**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

# UM MÉTODO DE INDEXAÇÃO DE FORMULÁRIOS WEB VISANDO CONSULTAS POR SIMILARIDADE

**WILLIAN VENTURA KOERICH**

**Florianópolis – SC**

**2013**

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

# UM MÉTODO DE INDEXAÇÃO DE FORMULÁRIOS WEB VISANDO CONSULTAS POR SIMILARIDADE

Willian Ventura Koerich

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Florianópolis – SC

2013

Willian Ventura Koerich

# UM MÉTODO DE INDEXAÇÃO DE FORMULÁRIOS WEB VISANDO CONSULTAS POR SIMILARIDADE

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Ronaldo dos Santos Mello, Dr.

Banca Examinadora

Profª. Dr. Carina Friedrich Dorneles

Prof. Dr. Renato Fileto

## Agradecimentos

Agradeço a Deus e minha família pela força e motivação frente às dificuldades, e apoio incondicional ao longo de toda minha vida e graduação. Aos meus grandes amigos que estiveram presentes em momentos de estudos, festas e descontração de forma que os anos que passaram se tornasse importantes em minha vida.

Ao meu orientador, professor Ronaldo pelo apoio e conselhos para realização deste trabalho. Aos professores da banca Carina e Renato por aceitarem meu convite e contribuírem comigo através de suas sujestões.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

## Abstract

Search engines do not support specific searches for web forms found on Deep Web. Within this context, the WF-Sim project, developed at UFSC database group (GBD/UFSC), proposes a query-by-similarity system for Web Forms to deal with this lack. This paper presents an indexing technique for querying-by-similarity web forms as a WF-Sim system component. This technique is centered on suitable index structures to the main kinds of queries applied to web forms, as well as some optimizations in these structures to reduce the number of index entries. To evaluate the indexes' performance, we ran experiments on two persistence strategies supported in WF-Sim project: file system and database. Furthermore, we also evaluated the performance of the proposed indexes versus the traditional keyword index, and the performance by using a cache structure proposed in this work.

**Palavras chave:** Web forms, Deep Web, indexes, query-by-similarity.

## Resumo

Motores de busca atuais não possuem suporte a buscas específicas por formulários Web relacionadas à Deep Web. Neste contexto, o projeto WF-Sim, em desenvolvimento no Grupo de Banco de Dados da UFSC (GBD/UFSC), propõe um processador de consultas por similaridade para formulários Web para lidar com esta limitação. Este trabalho apresenta uma técnica de indexação para buscas por similaridade em formulários web, atuando como um componente do sistema WF-Sim. Esta técnica está centrada em estruturas de índice adequadas aos principais tipos de consulta aplicados a formulários Web, bem como otimizações nestas estruturas para reduzir a quantidade de entradas no índice. Experimentos que validam estas estruturas foram executados sobre duas estratégias de persistência de dados suportados pelo WF-Sim: sistema de arquivos e banco de dados. Além disso, avaliou-se também o desempenho dos índices propostos contra o tradicional índice de palavras-chave, bem como o desempenho com a utilização de uma estrutura de cache proposta neste trabalho.

**Palavras chave:** formulários Web, Deep Web, consultas por similaridade, Lucene.

# 1. Introdução

## 1.1. Visão Geral

Uma grande quantidade de serviços está atualmente à disposição das pessoas através da Web, como locação e vendas de veículos, reserva de hotéis, compra de livros, oferta de empregos, etc. Esses serviços disponibilizam uma diversidade de informação para consultas aos usuários, porém, seu verdadeiro conteúdo se encontra invisível pelo fato dos seus resultados de consultas serem gerados dinamicamente através de especificação de filtros via formulários Web [Madhavan et al. 2009].

Esses formulários web, existentes em páginas web, são a porta de entrada ao usuário do conteúdo existente nos bancos de dados de determinado serviço. Devido à existência desses formulários web, mascarando o conteúdo dessas informações, Bergman [2001] em um artigo da Bright Planet, denominou a essa porção de conteúdo de informação escondida como *Deep web* (web oculta) ou *Hidden-databases* (bancos de dados escondidos).

A existência dessas informações presentes em formulários web e bancos de dados cujo conteúdo é recuperável através de páginas dinâmicas, torna o processo de consultas e recuperação de informação por web crawlers (indexadores web automatizados) muito difícil. De modo a recuperar essas informações e criar um repositório do conteúdo desses serviços é necessário criar um processo automatizado de investigação do conteúdo dos formulários, bem como de preenchimento dos mesmos que explore as infinidades de

combinações possíveis de valores de seus campos, visando indexar a informação recolhida para facilitar futuros acessos.

O projeto WF-Sim visa desenvolver uma solução para amenizar esta problemática: um processador de consultas por similaridade para formulários Web [Gonçalves, 2011]. Este processador caracteriza-se por ser um software responsável pela execução de todas as tarefas necessárias à geração de um resultado a uma consulta utilizando similaridade para recuperação de formulários web. Exemplos destas tarefas são a especificação de uma consulta por parte do usuário e métricas adequadas para definição do grau de similaridade entre os formulários.

Consultas no WF-Sim ocorrem sobre elementos de formulários web indexados, sendo um elemento um campo de formulário web. No contexto deste projeto, estruturas de índice são definidas a partir de clusters gerados por um processo de *matching* de elementos de formulários, permitindo recuperação de formulários com elementos similares. Estas estruturas de índice devem ser devidamente projetadas para facilitar as consultas típicas por formulários web. Esta é a principal contribuição deste trabalho de conclusão de curso.

Este trabalho apresenta uma estratégia de indexação por similaridade para formulários web desenvolvida para o WF-Sim, visando o acesso a dados mantidos em dois tipos de mecanismos de persistência: arquivos e banco de dados relacional. Avalia-se aqui não apenas o desempenho das estruturas de índice para cada tipo de persistência, mas também quais estruturas de índice apresentaram melhor desempenho e comportamento do sistema para análise

de escalabilidade. Uma estrutura de cache, proposta também neste trabalho, foi avaliada para fins de verificação de tempo de processamento das consultas.

### 1.2. Objetivos

O objetivo principal deste projeto é o desenvolvimento de uma estratégia de indexação para formulários Web visando realização de consultas por similaridade sobre estes formulários.

Abaixo são citados objetivos específicos:

- Realizar a indexação dos formulários Web a partir de clusters de elementos de formulários [Silva 2012];
- Utilizar estratégias que otimizem o processamento de consultas por similaridade para formulários web;
- Executar e avaliar um conjunto de experimentos sobre as formas de persistência (banco de dados relacional e arquivos serializados) usadas na clusterização dos formulários Web.
- Avaliar o desempenho das estruturas propostas contra o tradicional índice de palavra-chave, o uso de uma estrutura de cache e experimentos que testem a escalabilidade.

### 1.3. Metodologia

Buscando obter conhecimento necessário sobre o assunto, este trabalho adotou as seguintes etapas para seu desenvolvimento:

- Estudo sobre estruturas de indexação e seus respectivos enfoques de utilização;

- Estudo de ferramentas de indexação disponíveis no mercado e amplamente utilizadas para serviços de consulta;
- Estudo das métricas utilizadas nas etapas anteriores do projeto;
- Estudo do conceito de dicionário de sinônimos de modo a melhorar o desempenho do módulo de indexação.
- Implementação do módulo de indexação;
- Realização de experimentos que validem as estruturas desenvolvidas no módulo de indexação, comparando os desempenhos obtidos testando-se às duas estratégias de persistência e mensurar os ganhos da implementação da estrutura de cache.

Na primeira etapa foram estudadas as estruturas tradicionais de indexação, buscando escolher a estrutura mais adequada às características de formulários Web. Na segunda etapa, foi feito estudo sobre quais ferramentas disponíveis possuíam as características vistas como necessárias para a implementação de nossa solução pra a indexação. Na terceira etapa, foi realizado um estudo sobre as métricas utilizadas no trabalho anterior [Silva 2012], visando conhecer o processo de formação dos clusters a serem indexados. Na quarta e quinta etapa foram realizadas a implementação da solução desenvolvida, corresponde ao módulo de indexação de formulários Web e as melhorias realizadas com a inclusão de um dicionário de sinônimos no processo de busca. Na sexta etapa é realizado um comparativo do desempenho obtido para os conjuntos de consulta, quando submetidos às duas estratégias de persistência utilizadas no projeto, avaliar a relevância do uso de cache e os tempos obtidos em consultas submetidas a bases de formulário de tamanho diferentes de modo a medir a escalabilidade.

### 1.4. Estrutura do Trabalho

A forma de apresentação desse trabalho está dividida da seguinte forma:

- Capítulo 1: Introdução – Apresentação do trabalho, com uma contextualização da área de pesquisa, das tecnologias envolvidas e trabalhos correlatos.
- Capítulo 2: Deep Web – Apresentação dos conceitos e visão geral da Deep Web.
- Capitulo 3: WF-Sim – Apresentação do projeto ao qual esse trabalho de conclusão de curso está inserido.
- Capitulo 4: Módulo de Indexação – Apresentação das estruturas de índice propostas, bem como das técnicas e teoria utilizadas.
- Capítulo 5: Implementação – Apresentação das ferrementas utilizadas, como elas funcionam e modo como foram utilizadas para realização do projeto.
- Capítulo 6: Experimentos e Resultados – Apresentação da metodologia utilizada para avaliar as estruturas de índice e resultados obtidos.
- Capítulo 7: Trabalhos Relacionados – Apresentação de uma análise do trabalho desenvolvido com os trabalhos existentes na literatura.
- Capítulo 8: Conclusão – Apresentação das conclusões encontradas neste projeto quanto a sua relevância à comunidade científica.

## 2. Deep Web

*Deep Web* é o nome dado a uma grande parcela do conteúdo geral da *World Wide Web*. Esse termo surgiu a partir de um estudo elaborado nos laboratórios da BrightPlanet, por seu fundador Mike Bergman [2001], que afirma que buscas na Internet podem ser comparadas ao ato de pescar em um oceano. Ele afirma que os usuais motores de busca encontrados na Internet, como Google, Yahoo entre outros, podem ser vistos como barcos pesqueiros que conseguem capturar animais (analogia com conteúdo da Internet) presentes apenas na superfície do oceano, animais estes que correspondem a uma pequena porção de uma imensidão de conteúdo disponível a todos. A essa pequena porção a qual esses serviços de busca atuam denomina-se Surface Web.

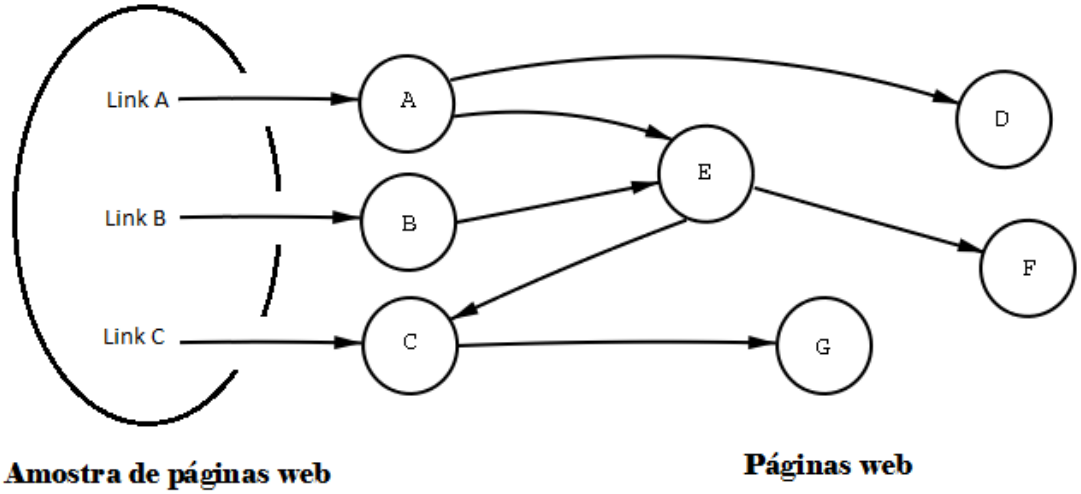


**Figura 1. Funcionamento de um Web Crawler Tradicional( Adaptado do original disponível em -http://www2002.org/CDROM/alternate/747/)**

A Surface Web diz respeito ao conteúdo indexado pelos serviços de busca, ou seja, páginas web estáticas que são referenciadas por outras

páginas web e/ou possuam permissão por parte do responsável de serem anexadas ao conteúdo de serviços de busca. Como pode ser visto na Figura 1, um web *crawler* (indexador automático de páginas estáticas), através de um conjunto inicial de links de páginas web, navega através desses links disponíveis no conjunto inicial indexando todo link alcançável e que não possua restrições de acesso e indexação.

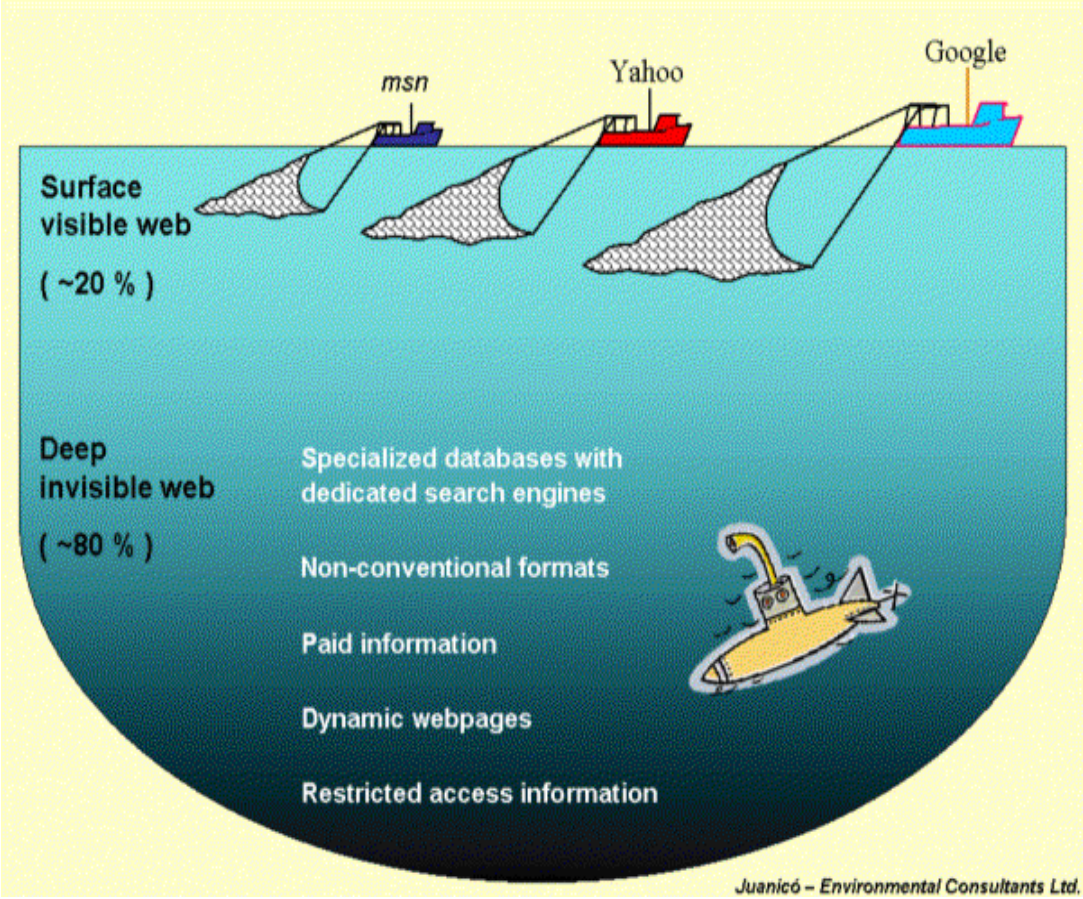


**Figura 2. Analogia da Web vista como um oceano.(disponível em http://cleberjbatista.blogspot.com.br/2012/01/deep-web-o-lado-obscuro-da-internet.html)**

Como definido anteriormente, a Deep Web é chamada “porção escondida” porque não é recuperada por consultas processadas por serviços de busca. Nesta porção “inalcançável” da web, como visto na Figura 2, são

encontradas informações principalmente relevantes comercialmente e para fins de pesquisa. A diversidade de informação escondida nessa porção da Web envolve principalmente bancos de dados de diferentes domínios do conhecimento, como prestadores de serviço (revendas, hotéis, etc...), bases de dados biológicos, arquivos de patentes, ofertas de emprego e negócios, acervos de livros e anúncios classificados, dentre outros.

De acordo com Bergman[2001], a Surface Web possuía, no ano de 2001, um número aproximado de 1 bilhão de documentos e tamanho total de 19 terabytes de informação. A Deep Web nesse mesmo ano, possuía um tamanho total de 7.500 terabytes de informação,sendo que um subconjunto dos 60 maiores *web sites* encontrados na Deep Web possuía tamanho de 750 terabytes, extrapolando com muita facilidade o tamanho obtido na época para a Surface Web.

A intenção do projeto WF-Sim, inserido no contexto da Deep Web, tem como foco a exploração de conteúdo escondido e acessível através de formulários web. Um formulário web, como visto na Figura 3, é a “porta de entrada” para um banco de dados escondido na Deep Web. Os formulários web são encontrados em inúmeras páginas que reúnem ricos conjuntos de informações sobre domínios específicos. Essas informações escondidas são acessíveis mediante interação do usuário e especificação de filtros representados pela escolha ou definição de valores para os campos dos formulários.

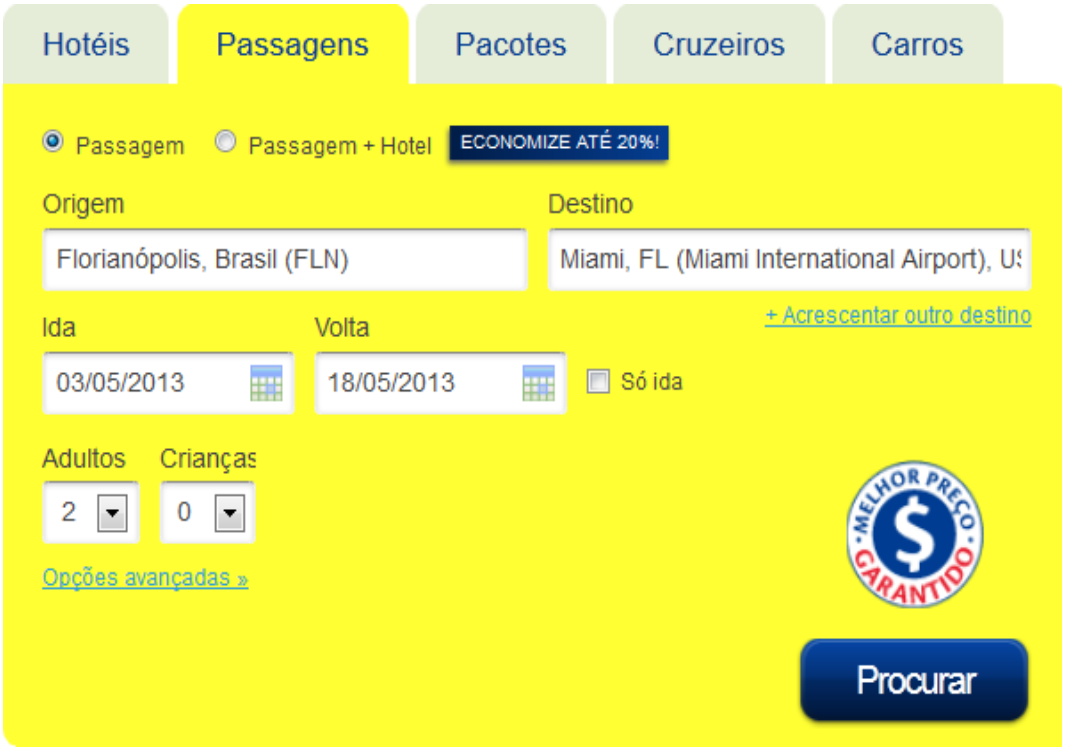


**Figura 3. Exemplo de Formulário Web no domínio de Passagens Aéreas (disponível em http://www.decolar.com)**

Formulários web não possuem apenas a função de definição de filtros de entrada para bancos de dados escondidos. Muitas vezes, são utilizados como recursos para o acesso a conta de email, páginas de cadastro, páginas de resposta a questionários, etc. Nesse caso, os *crawlers* voltados para a extração de dados de Deep Web devem ser especialistas nos determinados domínios de formulários presentes na Web, visando extrair apenas aqueles pertinentes a bancos de dados escondidos.

Os formulários web são divididos em elementos. Um elemento de formulário web é normalmente composto por um conjunto de valores que são atribuídos a um rótulo. Como pode ser visto na Figura 3, o rótulo *Destino* representa o elemento de formulário de aeroportos de destino para a emissão

de uma passagem aérea e seu respectivo conjunto de valores são os aeroportos cadastrados no sistema do serviço. Esse conceito de elemento é fundamental para o projeto WF-Sim, apresentado no próximo capítulo.

# 3. WF-Sim

## 3.1. Apresentação do Projeto

As técnicas de busca por formulários web atualmente utilizadas geralmente são baseadas no modelo de busca por palavra-chave, que executa o *matching* entre os termos de entrada com termos existentes nos formulários. O principal exemplo é a máquina de busca *DeepPeep*[1]. Visando adicionar capacidade de busca por similaridade a formulários web, o projeto WF-Sim, desenvolvido pelo GBD/UFSC[2], com financiamento do CNPq, se inspira no fato de que os dados disponíveis na Deep Web são relevantes para usuários que desejam encontrar formulários web que satisfaçam suas necessidades em termos de serviços online para os mais diversos fins.

A área de consulta à dados de formulários Web ou presentes em banco de dados escondidos é pouco explorada e mesmo os escassos trabalhos existentes não exploram consultas por similaridade. Esse é um problema em aberto e bastante relevante visto que os dados presentes em formulários Web de um mesmo domínio geralmente apresentam heterogeneidade em termos da definição de rótulos e valores.

Deste modo, o projeto WF-Sim propõe um processador de consultas por similaridade para formulários Web [Gonçalves, R. et. al,2011].Esse processador, através de um processo de clusterização de elementos de formulário similares,realiza a indexação dos elementos representativos de cada

[1] http://www.deeppeep.org/

[2] http://www.gbd.inf.ufsc.br

cluster (ditos centroides). A motivação da criação de clusters, além de reunir informações similares sobre campos de formulários, possibilita um mecanismo de recuperação eficaz por reduzir a grande quantidade de entradas de elementos de formulário no índice e, que com isso, prejudica o desempenho de acesso.

A Figura 4 mostra a arquitetura do sistema WF-Sim. Com base em uma consulta de entrada, no caso, um conjunto de elementos denominado *formulário template* (filtros), o sistema retorna um conjunto de formulários web que possuem elementos similares. O sistema possui os módulos de busca, indexação e clusterização.

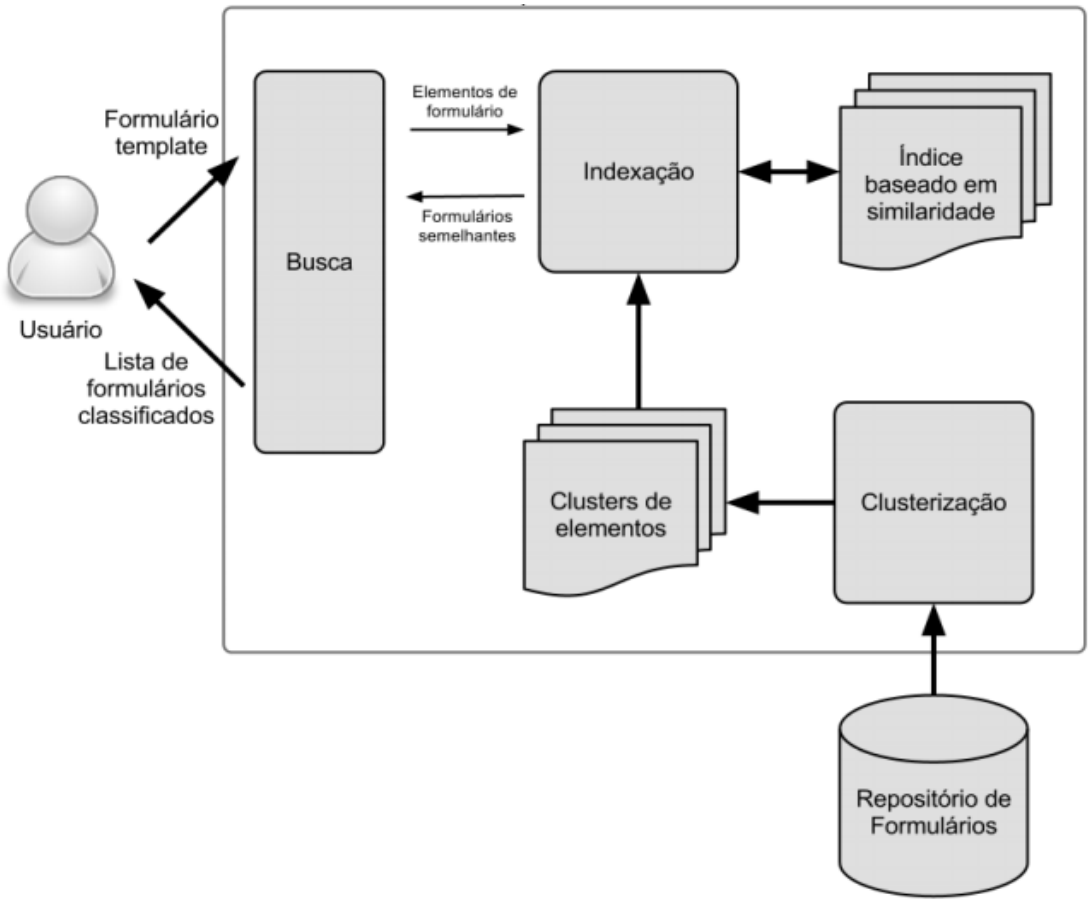


**Figura 4. Arquitetura do projeto WF-Sim.**

### 3.2. Módulo Clusterização

No módulo de clusterização são aplicadas métricas de similaridade de modo a agrupar os formulários em clusters com elementos similares [Silva, 2012]. Neste módulo, através da aplicação de algoritmos de clusterização e de métricas específicas de similaridade, elementos medianos são definidos como os centroides dos clusters como visto na Figura 5, para uma adaptação do algoritmo de clusterização K-Means adotado no sistema. Para fins de exemplo,poderíamos ter um cluster formado pelos elementos “Make”, “Brand”, “Select a Make”, sendo “Make” eleito como centróide.

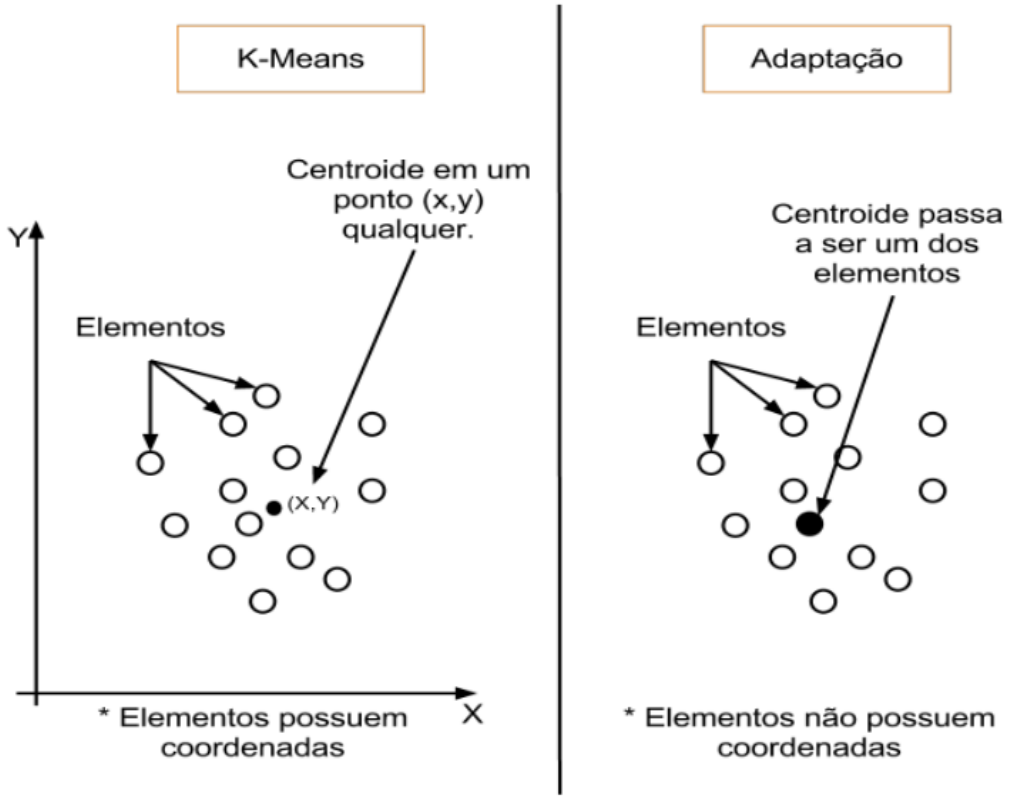


**Figura 5. Adaptação do K-Means [Silva, 2010]**

### 3.3. Estratégias de persistência dos clusters

Após a etapa de clusterização, os formulários classificados são armazenados em duas estratégias de persistência desenvolvidas para o projeto WF-Sim, que são sistemas de arquivos e banco de dados. Essas estratégias são detalhadas a seguir.

#### 3.3.1. Sistema de Arquivos

Sistema de arquivos foi a primeira estratégia de persistência adotada no projeto, sendo concebida para domínios da Deep Web com um número reduzido de formulários Web.

Como pode ser visto na Figura 6, essa estratégia armazena a informação dos processos de clusterização em diretórios cujos nomes são a discriminação dos pesos adotados para o rótulo e valores nas métricas de similaridade.

Nessas pastas é possível realizar o acesso a um determinado cluster de duas maneiras: através de um arquivo serializado chamado *keys.ser*, que contém uma lista dos elementos centroides, ou através do diretório chamado centroids, que contém arquivos nomeados com o rótulo do centroide.

Nas duas abordagens, é necessário carregar o centroide desejado de forma a recuperar o *id* que é o nome do diretório que possui os elementos pertencentes ao cluster que se deseja recuperar. A diferença das duas abordagens é que no diretório *centroid* é possível carregar apenas o arquivo com o nome do elemento centroide desejado, enquanto no arquivo *keys.ser* é necessário carregá-lo e efetuar uma busca pelo elemento desejado.

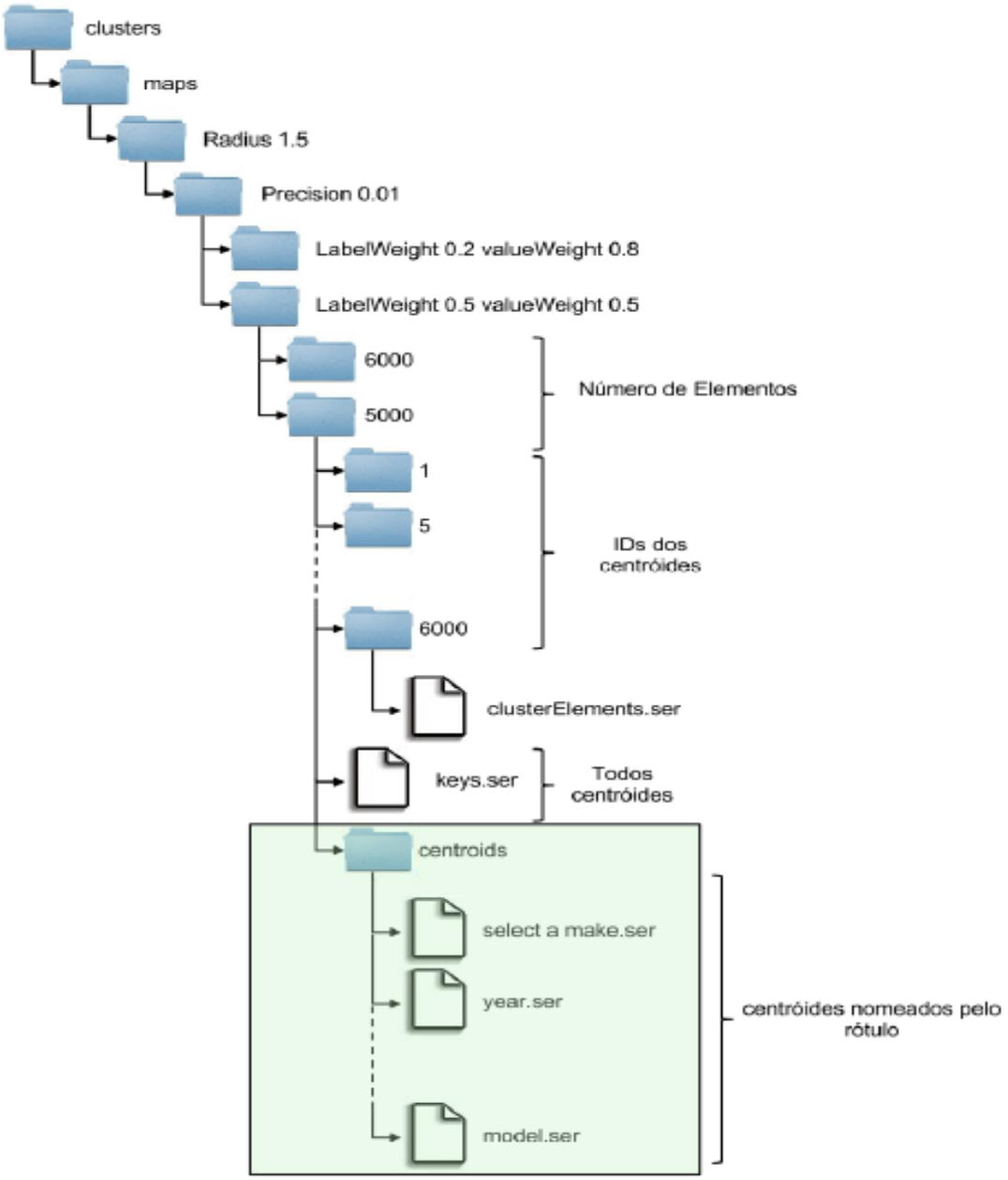


**Figura 6. Persistência do Sistema de Arquivos [Silva, 2012]**

### 3.3.2. Banco de Dados Relacional

Em um segundo momento, um sistema de banco de dados relacional foi adotado para realizar o armazenamento dos clusters de elementos. O esquema conceitual de banco de dados é formado basicamente pelas entidades *cluster, form_element* e *value*, como visto na Figura 7.

A entidade *form_element* possui informação da identificação do elemento de formulário como rótulo, URL de um formulário Web e a referência para um cluster. A entidade *value* possui a descrição de valores e a referência a um elemento. A entidade *cluster* mantém os dados utilizados para o cálculo das métricas de similaridade que foram utilizadas para a geração dos clusters, além do próprio identificador do cluster. A entidade *configuration* apenas contém as informações técnicas referentes ao processo de clusterização.

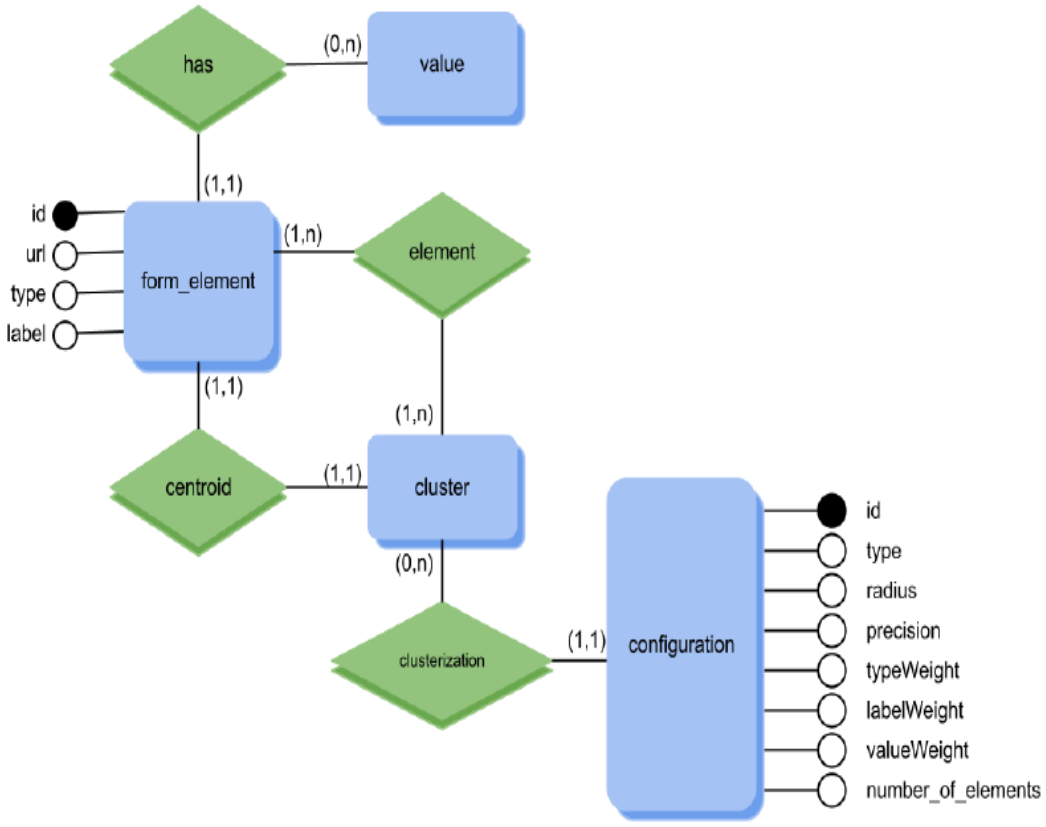


**Figura 7. Modelagem Conceitual do Banco de Dados para o Projeto WF-Sim [Silva, 2012]**

O módulo de indexação é o foco deste trabalho e visa criar estruturas de índice, garantir o acesso a estas estruturas e retornar os formulários relevantes. Estas estruturas foram desenvolvidas especialmente para manter

informações relevantes sobre formulários web e facilitar o mapeamento de elementos do *template* para os elementos centroides indexados. O próximo capítulo descreve o módulo de indexação.

# 4. Módulo de Indexação

Este capítulo descreve diversos pontos relacionados ao módulo de indexação, que é a contribuição deste trabalho, incluindo como foi utilizada a tecnologia de índice invertido, e implementadas as estruturas de índice e otimizações propostas.

## 4.1. Índice Invertido

Índice invertido é uma estrutura de índice amplamente utilizada na área de recuperação de informação para a indexação de textos. Essa estrutura é composta por um vocabulário, que é composto por termos presentes nos textos nas entradas do índice, e as ocorrências para cada entrada, que são as referências dos documentos nos quais o termo está presente.

## 4.2. Estruturas de Índice Propostas

Três estruturas de índice foram definidas para o acesso aos dados de formulários Web e baseadas no modelo de índice invertido: *palavra-chave*, *contexto* e *metadado*. A idéia destas estruturas foi obtida de um trabalho anterior do GBD/UFSC [Mello et. al. 2010] e adaptada à problemática de busca por similaridade em formulários Web. Estas estruturas, em particular os índices de contexto e metadado, são adequadas aos tipos de consulta mais frequentes sobre formulários web.

De acordo com a Figura 8, consultas por contexto associam um rótulo aos respectivos valores presentes que lhe podem ser atribuídos em um formulário. O diferencial desta estrutura é recuperar formulários com base na

contextualização de campos por domínio, ou seja, permite que o usuário recupere formulários com determinado tipo de conteúdo de campo que lhe interessa. As entradas desta estrutura de índice são representadas no formato *nome_valor###nome_rótulo.*

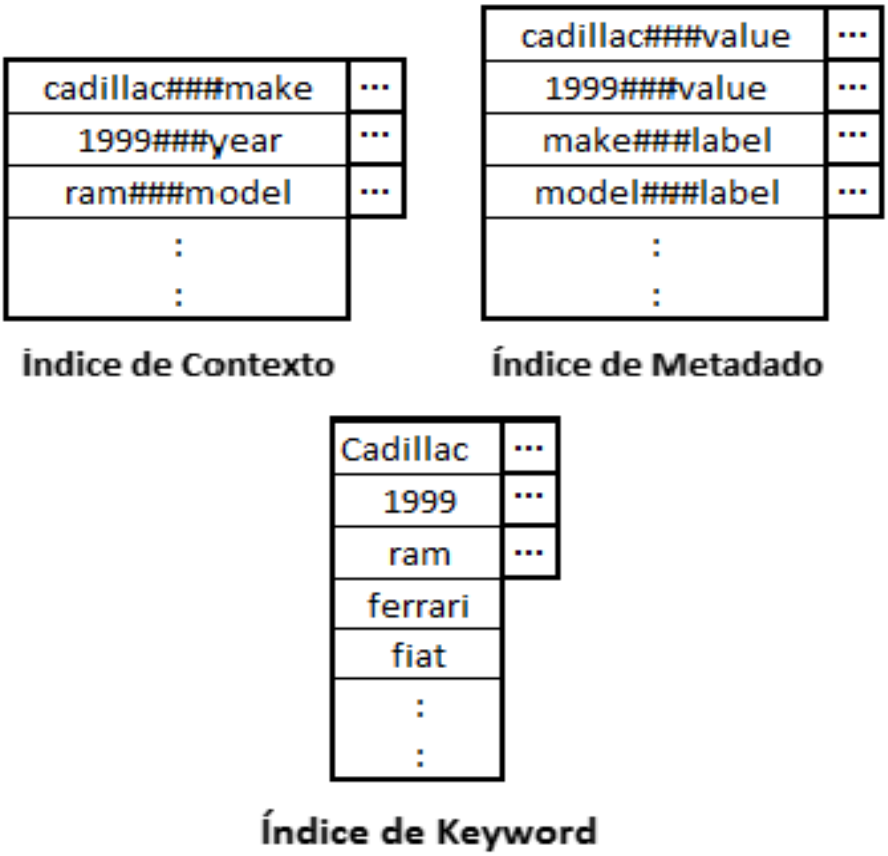


**Figura 8. Estrutura dos Índices Propostos**

Consultas por metadado, por outro lado, permitem especificar se um termo de interesse é um metadado do tipo *rótulo* ou *valor*, limitando o resultado de acordo com a natureza da informação armazenada. As entradas desta estrutura estão no formato *nome_valor###VALUE* ou *nome_valor###LABEL*.

A estrutura de índice de palavra-chave (*keyword*) possui entradas tradicionais de termos para cada possível valor ou rótulo existente. Ela foi

considerada para fins de futura avaliação de desempenho contra as outras duas estruturas propostas.

Abaixo, na Figura 9, podem ser vistos exemplos das entradas para cada estrutura de índice do WF-Sim.

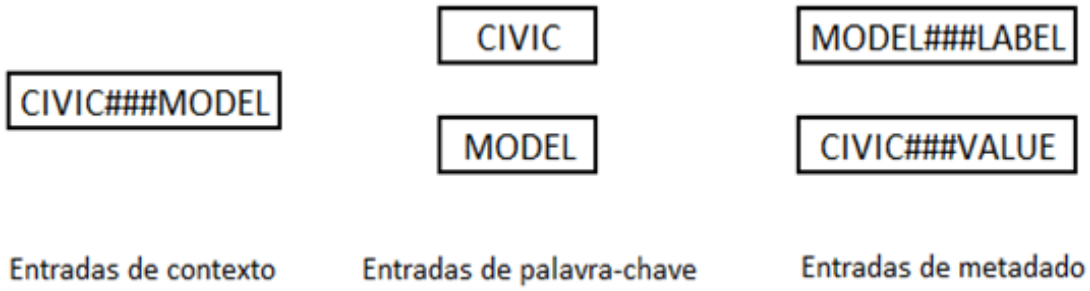


**Figura 9. Exemplos de entradas de Índice encontradas nas estruturas propostas**

Uma quarta estrutura de índice (índice de sinônimos) foi concebida de forma a exercer uma funcionalidade auxiliar as estruturas de índice apresentadas acima.

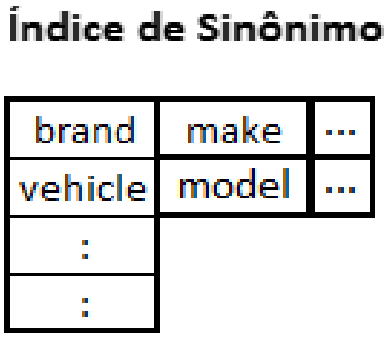


**Figura 10. Estrutura de Índice Auxiliar: Índice de Sinônimos**

Este índice auxilia o processamento de consultas possibilitando a busca por termos similares. Durante o processo de indexação dos rótulos, um banco de dados léxico da língua inglesa é acessado e deste são recuperados um

conjunto de sinônimos. A partir deste conjunto de termos sinônimos, são criadas entradas no índice de sinônimos para cada valor sinônimo, mapeando estes valores para o rótulo dos elementos centroides dos clusters que estão sendo indexado, como visto na Figura 10.

### 4.3. Otimizações propostas pelos índices de Contexto e Metadado

As estruturas de índice contexto e metadado garantem otimizações importantes no processamento de consultas sobre formulários Web.

De uma maneira simples, para processar uma consulta realizada na estrutura de índice de contexto, através de um filtro *CIVIC###MODEL,* por exemplo, é necessário realizar duas buscas pelos termos *CIVIC* e *MODEL* no índice de palavra-chave, realizar a recuperação das informações referentes aos clusters aos quais essa informação esteja presente e processar a intersecção entre os conjuntos resultantes formados por estes dois termos. Esse processamento do índice de palavra chave, em comparação com a estrutura de índice de contexto, realiza acessos adicionais ao índice e acessos adicionais a persistência, visto que os dois termos podem pertencer inclusive aos mesmos clusters de elementos similares. O processo de intersecção desses conjuntos não ocorre na estrutura de índice de contexto, uma vez que este índice já mantém os formulários que possuem o filtro exato pelo par (rótulo, valor) presente na entrada do índice.

O índice de metadado permite as mesmas otimizações apresentadas no caso anterior, ou seja, evita as mesmas intersecções de dados e acessos adicionais à persistência. Ele permite discriminar o tipo de metadado de um

formulário que se deseja recuperar e distinguindo a informação a ser recuperada entre valor (*VALUE*) e rótulo (*LABEL).*

### 4.4. Limpeza de Dados

O módulo de indexação é responsável também pela execução de procedimentos de limpeza nos dados para melhorar a indexação dos mesmos. Para tanto, esse módulo possui um analisador que faz a uniformização dos termos, passando as letras para o formato minúsculo e removendo variações indesejadas em rótulos de formulários através do processo de *stemming* e *remoção de stop words*.

*Remoção de stop words* é o processo de remover palavras que possuem elevada recorrência e que isoladas possuem baixo valor semântico. Essas palavras, devido as suas frequentes aparições, não são relevantes para fins de recuperação de informação.

A remoção de *stop words* reduz consideravelmente o tamanho das estruturas de índice, podendo atingir uma taxa de redução 40% ou ainda superior [Yates & Neto, 1999]. Essas palavras geralmente são pronomes, artigos, alguns advérbios e verbos.

*Stemming* é o processo de reduzir palavras aos seus radicais. Esse processo possibilita, no processamento da consulta, recuperar variações do termo informado na entrada que possam também interessar ao usuário.

Esses dois processos de limpeza de dados se aplicam aos rótulos dos formulários e são importantes considerando a alta heterogeneidade na

nomenclatura dos campos dos formulários. Essa falta de padronização viabiliza a existência de diversas grafias para a representação de um mesmo dado.

Um exemplo da inexistência de padrões de criação de campos de formulários e da importância desses processos de limpeza é retratado na Figura 11, quando um elemento com rótulo “Do ano” e outro elemento com rótulo “Ano” sem o processo de limpeza seriam indexados em posições diferentes no índice. Com a utilização de um analisador, os campos são indexados uma única vez, com letras minúsculas e sem a *stop word* “Do”.

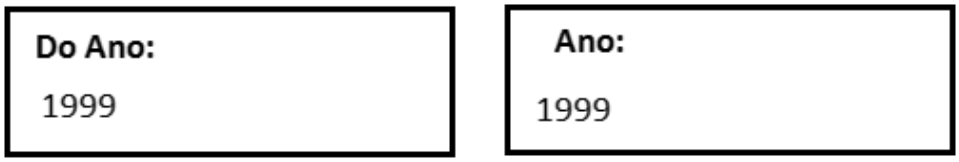


**Figura 11. Exemplo de rótulos extraídos de formulários**

Esse analisador também é utilizado no processo de consulta por formulários web. Neste caso, os rótulos dos elementos do *template* de consulta passam igualmente por um processo de limpeza. Sendo utilizados, os termos resultantes do processo de limpeza para o acesso aos índices.

### 4.5. Modelo de Recuperação de Informação e Acesso aos Índices

O acesso à formulários web relevantes através dos índices se baseia no tradicional modelo booleano de recuperação de informação [Yates & Neto, 1999]. Esse modelo possibilita a aplicação de teoria de conjuntos e álgebra booleana no processo de recuperação de informação.

Basicamente, uma consulta do usuário retorna um conjunto de elementos para cada filtro de entrada. Em seguida, através da análise dos

operadores lógicos AND, OR e NOT, é formado o conjunto resultante para a consulta.

Para ilustrar a sua utilização, suponha que um usuário necessita encontrar formulários que possuam veículos da marca (*brand*) GM e rótulo ano (*year*), ou seja, um formulário *template* que atenda a estas 2 restrições[3]. O módulo de Busca gera então a seguinte consulta: *GM###brand AND year###LABEL*. Estes filtros gerados são então passados para o módulo de Indexação. Considerando o filtro "GM###*brand*", o módulo de Indexação o caracteriza como sendo um filtro de contexto (formado por 2 termos – rótulo e um possível valor para ele), verifica a necessidade de limpeza de dados para o rótulo. Supondo que o termo "*brand*" tenha apenas um (1) centróide como correspondente no índice (*make*), o filtro é convertido para "GM###make" através de um acesso ao índice de sinônimo se a entrada correspondente a este filtro é então acessada no índice de contexto. O mesmo raciocínio se aplica ao filtro "year##LABEL" e o conjunto de formulários web resultante de cada filtro retornado pelo módulo de Indexação é processado posteriormente pelo módulo de busca conforme os operadores lógicos definidos na consulta.

[3]Maiores detalhes sobre como o módulo de busca processa templates e gera filtros de consulta para o módulo de indexação estão fora do escopo deste trabalho.

# 5. Implementação

A implementação do módulo de indexação foi realizada utilizando a linguagem de programação Java em conformidade com os demais módulos previamente desenvolvidos do sistema WF-Sim. As seções a seguir apresentam o conjunto de ferramentas, plugins e api's utilizadas neste trabalho, e como é realizada a implementação das estruturas de índice e da estrutura de cache.

## 5.1. Tecnologias utilizadas

Nessa seção são apresentadas as tecnologias utilizadas para a implementação dos índices, realização de analise e trato de sinônimos dos rótulos dos formulários.

### 5.1.1 Lucene

A ferramenta Lucene[4] foi escolhida para a implementação das estruturas de indexação propostas [Hatcher & Gospodnetic, 2005]. O Lucene é uma biblioteca de software para a recuperação de informação, sendo responsável pela indexação desta informação. Essa ferramenta possibilita a criação de índices invertidos, abordagem típica para recuperação de informação na web [Yates & Neto, 1999]. Como visto no capítulo 4, o índice invertido é uma estrutura de dados que mapeia um documento para uma entrada de índice através de um termo existente neste.

[4] http://lucene.apache.org/core/

O Lucene é um software de código aberto e licenciado através da Apache Software Foundation, estando disponível nas linguagens Java, C, C++, etc.

### 5.1.2. WordNet

WordNet[5] é um banco de dados léxico da língua inglesa bastante utilizado como um dicionário de suporte para análise de textos e aplicações envolvendo inteligência artificial. O WordNet agrupa palavras da língua inglesa em conjuntos de sinônimos e outros tipos de relações semânticas, além de fornecer definições gerais das mesmas.

### 5.1.3. JWI – JAVA WordNet Interface

Nesse trabalho foi utilizada a biblioteca Java WordNet Interface (JWI)[6], que fornece acesso ao banco de dados de informações do WordNet. Sua utilização foi com a finalidade de realizar o acesso aos conjuntos de termos sinônimos do WordNet para a criação dos índice de sinônimos.

## 5.2. Estruturas de Índice

O Lucene trabalha com o conceito de Documento. Um documento é uma abstração usada para representar os elementos persistidos que desejam adicionar ao índice. Cada documento possui um número opcional e variável de campos que são as informações que se deseja indexar, que neste trabalho, são as informações do elemento de formulário como descritas na Figura 12.

[5] http://wordnet.princeton.edu/

[6] http://projects.csail.mit.edu/jwi/

Esses campos armazenados definem a estrutura de índice e podem ser compostos por mais de um campo para a formação das entradas da estrutura de índice, como visto na construção do índice de contexto, que é a combinação do rótulo do formulário com todos os possíveis valores existentes no mesmo.

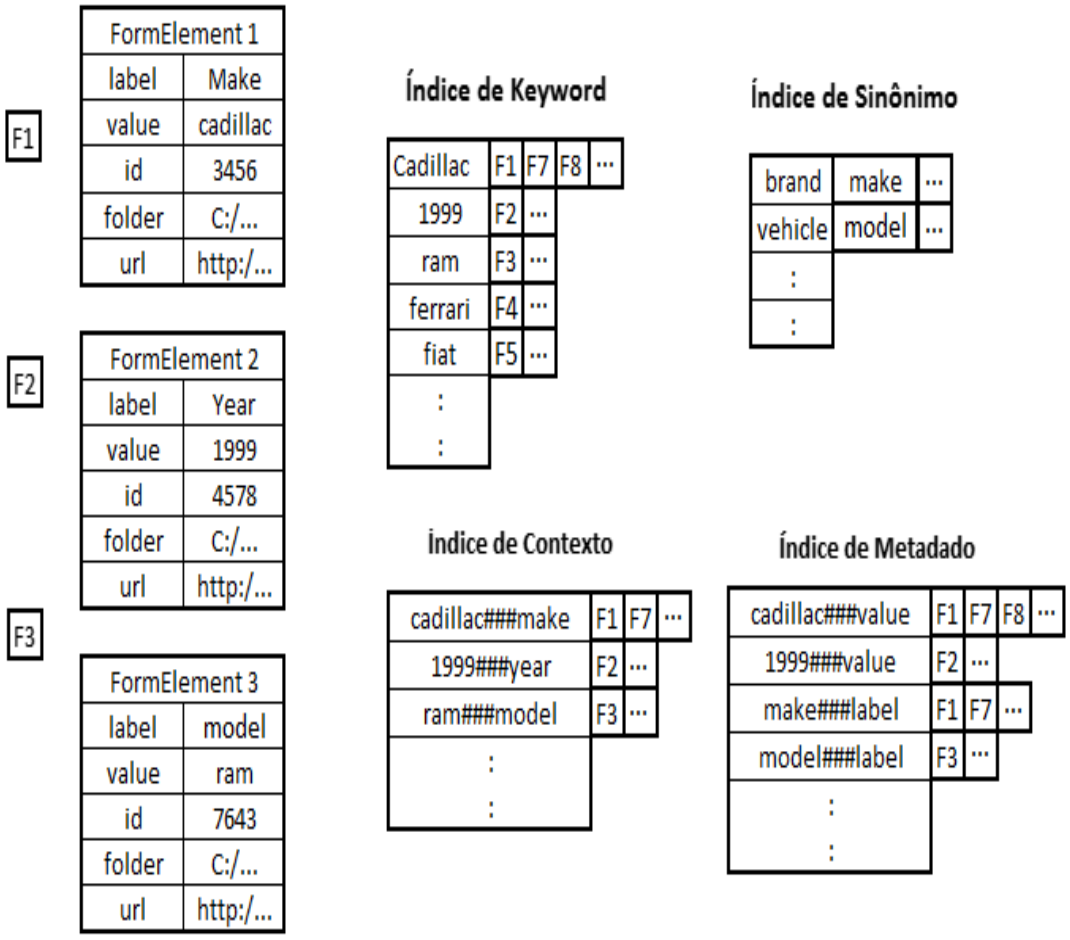


**Figura 12. Informações sobre elementos considerados nos índices**

No contexto deste trabalho, os índices invertidos mapeiam um termo de entrada, como por exemplo, um valor ou um rótulo, para documentos que possuem a referência de elementos de formulários centroides que contenham esse determinado termo. O WF-Sim persiste referências a um elemento de formulário centroide em documentos Lucene com as seguintes propriedades:

rótulo, valores que lhe podem ser atribuídos, endereço do formulário web original (URL), um identificador junto ao banco de dados e outro identificador junto ao sistema de arquivos. Neste caso, os índices definidos no Lucene apontam para a localização dos documentos que possuem a referência aos elementos persistidos em sistemas de arquivos ou banco de dados.

### 5.3. Analisador

Outra abstração importante do Lucene é o chamado *Analyzer* que é o analisador pelo qual as informações a serem indexadas passam por um processo de uniformização e tratamento para a extração de um texto de melhor qualidade e padronizado. O Lucene possibilita operar extensões dessa abstração e desse modo foi criado uma classe analisador que contemplasse as necessidades de limpeza de dados, conforme descrito na Seção 4.4.

A classe *WFSimAnalyzer* divide a entrada em tokens. Em seguida, executa um processo de uniformização dos tokens passando todas as letras para a forma minúscula. Depois, remove os tokens que sejam considerados termos *stopwords* e, por fim, para os tokens remanescentes, executa o processo de stemming. Ao final desse processo, os tokens resultantes são usados para a indexação dos formulários.

### 5.4. Índice de Sinônimos

No momento da criação dos índices de palavra-chave, metadado e contexto, através da biblioteca JWI, são recuperados junto ao banco de dados do Wordnet, os conjuntos de sinônimos dos rótulos dos formulários centroides. O índice de sinônimos é um índice invertido de estrutura semelhante aos três

anteriores, onde os termos sinônimos fazem referência ao termo existente no formulário centroide indexado.

Para realizar a recuperação de um determinado filtro de consulta, primeiro é feita uma consulta ao índice de sinônimos de modo a verificar o termo adequado, em caso de o filtro especificado ser um sinônimo do rótulo de um centróide, o filtro é tratado e seu rótulo substituído pelo valor adequado.

### 5.5. Cache

O processo de *caching* pode ser visto como uma porção de memória que é reservada e utilizada para armazenar informação de modo que, em futuros acessos, o processo de recuperação seja mais rápido. A contribuição do uso da técnica de *caching* é reduzir o número de requisições a um determinado dado, mantendo essa informação, quando muito relevante, em cache, eliminando requisições subseqüentes à persistência de um sistema.

Neste trabalho, o armazenamento dos conjuntos resultantes de consultas mais populares é realizado através da definição de um tipo específico de diretório no Lucene. Através do *RAMDirectory* existente no Lucene, a informação de um índice pode ser carregada em memória e ser utilizada como uma estrutura de cache.

Essa estrutura como visto na Figura 13 mapeia um filtro de consulta tratado pelo analisador a um conjunto de URLs.

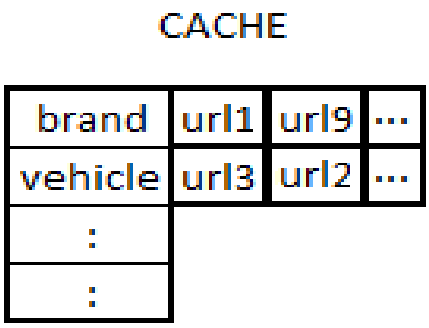


**Figura 13. Estrutura de Cache**

Outra estrutura utilizada para melhorar o desempenho de consultas, neste caso ao acesso à persistência de sistema de arquivos, é um *HashMap.* Essa estrutura foi utilizada de modo que os centroides já carregados em uma consulta fiquem armazenados de maneira volátil em memória até o final do processamento da consulta. Ela é aplicada ao sistema de arquivos, onde mais de um filtro de consulta pode compartilhar acesso a um mesmo centroide e assim evita um acesso repetido à persistência melhorando o desempenho da consulta.

Os dados são armazenados no HashMap onde a chave é o ID do centroide que mapeia para o conjunto das URLs presentes no cluster deste centroide. Com o processamento dos filtros subsequentes, é sempre verificado se essa estrutura possui ou não os centroides a serem recuperados.

# 6. Experimentos e Resultados

Foram realizados quatro experimentos de modo a validar a implementação da estratégia de indexação desenvolvida para o projeto WF-Sim. O primeiro experimento apresenta um comparativo do desempenho de consultas quanto ao tempo de resposta para as duas estratégias de persistência de modo a avaliar qual delas se mostra mais indicada para organizar e recuperar formulários Web.

O segundo experimento apresenta um comparativo entre os tempos de resposta obtidos no índice de palavra chave, tradicionalmente o mais utilizado em trabalhos voltados recuperação de informação na Web, frente às estruturas propostas. O terceiro experimento apresentara o impacto da utilização da estrutura de cache. O quarto e último apresenta o comparativo do desempenho das consultas quando submetidas a bases de formulários de tamanho diferentes de modo a avaliar a escalabilidade das consultas.

Os experimentos realizados foram desenvolvidos em um computador com processador Intel Core I7 2.3 GHz, memória 8 GB, 750 GB de disco rígido. Ele possui sistema operacional Windows 7 Home Premium SP1 64-bit e sistema gerenciador de banco de dados MySQL. As consultas foram realizadas sobre uma amostra de dados composta por 1090 formulários Web com 6157 elementos.

## 6.1. Comparativo do desempenho de acesso às estratégias de persistência de formulários Web

O objetivo desse experimento é avaliar o melhor desempenho entre as duas estratégias de persistência presentes no projeto WF-Sim. Os tempos das consultas apresentados nas tabelas são o valor médio dos tempos de execução repetidas vezes de cada consulta, desconsiderada a primeira execução.

Para que se possa avaliar posteriormente o desempenho do índice de palavra chave frente aos índices de contexto e metadado, foram formulados três conjuntos de consultas compostos por 5 consultas. Cada consulta de um conjunto para testar o índice de palavra chave é correspondente a de mesma posição nos outros dois conjuntos de maneira a acessar as mesmas informações presentes no banco de dados, para que se pudesse demonstrar as peculiaridades da especificação para cada estrutura de índice.

Analisado apenas as consultas mais simples e gerais que são às presentes no conjunto do índice de palavra chave, pode ser visto que as consultas possuem um, dois, quatro e 10 filtros. Esse número varia nos outros dois conjuntos: um, dois e cinco filtros no índice de contexto e, para o conjunto do índice de metadado, o mesmo número de filtros do índice de palavra chave foi utilizado.

| PALAVRA CHAVE | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 1999 | 627 | 12,46 ms | 687,48 ms |
| jeep AND make | 462 | 16,88 ms | 707,14 ms |
| jeep AND make AND corolla AND model | 194 | 21,29 ms | 737,87 ms |
| acura AND make AND 1999 AND year | 320 | 27,87 ms | 906,08 ms |
| audi AND make AND bmw AND car AND $12,000 AND price AND red AND color AND 1999 AND year | 4 | 45,96 ms | 967,72 ms |

**Tabela 1. Desempenho da execução de consultas realizadas junto a estrutura de índice de palavra-chave sem caching**

A Tabela 1 apresenta o resultado desse experimento para o conjunto de consultas ao índice de palavra chave. Eles mostram uma grande superioridade de desempenho de acesso ao banco de dados, que registra, para essa amostra, na média de todos os resultados obtidos, um desempenho 32 vezes superior aos registrados na abordagem de arquivos.

| CONTEXTO | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 1999###year | 266 | 6,9 ms | 120,05 ms |
| jeep###make | 355 | 7,44 ms | 368,3 ms |
| jeep###make AND corolla###model | 86 | 9,06 ms | 494,07 ms |
| acura###make AND 1999###year | 154 | 9,51 ms | 427,45 ms |
| audi###make AND bmw###car AND $12,000###price AND red###color AND 1999###year | 0 | 13,01 ms | 597,46 ms |

**Tabela 2. Desempenho do conjunto de consultas realizadas junto à estrutura de índice de contexto sem caching**

A Tabela 2 apresenta resultados referentes ao conjunto de consultas submetido ao índice de contexto e este conjunto também se mostra favorável a

abordagem de banco de dados. Esse experimento registra como média dos tempos obtidos para banco de dados uma superioridade de 43 vezes, se comparada com a abordagem de arquivos.

A Tabela 3 demonstra o conjunto de consultas realizadas ao índice de metadados e como registrado nas medições obtidas para os índices anteriores, manteve-se o bom desempenho da abordagem de banco de dados, com média 33 vezes superior. .

| METADADO | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 1999###value | 627 | 12,5 ms | 692,15 ms |
| jeep###value AND make###label | 353 | 14,98 ms | 564,46 ms |
| jeep###value AND make###label AND corolla###value AND model###label | 186 | 26,07 ms | 908,03 ms |
| acura###value AND make###label AND 1999###value AND year###label | 83 | 17,36 ms | 601,42 ms |
| audi###value AND make###label AND bmw###value AND car###label AND $12,000###value AND price###label AND red###value AND color###label AND 1999###value AND year###label | 1 | 40,86 ms | 936,36 ms |

**Tabela 3. Desempenho do conjunto de consultas realizadas junto à estrutura de índice de metadado sem caching.**

Essa superioridade registrada pela estratégia de persistência em banco de dados é fundamentada pelas otimizações presentes na estrutura dessa solução. Outra justificativa é a forma de recuperação de dados das duas abordagens: no banco de dados, ao se fazer uma requisição é possível recuperar a informação de múltiplos centroides a partir de uma única consulta, enquanto que no sistema de arquivos a recuperação é realizada individualmente.

## 6.2. Comparativo do Desempenho entre Ìndice de Palavra Chave e os Ìndices Propostos

De maneira a avaliar o desempenho obtido entre as estruturas de índice, é utilizado um histograma que compara as médias de tempos obtidas por cada conjunto de pesquisa aplicada às duas estratégias de persistência.

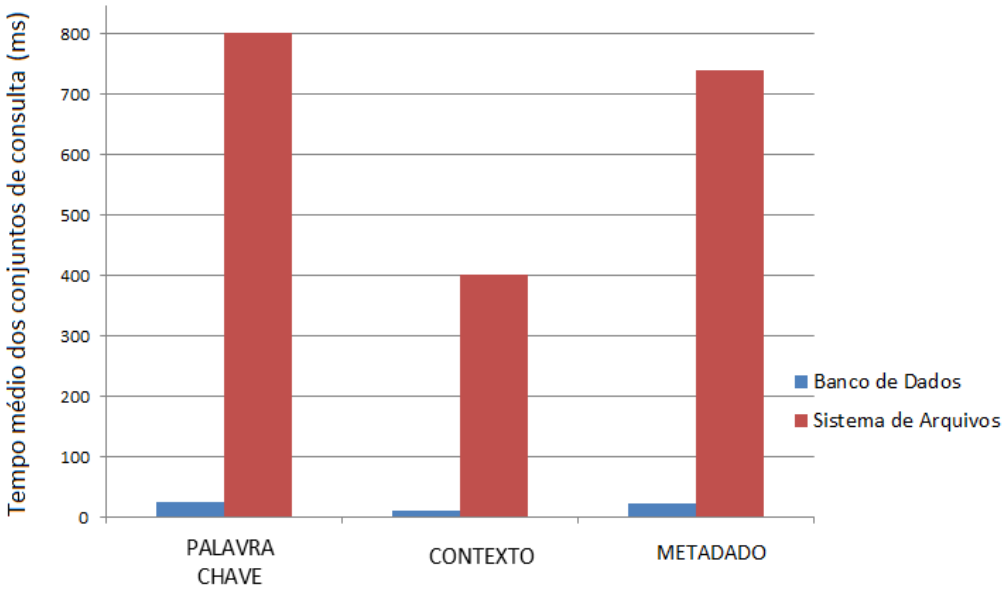


**Figura 14. Média dos tempos obtidos para conjunto de consultas no comparativo entre Banco de Dados e Sistema de Arquivos**

Através da Figura 14, é possível mensurar as otimizações que ocorrem na estrutura de índice de contexto que eliminam o elevado processamento adicional presente na estrutura de palavra chave. Com dito anteriormente, cada consulta de um conjunto é correspondente a de mesma posição nos outros dois conjuntos de consulta. Dessa forma, para cada filtro especificado no índice de contexto são necessários dois filtros e um operador lógico AND para que se possa representá-lo em uma consulta submetida ao índice de palavra chave.

Esse processamento adicional representa nos experimentos um incremento de 50% nos tempos obtidos.

Fazendo a comparação com o índice de metadado, percebe-se que o tempo de acesso deste índice e tempo de acesso do índice palavra chave é praticamente igual, com uma pequena diferença a favor do índice de metadado. Esse pequena diferença é explicável visto que todas as consultas dos dois conjuntos possuem o mesmo número de filtros e operadores, e a vantagem obtida pelo índice de metadado, mesmo discreta, é percebida pois ao ser especificado o tipo de metadado o conjunto recuperado é geralmente uma fração do obtido no índice de palavra chave, mostrando a sua vantagem.

### 6.3. Avaliação do desempenho de consultas com o uso ou não da estrutura de Cache

Neste tipo de experimento, todos os filtros que compõem as consultas dos testes estão mantidas em cache. Nas tabelas a seguir são realizadas comparações entre os tempos de recuperação das consultas efetuadas, utilizando persistência de banco de dados, com a utilização ou não da estrutura de cache proposta neste trabalho.

Na Tabela 4, é possível perceber um ganho substancial do uso de estrutura de cache visto que o tempo mais alto obtido com o uso dela é ainda menor que o menor tempo registrado sem o uso de cache.

| PALAVRA CHAVE | Qtde | BD Sem Cache Tempo | BD Com Cache Tempo |
|---|---|---|---|
| 1999 | 627 | 12,46 ms | 2,2 ms |
| jeep AND make | 462 | 16,88 ms | 3,89 ms |
| jeep AND make AND corolla AND model | 194 | 21,29 ms | 4,79 ms |
| acura AND make AND 1999 AND year | 320 | 27,87 ms | 6,45 ms |
| audi AND make AND bmw AND car AND $12,000 AND price AND red AND color AND 1999 AND year | 4 | 45,96 ms | 9,89 ms |

**Tabela 4. Desempenho do conjunto de consultas ao índice de palavra chave considerando ou não o uso da estrutura de cache**

A Tabela 5 se refere a consultas que utilizam o índice de contexto e, neste caso, a utilização da cache continua a apresentar melhor desempenho. Na comparação das médias obtidas, o desempenho com o uso de cache o é 4 vezes superior.

| CONTEXTO | Qtde | BD Sem Cache Tempo | BD Com Cache Tempo |
|---|---|---|---|
| 1999###year | 266 | 6,9 ms | 1,56 ms |
| jeep###make | 355 | 7,44 ms | 1,54 ms |
| jeep###make AND corolla###model | 86 | 9,06 ms | 2,1 ms |
| acura###make AND 1999###year | 154 | 9,51 ms | 2,36 ms |
| audi###make AND bmw###car AND $12,000###price AND red###color AND 1999###year | 0 | 13,01 ms | 3,68 ms |

**Tabela 5. Desempenho do conjunto de consultas ao índice de contexto considerando ou não o uso da estrutura de cache**

A Tabela 6, como as duas anteriores, confirma o melhor desempenho da utilização cache considerando o acesso ao índice de metadado.

| METADADO | Qtde | BD Sem Cache Tempo | BD Com Cache Tempo |
|---|---|---|---|
| 1999###value | 627 | 12,5 ms | 2,21 ms |
| jeep###value AND make###label | 353 | 14,98 ms | 3,63 ms |
| jeep###value AND make###label AND corolla###value AND model###label | 186 | 26,07 ms | 6,71 ms |
| acura###value AND make###label AND 1999###value AND year###label | 83 | 17,36 ms | 4,27 ms |
| audi###value AND make###label AND bmw###value AND car###label AND $12,000###value AND price###label AND red###value AND color###label AND 1999###value AND year###label | 1 | 40,86 ms | 10,58 ms |

**Tabela 6. Desempenho do conjunto de consultas ao índice de metadado considerando ou não o uso da estrutura de cache**

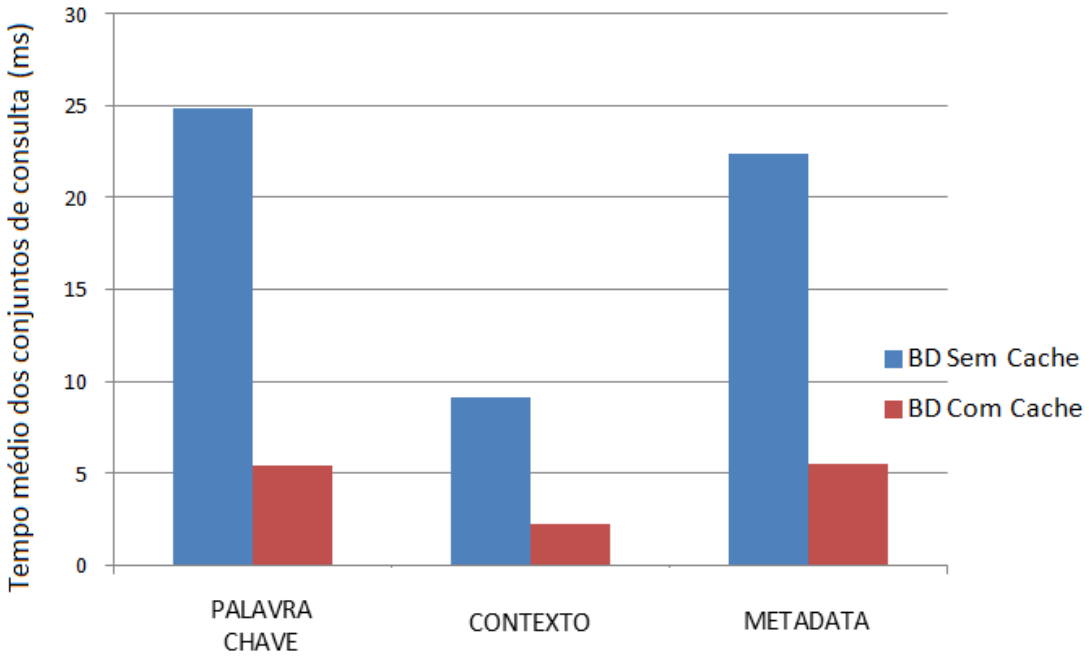


**Figura 15. Tempo médio para os conjuntos de consultas para as três estruturas de índice comparando as duas estratégias de persistência e utilizando a estrutura de Cache.**

Na Figura 15 são apresentadas as médias dos conjuntos de consultas submetidos às duas estratégias de persistência com o uso da estrutura de cache. Nessa tabela se percebe que os tempos entre as duas persistências são muito distintos, sendo o desempenho do uso da estrutura de cache 4 vezes

superior e, assim, comprovando a relevância dessa estrutura para a realização de consultas.

### 6.4. Avaliação de escalabilidade do sistema

Esse experimento propõe realizar uma comparação semelhante a executada no experimento 1, aplicado a duas bases de formulários com tamanho diferentes. Para esse experimento, foram utilizadas uma base com 1000 formulários Web e outra com 2000.

As duas bases possuem em sua maioria informação do domínio Book e desse modo foram desenvolvidos três novos conjuntos de consulta de modo a avaliar a escalabilidade do sistema.

As Tabelas 7, 8 e 9 apresentam o desempenho das consultas submetidas as três estruturas de índice da amostra de 1000 formulários. O desempenho obtido para consultas dessa base apresenta resultados similares aos obtidos no experimento 1. O índice de palavra chave mantém seus tempos maiores que aos apresentados nos índices propostos e com banco de dados sendo a estratégia de persistência mais eficaz.

| PALAVRA CHAVE | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 2005 | 93 | 12,64 ms | 107,45 ms |
| philosophy AND subject | 77 | 21,43 ms | 248,14 ms |
| philosophy AND subject AND german AND language | 4 | 28,77 ms | 265,11 ms |
| history AND category AND 2005 AND year | 12 | 23,46 ms | 234,96 ms |

**Tabela 7. Desempenho do conjunto de consultas ao índice de palavra chave submetido a amostra de 1000 formulários.**

| CONTEXTO | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 2005###year | 9 | 10,89 ms | 18,52 ms |
| philosophy###subject | 49 | 11,16 ms | 28,45 ms |
| philosophy###subject AND portuguese###language | 2 | 13,07 ms | 36,43 ms |
| history###category AND 2005###year | 1 | 10,56 ms | 47,62 ms |

**Tabela 8. Desempenho do conjunto de consultas ao índice de contexto submetido a amostra de 1000 formulários.**

| METADADO | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 2005###value | 93 | 12,35 ms | 105,78 ms |
| philosophy###value AND subject###label | 54 | 17,17 ms | 92,32 ms |
| philosophy###value AND subject###label AND portuguese###value AND language###label | 0 | 22,28 ms | 129,85 ms |
| history###value AND category###label AND 2005###value AND year###label | 2 | 18,17 ms | 171,96 ms |

**Tabela 9. Desempenho do conjunto de consultas ao índice metadado submetido a amostra de 1000 formulários.**

As Tabelas 10, 11 e 12 apresentam o desempenho das consultas submetidas as três estruturas de índice da amostra de 2000 formulários. Realizando um comparativo com os tempo obtidos na base de 1000, o maior aumento registrado foi na persistência do sistema de arquivos onde ocorreu um aumento de 84% na média das consultas do índice de palavra chave.

| PALAVRA CHAVE | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 2005 | 245 | 14,11 ms | 183,85 ms |
| philosophy AND subject | 135 | 32,5 ms | 490,49 ms |
| philosophy AND subject AND german AND language | 8 | 43,48 ms | 534,87 ms |
| history AND category AND 2005 AND year | 4 | 29,28 ms | 366,41 ms |

**Tabela 10. Desempenho do conjunto de consultas ao índice de palavra chave submetido a amostra de 2000 formulários.**

| CONTEXTO | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 2005###year | 88 | 11,89 ms | 46,71 ms |
| philosophy###subject | 103 | 12,15 ms | 42,11 ms |
| philosophy###subject AND portuguese###language | 6 | 14,46 ms | 50,2 ms |
| history###category AND 2005###year | 0 | 12,51 ms | 35,14 ms |

**Tabela 11. Desempenho do conjunto de consultas ao índice de contexto submetido a amostra de 2000 formulários.**

| METADADO | Qtde | Banco de Dados Tempo | Sist. Arquivos Tempo |
|---|---|---|---|
| 2005###value | 245 | 14,95 ms | 186,54 ms |
| philosophy###value AND subject###label | 109 | 17,36 ms | 125,3 ms |
| philosophy###value AND subject###label AND portuguese###value AND language###label | 0 | 23,09 ms | 188,07 ms |
| history###value AND category###label AND 2005###value AND year###label | 1 | 26,04 ms | 252,31 ms |

**Tabela 12. Desempenho do conjunto de consultas ao índice de metadado submetido a amostra de 2000 formulários.**

O maior aumento registrado para as consultas submetidas ao banco de dados foi de 38%, também no índice de palavra chave. Entretanto mesmo com o aumento, os tempos se mostram adequados visto que as médias dos conjuntos de consulta de banco de dados não chegaram a alcançar nem ao menos um centena de milisegundos de tempo de resposta.

## 7. Trabalhos Relacionados

As fontes de pesquisa disponíveis na literatura são bastante escassas se tratando em soluções para a indexação de formulários Web visando consultas por similaridade.

O trabalho que possivelmente mais se aproxima de nossa abordagem é o de [Shao et. All, 2003], onde também são extraídos dados de bancos de dados escondidos e estes armazenados em banco de dados relacional. Esse banco de dados mantém um índice para cada campo de um formulário Web que mantém seus valores, bem como apontadores para clusters de valores similares. Percebe-se várias limitações em comparação com a nossa proposta:

I. Um grande número de índices deve ser criado, se quisermos indexar valores de vários campos;
II. Eles assumem que um esquema de *matching* para formulários Web em um mesmo domínio foi previamente realizado para gerar um esquema global relacional através de um sistema de mediação. Tal suporte aumenta a complexidade da abordagem;
III. Eles são capazes de responder apenas consultas sobre valores de campos. Nossa abordagem possui um escopo mais amplo, podendo realizar consultas por rótulos de campos por similaridade (e seus valores), bem como termos que atuam como metadados específicos para um determinado formulário web.
IV. Nenhuma pesquisa de palavras-chave tradicional é também fornecida.

Além destas limitações, os seus experimentos enfatizam a eficiência de compressão de índices, bem como a precisão para evitar a recuperação de

falsos positivos. Nossos experimentos, por outro lado, enfatizam a eficiência em termos do tempo de processamento de acesso às estruturas de índice, analisando o impacto da realização de otimizações de consultas no momento de criação dos índices na forma das estruturas de contexto e metadado.

Outra abordagem relevante é o trabalho de [Dong, X., Halevy, A., 2007], que lida com o problemática mais geral de indexação de *dataspaces* e suporta apenas consultas por contexto. Nossa abordagem, além de índices de contexto para formulários Web, provê suporte adicional de índice de metadados às consultas. Pode-se argumentar que a nossa abordagem pode ser potencialmente útil para *dataspaces*, considerando dados que são similares aos de formulários Web.

## 8. Conclusão e Trabalhos Futuros

Este trabalho apresenta e valida uma estratégia de indexação visando buscas por similaridade para dados de formulários Web no contexto do projeto WF-Sim. Esta estratégia introduz mecanismos de refinamento para o contexto de formulários web. Esses mecanismos consideram a indexação de informações sobre elementos de formulários em estruturas de índice especificadas por contexto e metadado, além da tradicional busca por palavra chave. As duas primeiras estruturas garantem otimizações para o módulo de busca uma vez que as estruturas propostas já indexam os filtros de consultas mais frequentes para formulários. Buscas tradicionais considerando apenas palavras chave teriam que, no caso de uma busca por contexto, por exemplo, recuperar inicialmente informações sobre o rótulo, depois sobre o valor desejado e após computar uma intersecção dos formulários recuperados. Este *overhead* é desnecessário com a introdução do índice de contexto, e o mesmo vale para o índice de metadado.

O trabalho de [Mello et. al. 2010] foi a base escolhida para a construção das estruturas de índice aqui apresentadas. [Mello et. al. 2010] define índices de contexto e de metadado. Entretanto, o escopo do trabalho não é específico para o contexto de buscas por similaridade em formulários web, que é o objetivo do projeto WF-Sim. Este trabalho aplica e estende essas idéias para o propósito do projeto. Nenhum trabalho relacionado na literatura se propõe a definir estruturas de índice para buscas por similaridade sobre formulários web.

O próximo passo no contexto deste trabalho é avaliar o desempenho do módulo de indexação com um repositório maior de formulários web. A amostra

de testes do projeto conta com um número aproximado de 1090 formulários. Entretanto, através de uma parceria do GBD/UFSC com a Universidade de Utah, uma amostra de aproximadamente 40.000 formulários está sendo disponibilizada.

Outra atividade futura é considerar consultas que testem dependências entre elementos de formulários Web, como por exemplo, um campo “Marca” cujos valores determinam os valores de um campo “Modelo”, supondo um domínio de veículos. A intenção é considerar filtros que testem a existência de tais dependências e definir estruturas de indexação que facilitem buscas por similaridade neste contexto. Percebe-se que este tipo de consulta é relevante para casos em que o usuário deseja acessar formulários que implementam automaticamente uma cadeia de dependências entre campos, facilitando, assim, a sua intenção de busca por alguma informação.

Uma atividade futura é o aperfeiçoamento da estrutura de cache, visto que não foi possível implementar todas as operações de gerenciamento da cachê, como por exemplo, a permanência dos filtros mais utilizados em memória.

Esse trabalho desenvolvido teve premiada sua relevância científica no evento da Escola Regional de Banco de Dados (ERBD) 2013, recebendo o prêmio de melhor artigo na categoria Pesquisa.

## Referências Bibliográficas


BERGMAN. K. Michael. The Deep web: Surfacing Hidden Value. Disponível em: http://brightplanet.com/images/uploads/12550176481-DeepWebwhitepaper.pdf

Hatcher, E.; Gospodnetic, O. (2005).“Lucene in Action”. Greenwich: Manning Publications Co, 2005.

Gonçalves, R. et. al. (2011). “A Similarity Search Approach for *Web forms*”. In: Proceedings of the IADIS International Conference IADIS WWW/Internet.

Madhavan, J. et al. (2009) “Harnessing the Deep Web: Present and Future”. In: 4th Biennial Conference on Innovative Data Systems Research (CIDR 2009).

Silva, F. R.; Mello, R. S. (2012). “Estratégias de Persistência de Clusters em uma Técnica de Casamento por Similaridade para Web Forms”. In: VIII Escola Regional de Banco de Dados (ERBD 2012).

Mello, R. S., Pinnamaneni, R., Freire, J., (2010) Indexing Web Form Constraints., In: Journal of Information and Data Management (JIDM), Vol. 5, nº. 3, p.348-358.

Baeza-Yates, R.; Ribeiro-Neto, R. Modern Information Retrieval. (1999). ACM Press / Addison-Wesley.

Qiu, J., Shao, F., Zatsman, M., Shanmugasundaram, J.: Index Structures for Querying the Deep Web. In: 6th International Workshop on Web and Databases(WebDB). San Diego, CA, USA (2003)

Dong, X., Halevy, A.: Indexing Dataspaces. In: SIGMOD Conference.pp.43–54. Beijing, China (2007)

Apache Lucene. Disponível em http://lucene.apache.org/core/

WordNet, A lexical database for English. Disponível em: http://wordnet.princeton.edu/

JWI – Java Wordnet Interface. Disponível em: http://projects.csail.mit.edu/jwi/

# Anexo 1 - Artigo

# Um Método para Indexação de Formulários Web visando Consultas por Similaridade[7]

**Willian Ventura Koerich, Ronaldo dos Santos Mello**

Centro Tecnológico (CTC) - Departamento de Informática e Estatística (INE)

Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC - Brasil

{willian.vkoerich,ronaldo}@inf.ufsc.br

**Abstract:** Search engines do not support specific searches for web forms found on Deep Web. Within this context, the WF-Sim project proposes a query-by-similarity system for Web Forms to deal with this lack. This paper presents an indexing technique for querying-by-similarity web forms as a WF-Sim system component. This technique is centered on suitable index structures to the main kinds of queries applied to web forms, as well as some optimizations in these structures to reduce the number of index entries. To evaluate the indexes' performance, we ran experiments on two persistence strategies: file system and database. The performance of accessing the database was higher. We also compare the performance of our indexes with the traditional keyword-based index, and the results were also satisfactory.

**Resumo:** Motores de busca atuais não possuem suporte à buscas específicas por formulários web relacionados à Deep Web. Neste contexto, o projeto WF-Sim propõe um processador de consultas por similaridade para formulários Web para lidar com esta limitação. Este artigo apresenta uma técnica de indexação para buscas por similaridade em formulários web, atuando como um componente do sistema WF-Sim. Esta técnica está centrada em estruturas de índice adequadas aos principais tipos de consulta aplicados a formulários web, bem como otimizações nestas estruturas para reduzir a quantidade de entradas no índice. Experimentos preliminares sobre duas estratégias de persistência de dados suportados pelo WF-Sim foram realizados: sistema de arquivos e banco de dados. O desempenho de acesso ao banco de dados foi superior. Comparou-se também o desempenho dos índices propostos contra o tradicional índice de palavras-chave, e o resultado também foi satisfatório.

## 1. Introdução

Uma grande quantidade de serviços está atualmente à disposição das pessoas através da Web, como locação e vendas de veículos, reserva de hotéis, compra de livros, oferta de empregos, etc. Esses serviços disponibilizam diversos dados para consultas aos usuários como por exemplo, veículos de diversos fabricantes e modelos, no caso de um web *site* de uma concessionária. O acesso a esses bancos de dados é possível através de formulários existentes em páginas Web. Estes formulários exibem atributos do banco de dados sobre os quais o cliente especifica filtros e então submete consultas. Estes bancos

[7] Este trabalho é parcialmente financiado pelo CNPq através do projeto WF-Sim (Nro. processo:481569/2010-3).

de dados na Web são denominados banco de dados escondidos (*hidden databases* ou *deep-Web*)[8], uma vez que a sua estrutura e o seu conteúdo não estão completamente visíveis ao usuário. Somente alguns atributos (e alguns eventuais valores que permitem a definição de filtros) estão visíveis nos formulários [Madhavan et al. 2009].

O projeto WF-Sim visa desenvolver uma solução para esta problemática: um processador de consultas por similaridade para formulários Web [Gonçalves, 2011]. Este processador caracteriza-se por ser um software responsável pela execução de todas as tarefas necessárias à geração de um resultado adequado a uma consulta por similaridade, como especificação de uma consulta por parte do usuário e métricas adequadas para definição do grau de similaridade entre os formulários. Este projeto propõe ainda um método de busca por campos dos formulários web, internamente chamados de *elementos* de formulários. A Figura 1 mostra um exemplo de formulário web no domínio de veículos. Cada atributo (elemento) de um formulário geralmente possui um rótulo e uma série de valores possíveis associados a ele, como por exemplo, *Make* e *Model*.

Buscas no WF-Sim ocorrem sobre elementos de formulários indexados. As estruturas de índice são definidas a partir de clusters gerados por um processo de *matching* de elementos de formulários, permitindo recuperação de formulários com elementos similares. Estas estruturas de índice foram devidamente projetadas para facilitar as consultas típicas por formulários web.

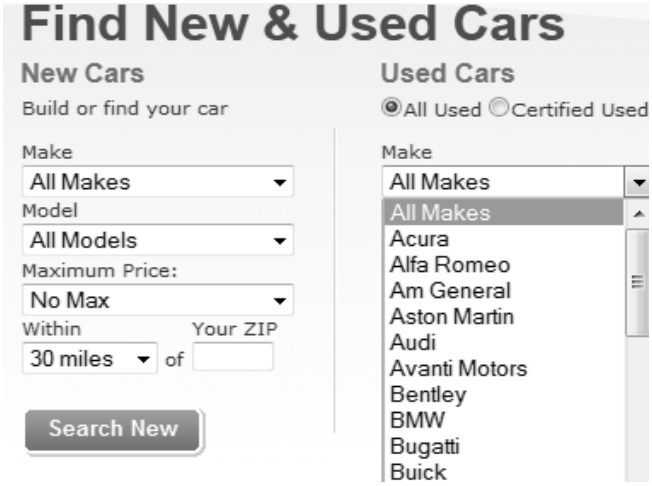


**Figura 16. Exemplo de Formulário Web**

Este artigo apresenta a estratégia de indexação por similaridade para formulários web desenvolvida para o WF-Sim, visando o acesso a dados mantidos em dois tipos de mecanismos de persistência: arquivos e banco de dados. Avalia-se aqui não apenas o desempenho das estruturas de índice para cada tipo de persistência, mas também quais estruturas de índice apresentaram melhor desempenho.

As demais seções detalham o desenvolvimento deste trabalho. A seção 2 aborda o projeto WF-Sim, com foco na atividade de indexação. A seção 3 detalha o módulo de

[8] http://brightplanet.com/wp-content/uploads/2012/03/12550176481-deepwebwhitepaper1.pdf

indexação e as estruturas de índice propostas. A seção 4 descreve os experimentos realizados e a seção 5 é dedicado às conclusões.

## 2. WF-Sim

As técnicas de busca por formulários web atualmente utilizadas geralmente são baseadas no modelo de busca por palavra chave, que executa o *matching* entre a entrada com termos existentes nos formulários. O principal exemplo é a máquina de busca *DeepPeep*[9]. Visando adicionar capacidade de busca por similaridade a formulários web, o projeto WF-Sim, desenvolvido na Universidade Federal de Santa Catarina pelo grupo de Banco de dados (GBD/UFSC)[10], com financiamento do CNPq, se inspira no fato de que os dados disponíveis na Deep Web são relevantes para usuários que desejam encontrar formulários web que satisfaçam suas necessidades em termos de serviços online para os mais diversos fins.

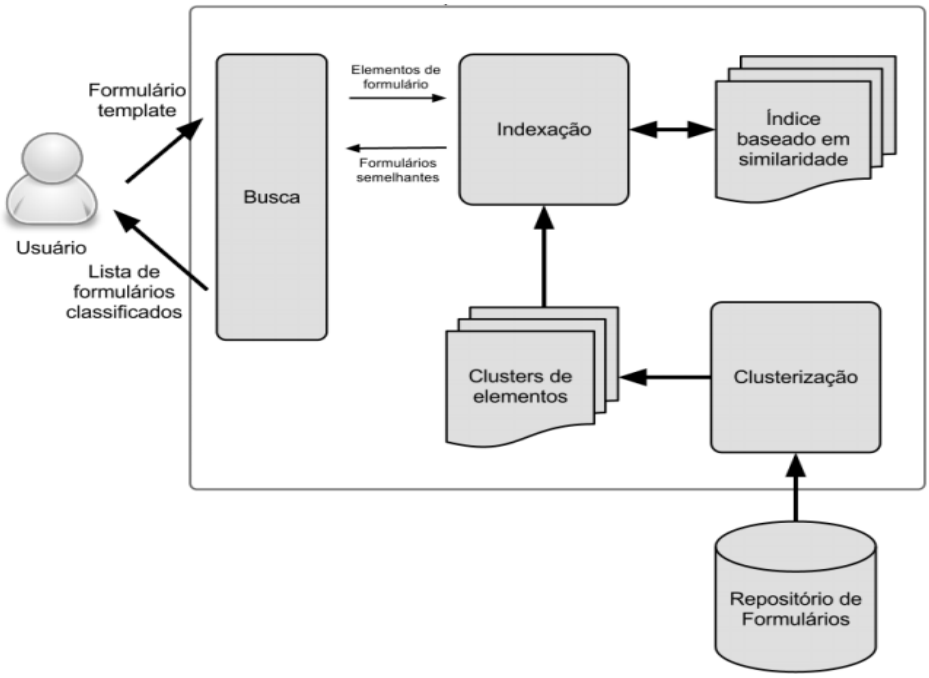


**Figura 17. Arquitetura do Sistema WF-Sim**

O WF-Sim é um processador de busca por similaridade para formulários web baseados nos seus elementos. A Figura 2 mostra a arquitetura do sistema. Com base em uma consulta de entrada, no caso, um conjunto de elementos denominado *formulário template*, o sistema retorna um conjunto de formulários web que possuem elementos similares. O sistema possui os módulos de busca, indexação e clusterização. O componente de clusterização foi alvo de um trabalho anterior [Silva & Mello, 2012], estando, assim, fora do escopo deste artigo. Neste componente são aplicadas métricas de similaridade de modo a agrupar os formulários em clusters com elementos similares.

[9] http://www.deeppeep.org/

[10] http://www.gbd.inf.ufsc.br

O módulo de indexação é o foco deste artigo e visa criar uma estrutura de índice, garantir o acesso a estas estruturas e a recuperar os formulários relevantes. Dado um *formulário template* de entrada, a busca por elementos similares acessa um dicionário de sinônimos que direciona elementos do *template* a elementos sinônimos ditos elementos centroides, ou seja, elementos representativos de um cluster de elementos similares. Um exemplo seria um cluster formado pelos elementos “Make”, “Brand”, “Select a Make”, sendo “Make” eleito como centroide. A próxima seção descreve o módulo de indexação.

## 3. Módulo de Indexação

Três estruturas de índice foram definidas para o acesso a dados de formulários Web: *palavra-chave*, *contexto* e *metadado*. A ideia destas estruturas foi obtida de um trabalho anterior do GBD/UFSC [Mello et. al. 2010] e adaptada à problemática de busca por similaridade em formulários Web. Estas estruturas, em particular os índices de contexto e metadado, são adequadas aos tipos de consulta mais frequentes sobre formulários web.

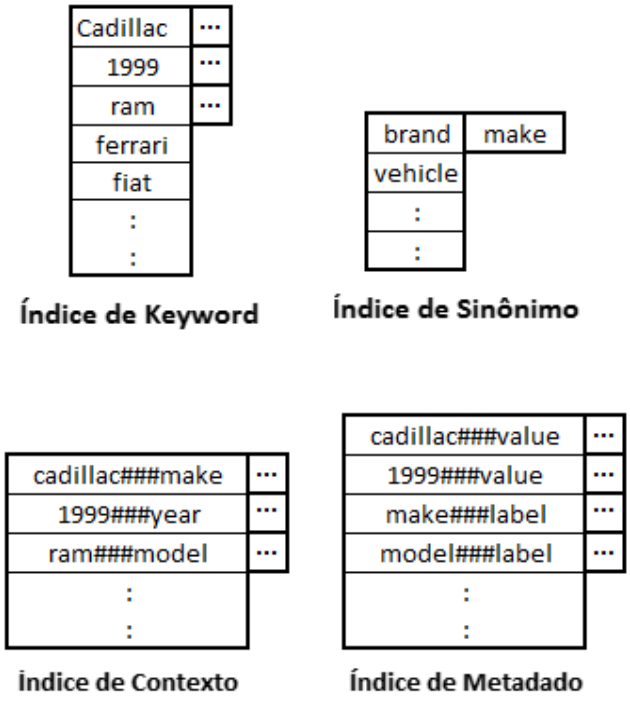


**Figura 18. Estrutura dos Índices Propostos**

Consultas por contexto associam um rótulo aos seus respectivos valores presentes em um formulário. A ideia é recuperar formulários com base na contextualização de campos por domínio, ou seja, permite que o usuário recupere formulários com determinado tipo de conteúdo de campo que lhe interessa. Já consultas por metadado permitem indicar se um termo de interesse é um metadado do tipo rótulo ou valor.

A Figura 3 mostra exemplos dos tipos de índice desenvolvidos. O índice de palavra-chave (*keyword*) possui entradas tradicionais de termos para cada possível valor ou rótulo existente. O índice de contexto define entradas de índice para as combinações de rótulos com seus respectivos valores presentes em elementos, no formato *nome_valor###nome_rótulo*. O índice de metadado classifica o tipo de informação que se deseja recuperar (rótulo ou valor), com entradas no formato *nome_valor###VALUE* ou *nome_valor###LABEL*.

O índice de sinônimos é uma estrutura auxiliar ao funcionamento dos índices de contexto, metadado e palavra-chave. No momento da indexação de um rótulo de um elemento de formulário Web, é feita uma consulta a um banco de dados de sinônimos para verificar se o rótulo é um sinônimo de um centroide. Em caso positivo, é adicionado ao índice de sinônimos uma entrada com o rótulo em questão e sua associação para o centroide sinônimo que está indexado nas outras estruturas de índice. Desta forma, quando o rótulo informado no *template* não estiver explicitamente indexado nas estruturas propostas, mas for um sinônimo de um centroide, é possível encontrar formulários similares através da procura por sinônimos, garantindo, assim, uma busca por similaridade. O índice de sinônimo é fundamental na estratégia de indexação proposta, visto permite que os índices de palavra-chave, contexto e metadado indexem apenas os rótulos dos elementos centroides de cada cluster, viabilizando, estruturas de índice com número reduzido de entradas.

### 3.1 Limpeza de Dados

O método de indexação é responsável também pela execução de procedimentos de limpeza nos dados para melhorar a indexação dos mesmos. Para tanto, esse módulo possui um analisador que faz a uniformização dos termos, passando as letras para o formato minúsculo e removendo variações indesejadas através do processo de *stemming* e remoção de *stop words*. Esses procedimentos garantem que campos de formulários representando uma mesma informação, mas com grafias diferentes, possam ser indexados de maneira uniforme. O processo de remoção de stop words possibilita a diminuição do número de entradas evitando criar entradas para termos existentes nos rótulos que não sejam relevantes para consultas e o processo de *stemming* possibilita reduzir número de modo que apenas a forma raiz dos elementos desejáveis dos rótulos sejam indexados. Um exemplo é mostrado na Figura 4. Um elemento com rótulo “Do ano” e outro elemento com rótulo “Ano” sem o processo de limpeza seriam indexados em posições diferentes no índice. Com a inclusão do analisador, os campos são indexados uma única vez, com letras minúsculas sem a *stop word* “Do”.

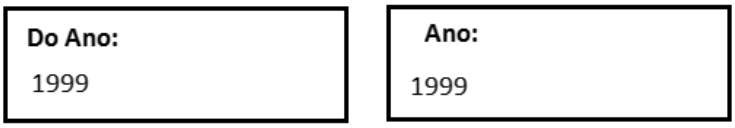


**Figura 4. Exemplo de rótulos extraídos de formulários**

O analisador também é utilizado no processo de busca por formulários web. Neste caso, os rótulos dos elementos do *template* passam igualmente por um processo de limpeza. Após, os termos resultantes são verificados nos índices.

### 3.2 Acesso aos Índices

O acesso a formulários web relevantes através dos índices se baseia no tradicional modelo booleano de recuperação de informação [Yates & Neto, 1999]. Esse modelo possibilita a definição de filtros utilizando os operadores lógicos AND, OR e NOT.

Para ilustrar a sua utilização, suponha que um usuário necessita encontrar formulários que possuam veículos da marca (*brand*) GM e rótulo ano (*year*), ou seja, um formulário *template* com esses 2 elementos[11]. O módulo de Busca gera então a seguinte consulta: *GM###brand AND year###LABEL*. Cada um dos filtros gerados é então passado para o módulo de Indexação. Considerando o filtro "GM###brand", o módulo de Indexação o caracteriza como sendo um filtro de contexto (formado por 2 termos – rótulo e um possível valor para ele), verifica a necessidade de limpeza de dados para o rótulo e então acessa o índice de sinônimos. Supondo que o termo *brand* tenha apenas um (1) centróide como sinônimo (*make*, por exemplo), o filtro é convertido para "GM###make" e a entrada correspondente a este filtro e então acessada no índice de contexto. Caso haja mais de um termo sinônimo para *brand*, as entradas do índice correspondentes a todos os sinônimos são acessadas e é feita uma união das URLs dos formulários web presentes em cada entrada. O mesmo raciocínio se aplica ao filtro "year##LABEL" e o conjunto de formulários web resultante de cada filtro retornado pelo módulo de Indexação é processado posteriormente pelo módulo de Busca conforme os operadores lógicos definidos na consulta.

### 3.3. Implementação

A ferramenta Lucene[12] foi utilizada para a implementação das estruturas de indexação propostas(Hatcher, E.; Gospodnetic, 2005). Lucene é uma biblioteca de software para a recuperação de informação, sendo responsável pela indexação e da informação indexada. Essa ferramenta possibilita a criação de índices invertidos, abordagem típica para recuperação de informação na web [Yates & Neto, 1999], [Elmasri & Shamkant]. O índice invertido é uma estrutura de dados que mapeia um conteúdo para os documentos que o contém.

No contexto deste trabalho, os índices invertidos mapeiam um termo, como por exemplo, um valor de um campo, para os formulários que contenham esse determinado termo. Como pode ser visto na Figura 5, o WF-Sim persiste um elemento de formulário com as seguintes propriedades: rótulo, valores que lhe podem ser atribuídos, endereço do formulário web original (URL) e um identificador junto ao banco de dados. Neste caso, os índices definidos no Lucene apontam para a localização destes elementos persistidos em arquivos ou banco de dados.

[11] Maiores detalhes sobre como o módulo de busca processa templates e gera filtros de consulta para o módulo de indexação estão fora do escopo deste artigo.

[12] http://lucene.apache.org/core/

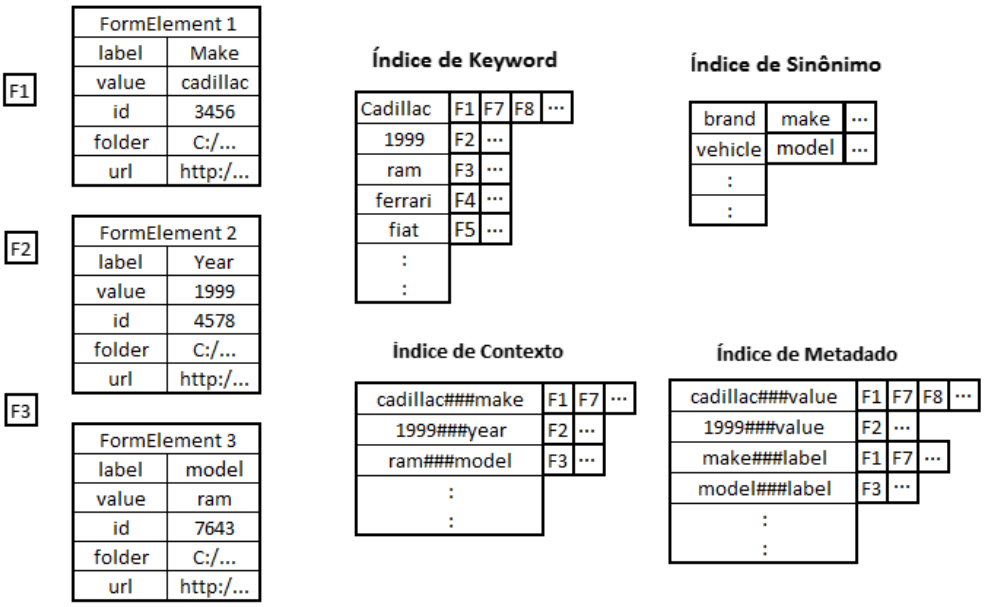


**Figura 5. Informações sobre elementos considerados nos índices**

Para a criação de um banco de dados de sinônimos foi adotada a ferramenta WordNet[13] é um banco de dados léxico bastante utilizado como um dicionário de suporte para análise de textos e aplicações envolvendo inteligência artificial. O WordNet agrupa palavras da língua inglesa em conjuntos de sinônimos e outros tipos de relações semânticas, além de fornecer definições gerais das mesmas.

Nesse trabalho foi utilizada a biblioteca Java WordNet Interface (JWI)[14], que fornece acesso ao banco de dados de informações do WordNet. Sua utilização foi necessária para determinar a similaridade entre elementos e construir o índice de sinônimos.

## 4. Experimentos

A implementação do módulo de indexação foi avaliada através de experimentos preliminares com o objetivo de medir o desempenho do processamento de filtros de consulta acessando as estruturas de índice desenvolvidas. Foram testados índices para elementos de formulários persistidos em arquivos de dados serializados em disco e elementos de formulários persistidos em um banco de dados relacional.

Para os experimentos foi utilizado um computador equipado com um processador Athlon II X2 de 2.9 GHz, memória de 4 GB, disco rígido de 500 GB, sistema operacional Windows 7 Home Basic SP1 de 64 bits e banco de dados MySQL. A amostra de formulários Web disponível para teste compreende 1090 formulários que possuem na totalidade 6157 elementos.

[13] http://wordnet.princeton.edu/

[14] http://projects.csail.mit.edu/jwi/

A Figura 6 mostra a média geral de tempo de execução de um mesmo conjunto de filtros de consulta, específico para cada tipo de índice, acessando arquivos em disco e o banco de dados. Já a Figura 7 mostra resultados de experimentos com uma quantidade incremental de filtros no domínio de veículos, ou seja, sobre um, dois e cinco elementos, visando o acesso ao índice de contexto. O índice de contexto é o índice mais acessado na prática, pois indexa filtros de consulta mais usuais no contexto de formulários web.

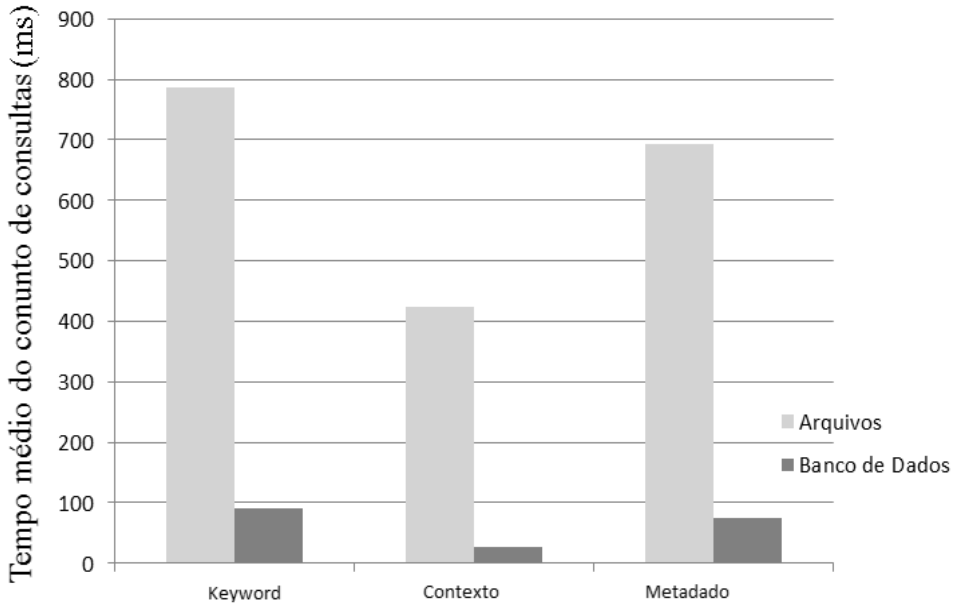


**Figura 6. Média dos tempos de processamento dos filtros de consulta conjunto de testes**

Os testes mostrados na Figura 6 registraram uma grande diferença de desempenho de acesso ao banco de dados frente aos arquivos serializados, ou seja, as consultas gastaram muito mais tempo (8x mais lento, em média) no acesso aos arquivos. Essa superioridade se deve ao fato de que o acesso a arquivos em disco não conta com as otimizações características dos sistemas gerenciadores de banco de dados.

Já com relação aos testes mostrados na Figura 7, percebe-se, para o acesso ao banco de dados, que o tempo de processamento dobrou a cada variação de quantidade de filtros conjuntivos, enquanto que o aumento de tempo foi bem menor no caso do acesso aos arquivos, inclusive com redução na passagem de 2 para 5 filtros. Essa situação precisa ser melhor avaliada com uma bateria mais ampla de testes sobre volumes maiores de dados. Mesmo assim, a diferença de tempo no acesso a arquivos e banco de dados continuou sendo bastante acentuada, sendo o acesso ao banco de dados de 10x a 30x (aproximadamente) mais rápido para filtros conjuntivos.

| CONTEXT | Qtde | Arquivos Tempo | Banco de Dados Tempo |
|---|---|---|---|
| jeep###make | 355 | 381 ms | 11 ms |
| corolla###model | 155 | 118 ms | 10 ms |
| jeep###make AND corolla###model | 86 | 486 ms | 19 ms |
| jeep###make OR corolla###model | 424 | 484 ms | 21 ms |
| jeep###make AND NOT corolla###model | 269 | 484 ms | 20 ms |
| acura###make AND audi###make AND chevrolet###make AND 1000###price AND maverick###vehicle" | 0 | 459 ms | 36 ms |
| acura###make OR audi###make OR bentley###make OR 1964###year OR 1999###year | 464 | 490 ms | 49 ms |
| acura###make OR audi###make AND bentley###make OR 1964###year AND NOT 1999###year | 355 | 488 ms | 53 ms |

**Figura 7. Conjunto de filtros de consulta submetidos ao índice de contexto**

Uma importante contribuição deste trabalho, no contexto de consultas sobre formulários Web, foram os menores tempos de busca para as consultas sobre os índices de contexto e metadados, se comparados com os tradicionais índices por palavras-chave. Este melhor desempenho está associado ao fato de que os dados de elementos passam, durante a construção destes índices, por um processo que gera automaticamente filtros por contexto e por metadado que ficam armazenados diretamente nestes índices. Esses tipos de estruturas são muito adequadas para consultas posteriores sobre formulários Web visto que garantem um mapeamento direto do filtro para uma entrada nestes índices. O uso do índice de sinônimo, visando garantir buscas por similaridade, contribuiu para uma redução no número de entradas nesses dois índices, diminuindo, consequentemente, o tamanho das estruturas e o *overhead* de acesso.

## 5. Conclusão

Este trabalho apresenta e valida uma estratégia de indexação visando buscas por similaridade para dados de formulários Web no contexto do projeto WF-Sim. Esta estratégia introduz mecanismos de refinamento para o contexto de formulários web. Esses mecanismos consideram a indexação de informações sobre elementos de formulários em estruturas de índice especificadas por contexto, metadado, além da tradicional busca por palavra-chave. As duas primeiras estruturas garantem otimizações para o módulo de busca uma vez que as estruturas já indexam os filtros de consultas mais frequentes para formulários. Buscas tradicionais considerando apenas palavras-chave teriam que, no caso de uma busca por contexto, por exemplo, recuperar informações primeiro sobre o rótulo, depois sobre o valor desejado, e após computar uma intersecção dos formulários recuperados. Este *overhead* é desnecessário com a introdução do índice de contexto, e o mesmo vale para o índice de metadado.

O trabalho de [Mello et. al. 2010] foi a base escolhida para a construção das estruturas de índice aqui apresentadas. [Mello et. al. 2010] define índices de contexto e de metadado. Entretanto, o escopo do trabalho não é específico para o contexto de buscas por similaridade em formulários web, que é o objetivo do projeto WF-Sim. Este artigo aplica e estende essas ideias para o propósito do projeto. Nenhum outro trabalho relacionado na literatura se propõe a definir estruturas de índice para buscas por similaridade sobre formulários web.

O próximo passo no contexto deste trabalho é avaliar o desempenho do módulo de indexação com um repositório maior de formulários web. A amostra de testes do projeto conta com um número aproximado de 1090 formulários. Essa base é composta de dados de serviços de origem pública e coletados para a utilização nesse projeto, não se encontrando disponível ao público e futuramente acessível através do funcionamento do sistema sendo desenvolvido. Entretanto, através de uma parceria do GBD/UFSC com a Universidade de Utah, uma amostra de aproximadamente 40000 formulários está sendo disponibilizada.

Sendo que natureza dos dados coletados nessas duas bases é de informação na língua inglesa, o WordNet cumpre seu papel de maneira satisfatória. Entretanto levando em consideração futuras adições de dados na língua portuguesa será necessário buscar uma ferramenta similar para expansão deste processador de consultas.

Outra atividade futura é considerar consultas que testam dependências entre elementos de formulários Web, como por exemplo, um campo "Marca" cujos valores determinam os valores de um campo "Modelo", supondo um domínio de veículos. A intenção é considerar filtros que testem a existência de tais dependências e definir estruturas de indexação que facilitem buscas por similaridade neste contexto. Percebe-se que este tipo de consulta é relevante para casos em que o usuário deseja acessar formulários que implementam automaticamente uma cadeia de dependências entre campos, facilitando, assim, a sua intenção de busca por alguma informação.

## Referências


Elmasri, R.; Shamkant B. (2011). "Sistemas de Banco de Dados"; tradução Daniel Vieira, revisão técnica Enzo Seraphim e Thatyana de Faria Piola Seraphim; 6ª ed. São Paulo: Pearson Addison Wesley, 2011.

Hatcher, E.; Gospodnetic, O. (2005). "Lucene in Action". Greenwich: Manning Publications Co, 2005.

Gonçalves, R. et. al. (2011). "A Similarity Search Approach for *Web forms*". In: Proceedings of the IADIS International Conference IADIS WWW/Internet.

Madhavan, J. et al. (2009) "Harnessing the Deep Web: Present and Future". In: 4th Biennial Conference on Innovative Data Systems Research (CIDR 2009).

Silva, F. R.; Mello, R. S. (2012). "Estratégias de Persistência de Clusters em uma Técnica de Casamento por Similaridade para Web Forms". In: VIII Escola Regional de Banco de Dados (ERBD 2012).

Mello, R. S., Pinnamaneni, R., Freire, J., (2010) Indexing Web Form Constraints., In: Journal of Information and Data Management (JIDM), Vol. 5, nº. 3, p.348-358.

Baeza-Yates, R.; Ribeiro-Neto, R. Modern Information Retrieval. (1999). ACM Press / Addison-Wesley.

# Anexo 2 - Código Fonte

```
package net.wfsim.indexer;

public class ConfiguracaoConsultas {

	private static ConfiguracaoConsultas config;

	private static String INDICE = "1";
	private int nroCluster = 1;
	private boolean stemm = true;
	private boolean cache = false;
	private static String tipoDaPersistencia = "BD";
	private int nroIndice = 2;

	public ConfiguracaoConsultas(){

	}

	public static ConfiguracaoConsultas getConfiguracao(){
		if(config == null){
			config = new ConfiguracaoConsultas();
		}
		return config;
	}

	public String logIndice(){
		return INDICE;
	}

	public int nroIndice(){
		return nroIndice;
	}

	public int nroCluster(){
		return nroCluster;
	}

	public void setNroCluster(int value){
		this.nroCluster = value;
	}

	public boolean isStemm(){
		return stemm;
	}

	public boolean withCache(){
		return cache;
	}

	public String tipoDePersistencia(){
		return tipoDaPersistencia;
	}

	public  String getDiretoriosKeyword(){
		if(nroCluster == 1){
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_1000/IndiceDeKeyword";
		}else{
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_2000/IndiceDeKeyword";
```

```
		}
	}

	public  String getDiretorioContexto(){
		if(nroCluster == 1){
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_1000/IndiceDeContexto";
		}else{
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_2000/IndiceDeContexto";
		}
	}

	public  String getDiretorioMetadado(){
		if(nroCluster == 1){
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_1000/IndiceDeMetadata";
		}else{
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_2000/IndiceDeMetadata";
		}
	}

	public  String getDiretorioDicionario(){
		if(nroCluster == 1){
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_1000/IndiceDicionarioSinonimosS
topWords";
		}else{
			return
"C:/Users/Jose/Desktop/Escalabilidade/Cluster_2000/IndiceDicionarioSinonimosS
topWords";
		}
	}
}
```

```
package net.wfsim.indexer;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;

import net.wfsim.structures.form.FormElement;

import org.apache.lucene.analysis.TokenStream;
import org.apache.lucene.index.CorruptIndexException;

public class IndexContruction {

	private static Indexer indiceKeyword;
	private static Indexer indiceContext;
	private static Indexer indiceMetadata;
	private static Indexer indiceDicionarioSinonimos;
	private static HashMap<String,List<String>> centroids = new
HashMap<String,List<String>>();

	public static List<Integer> loadCentroidNamesFromFileKey(){
		String[] integers;
		List<Integer> dir = new ArrayList<Integer>();
		String caminho =
"C:/Users/Jose/workspace/separetedClusters/maps/radius_1.5/precision_0.01/lab
elWeight_0.5_valueWeight_0.5/6157/";
		int i = 0;
		integers = null;
		File file = new File(caminho);
		integers = file.list();
		for(i= 0;i < integers.length; i++){
			if(!integers[i].equals("keys.ser"))
				dir.add(Integer.parseInt(integers[i]));
		}
		Collections.sort(dir);
		return dir;
	}

	public static List<String> loadCentroidsNamesFromFiles(){
		String[] centroidsDuplicados;
		List<String> centroids = new ArrayList<String>();
		String caminho =
"C:/Users/Jose/workspace/separetedClusters/maps/radius_1.5/precision_0.01/lab
elWeight_0.5_valueWeight_0.5/6157/centroids/";
		int i = 0;
		File file = new File(caminho);
		centroidsDuplicados = file.list();
		for(i = 0;i < centroidsDuplicados.length;i++)
			if(!centroidsDuplicados[i].contains("(1)") &&
!centroidsDuplicados[i].contains("(2)") &&
!centroidsDuplicados[i].contains("(3)")){
```

```
                        centroids.add(centroidsDuplicados[i]);
                }

                return centroids;
        }

        public static List<FormElement> deserializeFile(String folder) throws
IOException, ClassNotFoundException{
                FileInputStream fis = new FileInputStream(folder);
                ObjectInputStream ois = new ObjectInputStream(fis);
                @SuppressWarnings("unchecked")
                List<FormElement> saida = (List<FormElement>) ois.readObject();
                ois.close();
                return saida;
        }

        public static FormElement deserializeFileCentroid(String folder)
throws IOException, ClassNotFoundException{
                FileInputStream fis = new FileInputStream(folder);
                ObjectInputStream ois = new ObjectInputStream(fis);
                FormElement saida = (FormElement) ois.readObject();
                ois.close();
                return saida;
        }

        public static void createIndexesStructures(String
diretorioIndiceDeKeywordSerial, String diretorioIndiceDeContextoSerial,
                        String diretorioIndiceDeMetadataSerial, String
diretorioIndiceDicionarioDeSinonimosSerial){
                indiceKeyword = new Indexer(diretorioIndiceDeKeywordSerial);
                indiceContext = new Indexer(diretorioIndiceDeContextoSerial);
                indiceMetadata = new Indexer(diretorioIndiceDeMetadataSerial);
                indiceDicionarioSinonimos = new
Indexer(diretorioIndiceDicionarioDeSinonimosSerial);
        }

        public static void cleanIndexes(){
                indiceKeyword.cleanIndex();
                indiceContext.cleanIndex();
                indiceMetadata.cleanIndex();
                indiceDicionarioSinonimos.cleanIndex();
        }


        public static void optimizeAndCloseIndexes() throws
CorruptIndexException, IOException{
                indiceKeyword.optimize();
                indiceContext.optimize();
                indiceMetadata.optimize();
                indiceDicionarioSinonimos.optimize();
                indiceKeyword.close();
                indiceContext.close();
                indiceMetadata.close();
                indiceDicionarioSinonimos.close();
        }

        //Cria um arquivo com o registro de todos os valores encontrados em
formularios, ligado ao  numero do centroid.
        public static void escreveArquivoDeControle() throws IOException{
```

```
            String texto;
            FileOutputStream out = new
FileOutputStream("C:/Users/Jose/workspace/texto.txt");
            for(String aux: centroids.keySet()){
                  texto = aux + " == " + centroids.get(aux).size() + "\n" +
centroids.get(aux).toString()+"\n";
                  out.write(texto.getBytes());
                  out.flush();
            }
            out.close();

      }

      public static void verificarValoresDosRotulos(List<String> sinonimos)
throws ClassNotFoundException, IOException{
            FileOutputStream out = new
FileOutputStream("C:/Users/Jose/workspace/rotulos.txt");
            String texto;
            for(String nome: sinonimos){
                  texto = nome + "\n";
                  out.write(texto.getBytes());
                  out.flush();
            }
            out.close();
      }

      //Faz a verificacao apenas do rotulo do centroid e numero do centroid
      public static void verificarValoresDeCentroid() throws
ClassNotFoundException, IOException{
            String diretorioDaPastaDosCentroids =
"C:/Users/Jose/workspace/separetedClusters/maps/radius_1.5/precision_0.01/lab
elWeight_0.5_valueWeight_0.5/6157/centroids/";
            List<String> nomesCentroids =
IndexContruction.loadCentroidsNamesFromFiles();
            FileOutputStream out = new
FileOutputStream("C:/Users/Jose/workspace/centroids.txt");
            String texto;
            for(String nome: nomesCentroids){
                  List<String> filter = new ArrayList<String>();
                  FormElement main =
IndexContruction.deserializeFileCentroid(diretorioDaPastaDosCentroids +
nome);
                  texto = main.getId() + " == " + nome + " == ";
                  for(String value: main.getValues()){
                        if(!filter.contains(value)){
                              filter.add(value);
                              texto = texto + " " +value;
                        }
                  }
                  texto = texto +"\n";
                  out.write(texto.getBytes());
                  out.flush();
            }
            out.close();
      }

      public static List<String> rotulos(String rotulo) throws IOException{
            WFSimAnalyzer analisador = new WFSimAnalyzer();
```

```
            TokenStream result = analisador.tokenStream("keyword", new
StringReader(rotulo));
            List<String> token = Indexer.listTokens(result);
            analisador.close();
            return token;
      }


      public static void create() throws IOException,
ClassNotFoundException{
            WFSimDictionary dicionario = new WFSimDictionary();
            WFSimAnalyzer analisador = new WFSimAnalyzer();
            String diretorioDasPastasDosClusters =
"C:/Users/Jose/workspace/separetedClusters/maps/radius_1.5/precision_0.01/lab
elWeight_0.5_valueWeight_0.5/6157/";
            String diretorioDaPastaDosCentroids =
"C:/Users/Jose/workspace/separetedClusters/maps/radius_1.5/precision_0.01/lab
elWeight_0.5_valueWeight_0.5/6157/centroids/";
            List<String> nomesCentroids =
IndexContruction.loadCentroidsNamesFromFiles();
            List<String> filter;

            HashMap<String, List<String>> mapa = new HashMap<String,
List<String>>();

            String dirName = null;
            HashSet<String> aux = new HashSet<String>();
            try {
                  for(String nome: nomesCentroids){
                        filter = new ArrayList<String>();
                        FormElement main =
IndexContruction.deserializeFileCentroid(diretorioDaPastaDosCentroids +
nome);
                        aux.addAll(rotulos(main.getLabel()));
                  }
                  System.out.println("extração de tokens concluida");
                  for(String label: aux){
                        mapa.put(label, dicionario.getAllSynonym(label));
                  }
                  System.out.println("criacao hashMap concluida");
                  for(String nome: nomesCentroids){
                        filter = new ArrayList<String>();
                        FormElement main =
IndexContruction.deserializeFileCentroid(diretorioDaPastaDosCentroids +
nome);
                        dirName = diretorioDasPastasDosClusters +
main.getId();
                        List<FormElement> cluster =
IndexContruction.deserializeFile(dirName+"/clusterElements.ser");
                        for(FormElement element: cluster){
                              for(String value: element.getValues()){
                                    if(!filter.contains(value)){
                                          filter.add(value);
                                    }
                              }
                        }

                        indiceKeyword.indexKeywordLabel(main.getLabel(),
dirName, Integer.toString(main.getId()), main.getUrl());
```

```
                        indiceMetadata.indexMetadataLabel(main.getLabel(),
dirName, Integer.toString(main.getId()), main.getUrl());

                        for(String v: filter){
                              indiceContext.indexContext(main.getLabel(),
v, dirName, Integer.toString(main.getId()), main.getUrl());

      indiceMetadata.indexMetadata(main.getLabel(),v,dirName,
Integer.toString(main.getId()), main.getUrl());

      indiceKeyword.indexKeywordValue(main.getLabel(),v,dirName,
Integer.toString(main.getId()), main.getUrl());

                        }


                  }
                  System.out.println("criacao 3 indices concluida
concluida");
                  for(String value: mapa.keySet()){
                        for(String syn: mapa.get(value)){

      indiceDicionarioSinonimos.indexSinonimo(analisador.stemTerms(syn),anal
isador.stemTerms(value));
                        }
                  }
                  System.out.println("criacao do indice de sinonimos
concluida ");


            } catch (FileNotFoundException e) {
                  System.err.print("Cluster: '" + dirName + "' nao
encontrado\nO cluster sera calculado\n");
            } catch (IOException e) {
                  e.printStackTrace();
            } catch (ClassNotFoundException e) {
                  e.printStackTrace();
            }
      }

      public static HashMap<FormElement,List<FormElement>>
carregarHashMapDoCluster() throws ClassNotFoundException, IOException{
            //FileInputStream fis = new
FileInputStream("C:/Users/Jose/Desktop/Clusters/cluster_with_3383_elements.se
r");
            FileInputStream fis = new
FileInputStream("C:/Users/Jose/Desktop/Clusters/cluster_with_6976_elements.se
r");
            ObjectInputStream ois = new ObjectInputStream(fis);
            @SuppressWarnings("unchecked")
            HashMap<FormElement,List<FormElement>> cluster =
(HashMap<FormElement,List<FormElement>>) ois.readObject();
            ois.close();
            System.out.println();
            return cluster;
      }

      public static void createIndexFromUtahBase() throws IOException,
ClassNotFoundException{
```

```
            WFSimDictionary dicionario = new WFSimDictionary();
            WFSimAnalyzer analisador = new WFSimAnalyzer();
            String diretorioDasPastasDosClusters =
"C:/Users/Jose/Desktop/Clusters/separetedClusters/maps/radius_1.5/precision_0
.01/labelWeight_0.5_valueWeight_0.5/2/";
            HashMap<FormElement, List<FormElement>> cluster =
carregarHashMapDoCluster();
            List<String> filter;

            HashMap<String, List<String>> mapa = new HashMap<String,
List<String>>();

            String dirName = null;
            HashSet<String> aux = new HashSet<String>();
            try {
                  for(FormElement centroid: cluster.keySet()){
                        filter = new ArrayList<String>();
                        aux.addAll(rotulos(centroid.getLabel()));
                  }
                  System.out.println("extração de tokens concluida");
                  for(String label: aux){
                        mapa.put(label, dicionario.getAllSynonym(label));
                  }
                  System.out.println("criacao hashMap concluida");

                  for(FormElement main: cluster.keySet()){
                        dirName = diretorioDasPastasDosClusters +
main.getId();
                        filter = new ArrayList<String>();
                        List<FormElement> clusterS =
IndexContruction.deserializeFile(dirName+"/clusterElements.ser");
                        for(FormElement element: clusterS){
                              for(String value: element.getValues()){
                                    if(!filter.contains(value)){
                                          filter.add(value);
                                    }
                              }
                        }

                        indiceKeyword.indexKeywordLabel(main.getLabel(),
dirName, Integer.toString(main.getId()), main.getUrl());
                        indiceMetadata.indexMetadataLabel(main.getLabel(),
dirName, Integer.toString(main.getId()), main.getUrl());

                        for(String v: filter){
                              indiceContext.indexContext(main.getLabel(),
v, dirName, Integer.toString(main.getId()), main.getUrl());

      indiceMetadata.indexMetadata(main.getLabel(),v,dirName,
Integer.toString(main.getId()), main.getUrl());

      indiceKeyword.indexKeywordValue(main.getLabel(),v,dirName,
Integer.toString(main.getId()), main.getUrl());
                        }
                  }
                  System.out.println("criacao 3 indices concluida
concluida");

                  for(String value: mapa.keySet()){
```

```
                        for(String syn: mapa.get(value)){

      indiceDicionarioSinonimos.indexSinonimo(analisador.stemTerms(syn),anal
isador.stemTerms(value));
                        }
                  }
                  System.out.println("criacao do indice de sinonimos
concluida ");

            } catch (FileNotFoundException e) {
                  System.err.print("Cluster: '" + dirName + "' nao
encontrado\nO cluster sera calculado\n");
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
}
```

```
package net.wfsim.indexer;
import java.io.File;
import java.io.IOException;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;

import org.apache.lucene.analysis.TokenStream;
import org.apache.lucene.analysis.tokenattributes.CharTermAttribute;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.index.LogMergePolicy;
import org.apache.lucene.index.SegmentInfo;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.store.RAMDirectory;
import org.apache.lucene.util.Version;

public class Indexer{
	private IndexWriter writer;
	private WFSimAnalyzer  analyzer;
	private FSDirectory diretorio;
	private IndexWriterConfig configuration;
	private RAMDirectory dir;
	private String cache = "C:/Users/Jose/Desktop/C_Stem/Cache";

	public Indexer(String path){
		try {
			analyzer =  new WFSimAnalyzer();
			diretorio = FSDirectory.open(new File(path));
			configuration = new IndexWriterConfig(Version.LUCENE_33,
analyzer);
			writer = new IndexWriter(diretorio,configuration);

			LogMergePolicy log = new LogMergePolicy() {

				@Override
				protected long size(SegmentInfo arg0) throws
IOException {
					return 0;
				}
			};
			log.setMergeFactor(50);
			System.out.println("numero de docs do indice = " +
writer.numDocs() );
			System.out.println("Indexer -- Constructor -- SUCCESS\n");
		} catch (IOException e) {

			e.printStackTrace();
		}
	}

	public Indexer(){
		try {
			analyzer =  new WFSimAnalyzer();
			diretorio = FSDirectory.open(new File(cache));
```

```
                    dir = new RAMDirectory(diretorio);
                    configuration = new IndexWriterConfig(Version.LUCENE_33,
analyzer);
                    writer = new IndexWriter(dir,configuration);

                    LogMergePolicy log = new LogMergePolicy() {

                          @Override
                          protected long size(SegmentInfo arg0) throws
IOException {
                                return 0;
                          }
                    };
                    log.setMergeFactor(50);
                    System.out.println("numero de docs do indice = " +
writer.numDocs() );
                    System.out.println("Indexer -- Constructor -- SUCCESS\n");
             } catch (IOException e) {

                    e.printStackTrace();
             }
      }

      public Directory getDiretorio(){
             return dir;
      }


      public void commit() throws CorruptIndexException, IOException {
             this.writer.commit();
      }

      @SuppressWarnings("deprecation")
      public void close(){
             try {
                    System.out.println("numero de docs do indice = " +
writer.numDocs() );
                    System.out.print("closing index ...");
                    this.writer.optimize();
                    this.writer.close();
                    System.out.println(" complete!");
             } catch (CorruptIndexException e) {
                    e.printStackTrace();
             } catch (IOException e) {
                    e.printStackTrace();
             }
      }

      public List<String> indexKeywordLabel(String label, String folder,
String form_id, String url){

             TokenStream result = analyzer.tokenStream("keyword", new
StringReader(label));
             List<String> token = listTokens(result);
             //COM STEMMING
             List<String> stem = analyzer.stemTerms(token);

             for(String aux: stem){
```

```
                    Document doc = new Document();

                    Field keyword_field = new Field("keyword",
aux.trim().toLowerCase(),Field.Store.YES, Field.Index.NOT_ANALYZED);
                    Field form_folder = new Field("folder", folder,
Field.Store.YES, Field.Index.NO);
                    Field form_id_field = new Field("id", form_id,
Field.Store.YES, Field.Index.NO);
                    Field url_field = new Field("url", url, Field.Store.YES,
Field.Index.NO);

                    doc.add(keyword_field);
                    doc.add(form_folder);
                    doc.add(form_id_field);
                    doc.add(url_field);

                    try {
                          this.writer.addDocument(doc);
                    } catch (CorruptIndexException e) {
                          e.printStackTrace();
                    } catch (IOException e) {
                          e.printStackTrace();
                    }
              }
              return token;
        }


        public void indexKeywordValue(String label, String keyword, String
folder, String form_id, String url) throws IOException{

                    Document doc = new Document();

                    Field keyword_field = new Field("keyword",
keyword.trim().toLowerCase(), Field.Store.YES, Field.Index.NOT_ANALYZED);
                    Field form_folder = new Field("folder", folder,
Field.Store.YES, Field.Index.NO);
                    Field form_id_field = new Field("id", form_id,
Field.Store.YES, Field.Index.NO);
                    Field url_field = new Field("url", url, Field.Store.YES,
Field.Index.NO);

                    doc.add(keyword_field);
                    doc.add(form_folder);
                    doc.add(form_id_field);
                    doc.add(url_field);

                    try {
                          this.writer.addDocument(doc);
                    } catch (CorruptIndexException e) {
                          e.printStackTrace();
                    } catch (IOException e) {
                          e.printStackTrace();
                    }

        }
```

```
	public void indexContext(String label, String value, String folder,
String form_id, String url){

		TokenStream result = analyzer.tokenStream("keyword", new
StringReader(label));
		List<String> token = listTokens(result);
		//COM STEMMING
		token = analyzer.stemTerms(token);

		for(String lab: token){
			Document doc = new Document();
			Field labelval = new Field("context",
value.trim().toLowerCase() + "###" + lab.trim().toLowerCase(),
Field.Store.YES, Field.Index.NOT_ANALYZED);
			Field form_folder = new Field("folder", folder,
Field.Store.YES, Field.Index.NO);
			Field form_id_field = new Field("id", form_id,
Field.Store.YES, Field.Index.NO);
			Field url_field = new Field("url", url, Field.Store.YES,
Field.Index.NO);

			doc.add(labelval);
			doc.add(form_folder);
			doc.add(form_id_field);
			doc.add(url_field);

			try {
				this.writer.addDocument(doc);
			} catch (CorruptIndexException e) {
				e.printStackTrace();
			} catch (IOException e) {
				e.printStackTrace();
			}
		}

	}



	public List<String> indexMetadataLabel(String label, String folder,
String form_id, String url){

		TokenStream result = analyzer.tokenStream("keyword", new
StringReader(label));
		List<String> token = listTokens(result);
		//COM STEMMING
		List<String> stem = analyzer.stemTerms(token);

		for(String aux: stem){

			Document docAux = new Document();

			Field metadata_field = new Field("metadata",
aux.trim().toLowerCase()  + "###" + "label", Field.Store.YES,
Field.Index.NOT_ANALYZED);
			Field form_folder = new Field("folder", folder,
Field.Store.YES, Field.Index.NO);
			Field form_id_field = new Field("id", form_id,
Field.Store.YES, Field.Index.NO);
```

```
                    Field url_field = new Field("url", url, Field.Store.YES,
Field.Index.NO);

                    docAux.add(metadata_field);
                    docAux.add(form_folder);
                    docAux.add(form_id_field);
                    docAux.add(url_field);

                    try {
                          this.writer.addDocument(docAux);
                    } catch (CorruptIndexException e) {
                          e.printStackTrace();
                    } catch (IOException e) {
                          e.printStackTrace();
                    }
              }

              return token;
        }

        public void indexMetadata(String label, String value, String folder,
String form_id, String url){
              Document doc = new Document();

              Field metadata_field = new Field("metadata",
value.trim().toLowerCase()  + "###" + "value", Field.Store.YES,
Field.Index.NOT_ANALYZED);
              Field form_folder = new Field("folder", folder, Field.Store.YES,
Field.Index.NO);
              Field form_id_field = new Field("id", form_id, Field.Store.YES,
Field.Index.NO);
              Field url_field = new Field("url", url, Field.Store.YES,
Field.Index.NO);

              doc.add(metadata_field);
              doc.add(form_folder);
              doc.add(form_id_field);
              doc.add(url_field);

              try {
                    this.writer.addDocument(doc);
              } catch (CorruptIndexException e) {
                    e.printStackTrace();
              } catch (IOException e) {
                    e.printStackTrace();
              }

        }

        public List<String> getTokens(String label){
              TokenStream result = analyzer.tokenStream("keyword", new
StringReader(label));
              List<String> token = listTokens(result);
              return token;
        }

        public static List<String> listTokens(TokenStream arg) {
              List<String> token = new ArrayList<String>();
              try {
```

```
                  while(arg.incrementToken()){
                        token.add(new
String(arg.getAttribute(CharTermAttribute.class).toString()));
                  }
            } catch (IOException e) {
                  e.printStackTrace();
            }

            return token;
      }

      @SuppressWarnings("deprecation")
      public void optimize(){
            try {
                  this.writer.optimize();
            } catch (CorruptIndexException e) {
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

      //Limpeza total do indice, remo◆ao de todos os documents
      public void cleanIndex(){
            try {
                  if(writer != null){
                        System.out.println("INDEX CLEAN");
                        writer.deleteAll();
                        System.out.println("numero de docs do indice = " +
writer.numDocs() );
                  }
            } catch (IOException e) {
                  e.printStackTrace();
            }

      }

      public void indexSinonimo(String synonym, String elementCentroid) {
            List<String> token = null;
            //Remoção StopWords
            TokenStream result = analyzer.tokenStream("keyword", new
StringReader(synonym));
            token = listTokens(result);
            //Processo Stemming
            token = analyzer.stemTerms(token);

            for(String sin: token){
                  Document doc = new Document();

                  Field keyword_field = new Field("sinonimo",
sin.trim().toLowerCase(), Field.Store.YES,Field.Index.NOT_ANALYZED);
                  Field synonym_folder = new Field("centroid",
elementCentroid.trim().toLowerCase(),Field.Store.YES, Field.Index.NO);

                  doc.add(keyword_field);
                  doc.add(synonym_folder);

                  try {
                        this.writer.addDocument(doc);
```

```
                } catch (CorruptIndexException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }


    }



    public void indexSinonimos(String saida, List<String> pos) {
        int i = 0;
        while( i < 5 && i < pos.size()){

                Document doc = new Document();

                Field keyword_field = new Field("keyword",
saida.trim().toLowerCase(),Field.Store.YES,Field.Index.NOT_ANALYZED);
                Field synonym_folder = new Field("synonym",
pos.get(i).toLowerCase(), Field.Store.YES, Field.Index.NO);

                doc.add(keyword_field);
                doc.add(synonym_folder);

                try {
                        this.writer.addDocument(doc);
                } catch (CorruptIndexException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                i++;
        }
    }

    public void indexURLCache(String entrada, List<String>
conjuntoDeURLSdoCluster){

        for(String url: conjuntoDeURLSdoCluster){

                Document doc = new Document();

                Field keyword_field = new Field("key", entrada,
Field.Store.YES,Field.Index.NOT_ANALYZED);
                Field synonym_folder = new Field("url",
url,Field.Store.YES, Field.Index.NO);

                doc.add(keyword_field);
                doc.add(synonym_folder);
                try {
                        this.writer.addDocument(doc);

                } catch (CorruptIndexException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
```

```
                }
            }
        }
```

```java
package net.wfsim.indexer;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import net.wfsim.structures.form.FormElement;

public class IndexerOfDataBase {

	private static final String dbClassName = "com.mysql.jdbc.Driver";
	//private static final String CONNECTION =
"jdbc:mysql://localhost/wfsim_clusters";
	private static final String CONNECTION =
"jdbc:mysql://localhost/clusters_tcc";
	private Connection c;

	public IndexerOfDataBase() throws ClassNotFoundException,
SQLException{
		try {
			Class.forName(dbClassName).newInstance();
			Properties p = new Properties();
			p.put("user","root");
			p.put("password","09.bill.02.89");
			this.c = DriverManager.getConnection(CONNECTION,p);
		} catch (InstantiationException e) {
			e.printStackTrace();
		} catch (IllegalAccessException e) {
			e.printStackTrace();
		}
	}

	public void closeConection() throws SQLException{
		this.c.close();
	}

	public void cleanCache(){
		PreparedStatement busca;
		try {
			busca = c.prepareStatement("RESET QUERY CACHE");
			busca.clearParameters();
			busca = c.prepareStatement("FLUSH QUERY CACHE");
			busca.clearParameters();
			busca = c.prepareStatement("FLUSH STATUS");
			busca.clearParameters();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public List<FormElement> getCentroids() throws SQLException{
		ResultSet result = null;
		PreparedStatement busca = c.prepareStatement("select " +
```

```
                        "F.id as fid, F.url as furl, F.label as flabel from
cluster C join cluster_to_element CE on C.centroid = CE.cluster join
form_element F on F.id = CE.element");
            busca.clearParameters();
            result = busca.executeQuery();

            List<FormElement> centroids = new ArrayList<FormElement>();

            while(result.next()){
                  FormElement centroid = new FormElement();
                  centroid.setId(Integer.parseInt(result.getString("fid")));
                  centroid.setLabel(result.getString("flabel"));
                  centroid.setUrl(result.getString("furl"));

                  ResultSet values = null;
                  PreparedStatement buscaDosValores =
c.prepareStatement("select " +
                              "V.value as fvalue from form_element F join
value V on V.element = "+result.getString("fid"));
                  buscaDosValores.clearParameters();
                  values = buscaDosValores.executeQuery();
                  List<String> conjuntoDeValores = new ArrayList<String>();
                  while(values.next()){

      if(!conjuntoDeValores.contains(values.getString("fvalue")))

      conjuntoDeValores.add(values.getString("fvalue"));
                  }
                  values.close();
                  centroid.setValues(conjuntoDeValores);
                  centroids.add(centroid);
            }
            result.close();
            return centroids;
      }

      public List<FormElement> getElementosCentroid(String id) throws
SQLException {
            ResultSet result = null;
            PreparedStatement busca = c.prepareStatement("select " +
                        "F.id as fid, F.url as furl, F.label as flabel from
cluster C join cluster_to_element CE on C.centroid = CE.cluster join
form_element F on F.id = CE.element");
            busca.clearParameters();
            result = busca.executeQuery();
            List<FormElement> centroids = new ArrayList<FormElement>();

            while(result.next()){
                  FormElement centroid = new FormElement();
                  centroid.setId(Integer.parseInt(result.getString("fid")));
                  centroid.setLabel(result.getString("flabel"));
                  centroid.setUrl(result.getString("furl"));

                  ResultSet values = null;
                  PreparedStatement buscaDosValores =
c.prepareStatement("select " +
                              "V.value as fvalue from form_element F join
value V on V.element = "+Integer.parseInt(result.getString("fid")));
                  buscaDosValores.clearParameters();
```

```
                        values = buscaDosValores.executeQuery();
                        List<String> conjuntoDeValores = new ArrayList<String>();
                        while(values.next()){

      /*if(!conjuntoDeValores.contains(values.getString("fvalue")));*/
                              conjuntoDeValores.add(values.getString("fvalue"));
                        }
                        centroid.setValues(conjuntoDeValores);
                        centroids.add(centroid);
                  }
                  return centroids;
      }

      public List<String> getElementosCentroid(List<String> id) throws
SQLException {
                  List<String> urls = new ArrayList<String>();
                  ResultSet result = null;
                  String auxiliar = "";
                  for(int i=0;i<id.size();i++){
                        if(i < id.size()-1){
                              auxiliar = auxiliar + "CE.cluster = " + id.get(i) +
" OR ";
                        }else{
                              auxiliar = auxiliar + " CE.cluster = " + id.get(i);
                        }
                  }
                  PreparedStatement busca = c.prepareStatement("select " +
                              "F.id as fid, F.url as furl, F.label as flabel from
cluster C join (cluster_to_element CE join form_element F on F.id =
CE.element) on C.centroid = CE.cluster where " + auxiliar);
                  busca.clearParameters();
                  result = busca.executeQuery();
                  while(result.next()){
                        urls.add(result.getString("furl"));
                  }
                  return urls;
      }

      public List<String> getElementosCentroid(int nroCluster, List<String>
id) throws SQLException {
                  List<String> urls = new ArrayList<String>();
                  ResultSet result = null;
                  String auxiliar = "";
                  for(int i=0;i<id.size();i++){
                        if(i < id.size()-1){
                              auxiliar = auxiliar + "CE.cluster = " + id.get(i) +
" OR ";
                        }else{
                              auxiliar = auxiliar + " CE.cluster = " + id.get(i);
                        }
                  }
                  PreparedStatement busca = c.prepareStatement("select " +
                              "F.id as fid, F.url as furl, F.label as flabel from
cluster C join (cluster_to_element CE join form_element F on F.id =
CE.element) on C.centroid = CE.cluster where C.clusterization = " +
nroCluster + " AND ("+ auxiliar + ")");
                  busca.clearParameters();
                  result = busca.executeQuery();
                  while(result.next()){
```

```
                urls.add(result.getString("furl"));
            }
            return urls;
        }
}
```

```
package net.wfsim.indexer;

import java.io.IOException;
import java.sql.SQLException;

import net.wfsim.matching.ResultMatching;

import org.apache.lucene.queryParser.ParseException;

public class IndexSearcher {

	private static ResultMatching matching;

	private String diretorioIndiceDeKeywordSerial;
	private String diretorioIndiceDeContextoSerial;
	private String diretorioIndiceDeMetadataSerial;
	private String diretorioIndiceDicionarioDeSinonimosSerial;


	public IndexSearcher(String diretorioIndiceDeKeywordSerial,String
diretorioIndiceDeContextoSerial,
			String diretorioIndiceDeMetadataSerial, String
diretorioIndiceDicionarioDeSinonimosSerial){
		this.diretorioIndiceDeKeywordSerial =
diretorioIndiceDeKeywordSerial;
		this.diretorioIndiceDeContextoSerial =
diretorioIndiceDeContextoSerial;
		this.diretorioIndiceDeMetadataSerial =
diretorioIndiceDeMetadataSerial;
		this.diretorioIndiceDicionarioDeSinonimosSerial =
diretorioIndiceDicionarioDeSinonimosSerial;
	}

	public void closeIndex(){
		try {
			matching.closeIndexStructures();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public void createSearchers() throws IOException{
		matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
	}

	public void runDirtyQueries() throws IOException, ParseException,
SQLException{

		boolean STEMM = true;
		System.out.println();
		System.out.println();

	System.out.println("\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\");
		System.out.println("DIRTY QUERIES");
```

```
      System.out.println("////////////////////////////////////////");

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("language###label", STEMM ) ;


            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("1978 AND 2001###price AND 1995 AND
2001###price AND 2001###price", STEMM ) ;

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("2002###value AND 1934###value AND
1956###price", STEMM ) ;

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("2012###label AND 1983###label AND 1997",
STEMM ) ;

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("2000###year", STEMM ) ;


            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("book", STEMM ) ;


            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("title", STEMM ) ;
```

```
            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("subject", STEMM ) ;

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("author AND search AND series AND all AND
book AND isbn AND word AND volume", STEMM ) ;

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("1976###value AND year AND may AND spanish",
STEMM ) ;

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("1954###label", STEMM ) ;

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("1945###year", STEMM ) ;

            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("2003###year AND logic###subject AND
10###limit AND select AND new AND date AND book", STEMM ) ;

            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            matching.parseQuery("$5,000###trade AND trade AND item AND
#1,375.00###item AND reach AND 1395###reach AND audio###brows AND brows AND
14 AND 14###return AND $50,000###maximun AND maximun AND $50,000 AND category
", STEMM ) ;


      System.out.println("\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\");

      System.out.println("//////////////////////////////////////////");
            System.out.println();
            System.out.println();
```

```
	}

	public double consultasKeyword(boolean STEMM, int op) throws
IOException, ParseException, SQLException{
		System.out.println("================== KEYWORD
==============================");

		matching = null;
		matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
		switch(op){
			case 1: return matching.parseQuery("1999", STEMM );
			case 2: return matching.parseQuery("jeep AND make", STEMM
) ;
			case 3: return matching.parseQuery("jeep AND make AND
corolla AND model", STEMM ) ;
			case 4: return matching.parseQuery("acura AND make AND
1999 AND year", STEMM ) ;
			case 5: return matching.parseQuery("audi AND make AND bmw
AND car AND $12,000 AND price AND red AND color AND 1999 AND year", STEMM ) ;
		}
		return 0.0;
	}

	public double consultasContexto(boolean STEMM,int op) throws
IOException, ParseException, SQLException{

		System.out.println("================== CONTEXT
==============================");

		matching = null;
		matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
		switch(op){
			case 1: return matching.parseQuery("1999###year", STEMM )
;
			case 2: return matching.parseQuery("jeep###make", STEMM )
;
			case 3: return matching.parseQuery("jeep###make AND
corolla###model", STEMM ) ;
			case 4: return matching.parseQuery("acura###make AND
1999###year", STEMM ) ;
			case 5: return matching.parseQuery("audi###make AND
bmw###car AND $12,000###price AND red###color AND 1999###year", STEMM ) ;
		}
		return 0.0;
	}



	public double consultasMetadado(boolean STEMM,int op) throws
IOException, ParseException, SQLException{
```

```
            System.out.println("================== METADADO
============================");

            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            switch(op){
                  case 1: return matching.parseQuery("1999###value", STEMM )
;
                  case 2: return matching.parseQuery("jeep###value AND
make###label", STEMM ) ;
                  case 3: return matching.parseQuery("jeep###value AND
make###label AND corolla###value AND model###label", STEMM );
                  case 4: return matching.parseQuery("acura###value AND
make###label AND 1999###value AND year###label", STEMM );
                  case 5: return matching.parseQuery("audi###value AND
make###label AND bmw###value AND car###label AND $12,000###value AND
price###label AND red###value AND color###label AND 1999###value AND
year###label", STEMM ) ;
            }
            return 0.0;
      }

      public double consultaEquivalentes(boolean STEMM, int tipoIndice, int
consulta) throws IOException, ParseException, SQLException{
            runDirtyQueries();
            switch(tipoIndice){
                  case 1: return consultasKeyword(STEMM, consulta);
                  case 2: return consultasContexto(STEMM, consulta);
                  case 3: return consultasMetadado(STEMM, consulta);
            }
            return 0.0;
      }

      public double consultaEquivalentesEscalabilidade(boolean STEMM, int
tipoIndice, int consulta) throws IOException, ParseException, SQLException{
            switch(tipoIndice){
                  case 1: return consultasKeywordEscalabilidade(STEMM,
consulta);
                  case 2: return consultasContextoEscalabilidade(STEMM,
consulta);
                  case 3: return consultasMetadadoEscalabilidade(STEMM,
consulta);
            }
            return 0.0;
      }

      public double consultasKeywordEscalabilidade(boolean STEMM, int op)
throws IOException, ParseException, SQLException{
            System.out.println("================== KEYWORD
============================");
            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            switch(op){
```

```
                  case 1: return matching.parseQuery("2005", STEMM ) ;
                  case 2: return matching.parseQuery("philosophy AND
subject", STEMM ) ;
                  case 3: return matching.parseQuery("philosophy AND subject
AND german AND language", STEMM ) ;
                  case 4: return matching.parseQuery("history AND category
AND 2005 AND year", STEMM ) ;
            }
            return 0.0;
      }

      public double consultasContextoEscalabilidade(boolean STEMM,int op)
throws IOException, ParseException, SQLException{
            System.out.println("================== CONTEXT
============================");
            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            switch(op){
                  case 1: return matching.parseQuery("2005###year", STEMM )
;
                  case 2: return matching.parseQuery("philosophy###subject",
STEMM ) ;
                  case 3: return matching.parseQuery("philosophy###subject
AND portuguese###language", STEMM ) ;
                  case 4: return matching.parseQuery("history###category AND
2005###year", STEMM ) ;
            }
            return 0.0;
      }

      public double consultasMetadadoEscalabilidade(boolean STEMM,int op)
throws IOException, ParseException, SQLException{
            System.out.println("================== METADADO
============================");
            matching = null;
            matching =
ResultMatching.getResultMatching(diretorioIndiceDeKeywordSerial,
diretorioIndiceDeContextoSerial, diretorioIndiceDeMetadataSerial,
diretorioIndiceDicionarioDeSinonimosSerial);
            switch(op){
                  case 1: return matching.parseQuery("2005###value", STEMM )
;
                  case 2: return matching.parseQuery("philosophy###value AND
subject###label", STEMM ) ;
                  case 3: return matching.parseQuery("philosophy###value AND
subject###label AND portuguese###value AND language###label", STEMM ) ;
                  case 4: return matching.parseQuery("history###value AND
category###label AND 2005###value AND year###label", STEMM ) ;
            }
            return 0.0;
      }
}
```

```java
package net.wfsim.indexer;

import java.io.File;
import java.io.IOException;

import org.apache.lucene.analysis.KeywordAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryParser.ParseException;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;

public class Searcher {
	private IndexSearcher searcher;
	private IndexReader reader;
	private Directory diretorio;

	public Searcher(String path){
		try {
			diretorio = FSDirectory.open(new File(path));
			reader = IndexReader.open(diretorio);
			searcher = new IndexSearcher(reader);
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public Searcher(IndexReader reader){
		searcher = new IndexSearcher(reader);
	}

	public Searcher(Directory dir){
		try {
			reader = IndexReader.open(dir);
		} catch (CorruptIndexException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		searcher = new IndexSearcher(reader);
	}

	public TopDocs searchKeywordLabel(String query) throws IOException,
ParseException{
		KeywordAnalyzer analyzer = new KeywordAnalyzer();
		QueryParser parser = new QueryParser(Version.LUCENE_33,
"keyword", analyzer);
		Query q = parser.parse(query.trim().toLowerCase());
		return searcher.search(q,100);
```

```
	}

	public TopDocs searchKeyword(String query) throws IOException,
ParseException{
		//WFSimAnalyzer analyzer = new WFSimAnalyzer();
		KeywordAnalyzer analyzer = new KeywordAnalyzer();
		QueryParser parser = new QueryParser(Version.LUCENE_33,
"keyword", analyzer);
		Query q = parser.parse(query.trim().toLowerCase());
		return searcher.search(q,100);
	}

	public Document getDocument(ScoreDoc score) throws
CorruptIndexException, IOException{
		return searcher.doc(score.doc);
	}

	public TopDocs searchMetadataValue(String query) throws IOException,
ParseException{
		KeywordAnalyzer analyzer = new KeywordAnalyzer();
		//WFSimAnalyzer analyzer = new WFSimAnalyzer();
		QueryParser parser = new QueryParser(Version.LUCENE_33,
"metadata", analyzer);
		Query q = parser.parse(query.trim().toLowerCase());
		return searcher.search(q,100);
	}

	public TopDocs searchMetadataLabel(String query) throws IOException,
ParseException{
		KeywordAnalyzer analyzer = new KeywordAnalyzer();
		//Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_33);
		QueryParser parser = new QueryParser(Version.LUCENE_33,
"metadata", analyzer);
		Query q = parser.parse(query.trim().toLowerCase());
		return searcher.search(q,100);
	}

	public TopDocs searchMetadata(String query) throws IOException,
ParseException{
		KeywordAnalyzer analyzer = new KeywordAnalyzer();
		//Analyzer analyzer = new KeywordAnalyzer();
		QueryParser parser = new QueryParser(Version.LUCENE_33,
"metadata", analyzer);
		Query q = parser.parse(query.trim().toLowerCase());
		return searcher.search(q,100);
	}

	public TopDocs searchContext(String query) throws IOException,
ParseException{
		KeywordAnalyzer analyzer = new KeywordAnalyzer();
		//Analyzer analyzer = new KeywordAnalyzer();
		QueryParser parser = new
QueryParser(Version.LUCENE_33,"context", analyzer);
		Query q = parser.parse(query.trim().toLowerCase());
		return searcher.search(q,100);
	}

	public Document getDocument(int value) throws CorruptIndexException,
IOException{
```

```
            return searcher.doc(value);

      }

      public TopDocs searchDicionario(String query) throws IOException,
ParseException{
            KeywordAnalyzer analyzer = new KeywordAnalyzer();
            //Analyzer analyzer = new KeywordAnalyzer();
            QueryParser parser = new
QueryParser(Version.LUCENE_33,"sinonimo", analyzer);
            Query q = parser.parse(query.trim().toLowerCase());
            return searcher.search(q,100);
      }

      public TopDocs searchCache(String query) throws IOException,
ParseException{
            KeywordAnalyzer analyzer = new KeywordAnalyzer();
            //Analyzer analyzer = new KeywordAnalyzer();
            QueryParser parser = new QueryParser(Version.LUCENE_33,"key",
analyzer);
            Query q = parser.parse(query.trim().toLowerCase());
            return searcher.search(q,100);
      }

      public void close() throws IOException{
            this.searcher.close();
      }

      public IndexReader getIndexReader(){
            return this.searcher.getIndexReader();
      }

      public Directory getDirectory(){
            return this.diretorio;
      }
}
```

```java
package net.wfsim.indexer;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.LetterTokenizer;
import org.apache.lucene.analysis.LowerCaseFilter;
import org.apache.lucene.analysis.StopFilter;
import org.apache.lucene.analysis.TokenStream;
import org.apache.lucene.util.Version;
import org.tartarus.snowball.ext.PorterStemmer;

public class WFSimAnalyzer extends Analyzer{

	private Set<Object> stopWords;

	public static final String[] ENGLISH_STOP_WORDS_SET  = {
	    "a", "an", "and", "are", "as", "at", "be","choose",
	    "but", "by","for", "if", "in", "into", "is",
	    "it", "do","you","no", "not", "of", "on", "or",
	    "s", "such","t", "that", "the", "their", "then",
	    "there", "these","they", "this", "to", "was",
	    "will", "with"," - ","--","---", ":","...","?",
	    "<",">","","%","===","[","]","&","(",")"};

	public WFSimAnalyzer() throws IOException{
		stopWords = StopFilter.makeStopSet(Version.LUCENE_33,
savedStopWords());
	}

	public WFSimAnalyzer(String[] stopWords){
		this.stopWords =
StopFilter.makeStopSet(Version.LUCENE_33,stopWords);
	}

	public static List<String> savedStopWords() throws IOException{
		String dirName = "C:/Users/Jose/workspace/stop_words.txt";
		List<String> stopWords = new ArrayList<String>();
		BufferedReader br = null;
		try {
			br = new BufferedReader(new FileReader(dirName));
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
		String line="";
		try {
			while((line = br.readLine()) != null){
				line = line.trim();
				stopWords.add(line);
			}

			br.close();
```

```
			} catch (IOException e) {
				e.printStackTrace();
			}
			for(String s: ENGLISH_STOP_WORDS_SET)
				stopWords.add(s);
			return stopWords;
		}



		@Override
		public TokenStream tokenStream(String arg0, Reader arg1) {
			TokenStream result = new LetterTokenizer(Version.LUCENE_33,
arg1);
			result = new LowerCaseFilter(Version.LUCENE_33, result );
			result = new StopFilter(Version.LUCENE_33,result, stopWords);
			return result;
		}

		public List<String> stemTerms(List<String> term) {
		    PorterStemmer stemmer = new PorterStemmer();
		    List<String> out = new ArrayList<String>();
		    for(String pos: term){
		       stemmer.setCurrent(pos);
		       stemmer.stem();
		       char[] stem = new char[stemmer.getCurrentBufferLength()];
		       pos.getChars(0, stemmer.getCurrentBufferLength(), stem, 0);
		       out.add(String.valueOf(stem));
		    }
		    return out;
		}

		public String stemTerms(String term) {
		    PorterStemmer stemmer = new PorterStemmer();
		stemmer.setCurrent(term);
		stemmer.stem();
		char[] stem = new char[stemmer.getCurrentBufferLength()];
		term.getChars(0, stemmer.getCurrentBufferLength(), stem, 0);
		return String.valueOf(stem);
		}

}
```

```java
package net.wfsim.indexer;

import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

import edu.mit.jwi.Dictionary;
import edu.mit.jwi.IDictionary;
import edu.mit.jwi.item.IIndexWord;
import edu.mit.jwi.item.ISynset;
import edu.mit.jwi.item.IWord;
import edu.mit.jwi.item.IWordID;
import edu.mit.jwi.item.POS;

public class WFSimDictionary{

	private IDictionary dict;

	public WFSimDictionary(){
		String wnHome = System.getenv("WNHOME");
		String path = wnHome + File.separator +"dict";
		System.out.println(path);
		try {
			URL url = new URL("file", null, path);
			dict = new Dictionary(url);
			dict.open();
		} catch (MalformedURLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

	public List<String> getSynonyms(String noum){
		List<String> synonyms = new ArrayList<String>();
		IIndexWord idxWord = dict.getIndexWord(noum, POS.NOUN);
		if(idxWord != null){
			for(IWordID aux : idxWord.getWordIDs()){
				//IWordID wordID = idxWord.getWordIDs().get(0); //
1st meaning
				//System.out.println(wordID.toString());
				IWord word = dict.getWord(aux);
				ISynset synset = word.getSynset();
				// iterate over words associated with the synset
				for(IWord w : synset.getWords()) {
					//System.out.println(w.getLemma());
					synonyms.add(w.getLemma());
				}
			}
		}
		return synonyms;


	}
```

```
	public String getSynonym(String noum){
		IIndexWord idxWord = dict.getIndexWord(noum, POS.NOUN);
		if(idxWord != null){
			IWordID wordID = idxWord.getWordIDs().get(0);
			IWord word = dict.getWord(wordID);
			ISynset synset = word.getSynset();
			for(IWord w : synset.getWords()) {
				return w.getLemma();
			}
		}
		return noum;
	}

	public List<String> getAllSynonym(String noum){
		List<String> synonyms = new ArrayList<String>();
		IIndexWord idxWord = dict.getIndexWord(noum, POS.NOUN);
		if(idxWord != null){
			for(int i = 0; i < idxWord.getWordIDs().size(); i++){
				IWordID wordID = idxWord.getWordIDs().get(i);
				IWord word = dict.getWord(wordID);
				ISynset synset = word.getSynset();
				for(IWord w : synset.getWords()) {
					if(!synonyms.contains(w.getLemma()))
						synonyms.add(w.getLemma());
				}
			}
		}
		return synonyms;
	}

	public String getNextSynonym(String noum, int i){
		IIndexWord idxWord = dict.getIndexWord(noum, POS.NOUN);
		if(idxWord != null){
			if( i < idxWord.getWordIDs().size() ){
				IWordID wordID = idxWord.getWordIDs().get(i);
				IWord word = dict.getWord(wordID);
				ISynset synset = word.getSynset();
				for(IWord w : synset.getWords()) {
					return w.getLemma();
				}
			}
		}
		return noum;
	}

}
```

```
package net.wfsim.matching;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.StringReader;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import net.wfsim.indexer.ConfiguracaoConsultas;
import net.wfsim.indexer.Indexer;
import net.wfsim.indexer.IndexerOfDataBase;
import net.wfsim.indexer.IndexContruction;
import net.wfsim.indexer.Searcher;
import net.wfsim.indexer.WFSimAnalyzer;
import net.wfsim.structures.form.FormElement;

import org.apache.lucene.analysis.TokenStream;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.queryParser.ParseException;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;


public class ResultMatching {

	private static FileOutputStream out;

	private static ConfiguracaoConsultas configuracao;
	private Searcher searcherKeyword;
	private Searcher searcherContext;
	private Searcher searcherMetadata;
	private Searcher searcherDicionarioSinonimos;
	private Searcher searcher;
	private IndexerOfDataBase conexao;

	private static ResultMatching resultMatching;

	private Indexer indexCache;
	private Indexer indexMem;

	private WFSimAnalyzer analisador;

	private List<String> terms = new ArrayList<String>();
	private List<String> operands = new ArrayList<String>();
	private List<Integer> positionOfNegativesOperandsWithAnd = new
ArrayList<Integer>();
	private List<Integer> positionOfNegativesOperandsWithOR = new
ArrayList<Integer>();
	private HashMap<String,List<FormElement>> cacheWithLoadedClusters =
new HashMap<String, List<FormElement>>();
	private HashMap<String,List<String>> cacheWithLoadedCentroids = new
HashMap<String, List<String>>();
```

```
	public static ResultMatching  getResultMatching(String
diretorioIndiceDeKeyword, String diretorioIndiceDeContexto, String
diretorioIndiceDeMetadata, String diretorioIndiceDicionarioDeSinonimos){
		if(resultMatching == null)
			resultMatching = new
ResultMatching(diretorioIndiceDeKeyword, diretorioIndiceDeContexto,
diretorioIndiceDeMetadata, diretorioIndiceDicionarioDeSinonimos);
		return resultMatching;
	}

	public ResultMatching  getResultMatching(){
		return resultMatching;
	}

	public void closeIndexStructures() throws IOException{
		if(configuracao.withCache()){
			indexCache.optimize();
			indexCache.close();
			indexMem.close();
		}
	}


	private ResultMatching(String diretorioIndiceDeKeyword, String
diretorioIndiceDeContexto, String diretorioIndiceDeMetadata, String
diretorioIndiceDicionarioDeSinonimos) {
		searcherKeyword = new Searcher(diretorioIndiceDeKeyword);
		searcherContext = new Searcher(diretorioIndiceDeContexto);
		searcherMetadata = new Searcher(diretorioIndiceDeMetadata);
		searcherDicionarioSinonimos = new
Searcher(diretorioIndiceDicionarioDeSinonimos);
		configuracao = ConfiguracaoConsultas.getConfiguracao();
		System.out.println("AQUI");
		try {
			analisador = new WFSimAnalyzer();
		} catch (IOException e) {
			e.printStackTrace();
		}
		if(configuracao.withCache()){
			indexCache = new
Indexer("C:/Users/Jose/Desktop/C_Stem/Cache");
			indexCache.close();
			indexCache = new
Indexer("C:/Users/Jose/Desktop/C_Stem/Cache");
			indexMem = new Indexer();
		}

	}

	private void clearVariables() {
		terms = new ArrayList<String>();
		operands = new ArrayList<String>();
		positionOfNegativesOperandsWithAnd = new ArrayList<Integer>();
	}

	public static void abreArquivo(String index) throws IOException{
		out = new
FileOutputStream("C:/Users/Jose/Desktop/"+configuracao.logIndice()+"/"+index+
".txt");
```

```
}

public static void fechaArquivo(){
	try {
		out.close();
	} catch (IOException e) {
		e.printStackTrace();
	}
}

public static void escreverNoArquivo(String index, List<String> list){
	String texto = "";
	try {
		out.write(("\n"+index+"\n").getBytes());
		out.flush();
	} catch (IOException e) {
		e.printStackTrace();
	}

	for(String aux: list){
		 texto = texto +  " " + aux;
	}
	try {
		out.write(texto.getBytes());
		out.flush();
	} catch (IOException e) {
		e.printStackTrace();
	}
}

public static void escreverNoArquivo(String index, String list){
	String texto = "";
	try {
		out.write(("\n"+index+"\n").getBytes());
		out.flush();
	} catch (IOException e) {
		e.printStackTrace();
	}


		 texto = texto +  " " + list;

	try {
		out.write(texto.getBytes());
		out.flush();
	} catch (IOException e) {
		e.printStackTrace();
	}
}


private List<String> recognitionTypesOfElements(String file){
	List<String> out = new ArrayList<String>();
	String[] tokens = file.toLowerCase().split(" ");
	String[] types = new String[tokens.length];
	for(int i = 0;i < tokens.length; i++){
		String elemento = tokens[i];
		if(elemento.contains("label")){
			types[i] = "metadata:".concat(tokens[i]);
```

```
                    }else if(elemento.contains("value")){
                          types[i] = "metadata:".concat(tokens[i]);
                    }else if(elemento.contains("###")){
                          types[i] = "context:".concat(tokens[i]);
                    }else if(elemento.equals("or") || elemento.equals("and")
|| elemento.equals("not") ){
                          types[i] = elemento.toUpperCase();
                    }else if(elemento.contains("_")){
                          types[i] =
"keyword:\"".concat(elemento.split("_")[0]).concat("
").concat(elemento.split("_")[1]).concat("\"");
                    }else if(elemento.contains(">") ||
elemento.contains("<")){
                          types[i] = types[i-
1].split(":")[1].concat(elemento).concat(tokens[i+1]);
                          types[i-1] = "-1";
                          tokens[i+1] = "-1";
                    }else if(elemento.equals("-1")){
                          types[i] = "-1";
                    }else{
                          types[i] = "keyword:".concat(tokens[i]);
                    }
              }
              for(int j = 0;j < types.length;j++){
                    if(!types[j].equals("-1"))
                          out.add(types[j]);
              }
              return out;
        }

        private boolean validationOfSequence(List<String> tokens) {
              for(int i = 0; i < tokens.size()-1; i++){
                    if(tokens.get(i).toUpperCase().equals("NOT") &&
(tokens.get(i+1).toUpperCase().equals("NOT")
                                || tokens.get(i+1).toUpperCase().equals("OR")
|| tokens.get(i+1).toUpperCase().equals("AND")))
                          return false;
                    if((tokens.get(i).toUpperCase().equals("OR") ||
tokens.get(i).toUpperCase().equals("AND")) &&
                                (tokens.get(i+1).toUpperCase().equals("OR")
|| tokens.get(i+1).toUpperCase().equals("AND")))
                          return false;
              }
              return true;
        }


        public List<String> getSinonimosDoIndice(String noum) throws
IOException, ParseException{
              List<String> synonym = new ArrayList<String>();
              ScoreDoc[] aux =
searcherDicionarioSinonimos.searchDicionario(noum).scoreDocs;
              if(aux.length > 0){

        synonym.addAll(ResultMatchingAuxiliarOperations.extractSynonyms(search
erDicionarioSinonimos, aux));
                    return synonym;
              }
              synonym.add(noum);
```

```
            return synonym;
      }

      public double parseQuery(String file, boolean stem) throws
IOException, ParseException, SQLException {
            long start, end;
            start = System.nanoTime();
            int i;
            List<List<String>> hitsVec = new ArrayList<List<String>>();
            List<String> resultVec = new ArrayList<String>();
            if(configuracao.withCache()){   searcher = new
Searcher(indexMem.getDiretorio()); }
            long a, b;
            a = System.nanoTime();
            abreArquivo(file);
            if(configuracao.tipoDePersistencia().equals("BD") &&
!configuracao.withCache()){
                  try {
                        conexao = new IndexerOfDataBase();
                        conexao.cleanCache();
                  } catch (ClassNotFoundException e) {
                        e.printStackTrace();
                  }
            }
            b = System.nanoTime();

      System.out.println("=========================================");
            System.out.println("Inicializando o Banco: "+ (b-a));

            clearVariables();
            List<String> tokens = recognitionTypesOfElements(file);
            filterOperators(tokens);

            if(validationOfSequence(tokens)){

                  for(i=0;i<operands.size();i++){
                        if(operands.get(i).equals("NOT")){
                              if(operands.get(i-1).equals("AND")){

      positionOfNegativesOperandsWithAnd.add(i);
                              }else{

      positionOfNegativesOperandsWithOR.add(i);
                              }
                              operands.remove(i);
                        }
                  }

                  ///Submete a consulta aos índices e recupera junto a
persistencia
                  for(i=0;i<terms.size();i++){
                        if(positionOfNegativesOperandsWithAnd.contains(i)){

      hitsVec.add(i,this.performSearch(terms.get(i)));
                        }else{

      hitsVec.add(i,this.performSearch(terms.get(i)));
                        }
                  }
```

```
                    a = System.nanoTime();

                    //Ordena as consultas composta apenas por operadores AND

                    if((!(operands.contains("OR") ||
positionOfNegativesOperandsWithAnd.size() > 0) && operands.size() > 0)){
                          hitsVec =
ResultMatchingAuxiliarOperations.ordenarList(hitsVec);
                    }

                    //Realiza a operacao de diferenca de conjunto para a
operacao (A AND NOT B)
                    for(i=0;i<positionOfNegativesOperandsWithAnd.size();i++){

      this.diference(hitsVec,positionOfNegativesOperandsWithAnd.get(i)-1);

      operands.remove(positionOfNegativesOperandsWithAnd.get(i)-1);
                    }


                    for(i=0; i<operands.size(); i++){
                          if(operands.get(i).equals("AND")){
                                this.intersection(i, hitsVec);
                          }
                    }

                    for(i=0; i<hitsVec.size(); i++){
                          this.union(resultVec,hitsVec.get(i));
                    }

                    resultVec =
ResultMatchingAuxiliarOperations.urlFilter(resultVec);

                    end = System.nanoTime();
                    double time = (end-start)/1000000.0;
                    this.output(resultVec, time);


                    if(configuracao.tipoDePersistencia().equals("BD") &&
!configuracao.withCache()){
                          conexao.closeConection();
                    }
                    if(configuracao.withCache()) {
                          indexarCache();
                          searcher.close();
                    }
                    cacheWithLoadedCentroids = new HashMap<String,
List<String>>();
                    cacheWithLoadedClusters = new HashMap<String,
List<FormElement>>();
                    //return resultVec;
                    fechaArquivo();
                    return time;
              }
              return 0.0;
      }

      private void indexarCache() {
```

```
            for(String index: cacheWithLoadedCentroids.keySet()){
                  indexCache.indexURLCache(index,
cacheWithLoadedCentroids.get(index));
            }
            try {
                  indexCache.commit();
            } catch (CorruptIndexException e) {
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

      private List<String> performSearch(String query) throws IOException,
ParseException, SQLException{
            Searcher searcherDeSessao;
            HashSet<ScoreDoc[]> listaDocumentosCentroids = new
HashSet<ScoreDoc[]>();
            boolean keyword = false;
            String rotuloOriginal;
            List<String> listaDeURLs = new ArrayList<String>();
            List<String> listaDosElementosRecuperar = new
ArrayList<String>();
            List<String> listaDosElementosRecuperar1 = new
ArrayList<String>();

            List<String> listaDeSinonimos = new ArrayList<String>();
            String consulta = query.split(":")[1];
            String consultaComStemming = "";
            if(query.contains("keyword:")){
                  searcherDeSessao = searcherKeyword;
                  rotuloOriginal = consulta;
                  keyword = true;
            }else if(query.contains("context:")){
                  searcherDeSessao = searcherContext;
                  rotuloOriginal = consulta.split("###")[1];
            }else{
                  searcherDeSessao = searcherMetadata;
                  rotuloOriginal = consulta.split("###")[0];
                  if(consulta.split("###")[1].equals("value"))
                        keyword = true;
            }

            //1 - Busca dos sinonimos do rotulo no indice de sinonimos.
            //2 - Com os sinonimos coletados, ver quais sinonimos sao os
adequados a consulta.
            //3 - Extracao da informacao dos Documents Lucene que
representam os centroids.
            getSinonimosRotuloOriginal(analisador, keyword, rotuloOriginal,
listaDeSinonimos);
            acessarDocumentosDoIndice(query, analisador, searcherDeSessao,
rotuloOriginal, listaDeSinonimos, consulta, listaDocumentosCentroids);

      listaDeURLs.addAll(ResultMatchingAuxiliarOperations.extractUrls(search
erDeSessao, listaDocumentosCentroids));


            if(!configuracao.tipoDePersistencia().equals("BD")){
```

```
      listaDosElementosRecuperar.addAll(ResultMatchingAuxiliarOperations.ext
ractFolder(searcherDeSessao,listaDocumentosCentroids));
                  listaDosElementosRecuperar1 =
ResultMatchingAuxiliarOperations.filtroElementos(listaDosElementosRecuperar);
            }else{

      listaDosElementosRecuperar.addAll(ResultMatchingAuxiliarOperations.ext
ractIDS(searcherDeSessao, listaDocumentosCentroids));
                  listaDosElementosRecuperar1 =
ResultMatchingAuxiliarOperations.filtroElementos(listaDosElementosRecuperar);
            }
            listaDosElementosRecuperar =
ResultMatchingAuxiliarOperations.filtroElementos(listaDosElementosRecuperar);

            //Collections.sort(listaDosElementosRecuperar);

            listaDocumentosCentroids = new HashSet<ScoreDoc[]>();
            int hits = 0;
            if(configuracao.withCache()){
                  for(String sin: listaDeSinonimos){
                        if(query.contains("context:")){
                              consultaComStemming =
consulta.split("###")[0].trim().toLowerCase()+"###"+sin;
                              TopDocs top =
searcher.searchCache(consultaComStemming);
                              listaDocumentosCentroids.add(top.scoreDocs);
                              hits = listaDocumentosCentroids.size();

                        }else if(query.contains("keyword:")){
                              consultaComStemming = sin;
                              TopDocs top =
searcher.searchCache(consultaComStemming);
                              listaDocumentosCentroids.add(top.scoreDocs);
                              hits = listaDocumentosCentroids.size();
                        }else{
                              consultaComStemming =
analisador.stemTerms(sin)+"###"+consulta.split("###")[1];
                              TopDocs top =
searcher.searchCache(consultaComStemming);
                              listaDocumentosCentroids.add(top.scoreDocs);
                              hits = listaDocumentosCentroids.size();
                        }
                  }
            }

            if(hits > 0){
                  long start = System.nanoTime();
                  //System.out.println("CACHE");

      listaDeURLs.addAll(ResultMatchingAuxiliarOperations.extractUrls(search
er, listaDocumentosCentroids));
                  long end = System.nanoTime();
                  System.out.println("Tempo acesso à Cache: " + (end-
start));

            }else{
                  if(listaDosElementosRecuperar1.size() > 0){
```

```
	if(!configuracao.tipoDePersistencia().equals("BD")){
				//System.out.println("ARQUIVOS");

	listaDeURLs.addAll(extrairDocumentsFromFilePersistence(listaDosElement
osRecuperar1));
			}else{
				long start = System.nanoTime();
				//System.out.println("BD");

	listaDeURLs.addAll(extrairDocumentsFromDatabase(listaDosElementosRecup
erar1));
				long end = System.nanoTime();
				System.out.println("Tempo De busca no BD:
"+((end-start)/1000000));
			}
			cacheWithLoadedCentroids.put(consultaComStemming,
listaDeURLs);
		}
	}
	/*
	//SEM CACHE
	if(listaDosElementosRecuperar.size() > 0){
		if(!tipoDaPersistencia.equals("BD")){
			List<String> op =
extrairDocumentsFromFilePersistence(listaDosElementosRecuperar);
			//System.out.println("Quantidade Recuperada:
"+op.size());
			listaDeURLs.addAll(op);
		}else{
			List<String> op =
extrairDocumentsFromDatabase(listaDosElementosRecuperar);
			listaDeURLs.addAll(op);
			//System.out.println("Quantidade Recuperada:
"+op.size());
		}
	}
	System.out.println("TEMPO PERS: "+ (end-start) );
	 */
	//analisador.close();
	escreverNoArquivo(query,listaDosElementosRecuperar);
	return
ResultMatchingAuxiliarOperations.filtroElementos(listaDeURLs);
	}

	/**
	 * Criei de modo a manter esse metodo guardado
	 * @param listaDocumentosCentroids
	 */
	/**
	 * @param query
	 * @param stemm
	 * @param analisador
	 * @param searcherDeSessao
	 * @param rotuloOriginal
	 * @param synonyms
	 * @param consulta
	 * @param dd
	 * @throws IOException
```

```
	 * @throws ParseException
	 */
	private String acessarDocumentosDoIndice(String query, WFSimAnalyzer
analisador, Searcher searcherDeSessao,
			String rotuloOriginal, List<String> synonyms, String
consulta, HashSet<ScoreDoc[]> dd) throws IOException, ParseException {
		long start = System.nanoTime();
		String consultaComStemming = null;
		for(int i=0;i<synonyms.size();i++){
			if(query.contains("context:")){
				//COM STEMMING
				if(configuracao.isStemm()){
					consultaComStemming =
consulta.split("###")[0].trim().toLowerCase()+"###"+analisador.stemTerms(syno
nyms.get(i).toLowerCase());
				}else{
					consultaComStemming =
consulta.split("###")[0].trim().toLowerCase()+"###"+synonyms.get(i).toLowerCa
se();
				}

	dd.add(searcherDeSessao.searchContext(consultaComStemming).scoreDocs);
			}else if(query.contains("metadata:")){
				//COM STEMMING
				if(configuracao.isStemm()){
					consultaComStemming =
analisador.stemTerms(synonyms.get(i))+"###"+consulta.split("###")[1];
				}else{
					consultaComStemming =
synonyms.get(i)+"###"+consulta.split("###")[1];
				}

	dd.add(searcherDeSessao.searchMetadata(consultaComStemming).scoreDocs)
;
			}else if(query.contains("keyword:")){
				//COM STEMMING
				if(configuracao.isStemm()){
					consultaComStemming =
analisador.stemTerms(rotuloOriginal);
				}else{
					consultaComStemming = rotuloOriginal;
				}

	dd.add(searcherDeSessao.searchKeyword(consultaComStemming).scoreDocs);
			}
		}
		long end = System.nanoTime();
		System.out.println(end-start);
		return consultaComStemming;
	}

	/**
	 * @param stemm
	 * @param analisador
	 * @param keyword
	 * @param rotuloOriginal
	 * @param synonyms
	 * @throws IOException
	 * @throws ParseException
```

```
	 */
	private void getSinonimosRotuloOriginal(WFSimAnalyzer analisador,
boolean keyword, String rotuloOriginal, List<String> synonyms) throws
IOException, ParseException {
		if(keyword == false){
			TokenStream result = analisador.tokenStream("keyword", new
StringReader(rotuloOriginal));
			List<String> token = Indexer.listTokens(result);
			for(String labels: token){
				//COM STEMMING
				if(configuracao.isStemm()){

	synonyms.addAll(getSinonimosDoIndice(analisador.stemTerms(labels)));
				}else{

	synonyms.addAll(getSinonimosDoIndice(labels));
				}

			}
		}else{
			synonyms.add(rotuloOriginal);
		}
	}

	public void output(List<String> resultVec, double time) {
		HashSet<String> result = new HashSet<String>();
		result.addAll(resultVec);

		System.out.println("Found " + result.size() + " results in " +
String.format("%.2f", time) + " ms");

	}

	/*
	 * Operacoes de Conjunto
	 * =========================
	 */

	private void union(List<String> resultVec, List<String> docVec) {
		resultVec.addAll(docVec);
		Set<String> auxiliar = new HashSet<String>(resultVec);
		resultVec.clear();
		resultVec.addAll(auxiliar);
	}

	private void intersection(int i, List<List<String>> hits) {

		Set<String> docVec1 = new HashSet<String>(hits.get(i));
		Set<String> docVec2 = new HashSet<String>(hits.get(i+1));

		docVec1.retainAll(docVec2);
		//escreverNoArquivo("A AND B = ", ""+ docVec1.size());
		hits.remove(i);
		hits.remove(i);
		hits.add(i, new ArrayList<String>(docVec1));
		hits.add(i, new ArrayList<String>());
	}

	private void diference(List<List<String>> hit, Integer index) {
```

```
			List<String> first = hit.get(index);
			List<String> second = hit.get(index+1);
			first.removeAll(second);
			List<String> intersection = new ArrayList<String>(first);
			hit.remove(index);
			hit.remove(index);
			hit.add(index, intersection);
		}

		/* =========================
		 */

		private void filterOperators(List<String> tokens) {
			for(String element:tokens){
				if(element.equals("AND") || element.equals("OR") ||
element.equals("NOT")){
					operands.add(element);
				} else{
					terms.add(element);
				}
			}
		}

		public List<String> extrairDocumentsFromFilePersistence(List<String>
doc){
			List<FormElement> cluster = new ArrayList<FormElement>();
			List<String> url = new ArrayList<String>();
			try {
				for(String main: doc){
					List<String> aux = new ArrayList<String>();
					if(cacheWithLoadedCentroids.containsKey(main)){

		url.addAll(cacheWithLoadedCentroids.get(main));
					}else{
						cluster =
IndexContruction.deserializeFile(main+"/clusterElements.ser");
						for(FormElement find: cluster){
							url.add(find.getUrl());
							aux.add(find.getUrl());
						}
						cacheWithLoadedCentroids.put(main,aux);
					}
				}

			} catch (FileNotFoundException e) {
				System.err.print("Cluster nao encontrado\nO cluster sera
calculado\n");
			} catch (IOException e) {
				e.printStackTrace();
			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			}
			return url;
		}

		public List<FormElement> extrairDocumentsFilePersistence(List<String>
doc){
			List<FormElement> cluster = new ArrayList<FormElement>();
			List<FormElement> elements = new ArrayList<FormElement>();
```

```
		String dirName1 = "";
		try {
			for(String main: doc){
				if(cacheWithLoadedClusters.containsKey(main)){

	elements.addAll(cacheWithLoadedClusters.get(main));
				}else{
					cluster =
IndexContruction.deserializeFile(main+"/clusterElements.ser");
					elements.addAll(cluster);
					cacheWithLoadedClusters.put(main,cluster);
				}
			}
			return elements;
		} catch (FileNotFoundException e) {
			System.err.print("Cluster: '" +
dirName1+"/clusterElements.ser" + "' nao encontrado\nO cluster sera
calculado\n");
		} catch (IOException e) {
			e.printStackTrace();
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
		return new ArrayList<FormElement>();
	}

	public List<String>
extrairDocumentsFromDatabaseSemNroCluster(List<String> doc) throws
SQLException{
		List<String> url = new ArrayList<String>();
		if(doc.size() > 0){
			List<String> aux = conexao.getElementosCentroid(doc);
			url.addAll(aux);
		}
		return url;
	}

	public List<String> extrairDocumentsFromDatabase(List<String> doc)
throws SQLException{
		List<String> url = new ArrayList<String>();
		if(doc.size() > 0){
			List<String> aux =
conexao.getElementosCentroid(configuracao.nroCluster(),doc);
			url.addAll(aux);
		}
		return url;
	}
}
```

```java
package net.wfsim.matching;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Vector;

import net.wfsim.indexer.Searcher;
import net.wfsim.structures.form.FormElement;

import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.search.ScoreDoc;

public class ResultMatchingAuxiliarOperations {

	public static boolean isInteger( String input ) {
		try {
			Integer.parseInt( input );
			return true;
		}
		catch( Exception e ) {
			return false;
		}
	}

	public static List<List<String>> ordenarList(List<List<String>>
hitsVec) {
		int menor,pos;
		List<List<String>> hits = new ArrayList<List<String>>();
		int tam = hitsVec.size();
		for(int i=0; i<tam;i++){
			menor = 100000;
			pos = 0;
			for(List<String> doc:hitsVec){
				if(doc.size() < menor){
					menor = doc.size();
					pos = hitsVec.indexOf(doc);
				}
			}
			hits.add(hitsVec.get(pos));
			hitsVec.remove(pos);
		}
		return hits;
	}

	public static List<String> filtroElementos(List<String> docs) {
		List<String> folders = new ArrayList<String>();
		for(String doc: docs){
			if(!folders.contains(doc)){
				folders.add(doc);
			}
		}
		return folders;
	}

	public static Vector<String> extractDomain(Searcher
searcher,ScoreDoc[] not) throws CorruptIndexException, IOException {
		Vector<String> docs = new Vector<String>();
```

```
		for(int i = 0; i < not.length; i++){

	if(!docs.contains(searcher.getDocument(not[i].doc).get("domain")))

	docs.add(searcher.getDocument(not[i].doc).get("domain"));
		}
		return docs;
	}

	public static Vector<String> extractUrls(Searcher searcher,ScoreDoc[]
not) throws CorruptIndexException, IOException {
		Vector<String> docs = new Vector<String>();
		for(int i = 0; i < not.length; i++){
			docs.add(searcher.getDocument(not[i].doc).get("url"));

		}
		return docs;
	}

	public static Vector<String> extractUrls(Searcher
searcher,HashSet<ScoreDoc[]> not) throws CorruptIndexException, IOException {
		Vector<String> docs = new Vector<String>();
		for(ScoreDoc[] doc: not){
			for(int i = 0; i < doc.length; i++){

	docs.add(searcher.getDocument(doc[i].doc).get("url"));
			}
		}
		return docs;
	}

	public static Vector<String> extractFolder(Searcher
searcher,ScoreDoc[] not) throws CorruptIndexException, IOException {
		Vector<String> docs = new Vector<String>();
		for(int i = 0; i < not.length; i++){
			docs.add(searcher.getDocument(not[i].doc).get("folder"));
		}
		return docs;
	}

	public static Vector<String> extractFolder(Searcher
searcher,HashSet<ScoreDoc[]> not) throws CorruptIndexException, IOException {
		Vector<String> docs = new Vector<String>();
		for(ScoreDoc[] doc: not){
			for(int i = 0; i < doc.length; i++){

	docs.add(searcher.getDocument(doc[i].doc).get("folder"));
			}
		}
		return docs;
	}

	public static Vector<String> extractIDS(Searcher
searcher,HashSet<ScoreDoc[]> not) throws CorruptIndexException, IOException {
		Vector<String> docs = new Vector<String>();
		for(ScoreDoc[] doc: not){
			for(int i = 0; i < doc.length; i++){

	docs.add(searcher.getDocument(doc[i].doc).get("id"));
```

```
                }
            }
            return docs;
      }

      public static HashSet<String> extractSynonyms(Searcher
searcher,ScoreDoc[] not) throws CorruptIndexException, IOException {
            HashSet<String> docs = new HashSet<String>();
            for(int i = 0; i < not.length; i++){

      docs.add(searcher.getDocument(not[i].doc).get("centroid"));

            }
            return docs;
      }

      public static List<String> urlFilter(List<String> documents) throws
CorruptIndexException, IOException {
            List<String> urls = new ArrayList<String>();
            for(String doc: documents){
                  if(!urls.contains(doc)){
                        urls.add(doc);
                  }
            }
            return urls;
      }

      public static Vector<FormElement> formFilter(Vector<FormElement>
elements) {
            Vector<FormElement> element = new Vector<FormElement>();
            List<String> url = new ArrayList<String>();
            for(FormElement file: elements){
                  if(!url.contains(file.getUrl())){
                        element.add(file);
                        url.add(file.getUrl());
                  }
            }
            return element;
      }



}
```