IMPLEMENTASI CONTENT-BASED RETRIEVAL PADA PERPUSTAKAAN DIGITAL BERBASIS OPEN SOURCE MENGGUNAKAN APACHE LUCENE

DAVID

Program Studi Teknik Informatika,
Sekolah Tinggi Manajemen Informatika dan Komputer Pontianak
Jln. Merdeka No 372 Pontianak, Kalimantan Barat
E-mail: **David_Liauw@yahoo.com** dan **DavidLiauw@gmail.com**

Abstract : *Applications are built can be used to classification search result documents and documents search easier. Documents that are only for the article from journal, thesis, ebook and other documents. Indexing and searching documents using Lucene as a search engine. An filing text document often needed for finding document that content special word or combination of some words. In this research, is made application that can save and retrieve text document, using java program language, db4O and Lucene Library. For efficient in saving, data stop word eliminated, and take account the existent of synonim words. In retrieving document it is possible using operator AND, OR and NOT with the number of words priority that exist in that document. The process of searching are divided into two, namely Simple Search and Advanced Search. Simple Search using a query to search Db4o while Advanced Search using the search terms in the index using Lucene library. In this system the test results obtained are accurate.*

Keywords *: Apache Lucene, Indexing, TF-IDF*

1. PENDAHULUAN

Perkembangan ilmu pengetahuan dan teknologi saat ini semakin pesat, hampir semua bidang teknologi sekarang berbasis internet. Salah satu teknologi yang saat ini kita rasakan manfaatnya adalah *World Wide Web* atau dikenal dengan nama *web.* Dewasa ini web menjadi sumber data terbesar dan sangat berharga untuk setiap pengguna karena di dalam web terdapat dokumen-dokumen digital yang dapat diakses melalui koneksi internet. Setiap orang di dunia bebas menambahkan konten dalam webnya, dan makin banyak informasi yang ditambahkan, makin besar ukuran web dan semakin sulit pula untuk mencari informasi yang benar-benar diinginkan di web oleh karena itu dibutuhkan suatu teknologi untuk mendapatkan informasi yang benar-benar diinginkan.

Hal tersebut juga dialami para pengguna informasi dalam memanfaatkan teknologi internet, salah satunya adalah perpustakaan digital (*digital library).* Perpustakaan digital semakin berkembang karena dapat menyimpan koleksi diperpustakaan seperti buku, jurnal ilmiah, majalah, tugas akhir, skripsi dan tesis maupun disertasi dalam format digital. Oleh karena itu diperlukan teknologi yang tepat untuk mengembangkan perpustakaan digital yang sesuai dengan harapan pengguna dan menjadi tujuan di masa depan oleh semua pihak.

Peneliti mengusulkan untuk mengembangkan perpustakaan digital yang berbasis open source yang khusus menangani dokumen-dokumen perpustakaan sehingga pengguna dapat terfokus pada pencarian dokumen-dokumen perpustakaan tersebut.

Secara keseluruhan tujuan dari penelitian ini adalah untuk membangun sebuah program perpustakaan digital yang berbasis open source.

Penulis merumuskan masalah sebagai berikut: 1) Bagaimana mengimplementasikan aplikasi perpustakaan digital, 2) Dokumen apa saja yang disimpan, 3) Bagaimana melakukan pencarian dokumen berdasarkan isi dokumen (content-based retrieval), dan 4) Bagaimana menghitung similarity dokumen hasil pencarian.

Batasan masalah yang digunakan dalam penelitian ini adalah: 1) Dokumen yang digunakan adalah file-file tesis, skripsi, jurnal dan prosiding, artikel ilmiah, modul Kuliah serta diktat kuliah, 2) Tipe-tipe file yang digunakan berformat PDF, TXT, DOC, ODT, RTF dan PPT dan, 3) Aplikasi diimplementasikan dengan bahasa pemrograman Java.

2. TINJAUAN PUSTAKA

2.1 Perpustakaan Digital

Perpustakaan adalah 'gudang ilmu' bukan gudang buku seperti selama ini dipersepsikan orang pada umumnya (Jacob, 1990). Perkembangan dunia perpustakaan, dari segi data dan dokumen yang disimpan, dimulai dari perpustakaan tradisional yang hanya terdiri dari kumpulan koleksi buku tanpa katalog, kemudian muncul perpustakaan semi modern yang menggunakan katalog (*index*). Perkembangan mutakhir adalah munculnya perpustakaan digital (*digital library*) yang memiliki keunggulan dalam kecepatan pengaksesan karena berorientasi ke data digital dan media jaringan komputer (internet).

Perpustakaan Digital (*digital library*) adalah suatu perpustakaan yang menyimpan data, baik itu buku (tulisan), gambar, suara dalam bentuk file elektronik dan mendistribusikannya dengan menggunakan protokol elektronik melalui jaringan computer (Subrata**,** 2009). Perpustakaan digital telah banyak dibuat tetapi pelayanan kepada pengguna belum maksimal karena pengaksesan dokumen masih terbatas pada informasi tentang buku, seperti judul, pengarang, penerbit. Dokumen yang sesungguhnya belum bisa diakses secara open source (Wahono, 2006).

3.1 TF-IDF

Vector space model merepresentasikan dokumen dengan *term* yang memiliki bobot. Bobot tersebut menyatakan kepentingan/kontribusi term terhadap suatu dokumen dan kumpulan dokumen. Kepentingan suatu kata dalam dokumen dapat dilihat dari frekuensi kemunculannya terhadap dokumen. Biasanya *term* yang berbeda memiliki frekuensi yang berbeda. Dibawah ini terdapat beberapa metode pembobotan (Tao, dkk. 2005)) :

- *Term Frequency*
  Term frequency merupakan metode yang paling sederhana dalam membobotkan setiap term. Setiap term diasumsikan memiliki kepentingan yang proporsional terhadap jumlah kemunculan *term* pada dokumen. Bobot dari *term t* pada dokumen *d* dihitung menggunakan persamaan (1).

$$TF(d,t) = f(d,t) \quad \text{..... (1)}$$

  Dimana $f(d,t)$ adalah frekuensi kemunculan *term t* pada dokumen *d*.

- *Inverse Document Frequency (IDF)*
  Bila *term frequency* memperhatikan kemunculan term di dalam dokumen, maka *IDF* memperhatikan kemunculan term pada kumpulan dokumen. Latar belakang pembobotan ini adalah term yang jarang muncul pada kumpulan dokumen sangat bernilai. Kepentingan tiap term diasumsikan memilki proporsi yang berkebalikan dengan jumlah dokumen yang mengandung term. Faktor *IDF* dari *term t* dihitung menggunakan persamaan (2).

$$IDF(t) = \log\left(\frac{n}{df(t)}\right) \quad \text{..... (2)}$$

  Di mana *n* adalah jumlah seluruh dokumen, *df(t)* jumlah dokumen yang mengandung *term t*.
- *Term Frequency- Inverse Document Frequency (TF-IDF)*
  Perkalian antara *term frequency* dan *IDF* dapat menghasilkan performansi yang lebih baik. *TF-IDF* merupakan kombinasi bobot dari *term t* pada dokumen *d* dapat dihitung pada persamaan (3).

$$TFIDF(d,t) = TF(d,t) \times IDF(t) \quad \text{..... (3)}$$

Setiap dokumen sebagai suatu vektor dengan satu komponen yang saling berkorepondensi dengan setiap term dalam dictionary. Untuk term-term yang tidak muncul dalam suatu dokumen diberi bobot nol (0). Bentuk vektor inilah yang menentukan *scoring* dan *ranking*. *Score* suatu dokumen d adalah total dari term-term yang dibandingkan dengan *query* termnya yang muncul dalam dokumen *d* (Seki, 2003).

$$Score(q,d) = \sum_{t \in q} tfidf_{t,d} \quad \text{..... (4)}$$

3.2 Apache Lucene

Tujuan dari penyimpanan indeks adalah untuk mengoptimalkan kinerja dan kecepatan dalam mencari dokumen yang relevan untuk permintaan pencarian. Tanpa indeks, mesin pencari akan mencari dokumen di setiap storage, sehingga memerlukan waktu yang lama dan daya komputasi yang besar. Kebanyakan mesin pencari menggunakan *B-Tree* untuk menyimpan indeks, mereka relatif stabil sehubungan dengan penyisipan data (operasi *lookups* dan *insertions*-nya *O(log n)*). Prasad dan Patel (2005) menyatakan bahwa Lucene mengambil pendekatan yang sedikit berbeda yaitu membangun beberapa indeks segmen dan menggabungkannya secara berkala daripada mempertahankan satu indeks. Lucene menggunakan *Inverted Index* untuk melakukan penyimpanan *index* (Prasad dan Patel, 2005). Struktur data *Inverted Index* merupakan komponen pusat yang khas pada algoritma *indexing* mesin pencari.

Untuk setiap dokumen baru yang akan diindeks, Lucene membuat indeks segmen baru, tetapi dengan cepat menggabungkan segmen kecil dengan segmen yang lebih besar sehingga mempertahankan jumlah total segmen kecil agar pencarian tetap cepat. Untuk mengoptimalkan indeks dalam pencarian cepat, Lucene dapat menggabungkan semua segmen menjadi satu, yang berguna untuk indeks yang jarang diperbaharui. Untuk mencegah konflik (atau penguncian *overhead*) antara indeks *reader* dan indeks *writer*, Lucene memodifikasi segmen tidak pada tempatnya, hanya membuat baru. Ketika menggabungkan segmen, Lucene menulis segmen yang baru dan menghapus segmen yang lama setelah setiap reader yang aktif ditutup. Pendekatan ini

menjadi pertimbangan yang baik, Lucene menawarkan derajat fleksibilitas yang tinggi dalam kecepatan pengindeksan untuk proses pencarian, dan memiliki karakteristik I/O untuk *merging* dan *searching*.

Segmen indeks dari Lucene terdiri dari beberapa file, yaitu sebuah kamus indeks yang berisi satu entri untuk setiap 100 masukan dalam kamus, sebuah kamus berisi satu entri untuk tiap kata dan sebuah file posting yang mengandung entri untuk setiap posting.

Selama segmen Lucene tidak pernah diupdate, mereka dapat disimpan dalam file flat yang tidak serumit *B-Tree*. Untuk kecepatan *retrieval*, kamus indeks yang berisi *offsets* ke dalam file kamus, dan kamus memegang *offsets* ke dalam file *posting*. Lucene juga mengimplementasikan berbagai cara untuk memampatkan file kamus dan file *posting* sehingga mengurangi I/O disk tanpa mengakibatkan *overhead* CPU yang besar.

3.4 Basis data obyek Db4o

Objek database mulai populer pada pertengahan tahun 1990-an. Bermula dari *Objek Oriented Programming(OOP)* yang kemudian dikembangkan menjadi *Objek Oriented Design (OOD)* dan pada akhirnya menjadi *Objek Oriented Analysis (OOA)*. Didalam konsep objek oriented database kita dapat melakukan pemodelan data dari semua phenomena dan dapat dinyatakan dalam bahasa umum (natural). *Objek Oriented Database* pada dasarnya merupakan konsep dari pemrograman berorientasi objek secara umum ditambah dengan database sebagai media penyimpanan datanya yang berbentuk *class-class*, sehingga dalam hal ini masih berhubungan erat dengan *E-R Model*.

OODB muncul karena kekomplekan dari penyimpanan objek-objek yang akan disimpan didalam database sehingga konsep dari *Relational Database Manejemen Sistem (RDBMS)* masih tetap digunakan. Mekanisme penyimpanan objek-objek didalam Relational Database Menejemen Sistem ini sering dikenal dengan istilah *ORDBMS (Objek Relational Database Managemen System*).

Keunikan dari Db4o adalah Db4o sebagai *native object database* sehingga sangat ideal untuk di-*embed* ke dalam *equipment* atau *device*, baik *mobile*, *desktop*, dan *server platform* (Irwanto, 2007). *Footprint Db4o* dapat dibilang cukup kecil sehingga membuat Db4o lebih efektif untuk di-*embed* ke dalam device yang memiliki kapasitas memori yang kecil. Walaupun memiliki footprint yang kecil, namun *class library Db4o* menyediakan fitur-fitur yang lengkap, seperti *concurrency control*, replikasi data, dan *native query* (Irwanto, 2007). Sistem Db4o tidak banyak membutuhkan administrasi sehingga db4o cocok jika di-*embed* ke dalam aplikasi.

3. METODOLOGI PENELITIAN

Dalam penelitian ini yang dijadikan obyek penelitian adalah perpustakaan digital di Perpustakaan Magister Ilmu Komputer UGM Yogyakarta. Di dalam mengumpulkan data-data penulis menggunakan metode sebagai berikut : (1) Observasi, yaitu melakukan pengamatan langsung dengan pihak-pihak yang menggunakan perpustakaan digital (2) Dokumentasi, yaitu mengambil data berupa dokumen-dokumen yang dibutuhkan sebagai dokumen pencarian.

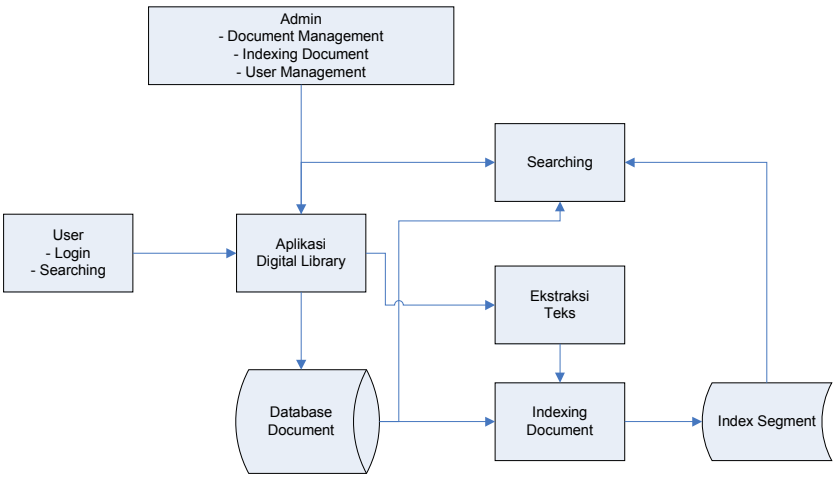
Penelitian ini menggunakan metode pengembangan perangkat lunak *Object Oriented Software Development* yang terdiri dari tahap-tahap sebagai berikut: 1) *Requirement Analysis* menggunakan Studi literature : Studi mengenai perpustakaan digital, java open source, java, *indexing* dan *searching* menggunakan Apache Lucene, pembobotan menggunakan TF-IDF dan lainnya untuk membangun perpustakaan digital

berbasis open source, 2) Analisis : perpustakaan digital dengan menggunakan metode berorientasi obyek yang mencakup use case model dan class diagram, 3) perancangan : Perancangan model bagaimana obyek berinteraksi satu sama lain saat system sedang dijalankan yang mencakup activity dan sequence diagram, 4) Implementasi: Rancangan model yang telah dibuat akan diimplementasikan dengan menggunakan J2SE sebagai antar muka dari sistem perpustakaan digital ini dalam tampilan desktop, dan 4) Pengujian aplikasi : testing aplikasi yang telah dibuat untuk mencari kelemahan dari sistem, kemudian disempurnakan kembali sampai didapatkan suatu sistem yang siap pakai.

4. HASIL PENELITIAN

4.1. Perancangan Sistem

Perancangan perangkat lunak (*software*) yang dipakai untuk menjalankan sistem ini dibagi menjadi dua bagian, yaitu program untuk admin dan program untuk user. Berikut ini blok diagram yang akan menggambarkan sistem dari aplikasi yang akan dibuat dapat dilihat pada gambar:



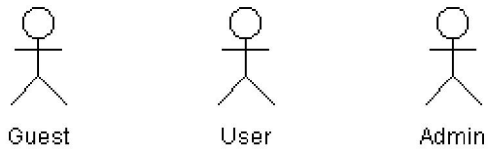
Gambar 1. Bagan Perancangan Perangkat Lunak Digital Library

Dokumen-dokumen disimpan ke dalam suatu folder pada path aplikasi dan semua atribut dari dokumen tersebut disimpan dalam Object Database Db4o. Semua dokumen yang tersimpan di-Indexing dengan sebelumnya diekstraksi teksnya dan disimpan ke dalam Index Segment menggunakan library Apache Lucene. Adapun dokumen yang dapat diterima oleh aplikasi ialah dokumen file teks (*.txt dan *.rtf), file Acrobat Document(*.pdf), file dokumen word atau *Ms. Office Document* (*.doc), file dokumen power point (*.ppt), file dokumen openoffice (*.odt), dan dokumen-dokumen e-book berformat file *.chm. File-file dokumen seperti file tesis, proposal tesis, artikel jurnal, ebook dan dokumen lainnya terlebih dahulu melalui proses ekstraksi teks (*Parser*) untuk kemudian dianalisis menggunakan *library lucene*, seperti yang terlihat pada gambar 1. Tahapan analisis meliputi memproses stopword, analisa standar, lower case dan porter stemming. Hasil analisis kemudian diindex dan tersimpan dalam file index. Proses searching dibedakan menjadi dua, yaitu Simple Search dan *Advanced*

Search. *Simple Search* menggunakan *query Db4o* untuk melakukan pencarian sedangkan *Advanced Search* menggunakan pencarian term pada index menggunakan *library lucene*.

4.2 Identifikasi Actor

Actor yang berinteraksi dengan Aplikasi ini adalah guest, pengguna (user) dan Pengurus perpustakaan (sebagai Administrator). Pengurus perpustakaan merupakan actor yang menggunakan aplikasi untuk mengelola isi dari aplikasi tersebut. User merupakan actor yang memiliki hak terbatas dalam mengakses aplikasi tersebut. keterbatasan hak akses yang dimiliki oleh member menyebabkan user tidak dapat mengubah isi dari aplikasi tersebut. Berikut ini gambaran actor yang terdapat dalam aplikasi.



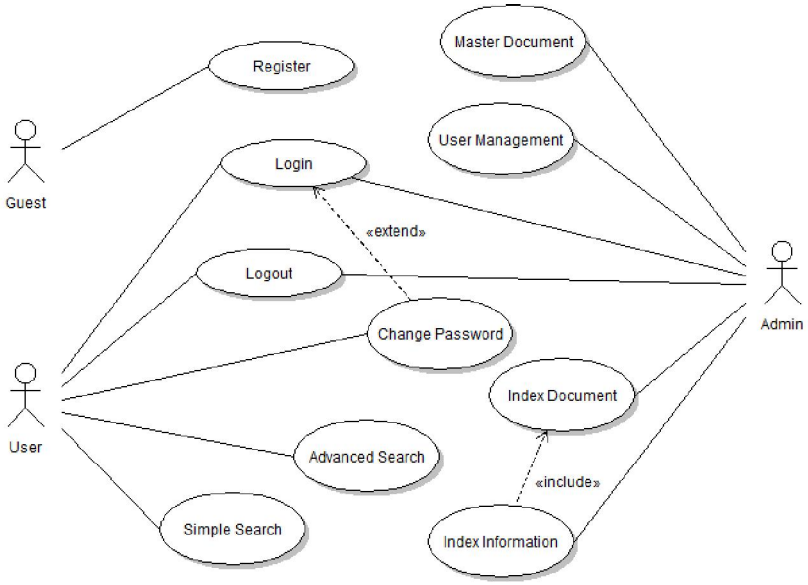
Gambar 2. Actor

4.3 Analisis Sistem

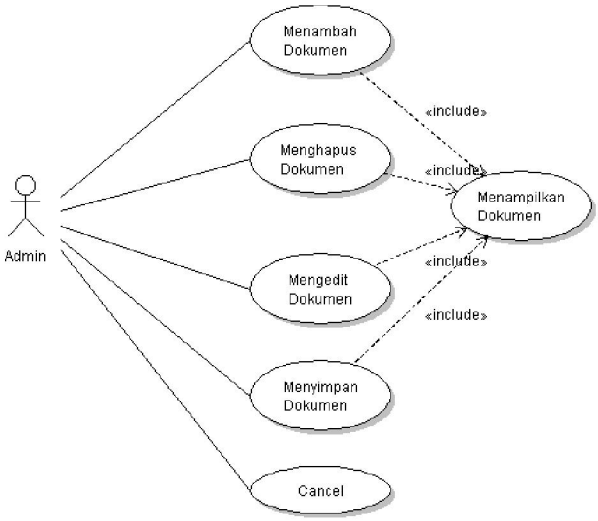
Metode yang digunakan adalah metode perancangan sistem berorientasi obyek. Dalam hal ini penggunaan UML (*Unified Modelling Language*) sangat sesuai karena aplikasi yang dibangun di atas teknologi Java yang mengimplementasikan paradigma pengembangan sistem berorientasi obyek.

4.4 Use Case Diagram

Use Case Diagram berisi gambaran fungsionalitas yang diharapkan dari sebuah sistem dengan fokus penekanan pada apa yang dilakukan oleh sistem, bukan bagaimana sistem melakukan sesuatu. Dalam use case diagram terdapat dua phak yang saling berhubungan yaitu actor dan use case yang berkaitan dengan actor. Dengan diagram ini dapat diketahui cakupan dari sistem, siapa saja aktor yang berperan dalam sistem dan interaksi antara aktor dengan elemen-elemen use case dalam sistem. Diagram use case yang memperlihatkan aktor-aktor yang berperan dalam sistem, diperlihatkan pada Gambar.



Gambar 3. Use Case Aplikasi Digital Library



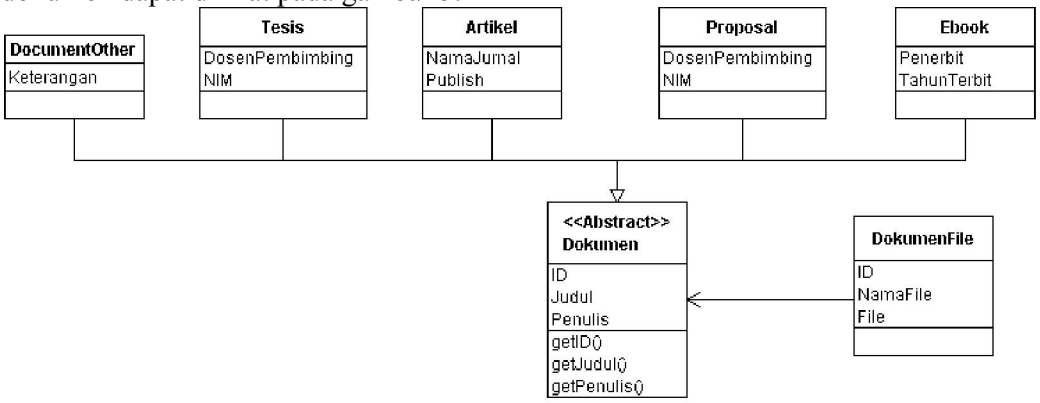
Gambar 4. Use Case Master Document

4.5 Class Diagram

Diagram class menggambarkan struktur clas di dalam sistem. Class merepresentasikan sesuatu yang ditangani oleh sistem. Class dapat berhubungan dengan yang lain melalui berbagai cara yaitu associated (terhubung satu sama lain), dependent (satu class tergantung/menggunakan class yang lain), specialed (satu class merupakan spesialisasi dari class lainnya), atau package (grup bersama sebagai satu unit).

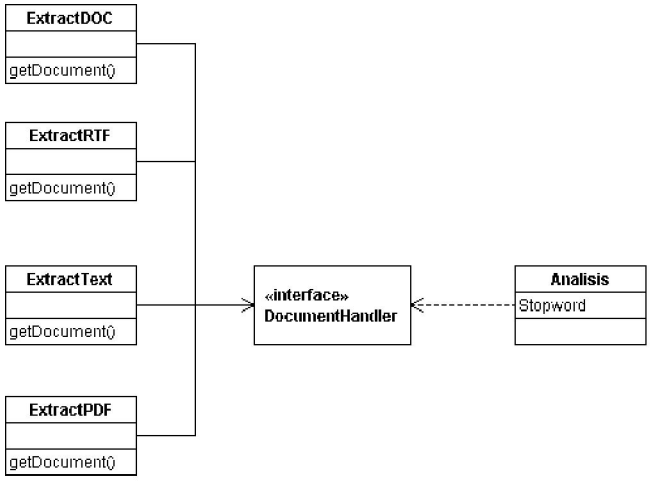
Pada package dokumen, class dokumen memiliki hubungan struktur *generalization*. *Generalization* mengekspresikan *inheritance* (pewarisan sifat) dari *super class* ke *sub class*. Dalam *generalization*, hubungan *super class* dengan *sub class* secara *linguistic* diformulasikan dengan *"is a"*. Oleh karena itu, secara otomatis instance dari *sub class* merupakan *instance* bagi *super class*.

Class dokumen merupakan *super class* yang berupa *abstract class*. *Abstract class* merupakan class yang tidak mempunyai object. Eksistensi *abstract clas*s adalah untuk mendefinisikan suatu frame. Object dari sebuah *abstract class* direalisasikan melalui sub classnya. *Sub class* dari *abstract class* dokumen adalah class Tesis, class Proposal, class EBook, class Jurnal dan class DocumentOther. Pada package dokumen terdapat class DokumenFile yang berhubungan struktur *navigation association* dengan *abstract class* dokumen. *Navigation association* (*unindirectional association*) adalah bentuk hubungan asosiasi antar object yang sifatnya satu arah. Hubungan object yang memiliki navigational association direalisasikan dengan korespondensi attribute dari object key yang berhubungan. Selain korespondensi, attribute salah satu object harus memiliki akses operation terhadap object yang satunya. Rancangan class diagram dokumen dapat dilihat pada gambar 5.



Gambar 5. Rancangan Class Diagram Dokumen

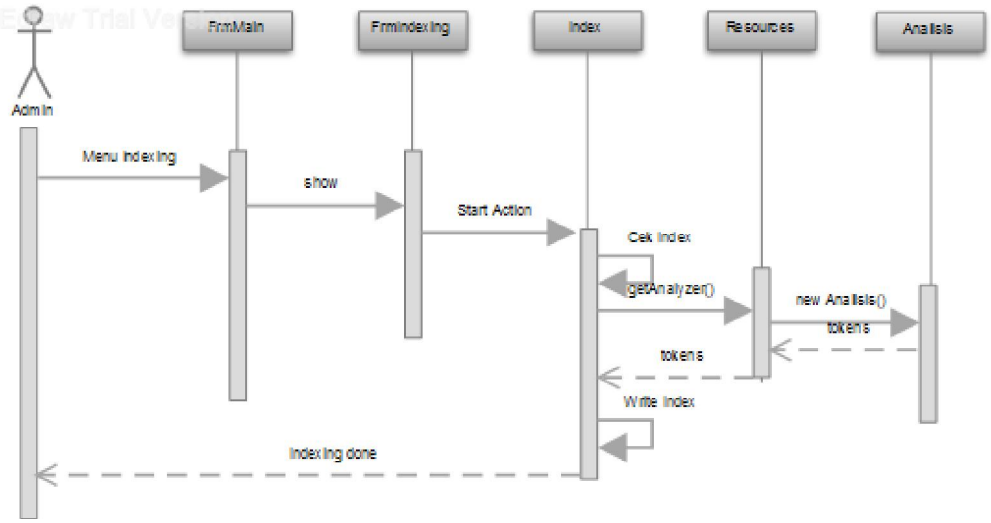
Analisis dokumen terlebih dahulu dilakukan proses ekstraksi. Class ExtractDoc, ExtractPDF, ExtractRTF dan ExtractTXT mengunakan interface DocumentHandler agar mudah dihubungkan ke class Analisis. Rancangan class diagram untuk analisis dapat dilihat pada gambar 6.



Gambar 6. Rancangan Class Diagram Analisis

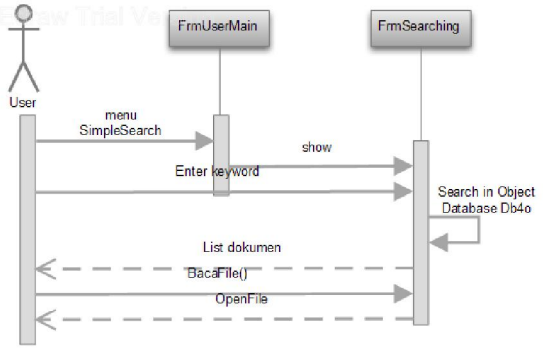
4.6 *Sequence Diagram*

Pada gambar 7 terlihat ketika admin memilih menu *Indexing* maka maka akan ditampilkan form *Indexing*. Pada form ini admin dapat melakukan pengindeksan terhadap semua dokumen yang ada di direktori dokumen.



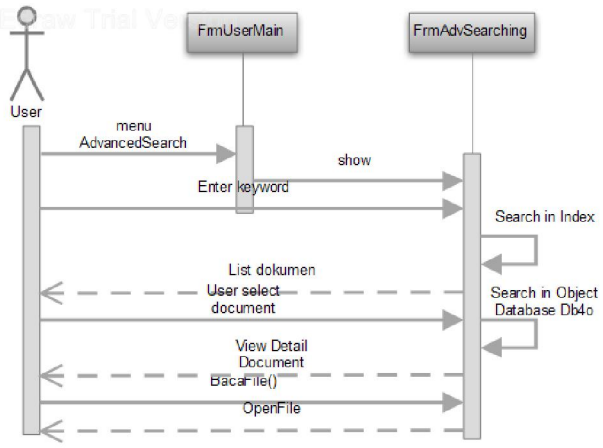
Gambar 7. Sequence Diagram Indexing

Pada gambar 9 terlihat ketika pengguna menampilkan *Form Simple Search* maka pada form ini pengguna dapat langsung melakukan pencarian berdasarkan judul atau nama penulis.



Gambar 8. *Sequence Diagram Simple Searching*

Pada gambar 9 terlihat ketika pengguna menampilkan *Form Advanced Search* maka pada form ini pengguna dapat langsung melakukan pencarian berdasarkan isi dokumen.



Gambar 9. Sequence Diagram Advanced Searching

4.7 Pengujian Sistem

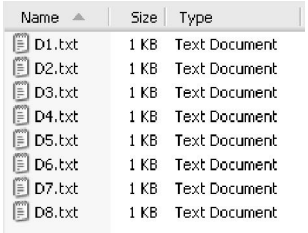
Tujuan dari tahap pengujian ini adalah pengujian sistem yang meliputi pengujian *indexing* dan pengujian *searching*, dengan studi kasus menggunakan data dokumen. Pengujian ini dilakukan sehingga dapat membuktikan kinerja sistem aplikasi yang sudah disusun pada tahap implementasi.

Pada pengujian ini, dokumen yang digunakan berupa delapan dokumen teks yang disimpan pada path "C:\D". Nama dokumen teks beserta isi dokumennya dapat dilihat pada Tabel 1.

Tabel 1 Dokumen Teks

No	Nama File Dokumen	Isi Dokumen Teks
1	D1.txt	How Ant Communicate Each Other
2	D2.txt	Computer And Ant
3	D3.txt	Ant Colony System
4	D4.txt	A Little Facts About Ant
5	D5.txt	The Ant Colony System
6	D6.txt	Pheromone Trail
7	D7.txt	multi agent system
8	D8.txt	swarm intelligence

Tampilan direktori dokumen teks yang disimpan pada path "C:\D" dapat dilihat pada gambar 10.

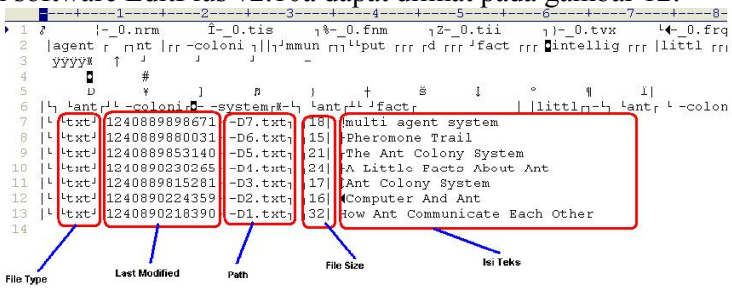


Gambar 10. Direktori Document

Dokumen-dokumen teks tersebut memiliki tipe file Plain Text Document (file txt). Dari aplikasi dilakukan proses pengindeksan sehingga dihasilkan beberapa file pada direktori aplikasi. Gambar 11 menunjukkan tampilan file index yang terbentuk.

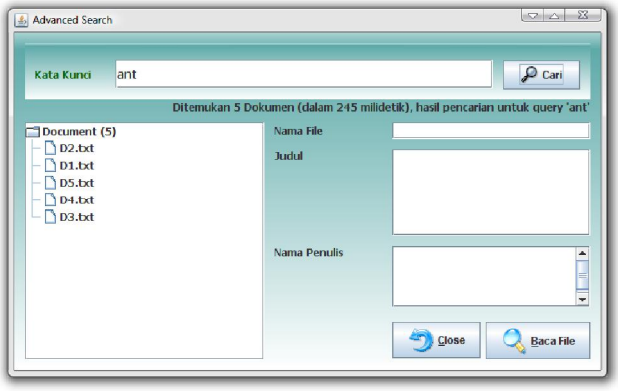
Gambar 11. File Index

Pada isi file _0.cfs terdapat sejumlah term-term, tipe file berupa txt, waktu terakhir file dokumen dimodifikasi (last modified), nama file dokumen , ukuran dokumen dan isi dari dokumen-dokumen tersebut. Tampilan isi file _0.cfs menggunakan software EditPlus v2.10a dapat dilihat pada gambar 12.



Gambar 12. Isi file _0.cfs

Pengujian pencarian berdasarkan isi dari dokumen, digunakan Advanced Search pada aplikasi. Misalkan dilakukan pencarian dokumen yang isinya terdapat term "ant", maka program aplikasi akan mencarinya pada index dan diketemukan sebanyak lima dokumen, yaitu D1.txt, D2.txt, D3.txt, D4.txt dan D5.txt. Pengujian advanced search diperlihatkan pada gambar 13.



Gambar 13 Hasil Pencarian berdasarkan kata kunci "ant"

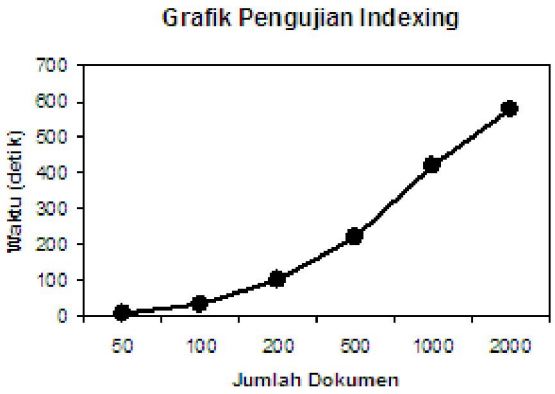
Pengujian indexing dilakukan dengan dokumen yang kompleks dan diharapkan dapat diketahui kekurangan-kekurangan dari sistem untuk kemudian diperbaiki sehingga kesalahan dari sistem dapat diminimalisasi atau bahkan dihilangkan. Pengujian sistem ini dilakukan untuk mendapatkan hasil yang akurat.

Pengujian indexing dilakukan terhadap variasi jumlah dokumen pada suatu direktori dokumen. Dari 6 kali pengujian yang dilakukan terdapat beberapa dokumen yang gagal diindeks. Hal ini disebabkan karena ada beberapa dokumen tersebut

diproteksi sehingga proses pengekstaksian teks tidak dapat dilakukan. Pengujian indexing dapat dilihat pada tabel 2. Adapun grafik pengujian indexing dapat dilihat pada gambar 14.

Tabel 2 Tabel Pengujian Indexing

Pengujian	Jumlah Dokumen	Ukuran	Waktu Index	Ukuran Index	Jumlah Dokumen yang Gagal diIndex
1	50	3,53 MB	7 detik	670 KB	1
2	100	26,9 MB	35 detik	2,59 MB	2
3	200	64,2 MB	1 menit 41 detik	6,54 MB	3
4	500	182 MB	3 menit 39 detik	14,5 MB	12
5	1000	365 MB	7 menit	28,5 MB	24
6	2000	694 MB	9 menit 37 detik	70, 3 MB	44



Gambar 14. Grafik Pengujian Indexing (—●— detik)

5. KESIMPULAN

Dari hasil percobaan yang telah dilakukan dapai diambil beberapa kesimpulan, yaitu dengan adanya aplikasi digital library yang berbasis GUI (Grapical User Interface) seorang user akan lebih mudah melakukan pencarian dokumen berdasarkan content dokumen (*Information Retrieval*). Pada pengujian sistem ini didapatkan hasil pencarian yang akurat. Berdasarkan pada pengujian yang telah dilakukan pada perangkat lunak yang dibuat, masih banyak ekurangan dan kelemahan sehingga perlu dikembangkan lagi agar kinerjanya lebih baik, oleh karena itu saran yang diberikan adalah pengujian dengan dokumen yang besar memerlukan komputasi yang lebih efisien, oleh karena itu disarankan pengembangan aplikasi ini selanjutnya menggunakan teknik pemrograman dan struktur data yang bisa menangani perhitungan matriks yang besar dan lebih cepat dalam memproses *indexing* dan *searching* dokumen.

DAFTAR RUJUKAN

Db4o-5.2 Tutorial, db4objects Inc., USA

Irwanto, Djon., 2007, *Membangun Object Oriented Software dengan Java dan Object Database*, PT Elex Media Komputindo, Jakarta

Prasad, A., dan Patel, D., 2005, "Lucene Search Engine: An Overview", *DRTC-HP International Workshop on Building Digital Libraries using DSpace*, 7th – 11th March, 2005, DRTC, Bangalore.

Seki, Y., 2003, "Sentence Extraction by tf/idf and Position Weighting from Newspaper Articles", *Proceeding of the Third NTCIR Workshop*, National Institute of Informatics

Subrata, Gatot., 2009, *Perpustakaan Digital*, http://library.um.ac.id/images/stories/pustakawan/kargto/Perpustakaan%20Digital.pdf, diakses pada tanggal 24 Maret 2009

Tao, Z.Y., Ling, G., dan Cheng, W.Y., 2005, "An improved TF-IDF approach for text classification", *Journal of Zhejiang University SCIENCE*, 2005 6A(1):49-55, ISSN 1009-3095.

The Apache Software Foundation, 2009, "*Apache Lucene – Overview*", http://lucene.apache.org/java/docs/, diakses pada tanggal 19 Februari 2009.

Wahono, R. S, 2006, "Teknologi Informasi untuk Perpustakaan: Perpustakaan Digital dan Sistem Otomasi Perpustakaan", http://www.scribd.com/doc/3020850/Perpustakaan-Digital-dan-Sistem-Otomasi-Perpustakaan, diakses tanggal 24 Maret 2009.