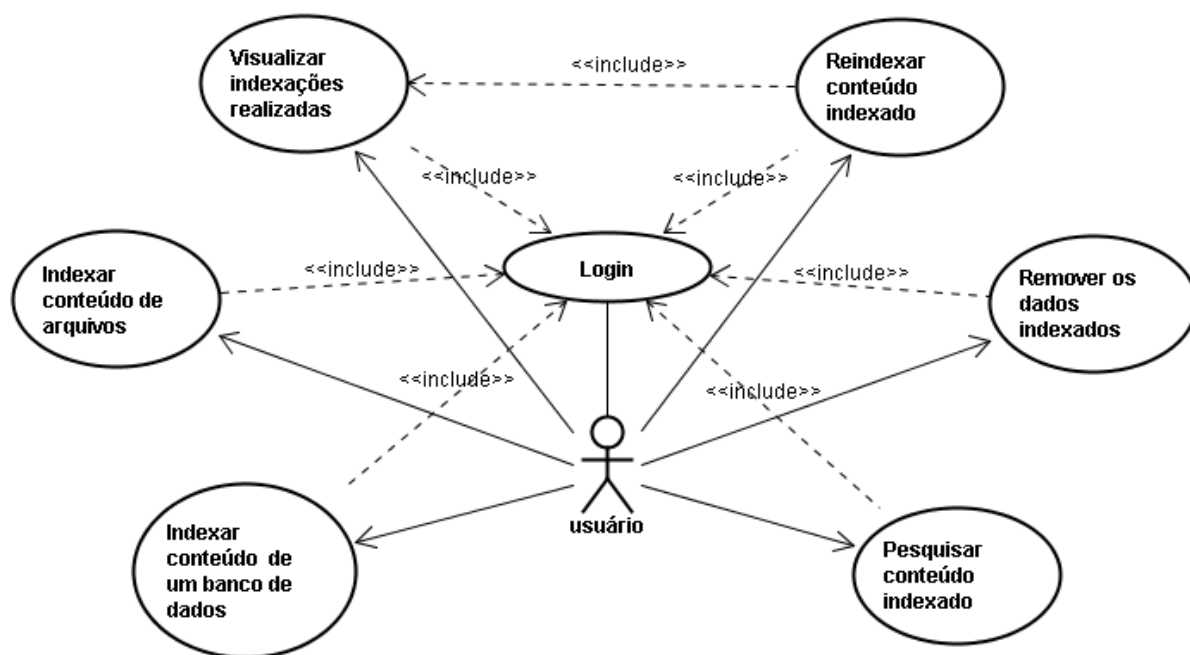


Figura 10 - Diagrama de casos de uso da aplicação exemplo

A aplicação exemplo faz uso de dois motores de indexação, o *Windows Search* e o *Lucene*. O primeiro é responsável por indexar arquivos comuns (tais como documentos, planilhas, apresentações, etc) e o segundo, por indexar dados provenientes de bancos de dados relacionais.

A aplicação consiste em uma interface *Web* montada sobre o modelo proposto. Seu desenvolvimento foi dividido em etapas de forma a exemplificar cada passo da construção de uma aplicação sobre o modelo de integração e elucidar a possibilidade de extensão do modelo.

A etapa inicial consistiu na criação das classes do *Windows Search*, responsável por permitir a recuperação de informações provenientes dos arquivos comuns. Estas classes são *WDSSearch*, *WDSResult* e *WSDDocumentIndex* que implementam respectivamente as interfaces do módulo de busca *ISearch* e *IResult* e a do módulo de indexação *IDocumentIndex*, como pode ser observado em vermelho no diagrama de classes apresentado na Figura 11.

Além destas, também foram implementados os métodos da classe *EngineFactory*, responsáveis por instanciar os objetos de indexação e busca de informações do motor *Windows Search*.

A partir dessas implementações, foi concebida uma aplicação *Web* com o objetivo de proporcionar a interação com o usuário final. Esta aplicação permite que sejam realizadas indexações e busca de informações previamente indexadas. Para comprovar tal fato, foram indexados 360.543 arquivos em seis computadores interligados através de uma rede local. A Figura 12 apresenta o resultado de uma consulta realizada nesta aplicação após a indexação dos arquivos.

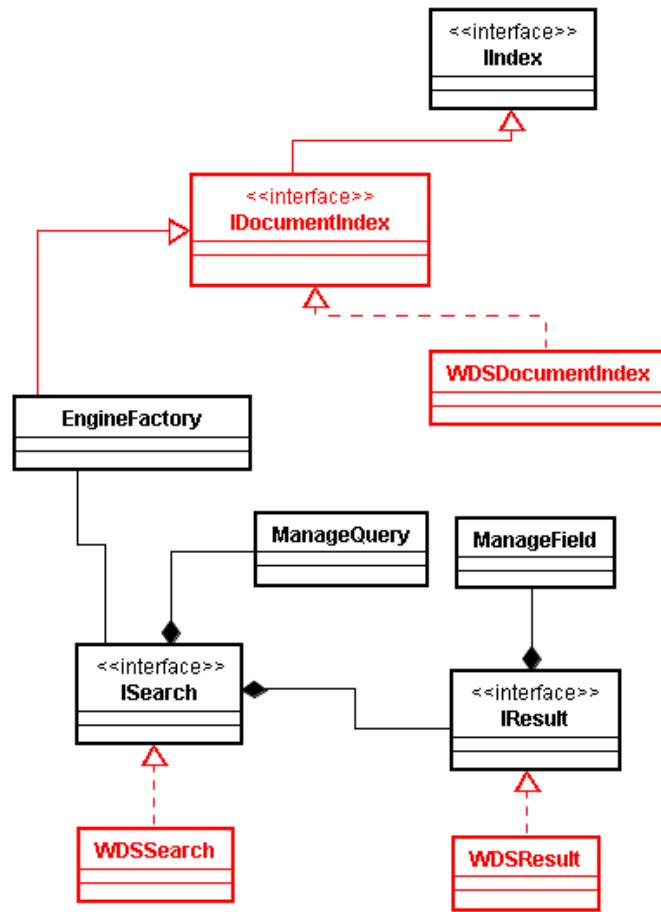


Figura 11 - Diagrama de classes da aplicação-exemplo com o motor *Windows Search*

Consulta

Buscar

Foram encontrados 43 registros

Id	Tipo	Descrição
1	Arquivo	Caminho: d:\camtasia studio\Windows Media.xml;
2	Arquivo	Caminho: d:\Camtasia Studio\testeWMV2\testeWMV2.wmv;
3	Arquivo	Caminho: d:\Camtasia Studio\testeWMV\testeWMV.wmv;
4	Arquivo	Caminho: c:\tdt\discovery\Discovery\Index\WDS\WDSDocumentIndex.cs;
5	Arquivo	Caminho: c:\tdt\discovery\Discovery\Search\Searchers\WDS\WDSSearch.cs;

Figura 12 - Consulta realizada na aplicação exemplo com o motor *Windows Search*

A etapa seguinte no desenvolvimento da aplicação consistiu na inclusão do motor de indexação e busca *Lucene* com o intuito de indexar dados provenientes de um banco de dados relacional. Para isso, as classes *LuceneSearch*, *LuceneResult* e *LuceneDBIndex* foram implementadas de acordo com as especificações das interfaces *ISearch*, *IResult* e *IDBIndex*, respectivamente. Estas modificações podem ser visualizadas em vermelho no diagrama de classes apresentado na Figura 13.

Além disso, foram efetuadas as alterações necessárias na classe *EngineFactory* para que as novas classes pudessem ser instanciadas.

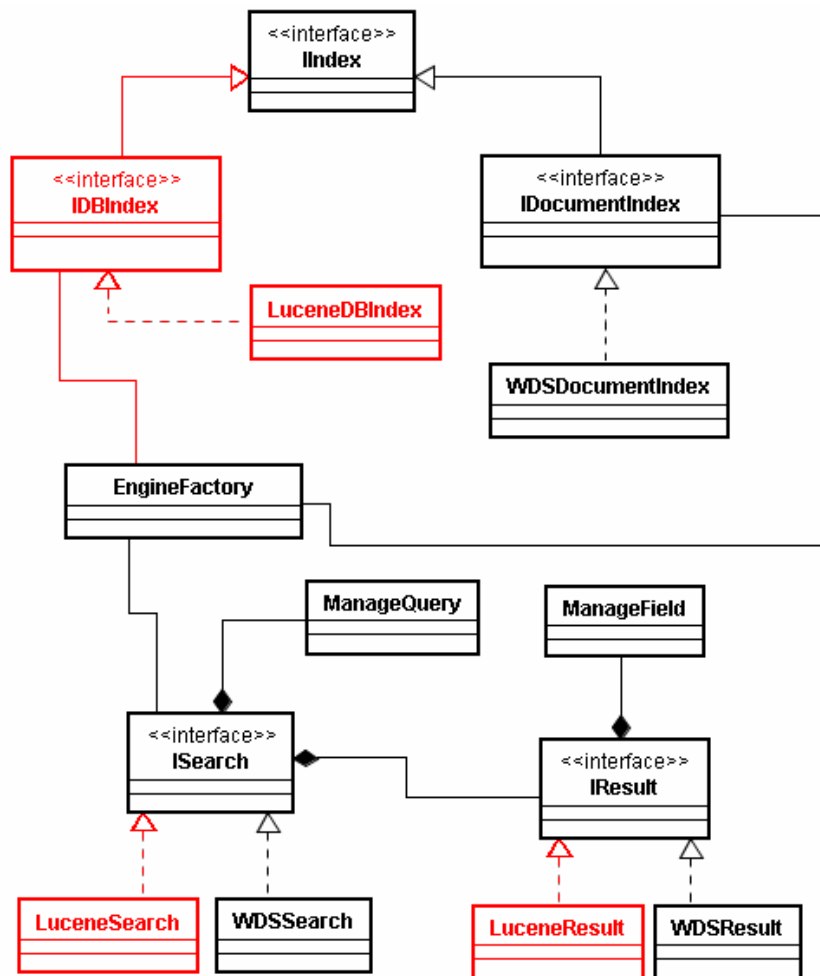


Figura 13 – Diagrama de classes da aplicação-exemplo com a inclusão do motor de indexação *Lucene*

Após esta extensão, foi indexada uma base de dados proveniente de um sistema de *helpdesk* armazenado em um banco de dados relacional. Este banco de dados estava contido em um servidor presente em uma das seis máquinas que tiveram seus arquivos indexados na etapa anterior do desenvolvimento.

Apesar da aplicação fazer uso dos dois motores de indexação após a extensão, a realização da busca é feita de forma integrada e transparente para o usuário da aplicação, já que os resultados de quaisquer fontes de dados são padronizados e retornados em uma só listagem, como pode ser observado na Figura 14. Esta figura

apresenta informações recuperadas do servidor de banco de dados *Harry* em uma base de dados denominada *DtHelp*, e também dados advindos de uma das máquinas utilizadas nas operações de indexação e busca. Estes resultados foram obtidos a partir de uma consulta executada através do mesmo campo de entrada de dados, que também pode ser observado na Figura 14.

Consulta

Buscar

Foram encontrados 74 registros









Id	Tipo	Descrição	
1	Banco de Dados	Servidor: harry; Base de dados: DtHelp; Tabela: chamado; Registro: 50;	
2	Banco de Dados	Servidor: harry; Base de dados: DtHelp; Tabela: chamado; Registro: 45;	
3	Banco de Dados	Servidor: harry; Base de dados: DtHelp; Tabela: chamado; Registro: 52;	
4	Arquivo	Caminho: d:\camtasia studio\Windows Media.xml;	
5	Arquivo	Caminho: d:\Camtasia Studio\testeWMV2\testeWMV2.wmv;	
6	Arquivo	Caminho: d:\Camtasia Studio\testeWMV\testeWMV.wmv;	
7	Arquivo	Caminho: c:\dtdiscovery\Discovery\Index\WDS\WDS\DocumentIndex.cs;	
8	Arquivo	Caminho: c:\dtdiscovery\Discovery\Search\Searchers\WDS\WDS\Search.cs;	

Figura 14 - Consulta realizada na aplicação-exemplo com a inclusão do motor de indexação *Lucene*

A listagem exibida pela Figura 14, assim como a da Figura 12, foi obtida através do trecho de código apresentado na Figura 15. Percebe-se nesta figura que o uso da interface *ISearch* permite que sejam realizadas as pesquisas por todos os motores, o que deixa claro que o uso do arcabouço proporciona um nível de abstração a ponto de não ser necessário o conhecimento dos motores utilizados em cada pesquisa, nem quantos são eles. Este fato fica claro na utilização do mesmo código para recuperação de dados ao se utilizar um ou mais motores. A cada interação, os resultados de cada motor são inseridos em uma lista, que possui os resultados provenientes dos índices de todos os motores utilizados e, assim, a lista final de resultados é obtida.


```
1 //Cria a lista de resultados que será exibida
2 List<IResult> listResult = new List<IResult>();
3
4 //Cria uma nova instância da classe EngineFactory
5 EngineFactory engFact = new EngineFactory();
6
7 //Recupera a lista dos buscadores implementados na aplicação exemplo
8 List<ISearch> listSearch = engFact.CreateSearchEngines();
9
10 //Cria o objeto com os dados a serem buscados
11 ManageQuery mgQuery = new ManageQuery();
12 mgQuery.Sentence = TextBoxQuery.Text;
13
14 //Recupera os resultados de cada buscador e os insere na lista
15 foreach (ISearch search in listSearch)
16 {
17     listResult.AddRange(search.Search(mgQuery));
18 }
```

Figura 15 - Trecho de código de realização de busca de informações

A partir do desenvolvimento e uso desta aplicação exemplo foi possível validar o modelo proposto, permitindo o uso integrado de dois ou mais motores de indexação, possibilitando também a independência entre a aplicação e os motores utilizados.

Como pode ser observado na Figura 14, as buscas realizadas obtiveram retorno proveniente de diversas fontes de dados como banco de dados relacional, arquivos estruturados em formato *xml*, arquivos de vídeo em formato *wmv* e arquivos de código em formato *cs*.

5. CONCLUSÃO

Sabe-se que a geração de dados ocorre em uma escala cada vez maior e a exploração destes é uma tarefa cada vez mais complexa devido à heterogeneidade das fontes de dados em que eles se encontram. Por isso, o aproveitamento das informações digitais acontece geralmente de maneira superficial. A solução desse problema encontra resposta parcial nos motores de indexação de dados existentes, conforme apresentado na seção 4.2.

A indexação dos dados facilita o acesso aos mesmos, o que agiliza a recuperação de informações e reduz a quantidade de informação não aproveitada.

Contudo, apesar de possuírem características que contribuem para a solução do problema, esses motores não apresentam soluções completas, visto a atual demanda aos dados provenientes de diversas fontes.

Neste contexto, o modelo de integração de motores de indexação apresentado permite o desenvolvimento de soluções que possibilitem a realização de recuperação de

informações em um ambiente de fontes de dados heterogêneas. Isto é possível devido à possibilidade de utilização de diversos motores responsáveis pela indexação e busca de dados em fontes específicas. Assim, por trabalharem em conjunto, estes motores permitem a indexação da quase totalidade dos tipos de dados gerados. O modelo ainda contempla a dissociação entre os motores de indexação e as aplicações desenvolvidas sobre ele. Isto torna possível a substituição de qualquer um desses motores, ou ainda, a incorporação de um novo, sem comprometer qualquer aplicação que já tenha sido desenvolvida sobre a estrutura anterior.

Após a avaliação da proposta com o desenvolvimento e uso da aplicação exemplo, foi observado que o uso do modelo de integração cumpre o objetivo de possibilitar a utilização integrada de diferentes motores de indexação. Além disso, foi observada também a possibilidade de inclusão de novos motores de indexação às aplicações, sem a necessidade de alterações em demais partes destas. Assim, conclui-se que o modelo cumpre suas metas e possibilita o desenvolvimento de SRIs capazes de recuperar informações em ambientes de fontes de dados heterogêneas.

Vale lembrar que a concepção desse modelo, não levou em consideração questões referentes ao desempenho na indexação e busca dos dados. Esta questão está fora do escopo desse trabalho, uma vez que a questão do desempenho está intrinsecamente relacionada às características de cada um dos motores de indexação utilizados nos SRIs desenvolvidos sobre o modelo. No entanto, a utilização do modelo de integração não impede que sejam feitos testes de desempenho em SRIs desenvolvidos sobre ele.

Por fim, vale ressaltar que este artigo é um dos resultados do projeto de pesquisa e desenvolvimento DT-Discovery desenvolvido pelo Grupo de Pesquisas em Aplicações Multimídias Avançadas – GAMA do Núcleo de Pesquisa em Redes de Computadores – NUPERC da Universidade Salvador – UNIFACS em parceria com Daten Tecnologia Ltda (DATEN, 2008).

6. TRABALHOS FUTUROS

Como trabalho futuro está sendo desenvolvida uma aplicação baseada no modelo de integração proposto neste artigo estendendo-o para tornar possível a indexação de dados provenientes de outras fontes de dados, até então, não abordadas.

Outra possibilidade de prosseguimento do trabalho consiste no desenvolvimento de um módulo de mineração dos dados recuperados por um SRI construído sobre o modelo, já que a padronização dos dados realizada pela implementação do arcabouço de integração pode ser considerada um pré-processamento dos dados, uma das etapas fundamentais da mineração de dados (MARQUES NETO, 2004). Este módulo poderia incluir mineração visual dos dados, por exemplo, ou ainda outra técnica que pudesse ser utilizada para a descoberta de informações. Além disso, também poderiam ser incluídas técnicas que levassem em consideração a relevância das informações retornadas, a fim de aumentar a precisão e a revocação das informações recuperadas.

Como os mapas de conhecimento das organizações são vistos como responsáveis por identificar conhecimento interno a elas (COSTA, 2004), uma aplicação gerada a partir do modelo de integração proposto também pode ser útil na sua

elaboração. Isso é corroborado se esta aplicação for considerada parte integrante de um sistema de gestão do conhecimento, que segundo Mesquita Mota (2003) serve de ferramenta para criação desses mapas.

Esses trabalhos são apenas algumas das possibilidades expostas para realização de estudos futuros sobre o modelo de integração proposto. Diante do problema que se apresenta, existem inúmeros caminhos que podem ser seguidos a partir da definição do modelo de integração de tecnologias apresentada nesse trabalho.

7. REFERÊNCIAS

AMORIM, S. R. L.; CHERIAF, Malik. **Sistema de Indexação e Recuperação de Informação em Construção Baseado em Ontologia**. III Encontro Tecnologia da Informação e Comunicação na Construção Civil - TIC2007, Porto Alegre, Brasil, 2007.

APACHE – The Apache Software Foundation. **Apache Lucene – Overview**. Disponível em: <<http://lucene.apache.org/java/docs/>>. Acesso em: 05 nov. 2007.

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. **Modern Information Retrieval**. Harlow, Reino Unido: Addison Wesley, 1999. 544p.

BEPPLER, Fabiano D et al. **Uma Arquitetura Para Recuperação de Informação Aplicada ao Processo de Cooperação Universidade - Empresa**. KM Brasil, São Paulo, Brasil, 2005.

COLE, Bernard. **Search Engines Tackle the Desktop**. IEEE Computer, Los Alamitos, EUA, Vol. 38, p. 14-17, mar. 2005.

CORMEN, Thomas H. et al. **Algoritmos: teoria e prática**. Rio de Janeiro, Brasil, Editora Campus, 2002. 916p.

COSTA, M. D. ; KRUCKEN, Lia. **Aplicações de mapeamento do conhecimento para a competitividade empresarial**. KM BRASIL, São Paulo, Brasil, 2004.

DATEN – Daten Tecnologia Ltda. Disponível em: <<http://www.daten.com.br>>. Acesso em: 26 mar. 2008.

DRUCKER, Peter. **The coming of the new organization**. Harvard business review on knowledge management. Boston, EUA: Harvard Business School Press, p. 1-19, 1998.

FREEMAN, Eric et al. **Use a cabeça – Padrões de Projeto**. Rio de Janeiro, Brasil: Alta Books, 2005. 496p.

GAMMA, Erich et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. First Edition. Upper Saddle River, EUA: Addison-Wesley Professional, 1994. 416p.

GANTZ, John F. **The Expanding Digital Universe, A Forecast of Worldwide Information Growth Through 2010**. IDC White Paper, Framingham, EUA, mar.

2007.

GOOGLE. **Google Desktop Features.** Disponível em: <<http://desktop.google.com/en/features.html>>. Acesso em: 20 nov. 2007a.

GOOGLE. **Google Desktop SDK.** Disponível em: <<http://desktop.google.com/dev/>>. Acesso em: 20 nov. 2007b.

GOSPODNETIC, Otis; HATCHER, Erik. **Lucene in Action.** Greenwich, Reino Unido: Manning Publications, 2005. 421p.

LAU, Lawrence J. **Economic Growth in the Digital Era.** Symposium on “Welcoming the Challenge of the Digital Era” 2003 Kwoh-Ting Li Forum, Taipei, China, nov. 2003.

LIMA, Gercina Ângela Borém. **Interfaces between information science and cognitive science.** Ci. Inf., Brasília, Brasil, v. 32, n. 1, 2003 . Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652003000100008&lng=en&nrm=iso>. Acesso em: 11 jan 2008.

MA, Yan; LI, Ming. **Design of an Intelligent Meta Search Engine.** Journal of Communication and Computer, EUA, Volume 2, No.7, USA, jul. 2005. ISSN1548-7709.

MESQUITA MOTA, M. **Tecnologias de Gestão do Conhecimento e sua Relação com a Inovação nas Organizações: O Caso de uma Multinacional de Consultoria.** Dissertação (Grau de Mestre em Administração de Empresas), Universidade Federal da Bahia, Salvador, Bahia, Brasil, 2003.

MEYER, Bertrand. **Object-Oriented Software Construction.** Santa Barbara, EUA: Prentice Hall, 2000. 1296p.

MICROSOFT. **Windows Search.** Disponível em: <<http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx>>. Acesso em: 05 nov. 2007a.

MICROSOFT. **Development Platform Overview.** Disponível em: <<http://msdn2.microsoft.com/en-us/library/bb331575.aspx> >. Acesso em: 15 nov. 2007b.

NASCIMENTO, Luiz Antonio. **Proposta de um Sistema de Recuperação de Informação para Extranet de Projeto.** Dissertação (Grau de Mestre), Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2004.

NUNES, A. M.; FILETO, Renato. **Uma Arquitetura para Recuperação de Informação Baseada em Semântica e sua Aplicação no Apoio a Jurisprudência.** Escola Regional de Banco de Dados (ERBD), Caxias do Sul, Brasil, 2007.

PENG, Fuchun et al. **Context sensitive stemming for web search.** Proceedings of the 30th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, p. 639-646. ACM, 2007.

SOUZA, Renato Rocha. **Sistemas de Recuperação de Informações e Mecanismos de Busca na web: panorama atual e tendências**. *Perspect. ciênc. inf.*, Belo Horizonte, Brasil, v.11, n.2, p.161 -173, ago. 2006.

UCHÔA, Elvira Maria Antunes; MELO, Rubens Nascimento. **Integração de Sistemas de Bancos de Dados Heterogêneos Usando Frameworks**. *Anais do Simpósio Brasileiro de Banco de Dados*, Florianópolis, Brasil, 1999.

ZOBEL, Justin; MOFFAT, Alistair. **Inverted Files for Text Search Engines**. New York, EUA: ACM Pres, 2006.

