BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis

3.1.1 Analisis Masalah

Permasalahan pada penelitian ini adalah sulitnya membangun sistem pencarian untuk grup milis Linux di Indonesia. Hal ini dikarenakan besarnya data yang ada pada grup milis di Indonesia, sehingga menyebabkan lamanya proses pengolahan data.

3.1.2 Analisis Kebutuhan Data

Sistem yang akan dibangun merupakan sistem pencarian untuk grup milis Linux yang ada di Indonesia. Sebuah sistem pencarian hanya dapat melakukan proses pencarian jika tempat untuk pencarian ada, dan tempat pencarian tersebut adalah data tulisan grup milis, sehingga untuk dibangunnya sistem ini, dibutuhkan data tulisan yang telah ada pada grup milis.

3.1.3 Analisis Kebutuhan Non Fungsional

Kebutuhan non fungsional yang dibutuhkan untuk membangun sistem ini mencakup dua hal yaitu; kebutuhan perangkat keras dan perangkat lunak.

3.1.3.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang diperlukan untuk membangun dan menjalankan sistem ini adalah :

1. Processor Dual Core untuk komputer server.
2. RAM 1 GB atau lebih untuk komputer server.

3. Hard driver 100 GB atau lebih untuk setiap komputer.
4. Jariangan komputer berkecepatan tinggi untuk proses transfer data antar komputer.

3.1.3.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk membangun sistem pencarian ini adalah :

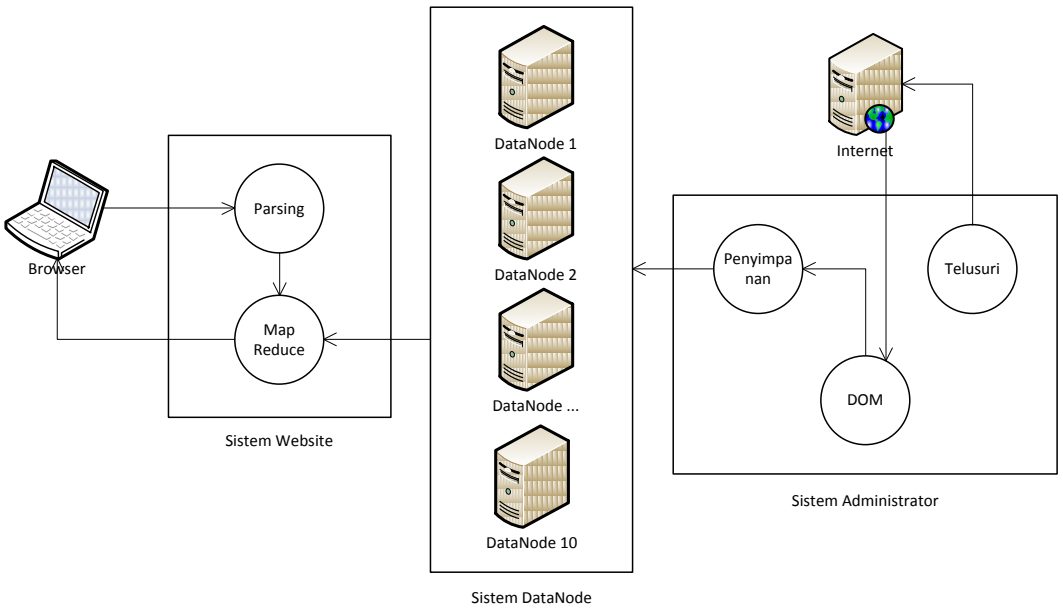
1. Java Development Kit, sebagai sistem kompilasi kode-kode program.
2. Java Runtime Edition, sebagai *platform* untuk menjalankan sistem.
3. Glassfish, sebagai *webserver* untuk sistem website.
4. NetBeans IDE, sebagai perangkat lunak yang digunakan untuk membangun sistem.
5. Ubuntu Linux dan Windows, sebagai sistem operasi.
6. Aplikasi *browser* (seperti FireFox, Chrome, Opera dan Safari) sebagai aplikasi perambah untuk pengguna.

3.1.4 Analisis Sistem

3.1.4.1 Arsitektur Sistem

Sistem dibagi menjadi tiga sistem, yaitu Sistem Website, Sistem Administrator dan sistem DataNode. Sistem website digunakan oleh pengguna untuk melakukan proses pencarian. Sistem Administrator digunakan oleh Administrator untuk melakukan proses pengolahan data, dan Sistem DataNode

digunakan untuk menyimpan seluruh data milis. Gambaran arsitektir sistem dapat dilihat pada gambar 3.1.



Gambar 3.1 Arsitektur Sistem Pencarian

Terdapat banyak proses yang terjadi pada sistem website, sistem administrator dan sistem HDFS, yaitu :

1. Proses *parsing*
2. Proses *MapReduce*
3. Proses penelusuran.
4. Proses DOM
5. Proses penyimpanan data.

3.1.4.2 Analisis Parsing Query

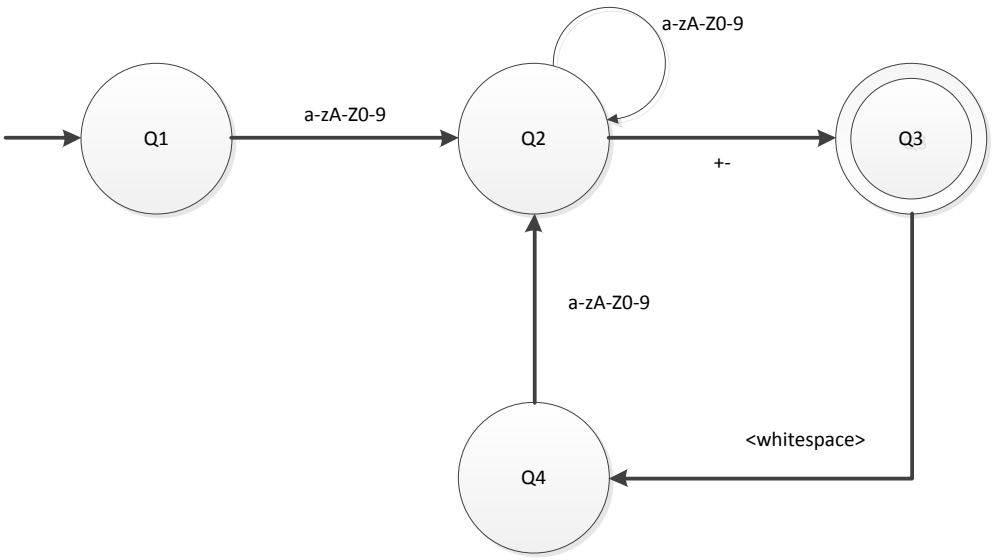
Proses *parsing* (penguraian kalimat) *query* (pertanyaan) merupakan proses yang dilakukan untuk menterjemahkan masukkan pertanyaan dari pengguna agar dapat dimengerti oleh sistem. Secara *default*, seluruh kalimat pertanyaan

pengguna yang dimasukkan akan dianggap sebagai kata-kata yang harus ada pada data yang akan dicari, misal pengguna memasukkan kalimat sebagai berikut :

pemrograman java

Maka sistem akan melakukan pencarian data pada dokumen yang memiliki kata *pemrograman* **atau** *java*. Artinya jika terdapat dokumen yang mengandung kata *pemrograman delphi* maka dokumen tersebut dianggap sesuai dengan yang dicari pengguna, karena mengandung kata *pemrograman*. Hal ini membuat pengguna yang ingin mencari informasi tentang *pemrograman java* akan mendapatkan hasil yang tidak sesuai dengan yang diinginkan.

Jika pengguna ingin melakukan proses pencarian yang lebih spesifik, agar informasi yang didapat lebih tepat sasaran, maka pengguna harus mengikuti aturan proses penguraian kalimat yang dimiliki sistem, yaitu :



Gambar 3.2 Finite State Automata Proses Parsing

Dari diagram state pada gambar 3.2, didapat aturan produksi :

T = {+, -, a-z, A-Z, 0-9, ‘ ‘}

S = A

A → aB...zB | AB...ZB | 0B...B

B → aB...zB | AB...ZB | 0B...B | +C | -C | + | -

C → <spasi>D

D → aB...zB | AB...ZB | 0B...B

Dimana :

A = Q1

B = Q2

C = Q3

D = Q4

Maka diperoleh :

V = {S, A, B, C, D }

P = { S = A, A → aB...zB | AB...ZB | 0B...B , B → aB...zB | AB...ZB | 0B...B | +C | -C | + | - , C → <spasi>D, D → aB...zB | AB...ZB | 0B...B}

Aturan produksi diatas menggunakan metode *Right-to-Left* sehingga jika pengguna memasukkan pertanyaan "+hello", maka proses akan dilakukan dari belakang (*right*) ke depan (*left*) menjadi "olleh+".

S → A

S → oB

S → olB

S → ollB

S → olleB

S → ollehB

S → olleh+

Sehingga “+hello” diterima oleh proses parsing. Bila pengguna memasukkan pertanyaan “eko”, maka pohon penurunannya akan terlihat seperti :

S → oB

S → okB

S → okeB

Maka “eko” tidak diterima oleh proses parsing. Karena masih menghasilkan B, sedangkan B bulankah *final state*.

Pada state diatas, setiap kata diawali dengan tanda (+) atau (-) , dalam sistem tanda (+) diartikan bahwa kata tersebut harus ada dalam kriteria informasi yang akan dicari, dan tanda (-) diartikan kata tersebut tidak boleh ada dalam informasi yang akan dicari.

Agar sistem *fault tolerance*, maka sebelum proses *parsing*, dilakukan proses penyaringan dimana, pada proses penyaringan terjadi proses penghapusan karakter yang tidak diperlukan dan penambahan operasi yang diperlukan. Kumpulan karakter yang akan dihapuskan pada saat proses penyaringan terlihat pada tabel 3.1.

Tabel 3.1 Kumpulan Karakter yang Disaring

Karakter
~
!
@
#
$
%
^
&
*
(
)
_
=
`
<
>
?
/
;
‘
:
“
[
]
{
}
\
|
white space (tab, spasi, enter, carriage return)

Selain penghapusan karakter para proses penyaringan terjadi penambahan operasi + secara otomatis jika pengguna tidak memasukkan operasi tersebut dimana jika pengguna memasukkan pertanyaan :

pemrograman java

Maka sistem akan menambahkan tanda + pada awal kata yang tidak memiliki operasi, sehingga hasilnya adalah :

+pemrograman +java

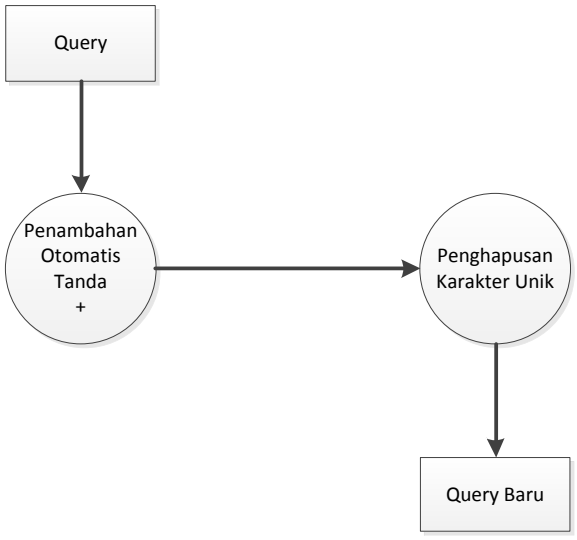
Setelah proses penghapusan karakter yang tidak diperlukan, maka hasil akhirnya adalah :

+pemrograman+java

Dengan begitu pertanyaan yang diberikan oleh pengguna dapat dimengerti oleh sistem *parsing*.

3.1.4.3 Analisis Perbaikan Input Parsing

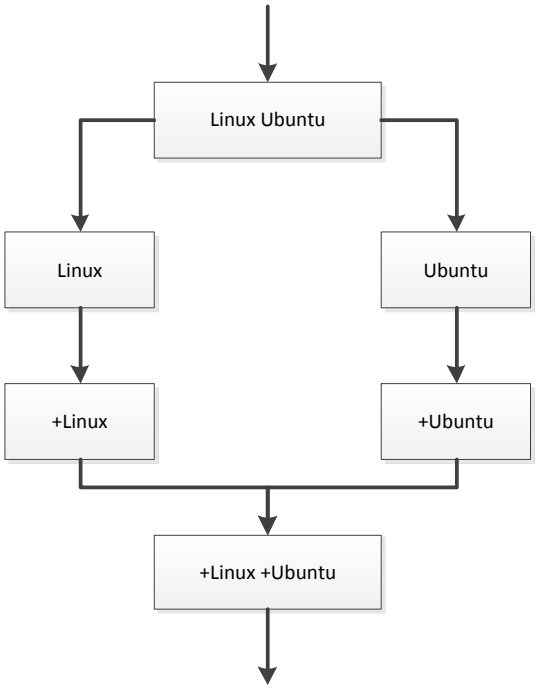
Proses perbaikan input parsing dilakukan agar sistem dapat mentolelir berbagai kesalahan yang dilakukan oleh pengguna saat memasukkan *query* pada saat proses pencarian data. Terdapat dua proses yang terjadi pada saat proses perbaikan input parsing, proses tersebut dapat dilihat pada gambar 3.5.



Gambar 3.3 Tahapan Proses Perbaikan Input Parsing

1. Penambahan otomatis karakter +. Saat pengguna memasukkan data "linux", maka otomatis sistem tidak dapat mengerti *query* tersebut, hal ini karena kata "linux" tidak dapat diterima oleh aturan produksi, oleh karena

itu maka sistem otomatis menambahkan karakter + pada setiap kata yang tidak diawali oleh tanda + atau -, seperti yang terlihat pada gambar 3.6.



Gambar 3.4 Gambar Tahapan Penambahan Tanda +

2. Penghapusan karakter unik. Dikarenakan sistem tidak mendukung karakter-karakter unik seperti yang terdapat ditabel 3.1, maka sistem harus dapat secara otomatis melakukan penghapusan terhadap data karakter unik tersebut dari query, sehingga saat pengguna memasukkan data "pemr%ogram&an" maka sistem harus melakukan penghapusan karakter unik, sehingga query menjadi "pemrograman" (karakter % dan & dihilangkan).

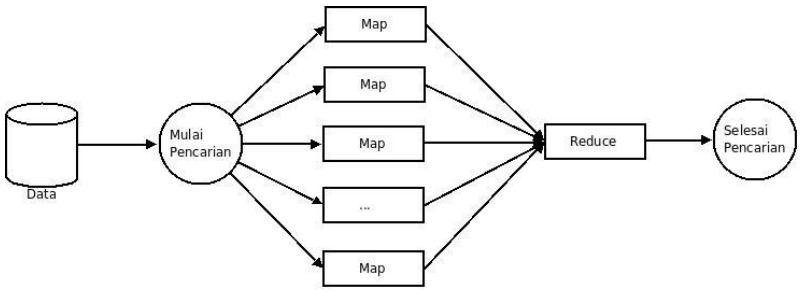
3.1.4.4 Analisis Proses MapReduce

Proses *MapReduce* merupakan proses inti pada sistem yang akan dibangun ini. Proses ini merupakan proses yang dimana terjadi proses pencarian data pada

berkas-berkas yang telah disimpan. Dimana proses pencarian tersebut harus cepat agar hasil bisa dengan cepat diberikan kepada pengguna.

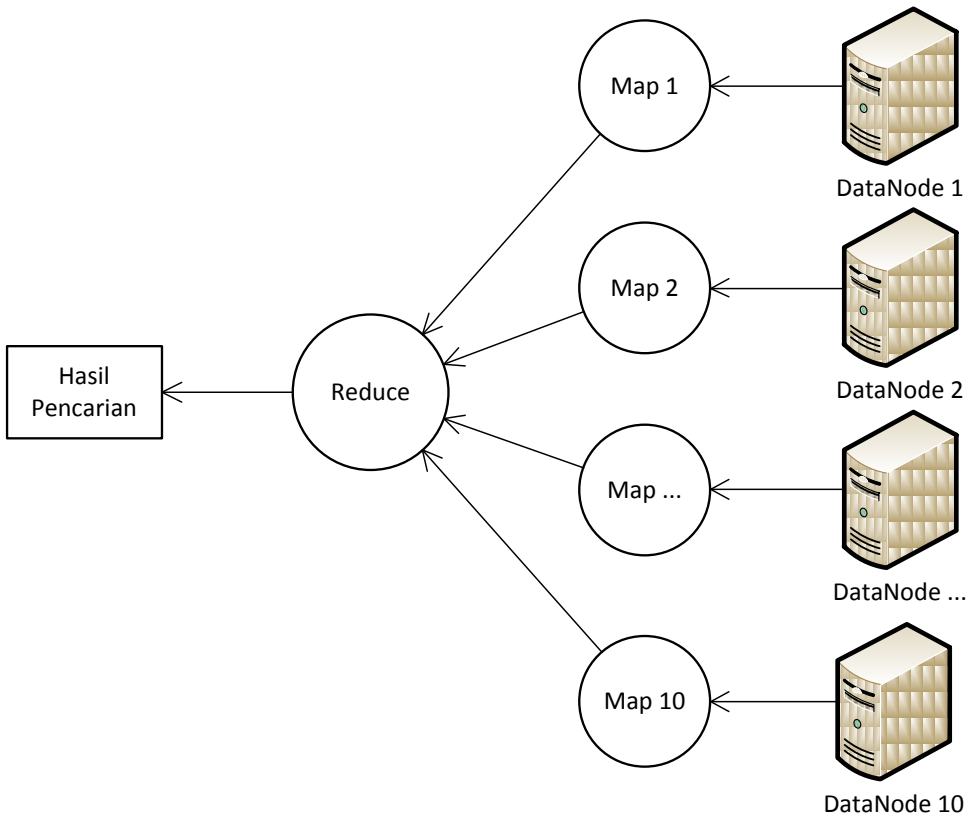
Proses *MapReduce* terjadi setelah sistem berhasil menguraikan kalimat dari pertanyaan yang dimasukkan oleh pengguna. Setelah sistem mengerti dokumen dengan karakteristik apa saja yang dianggap sesuai, maka proses pencarian data dengan model pemrograman *MapReduce* pun berjalan.

Proses pencarian berjalan dengan masukkan berupa berkas-berkas yang akan dilakukan proses pencarian. Setelah itu proses dipecah menjadi beberapa proses dimana proses tersebut dinamakan *Map*, pada proses *Map* tersebut, dilakukan pencarian terhadap masukkan data berkas dengan kriteria yang telah dimasukkan oleh pengguna. Proses *Map* ini berjalan secara parallel, sehingga proses ini akan sangat cepat dibandingkan proses yang sekuensial seperti terlihat pada gambar 3.3.



Gambar 3.5 Proses MapReduce

Proses *Map* dilakukan pada sistem *DataNode*, setiap sistem *DataNode* akan melakukan proses satu proses *Map* untuk satu perintah pencarian seperti terlihat pada gambar 3.4.



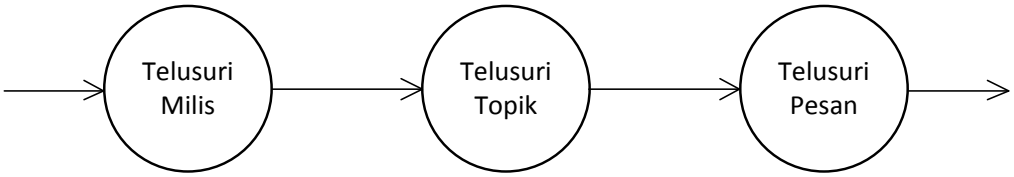
Gambar 3.6 Proses MapReduce Sistem Pencarian pada DataNode

Hasil dari setiap proses pencarian yang terjadi pada proses *Map* akan di gabungkan dalam proses *Reduce*. Dengan melakukan proses pencarian secara paralel, atau berbarengan dalam beberapa sistem, maka akan mempercepat proses pencarian, namun karena sistem terdapat pada komputer yang berbeda, maka sangat diperlukan kecepatan jaringan yang sangat tinggi, agar proses pertukaran data antar komputer berjalan cepat.

3.1.4.5 Analisis Proses Penelusuran

Proses penelusuran merupakan proses *crawling* halaman-halaman diskusi yang terdapat pada grup milis. Proses ini bekerja dengan menelusuri setiap tautan-tautan yang merupakan topik diskusi pada grup milis. Proses penelusuran dimulai dengan menelusuri halaman tautan milis yang dimasukkan oleh Administrator.

Halaman yang pertama kali ditelusiri oleh sistem adalah halaman milis tersebut, setelah halaman milis tersebut ditemukan, sistem akan menelusuri halaman-halaman topik diskusi yang terjadi dalam milis tersebut. Setiap topik diskusi pasti memiliki satu atau lebih pesan yang berasal dari pengguna milis tersebut, sehingga proses selanjutnya adalah melakukan penelusuran terhadap data pesan-pesan yang ada pada topik diskusi tersebut. Tahapan proses penelusuran dapat dilihat pada gambar 3.5.



Gambar 3.7 Proses Penelusuran

Sambil proses penelusuran terjadi, jika topik ditemukan dan pesan ditemukan, maka dilakukan proses DOM (*Document Object Model*), yang kemudian hasil dari DOM tersebut nantinya disimpan dalam *DataNode*.

3.1.4.6 Analisis Proses Document Object Model

Proses DOM (Document Object Model) merupakan proses pengambilan informasi dari halaman-halaman milis yang ditelusuri. Karena sistem melakukan penelusuran lewat website, maka hasil penelusuran tersebut adalah teks HTML, sehingga perlu dilakukan penyaringan data yang akan disimpan dalam *DataNode*. Proses DOM bermulai dengan menkonversi halaman HTML menjadi objek DOM, dengan begitu maka sistem akan dengan mudah mengambil informasi yang ada pada halaman HTML tersebut.

Proses DOM ini perlu dilakukan agar sistem dapat menentukan manakah data yang perlu disimpan, dan manakah data yang tidak perlu disimpan. Data yang

perlu disimpan merupakan data topik dan data pesan-pesan yang ada dalam topik. Saat DOM menemukan sebuah topik pada halaman HTML, maka DOM akan mengkonversinya menjadi objek *Topics*, dimana objek *Topics* tersebut akan disimpan dalam *DataNode*. Contoh halaman HTML topik dalam milis dapat dilihat pada gambar 3.6.



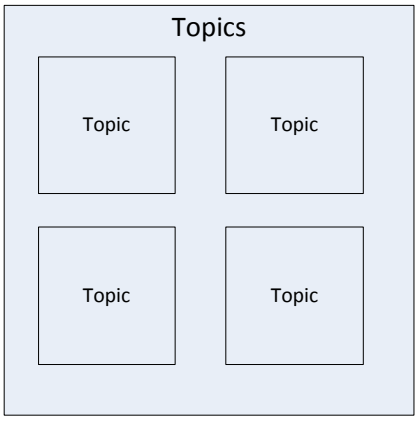
Gambar 3.8 Contoh Halaman Topik Diskusi

Di dalam sebuah topik, akan terdapat minimal satu atau lebih pesan, dan DOM akan mengkonversi setiap pesan yang ada dalam topik menjadi objek *Topic*, dimana objek *Topic* inipun akan disimpan dalam *DataNode*. Pembagian topik dan pesan-pesan yang ada didalamnya dapat dilihat pada gambar 3.7.



Gambar 3.9 Pembagian Topik dan Pesan

Dengan demikian isi sebuah objek *Topics* adalah isi dari satu atau lebih dari objek *Topic* seperti yang terlihat pada gambar 3.8.

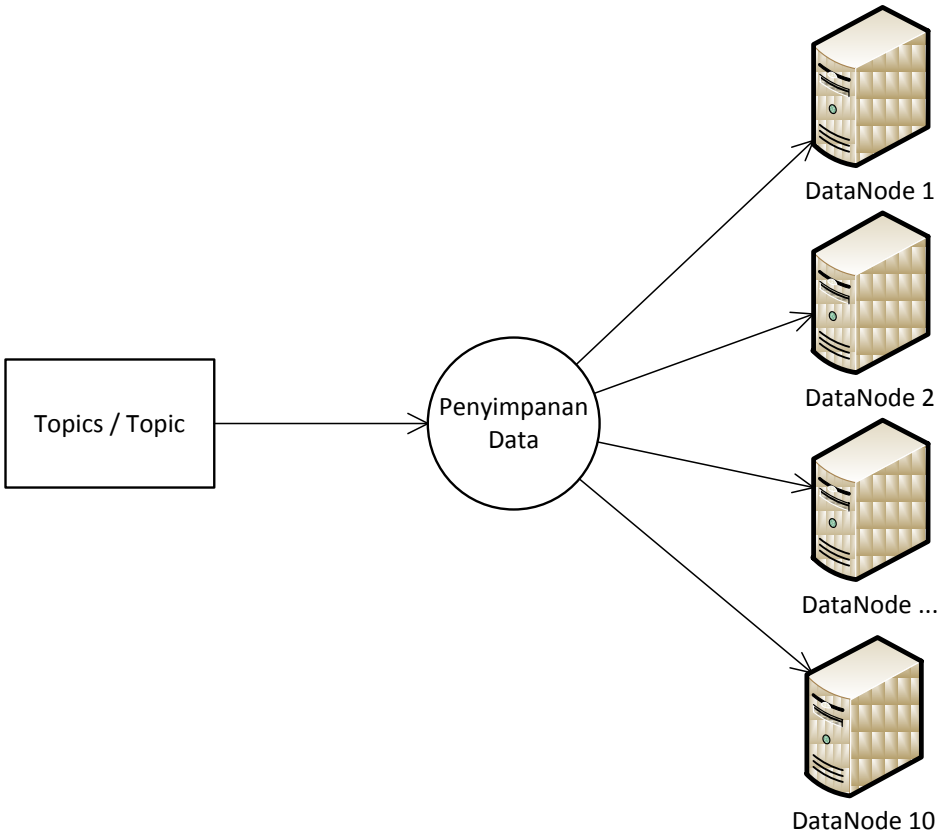


Gambar 3.10 Isi Topics

3.1.4.7 Analisis Proses Penyimpanan Data

Proses penyimpanan data merupakan proses data *Topics* dan *Topic* hasil dari proses DOM sebelumnya disimpan. Data *Topics* dan *Topic* perlu disimpan untuk dijadikan data pencarian. Seluruh data Topics dan Topic yang didapat tersebut akan disimpan pada setiap *DataNode* yang ada, hal ini berakibat data akan

membengkak sebanyak *DataNode* yang ada, jadi jika terdapat data 100MB dan jika memiliki 10 *DataNode*, maka data akan menjadi 1GB. Tahapan penyimpanan data ke *DataNode* dapat dilihat pada gambar 3.8.



Gambar 3.11 Proses Penyimpanan Data

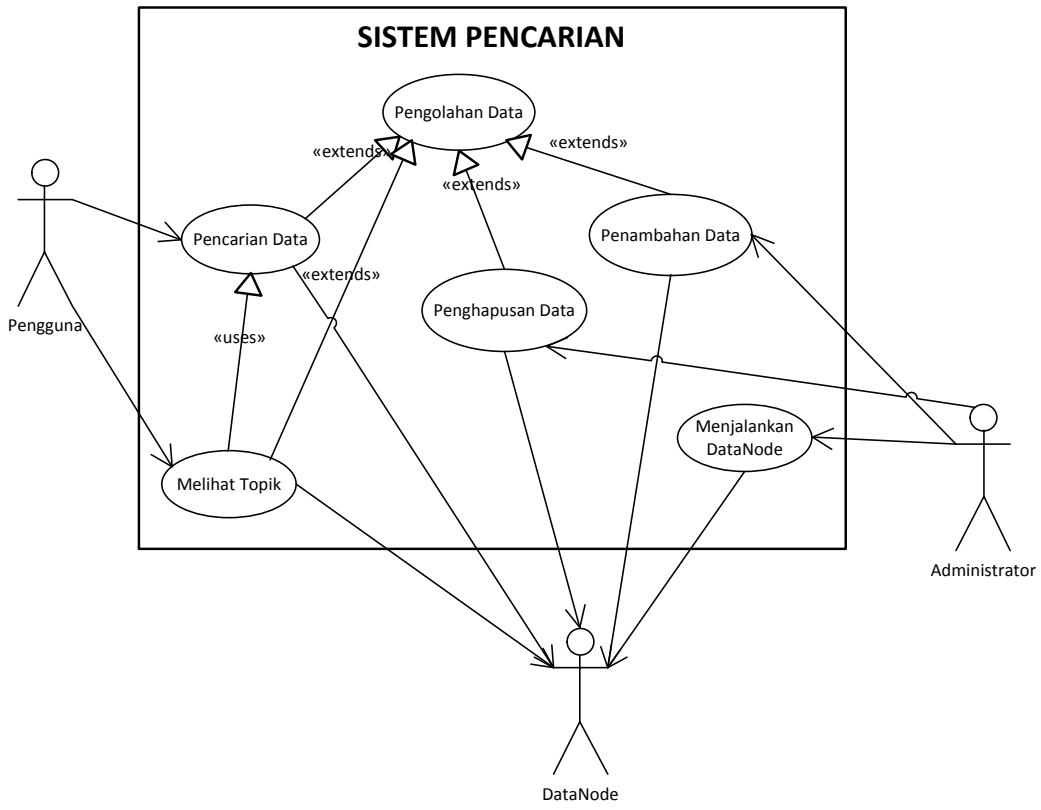
3.2 Perancangan

3.2.1 Perancangan Perangkat Lunak

3.2.1.1 Diagram Use Case

Terdapat dua aktor dalam sistem pencarian yang akan dibangun ini, pengguna dan administrator. Pengguna adalah seseorang yang berperan sebagai pencari informasi, sedangkan Administrator adalah pengguna yang melakukan proses penambahan data, penghapusan data dan yang menjalankan sistem

DataNode. Tugas setiap aktor dapat terlihat pada diagram *Use Case* pada gambar 3.10.



Gambar 3.12 Diagram *Use Case* Sistem Pencarian

3.2.1.2 Diagram Activity

3.2.1.2.1 Pencarian Data

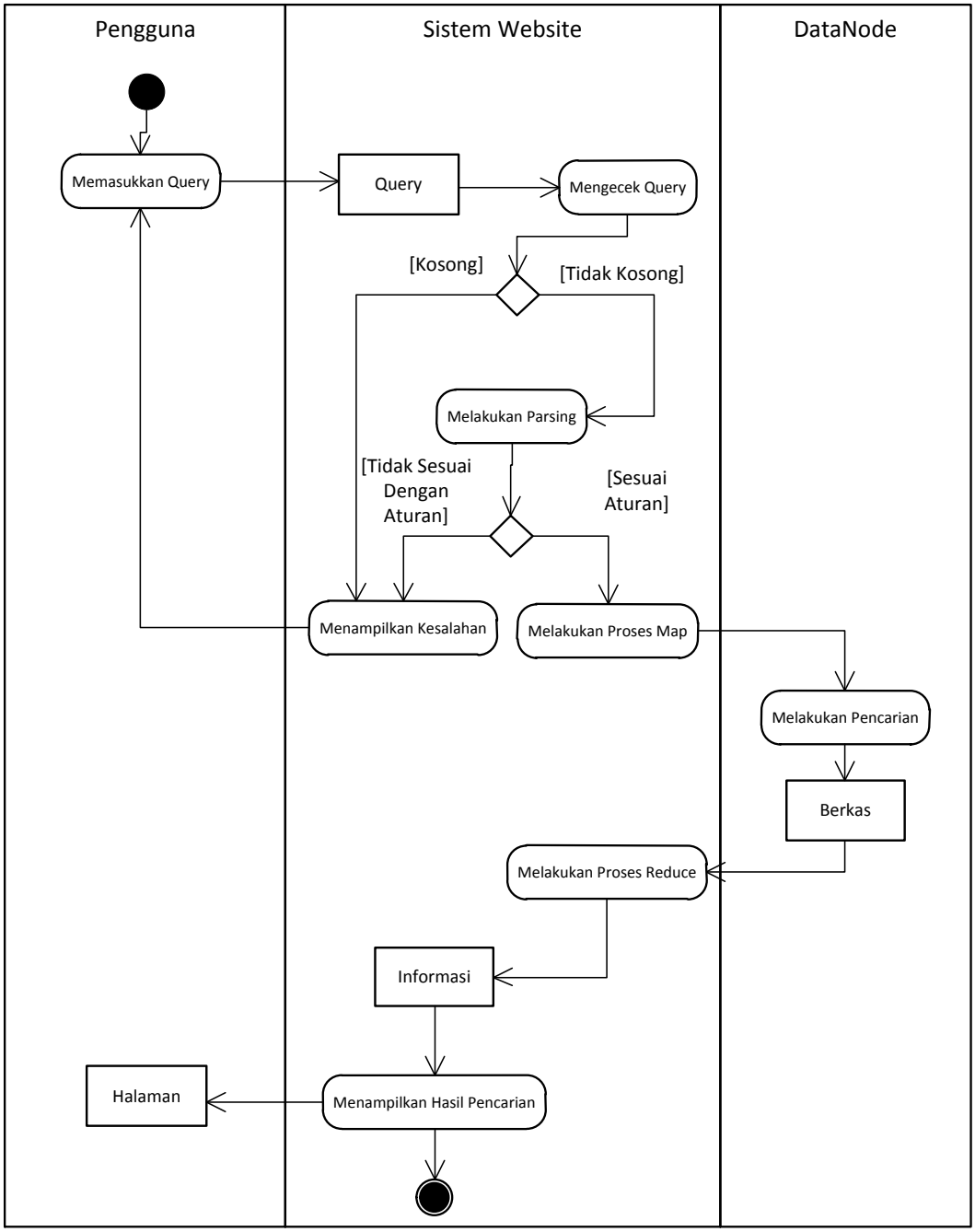
Aktifivas pencarian data merupakan aktifitas pengguna mencari informasi yang dibutuhkan, aktifitas ini inti dari sistem pencarian yang akan dibangun. Tahapan-tahapan aktifitas yang akan dilakukan pada saat pencarian data adalah :

1. Memasukkan *query*, pada aktifitas ini pengguna harus memasukkan *query* yang berisikan kriteria informasi yang diinginkan, *query* harus sesuai dengan ketentuan aturan produksi sistem pada saat proses parsing.

2. Mengecek *query*, pada aktifitas ini sistem melakukan proses pengecekan *query* apakah kosong atau tidak, jika kosong artinya pengguna tidak memasukkan data *query* dan dianggap tidak valid, maka sistem akan menampilkan kesalahan jika tidak kosong, proses selanjutnya akan dilakukan.
3. Melakukan *parsing*, pada aktifitas ini sistem melakukan proses *parsing* terhadap data *query* yang dimasukkan oleh pengguna, hal ini dilakukan agar *query* yang dimasukkan dapat dimengerti oleh sistem. Jika data yang di-*parsing* sesuai dengan aturan produksi, maka data dianggap valid, jika tidak maka data dianggap tidak valid dan sistem akan menampilkan kesalahan.
4. Menampilkan kesalahan, aktifitas ini merupakan aktifitas menampilkan kesalahan dikarenakan *query* yang dimasukkan pengguna tidak sesuai dengan aturan produksi.
5. Melakukan proses *Map*, aktifitas ini merupakan proses *Map* dalam model pemrograman *MapReduce*, dimana pada proses ini sistem melakukan pencarian terhadap berkas yang ada dalam DataNode.
6. Melakukan pencarian, aktifitas ini dilakukan jika data *query* valid. Sistem akan melakukan proses pencarian terhadap informasi-informasi yang ada sesuai dengan kriteria *query* yang dimasukkan pengguna.
7. Melakukan proses *Reduce*, dktifitas ini merupakan proses *Reduce* dalam model pemrograman *MapReduce*, dimana pada proses ini sistem menggabungkan hasil pencarian dari proses *Map*.

8. Menampilkan hasil pencarian, pada aktifitas ini sistem menampilkan seluruh hasil informasi yang didapat dalam proses pencarian.

Seluruh aktifitas pencarian dapat tergambar tahapannya pada diagram *Activity* pada gambar 3.11.



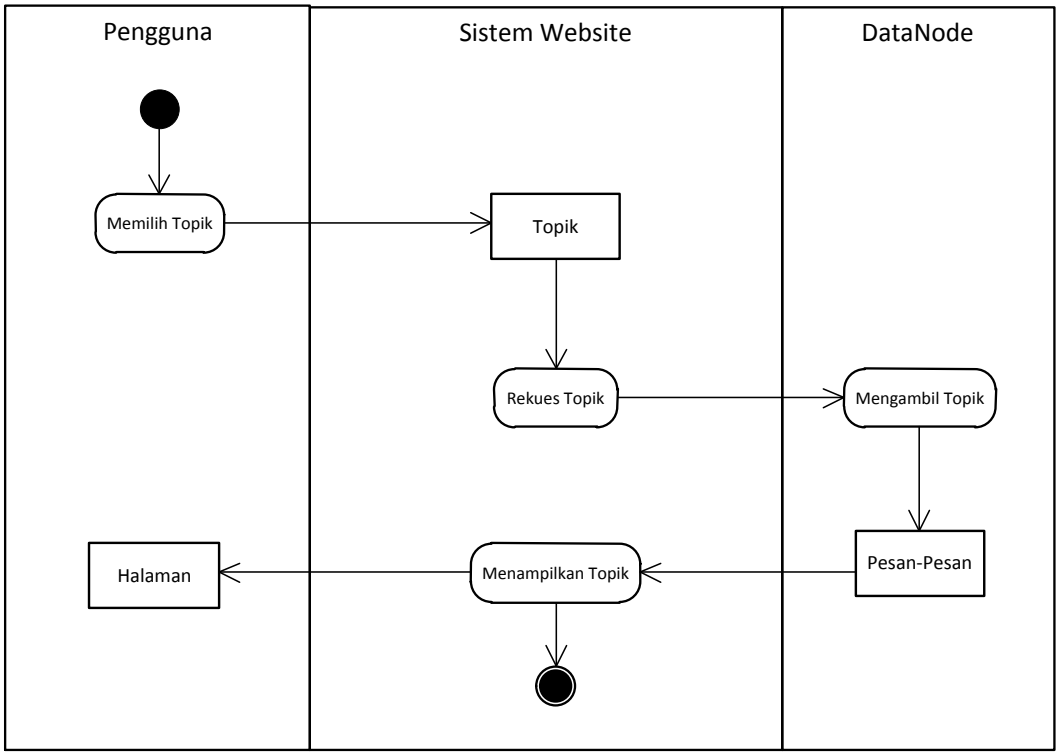
Gambar 3.13 Diagram *Activity* Pencarian Data

3.2.1.2.2 Melihat Topik

Aktifitas melihat topik merupakan aktifitas yang dilakuka oleh pengguna untuk melihat detail data yang dicari. Detail data tersebut berisikan seluruh pesan-pesan milis pada topik, sehingga pengguna tidak perlu sulit untuk mengunjungi lokasi milis untuk melihat pesan-pesan yang ada pada topik tersebut. Dalam aktifitas ini terdapat beberapa tahapan aktifitas, yaitu :

1. Memilih topik, dalam aktifitas ini pengguna memilih topik yang akan dilihat dari hasil pencarian sebelumnya.
2. Rekues topik, dalam aktifitas ini, sistem *website* melakukan *request* data topik ke *DataNode*.
3. Mengambil topik, dalam aktifitas ini *DataNode* akan mengambil topik dari berkas, sesuai dengan yang diminta oleh siste *website*.
4. Menampilkan data, dalam aktifitas ini, sistem menampilkan data seluruh pesan yang ada pada topik yang dipilih kepada pengguna.

Seluruh tahapan aktifitas melihat topik dapat dilihat pada diagram *Activity* pada gambar 3.12.



Gambar 3.14 Diagram *Activity* Melihat Topik

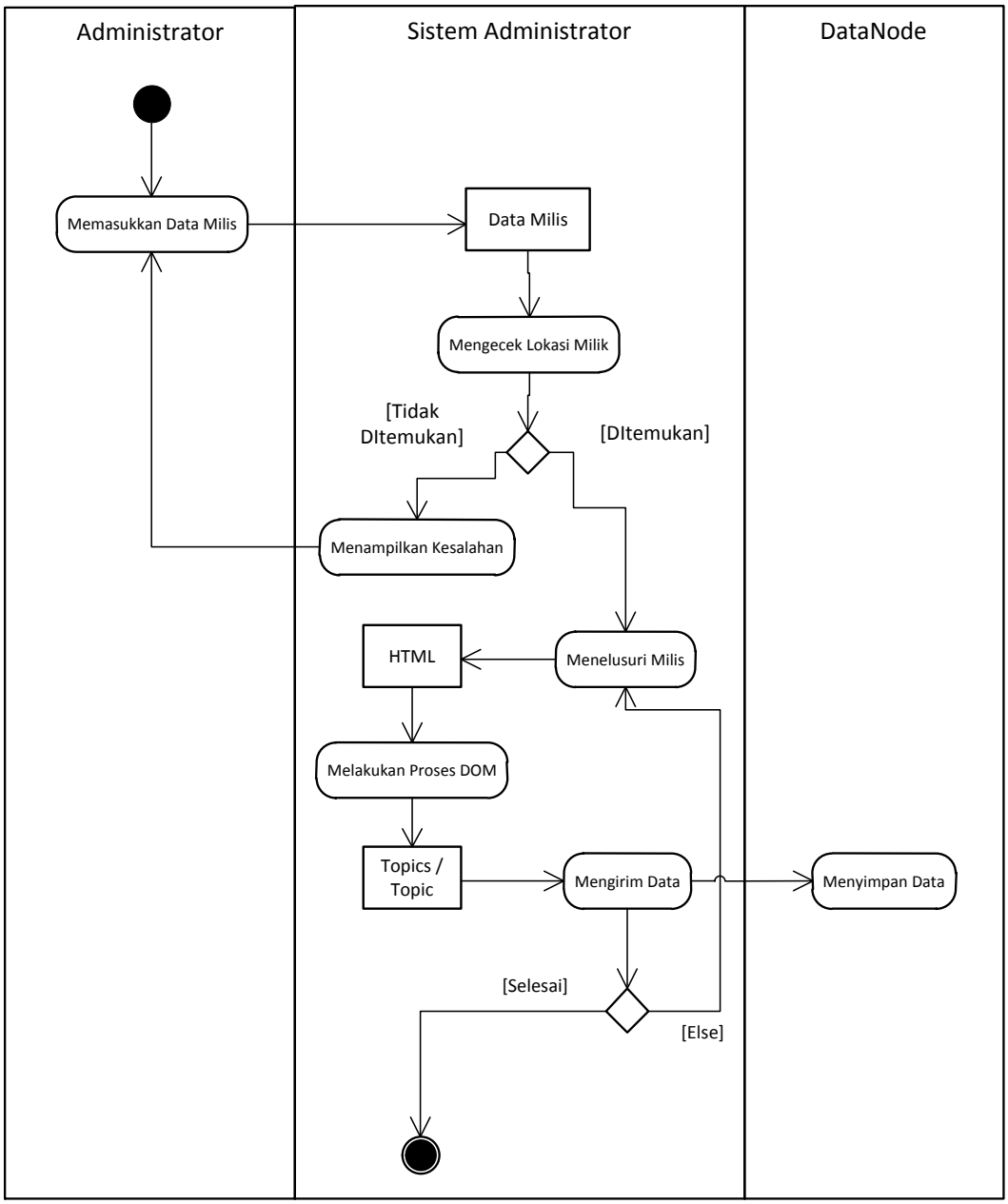
3.2.1.2.3 Penambahan Data

Aktifitas penambahan data merupakan aktifitas yang dilakukan oleh administrator untuk menambahkan data milis yang akan diindeks agar sistem dapat melakukan proses pencarian data. Dalam aktifitas ini terdapat beberapa tahapan aktifitas, yaitu :

1. Memasukkan data milis, dalam aktifitas ini administrator memasukkan data milis yang akan ditambahkan untuk di indeks, agar sistem dapat melakukan proses pencarian terhadap milis tersebut.
2. Mengecek lokasi milis, aktifitas ini merupakan aktifitas yang dilakukan sistem untuk melakukan pengecekan milis yang dimasukkan ada atau tidak ada, jika lokasi milis tidak ditemukan, maka data tidak valid dan sistem akan menampilkan data kesalah, jika ditemukan berarti data valid.

3. Menampilkan kesalahan, pada aktifitas ini sistem menampilkan kesalahan ke administrator dikarenakan data milis yang ditambahkan oleh administrator tidak valid.
4. Menelusuri milis, pada aktifitas ini Sistem Administrator melakukan proses penelusuran terhadap milis, penelusuran tersebut dilakukan untuk mendapatkan berkas-berkas HTML untuk tiap halaman-halaman topik.
5. Melakukan proses DOM, pada aktifitas ini sistem Administrtor merubah halaman HTML yang telah ditelusuri menjadi objek DOM, hal ini dilakukan agar dapat diambil data yang diperlukan dari halaman HTML yaitu topik dan isi pesannya.
6. Mengirim data, dalam aktifitas ini Sistem Administrator mengirimkan data ke *DataNode*. Data yang dikirimkan adalah data topik dan pesan-pesan yang ada dalam milis yang ditelusuri.
7. Menyimpan data, dalam aktifitas ini *DataNode* melakukan proses penyimpanan terhadap seluruh data topik dan tulisan yang ada pada milis yang diberikan oleh Sistem Administrator.

Seluruh tahapan aktifitas penambahan data dapat terlihat pada diagram *Activity* gambar 3.13.



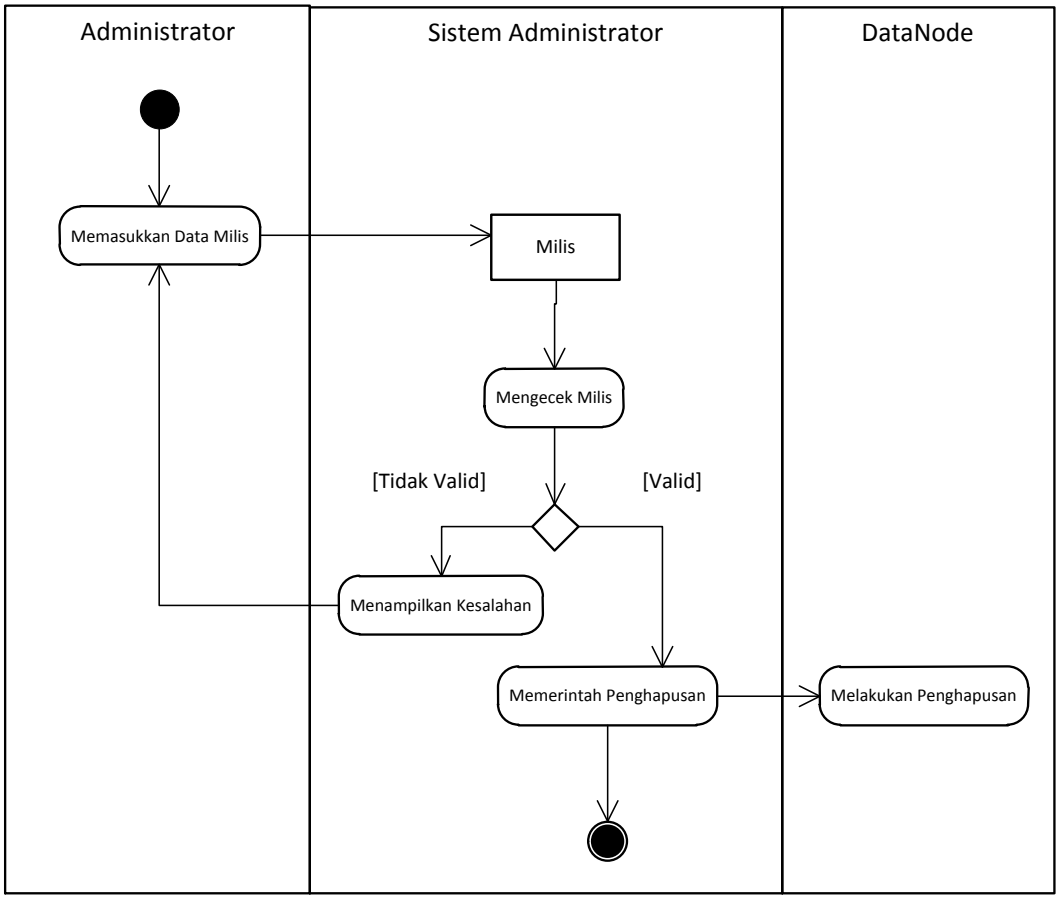
Gambar 3.15 Diagram *Activity* Penambahan Data

3.2.1.2.4 Penghapusan Data

Penghapusan data merupakan aktifitas melakukan penghapusan data milis yang terdapat pada DataNode. Aktifitas ini hanya dapat dilakukan oleh administrator. Dalam aktifitas ini terdapat beberapa tahapan aktifitas, yaitu :

1. Memasukkan data milis, aktifitas ini merupakan aktifitas administrator memasukkan data milis yang akan dihapus.
2. Mengecek milis, aktifitas ini merupakan proses sistem mengecek apakah, data milis yang dimasukkan oleh administrator valid atau tidak, jika valid maka aktifitas penghapusan akan dilakukan, jika tidak maka akan dibatalkan.
3. Menampilkan kesalahan, aktifitas ini merupakan proses sistem menampilkan kesalahan atas ketidak validan data milis yang dimasukkan oleh administrator. Agar proses penghapusan bisa dilakukan, administrator harus mengulangin memasukkan data milis lagi.
4. Memerintah penghapusan, aktifitas ini merupakan aktifitas sistem memerintah seluruh DataNode untuk melakukan proses penghapusan berdasarkan data milis yang telah dimasukkan oleh pengguna.
5. Melakukan penghapusan, aktifitas ini merupakan aktifitas DataNode menghapus seluruh berkas milis yang diperintahkan oleh sistem.

Seluruh tahapan aktifitas penghapusan data, dapat dilihat pada diagram *Activity* gambar 3.14.



Gambar 3.16 Diagram *Activity* Menghapus Data

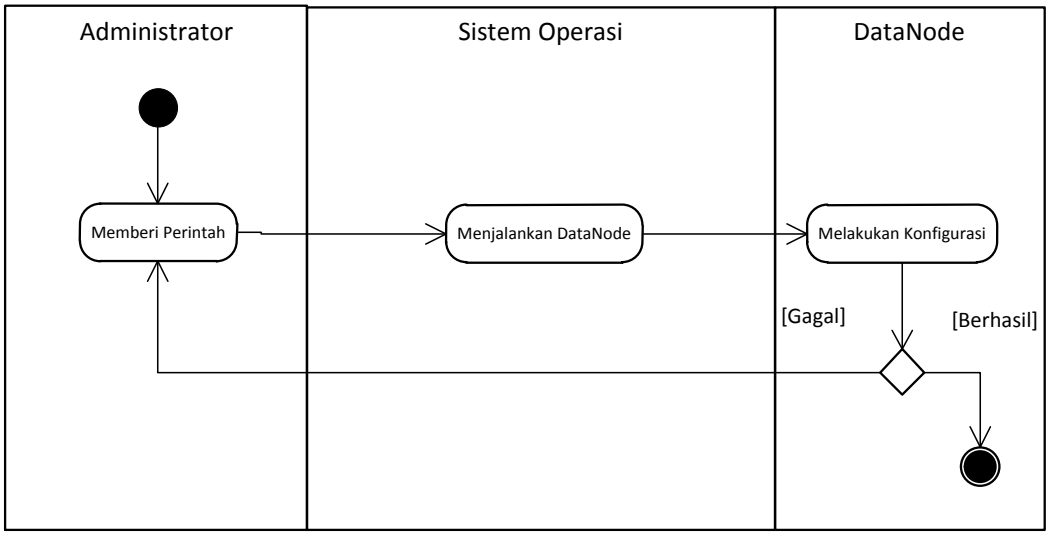
3.2.1.2.5 Menjalankan DataNode

Aktifitas ini merupakan aktifitas administrator menjalankan sistem DataNode, dimana DataNode berperan sebagai sistem yang mengelola berkas hasil dari pengindeksan dan sistem yang bertugas melakukan proses pencarian. Dalam aktifitas ini terdapat beberapa tahapan aktifitas, yaitu :

1. Memberi perintah, aktifitas ini adalah aktifitas administrator untuk memberi perintah sistem operasi agar menjalankan aplikasi DataNode. Aktifitas ini dilakukan oleh administrator dengan cara mengetikkan perintahnnya pada terminal.

2. Menjalankan DataNode, aktifitas ini dilakukan oleh sistem operasi untuk menjalankan DataNode, sesuai dengan yang diperintahkan oleh administrator.
3. Melakukan konfigurasi, aktifitas ini dilakukan oleh DataNode untuk melakukan proses konfigurasi, jika proses konfigurasi berhasil, maka sistem DataNode akan siap berjalan, jika gagal maka sistem DataNode akan mati dan administrator harus mengulangi menjalankannya.

Seluruh tahapan aktifitas menjalankan DataNode dapat dilihat pada diagram *Activity* gambar 3.15.



Gambar 3.17 Diagram *Acticity* Menjalankan DataNode

3.2.1.3 Diagram Sequence

3.2.1.3.1 Pencarian Data

Pencarian data merupakan proses yang dilakukan oleh pengguna untuk mendapatkan informasi yang diperlukan, skenario pencarian data dapat dilihat pada tabel 3.2.

Tabel 3.2 Skenario Pencarian Data

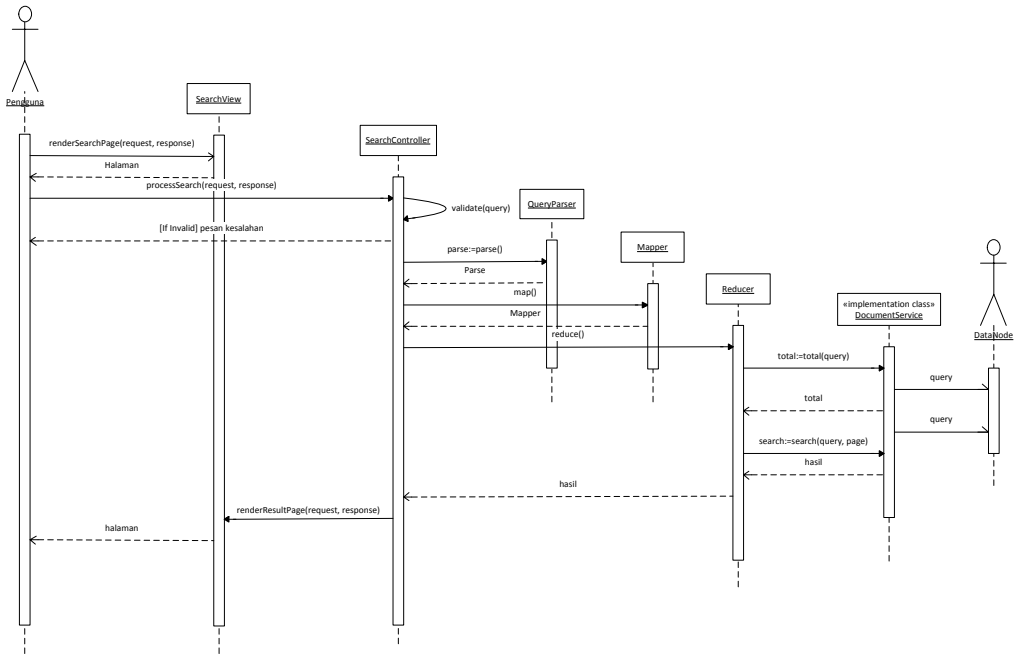
Identifikasi	
Nomor	1
Nama	Pencarian Data
Tujuan	Melakukan proses pencarian informasi yang diperlukan oleh pengguna.
Deskripsi	Pengguna memasukkan *query* yang merupakan kriteria informasi yang akan dicari, setelah itu sistem melakukan proses parsing dan validasi terhadap *query* yang dimasukkan, jika valid maka informasi hasil dari pencarian akan ditampilkan, jika tidak valid maka proses dibatalkan dan akan diberi peringatan kesalahan *query*.
Aktor	Pengguna
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
Pengguna memasukkan data *query*.	Mengirimkan data ke sistem
	Sistem melakukan proses *parsing* terhadap data *query* yang dimasukkan oleh pengguna.
	Sistem melakukan proses validasi terhadap data query, jika data valid maka proses pencarian akan dilakukan namun jika tidak valid maka proses berhenti dan sistem akan memberi peringatan kesalahan.
	Sistem melakukan proses pencarian dengan cara *MapReduce*, melalui DataNode yang ada pada sistem.
	Sistem menampilkan seluruh data yang didapat, jika data yang didapat tidak ada, maka sistem akan menampilkan pesan bahwa data yang dicari tidak ditemukan
Skenario Alternatif – Validasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Pengguna harus melakukan proses yang sama dengan skenario utama.	Sistem akan melakukan reaksi yang sama dengan skenario utama.
Kondisi Akhir	Sistem akan menampilkan informasi yang ditemukan berdasarkan kriteria data *query* yang dimasukkan oleh pengguna.

Interaksi antar kelas yang terjadi saat proses pencarian data adalah sebagai berikut :

1. Pengguna membuka halaman pencarian menggunakan *browser*, lalu *browser* memberi *request* ke sistem.
2. *Request* dari *browser* diterima oleh sistem dan ditangani oleh kelas *SearchView*. *SearchView* menampilkan halaman pencarian dengan menggunakan metode *renderSearchPage()*.
3. Pengguna memasukkan data kriteria yang akan dicari, setelah itu pengguna menekan tombol cari pada halaman pencarian. *Browser* akan memberi *request* ke sistem.
4. *Request* pencarian akan ditangani oleh kelas *SearchController* menggunakan metode *processSearch()*. Kriteria hasil pencarian akan divalidasi oleh kelas *SearchController* menggunaan metode *validate()*, jika data tidak valid, maka sistem akan menampilkan pesan kesalahan, jika valid maka proses pencarian akan dilakukan.
5. Agar kriteria yang diberikan dimengerti oleh sistem, maka sistem melakukan proses parsing terlebih dahulu. Proses parsing dilakukan oleh kelas *QueryParser* menggunakan metode *parse()*.
6. Hasil dari proses parsing merupakan kriteria yang dimengerti oleh sistem, setelah itu sistem melakukan proses pencarian dengan menggunakan metode *MapReduce*. Pencarian *MapReduce* dilakukan mengugnakan kelas *Mapper* dan kelas *Reducer*.

7. Pada saat proses *Reducer*, sistem mengambil data pencarian data DataNode mengugnakan kelas *DocumentService*. Setelah itu hasil dari pencarian diberikan ke kelas *SearchController*.
8. Hasil dari pencarian yang diterima *SearchController* diberikan ke kelas *SearchView* untuk ditampilkan ke pengguna.
9. *SearchView* menampilkan hasil pencarian ke pengguna menggunakan metode *renderResultPage()*.

Seluruh interaksi antar kelas pada saat proses pencarian data dapat dilihat pada diagram *Sequence* gambar 3.16.



Gambar 3.18 Diagram *Sequence* Pencarian Data

3.2.1.3.2 Melihat Topik

Melihat topik merupakan proses yang dilakukan oleh pengguna setelah proses pencarian selesai. Pengguna data memilih topik yang akan dilihat

berdasarkan hasil dari pencaraian yang telah dilakukan. Skenario proses melihat topik dapat dilihat pada tabel 3.3.

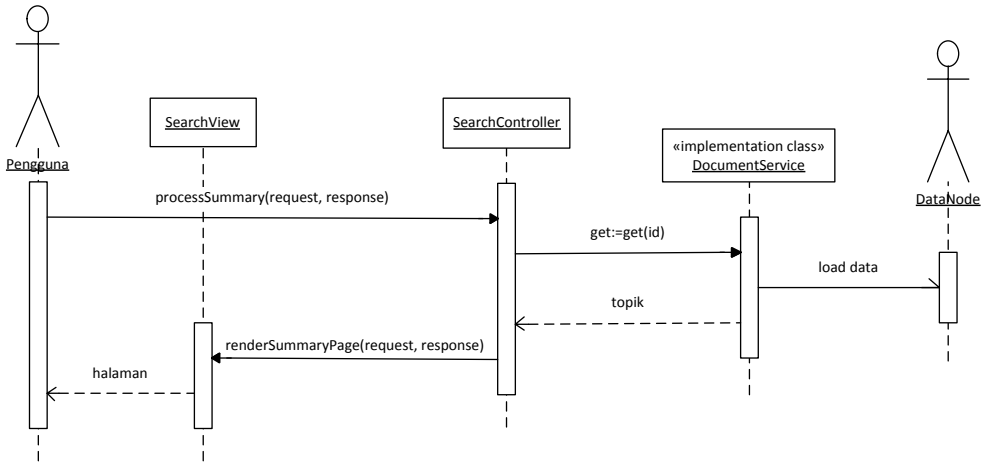
Tabel 3.3 Skenario Melihat Topik

Identifikasi	
Nomor	2
Nama	Melihat Topik
Tujuan	Melihat topik hasil pencarian.
Deskripsi	Pengguna memilih data topik yang akan ditampilkan, lalu sistem akan menampilkan topik yang dipilih.
Aktor	Pengguna
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
Pengguna memilih topik yang akan dilihat.	Mengirim id topik ke sistem
	Sistem meminta data ke DataNode untuk topik yang dipilih.
	DataNode memberikan topik yang diminta oleh sistem.
	Sistem menampilkan topik yang diminta dipilih oleh pengguna.
Kondisi Akhir	Sistem akan menampilkan topik yang dipilih oleh pengguna dari hasil pencarian yang dilakukan.

Interaksi antar kelas yang terjadi pada saat proses melihat topik adalah sebgai berikut :

1. Pengguna melakukan memilih topik yang akan dilihat dengan cara mengklik topik yang akan dilihat pada halaman browser, setelah itu *browser* akan mengirimkan request ke server.
2. Request yang diterima sistem akan di ditangani oleh kelas *SearchController* melalui metode *processSummary()*. SearchController lalu melakukan request ke *DocumentService* untuk meload data topik yang diminta.
3. *DocumentService* mengambil data yang diminta oleh *SearchController* dari *DataNode* menggunakan metode *get()*, lalu memberikan hasil datanya ke *SearchController*.
4. Setelah mendapatkan data topik yang diminta, *SearchController* memberikan data topik tersebut ke *SearchView* untun ditampilkan ke pengguna.
5. *SearchView* menampilkan data topik yang diberikan oleh *SearchController* menggunakan metode *renderSummaryPage()*.

Seluruh interaksi antar kelas pada saat proses melihat topik dapat dilihat pada diagram *Sequence* gambar 3.17.



Gambar 3.19 Diagram *Sequential* Melihat Topik

3.2.1.3.3 Penambahan Data

Penambahan data merupakan proses yang dilakukan oleh administrator untuk menambahkan data milis yang akan di index, dimana milis tersebut merupakan sumber data yang digunakan pada proses pencarian. Skenario proses penambahan data dapat dilihat pada tabel 3.4.

Tabel 3.4 Skenario Penambahan Data

Identifikasi	
Nomor	3
Nama	Penambahan Data
Tujuan	Melakukan penambahan data milis.
Deskripsi	Administrator memasukkan data milis yang akan ditambahkan, jika data milis valid maka sistem akan melakukan proses penelusuran dan pengindeksan, jika tidak valid maka sistem akan menampilkan pesan kesalahan.
Aktor	Administrator
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
Administrator memasukkan data milis.	Membuka konfigurasi, jika konfigurasi benar maka sistem akan meneruskan program, jika konfigurasi gagal maka sistem akan menghentikan program.

	Melakukan validasi terhadap data milis yang dimasukkan oleh administrator. jika data milis yang dimasukkan tidak valid, sistem akan menampilkan pesan kesalahan, jika valid, sistem akan melanjutkan proses penambahan data.
	Sistem melakukan proses penelusuran terhadap data milis yang dimasukkan.
	Jika sistem menemukan pesan pada milis yang sedang ditelusuri, sistem akan menyimpannya ke DataNode.
	Jika sistem menemukan topik pada milis yang sedang ditelusuti, sistem akan mengindeksnya ke DataNode.
Skenario Alternatif – Validasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Administrator harus melakukan proses yang sama dengan skenario utama.	Sistem akan melakukan reaksi yang sama dengan skenario utama.
Skenario Alternatif – Konfigurasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Administrator harus melakukan proses yang sama dengan skenario utama.	Sistem akan melakukan reaksi yang sama dengan skenario utama.
Kondisi Akhir	Sistem akan menampilkan progress proses penambahan data.

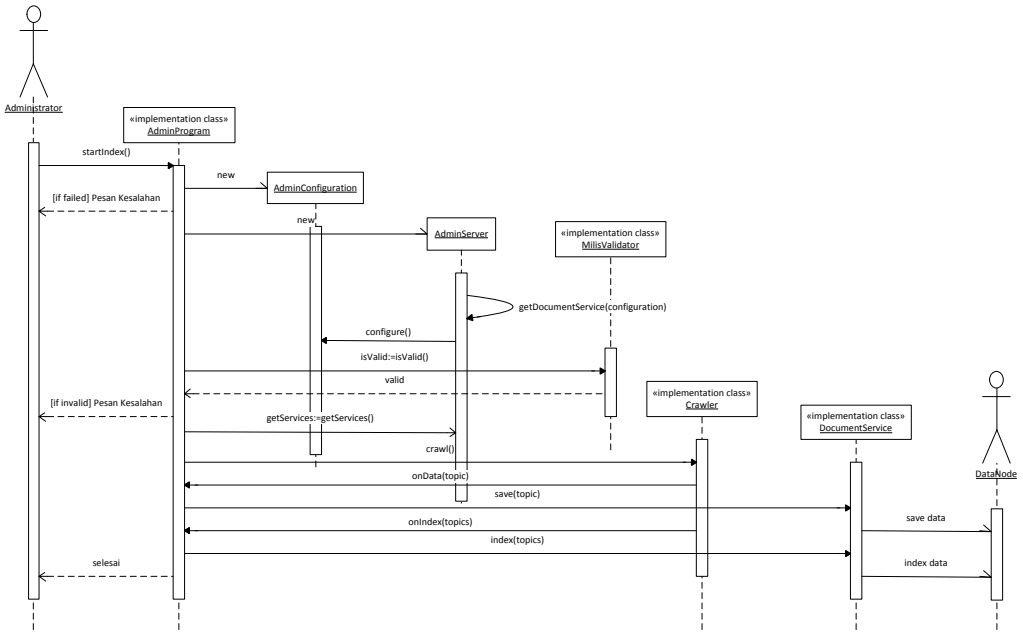
Interaksi antar kelas yang terjadi pada saat proses penambahan data adalah sebagai berikut :

1. Administrator melakukan perintah penambahan data dengan cara mengetikkan perintah pada terminal (console), lalu sistem operasi akan menjalankan sistem administrator.
2. *AdminProgram* merupakan kelas program untuk sistem administrator, saat melakukan proses penambahan data, metode yang pertama kali dipanggil adalah *startIndex()*.

3. *AdminProgram* membuat *AdminConfiguration* untuk melakukan konfigurasi, jika proses konfigurasi gagal, maka sistem akan menghentikan proses dan menampilkan pesan kesalahan
4. *AdminProgram* menjalankan *AdminServer* untuk melakukan koneksi jaringan ke DataNode. AdminServer memanggil metode *getDocumentService()* untuk membaca data configurasi data *AdminConfiguration*.
5. *AdminProgram* melakukan proses validasi terhadap data milis yang akan ditambahkan menggunakan kelas *MilisValidator*. Jika metode *isValid()* menghasilkan nilai *false*, artinya data milis tidak valid dan jika menghasilkan nilai *true*, maka data milis valid.
6. Jika data milis tidak valid, maka *AdminProgram* akan menghentikan proses penambahan data dan memberikan pesan kesalahan kepada Administrator. Jika data milis valid, maka *AdminProgram* akan melakukan proses penelusuran.
7. Proses penelusuran dilakukan menggunakan kelas *Crawler*. Perintah penelurusan akan dilakukan menggunakan metode *crawl()*. Pada saat penelusuran, jika data pesan ditemukan, maka *Crawler* akan memanggil metode *onData()* milik *AdminProgram*, jika data topik ditemukan, maka Crawler akan memanggil metode *onIndex()* milik *AdminProgra*m.
8. Saat metode *onData()* dipanggil, maka *AdminProgram* akan memerintahkan kelas *DocumentService* untuk melakukan proses penyimpanan data menggunakan metode *save()*.

9. Saat metode *onIndex()* dipanggil, maka *AdminProgram* akan memerintahkan kelas *DocumentService* untuk melakukan proses pengindeksan menggunakan metode *index()*.

Seluruh interaksi antar kelas yang terjadi pada saat proses penambahan data dapat dilihat pada diagram *Sequence* gambar 3.18.



Gambar 3.20 Diagram *Sequence* Penambahan Data

3.2.1.3.4 Penghapusan Data

Penghapusan Data merupakan proses administrator menghapus data milis yang telah disimpan. Skenario proses penghapusan data dapat dilihat pada tabel 3.5.

Tabel 3.5 Skenario Penghapusan Data

Identifikasi	
Nomor	4
Nama	Penghapusan Data
Tujuan	Melakukan penghapusan data milis yang telah disimpan.

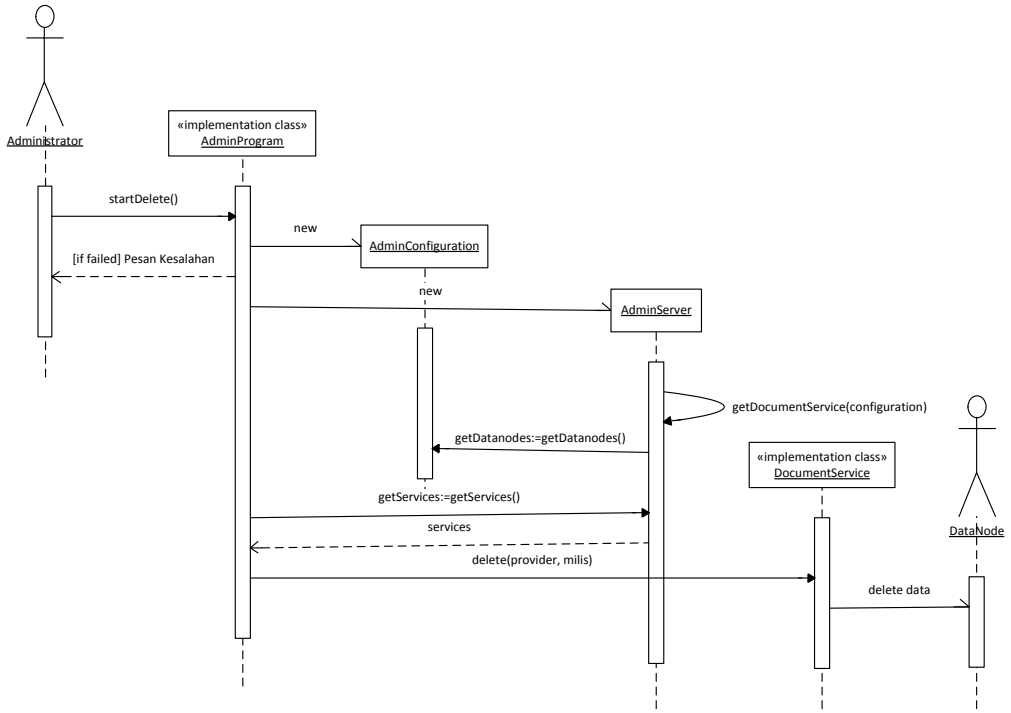
Deskripsi	Administrator memasukkan data milis yang akan dihapus, jika data valid maka sistem akan melakukan proses penghapusan, jika data tidak valid maka sistem akan membatalkan penghapusan.
Aktor	Administrator
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
Administrator memasukkan data milis yang akan dihapus.	Melakukan konfigurasi, jika konfigurasi benar, maka sistem akan melanjutkan program, jika gagal maka sistem akan menghentikan program.
	Melakukan validasi terhadap data milis yang dimasukkan oleh administrator. jika data milis yang dimasukkan tidak valid, sistem akan menampilkan pesan kesalahan, jika valid, sistem akan melanjutkan proses penghapusan data.
	Sistem memerintahkan DataNode untuk melakukan penghapusan data milis yang dimasukkan oleh administrator.
Skenario Alternatif – Validasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Administrator harus melakukan proses yang sama dengan skenario utama.	Sistem akan melakukan reaksi yang sama dengan skenario utama.
Skenario Alternatif – Konfigurasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Administrator harus melakukan proses yang sama dengan skenario utama.	Sistem akan melakukan reaksi yang sama dengan skenario utama.
Kondisi Akhir	Sistem akan menghapus seluruh data milis yang dimasukkan oleh Administrator

Interaksi antar kelas yang terjadi pada saat proses penghapusan data adalah sebagai berikut :

1. Administrator memberi perintah penghapusan data lewat terminal (console), lalu sistem akan menjalankan sistem Administrator lewat kelas *AdminProgram*.

2. Jika proses penghapusan terjadi, maka metode yang pertama kali dipanggil untuk oleh *AdminProgram* adalah metode *startDelete(),* metode ini digunakan untuk memulai penhapusan data.
3. *AdminProgram* melakukan konfigurasi menggunakan kelas *AdminConfiguration*, jika proses konfigurasi berhasil, maka sistem akan melakukan proses penghapusan, jika gagal maka sistem akan berhenti dan memberikan pesan kesalahan.
4. AdminProgram menjalankan AdminServer, kelas tersebut digunakan untuk komunikasi dengan DataNode lewat jaringan. Saat pertama kali dibuat AdminServer akan memanggil metode getDocumentService() dan menggunakan data yang ada di AdminConfiguration melalui metode getDatenodes().
5. *AdminProgram* memerintahkan *DocumentService* yang didapat dari metode *getServices()* milik *AdminServer* untuk melakukan penghapusan data melalui metode *delete()*.
6. *DocumentService* melakukan penghapusan data yang ada dalam DataNode.

Seluruh interaksi antar kelas yang terjadi pada saat proses penghapusan data dapat dilihat pada diagram *Sequence* gambar 3.19.



Gambar 3.21 Diagram Sequential Penghapusan Data

3.2.1.3.5 Menjalankan DataNode

Menjalankan DataNode merupakan proses administrator menghidupkan sistem DataNode yang berguna sebagai sistem pencari dan sistem yang mengolah data. Skenario proses menjalankan *DataNode* dapat terlihat pada tabel 3.6.

Tabel 3.6 Skenario Menjalankan DataNode

Identifikasi	
Nomor	5
Nama	Menjalankan *DataNode*
Tujuan	Menjalankan *DataNode* yang berguna sebagai sistem pencari dan sistem pengelola data.
Deskripsi	Administrator memberi perintah untuk menjalankan sistem *DataNode*, jika konfigurasi benar, maka program akan berjalan jika gagal maka program tidak akan berjalan.
Aktor	Administrator
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**

Administrator memerintahkan untuk menjalankan sistem.	Melakukan konfigurasi, jika konfigurasi benar, maka sistem akan melanjutkan program, jika gagal maka sistem akan menghentikan program.
	Membuat *DocumentService* yang akan disharing melalui jaringan. *DocumentService* akan digunakan untuk berinteraksi dengan klien yaitu web sistem pencarian dan sistem administrator
Skenario Alternatif – Konfigurasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Administrator harus melakukan proses yang sama dengan skenario utama.	Sistem akan melakukan reaksi yang sama dengan skenario utama.
Kondisi Akhir	Sistem *DataNode* akan berjalan dan siap menerima perintah dari sistem administrator dan web sistem pencarian.

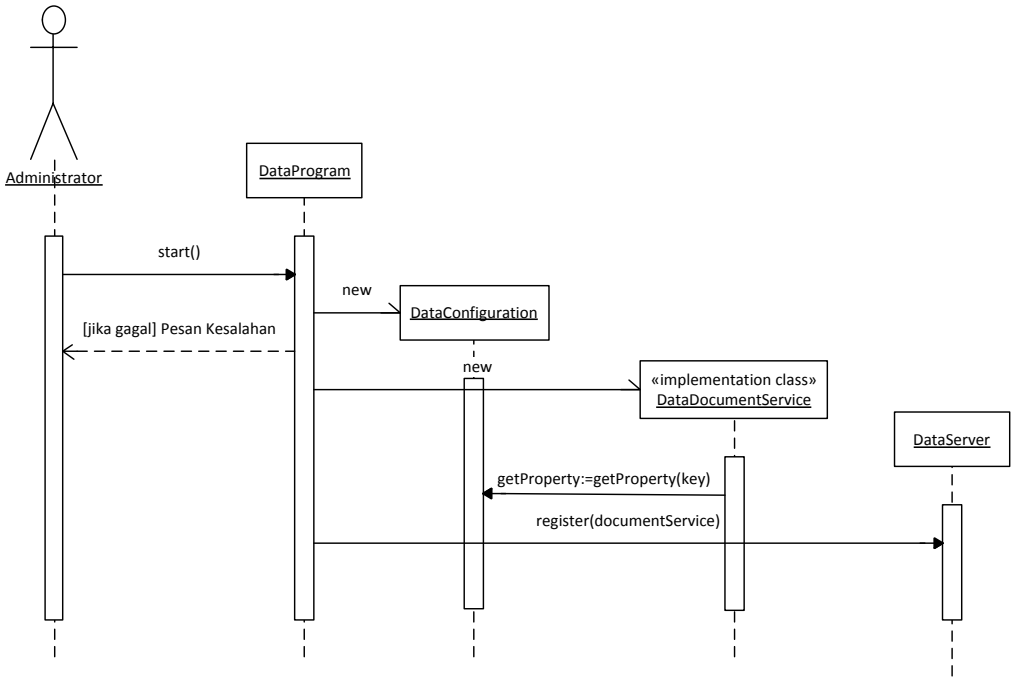
Interaksi antar kelas yang terjadi pada saat prose menjalankan DataNode adalah sebagai berikut :

1. Administrator memberikan perintah untuk menjalankan sistem *DataNode* lewat terminal (console), lalu sistem operasi akan menjalankan sistem *DataNode*.
2. Saat pertama kali sistem *DataNode* berjalan, kelas yang pertama kali dijalankan adalah kelas *DataProgram*, dan metode yang digunakan untuk menjalankan program adalah metode *start()*.
3. Setelah sistem berjalan, *DataProgram* melakukan konfigurasi menggunakan kelas *DataConfiguration.* Jika konfigurasi ada yang salah, maka sistem akan menghentikan program dan memberikan pesan kesalahan. Jika konfigurasi berhasil, maka *DataProgram* akan membuat

DataDocumentService yang akan digunakan sebagai komunikasi dengan klien (Sistem Administrator dan Web Sistem Pencarian).

4. Setelah *DataDocumentService* dibuat, *DataDocumentService* di daftarkan ke *DataServer* agar siap diakses lewat jaringan menggunakan metode *register()*.

Seluruh interaksi antar kelas proses menjalankan *DataNode* dapat dilihat pada diagram *Sequence* gambar 3.20.

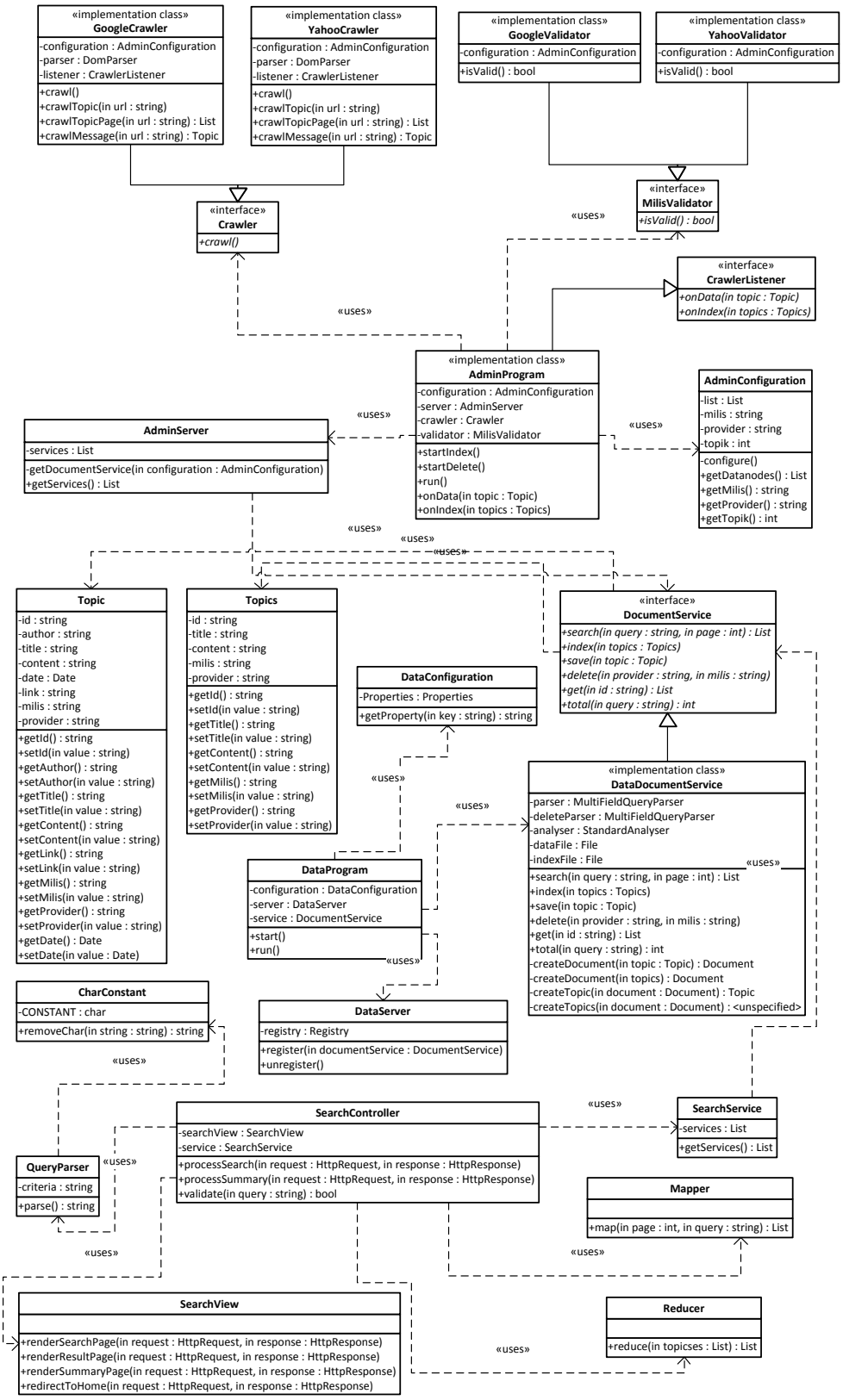


Gambar 3.22 Diagram Sequential Menjalankan DataNode

3.2.1.4 Diagram Class

Terdapat tiga jenis kelas dalam sistem, yaitu kelas *Data*, kelas *Controller* dan kelas *Interface*. Kelas *Data* merupakan kelas yang digunakan sebagai penampung data atau representasi data itu sendiri. Kelas *Controller* merupakan kelas yang digunakan sebagai tempat terjadinya seluruh proses yang ada dalam

sistem, dan kelas *Interface* merupakan kelas yang digunakan untuk membangun tampilan antarmuka sistem. Seluruh kelas-kelas yang ada dalam sistem pencarian dapat dilihat pada gambar 3.21



Gambar 3.23 Diagram *Class* Sistem Pencarian

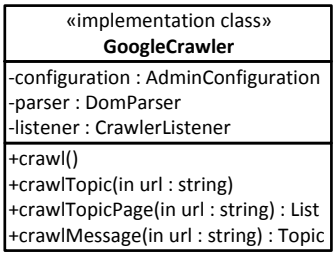
Setiap kelas-kelas pada diagram Class gambar 3.21 memiliki jenis kelasnya masing-masing seperti yang terlihat pada tabel 3.7.

Tabel 3.7 Jenis-Jenis Kelas

Nama Kelas	Jenis Kelas
Crawler	Controller
YahooCrawler	Controller
GoogleCrawler	Controller
Validator	Controller
YahooValidator	Controller
GoogleValidator	Controller
AdminConfiguration	Data
CrawlerListener	Controller
AdminProgram	Controller
AdminServer	Controller
Topic	Data
Topics	Data
DataConfiguration	Data
DocumentService	Controller
DataDocumentService	Controller
DataProgram	Controller
CharConstant	Data
DataServer	Controller
SearchController	Controller
QueryParser	Controller
SearchService	Controller
Mapper	Controller
Reducer	Controller
SearchView	Interface

3.2.1.4.1 Kelas GoogleCrawler

Kelas *GoogleCrawler* merupakan kelas yang digunakan untuk melakukan proses penelusuran terhadap milis-milis yang terdapat pada Google Groups. Seluruh atribut dan operasi yang dimiliki kelas *GoogleCrawler* dapat dilihat pada gambar 3.22.



Gambar 3.24 Kelas *GoogleCrawler*

3.2.1.4.1.1 Deskripsi Atribut

Tabel 3.8 Tabel Deskripsi Atribut Kelas *Google Crawler*

Nama	Tipe	Panjang	Data	Keterangan
configuration	AdminConfiguration	-	Objek	Konfigurasi yang digunakan untuk proses penelusuran
parser	DomParser	-	Objek	DomParser yang digunakan untuk parsing HTML
listener	CrawlerListener	-	Objek	*Trigger* untuk memicu aksi penyimpanan dan pengindeksan

3.2.1.4.1.2 Deskripsi Operasi

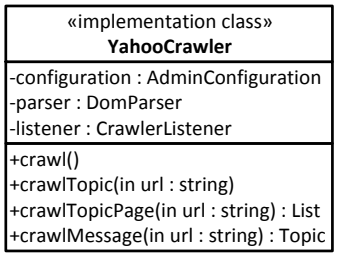
Terdapat empat operasi yang ada pada kelas *GoogleCrawler*, yaitu :

1. *crawl()*, operasi ini digunakan untuk menjalankan proses penelusuran.
2. *crawlTopic(in url : string)*, operasi ini digunakan untuk melakukan penelusuran terhadap topik-topik yang ada pada milis.
3. *crawlTopicPage(in url : string) : List*, operasi ini digunakan untuk menemukan lokasi halaman pesan pada saat penelusuran halaman topik.

4. *crawlMessage(in url : string) : Topic*, operasi ini digunakan untuk mengambil data pesan pada sebuah topik.

3.2.1.4.2 Kelas YahooCrawler

Kelas *YahooCrawler* merupakan kelas yang digunakan untuk melakukan proses penelusuran terhadap milis-milis yang terdapat pada Yahoo Groups. Seluruh atribut dan operasi yang terdapat pada kelas *YahooCrawler* dapat terlihat pada gambar 3.23.



Gambar 3.25 Kelas *YahooCrawler*

3.2.1.4.2.1 Deskripsi Atribut

Tabel 3.9 Deskripsi Atribut Kelas *YahooCrawler*

Nama	Tipe	Panjang	Data	Keterangan
configuration	AdminConfiguration	-	Objek	Konfigurasi yang digunakan untuk proses penelusuran
parser	DomParser	-	Objek	DomParser yang digunakan untuk parsing HTML
listener	CrawlerListener	-	Objek	*Trigger* untuk memicu aksi penyimpanan dan pengindeksan

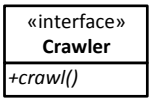
3.2.1.4.2.2 Deskripsi Operasi

Terdapat empat operasi yang ada pada kelas YahooCrawler, yaitu :

5. *crawl()*, operasi ini digunakan untuk menjalankan proses penelusuran.
6. *crawlTopic(in url : string)*, operasi ini digunakan untuk melakukan penelusuran terhadap topik-topik yang ada pada milis.
7. *crawlTopicPage(in url : string) : List*, operasi ini digunakan untuk menemukan lokasi halaman pesan pada saat penelusuran halaman topik.
8. *crawlMessage(in url : string) : Topic*, operasi ini digunakan untuk mengambil data pesan pada sebuah topik.

3.2.1.4.3 Kelas Crawler

Kelas *Crawler* merupakan kelas *prototype* yang digunakan untuk melakukan proses penelusuran. Seluruh atribut dan operasi kelas *Crawler* dapat dilihat pada gambar 3.24.



Gambar 3.26 Kelas Crawler

3.2.1.4.3.1 Deskripsi Atribut

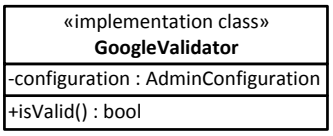
Kelas *Crawler* tidak memiliki atribut.

3.2.1.4.3.2 Deskripsi Operasi

Terdapat satu operasi yang ada pada kelas *Crawler*, yaitu : *crawl()*, operasi ini digunakan untuk menjalankan proses penelusuran.

3.2.1.4.4 Kelas GoogleValidator

Kelas *GoogleValidator* merupakan kelas yang digunakan untuk melakukan proses validasi terhadap data milis yang ada di Google Groups. Seluruh atribut dan operasi yang dimiliki kelas *GoogleValidator* dapat dilihat pada gambar 3.25.



Gambar 3.27 Kelas GoogleValidator

3.2.1.4.4.1 Deskripsi Atribut

Tabel 3.10 Deskripsi Atribut Kelas *GoogleConfiguration*

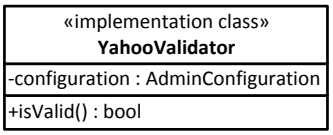
Nama	Tipe	Panjang	Data	Keterangan
configuration	AdminConfiguration	-	Objek	Konfigurasi yang digunakan untuk proses validasi

3.2.1.4.4.2 Deskripsi Operasi

Terdapat satu operasi yang ada pada kelas *GoogelValidator* yaitu : *isValid() : bool*, metode ini digunakan untuk melakukan validasi terhadap data milis, jika valid maka akan menghasilkan nilai *true*, jika tidak valid maka akan menghasilkan nilai *false*.

3.2.1.4.5 Kelas YahooValidator

Kelas *YahooValidator* merupakan kelas yang digunakan untuk melakukan proses validasi terhadap data milis yang ada di Yahoo Groups. Seluruh atribut dan operasi kelas *YahooValidator* dapat dilihat pada gambar 3.26.



Gambar 3.28 Kelas *YahooValidator*

3.2.1.4.5.1 Deskripsi Atribut

Tabel 3.11 Deskripsi Atribut Kelas *YahooValidator*

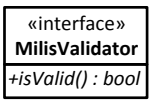
Nama	Tipe	Panjang	Data	Keterangan
configuration	AdminConfiguration	-	Objek	Konfigurasi yang digunakan untuk proses validasi

3.2.1.4.5.2 Deskripsi Operasi

Terdapat satu operasi yang ada pada kelas *YahooValidator* yaitu : *isValid() : bool*, metode ini digunakan untuk melakukan validasi terhadap data milis, jika valid maka akan menghasilkan nilai *true*, jika tidak valid maka akan menghasilkan nilai *false*.

3.2.1.4.6 Kelas MilisValidator

Kelas *MilisValidator* merupakan kelas *prototype* yang digunakan untuk melakukan proses validasi terhadap data milis. Seluruh atribut dan operasi yang ada pada kelas *MilisValidator* dapat dilihat pada gambar 3.27.



Gambar 3.29 Kelas *MilisValidator*

3.2.1.4.6.1 Deskripsi Atribut

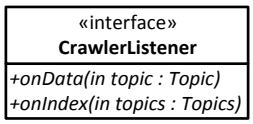
Kelas *MilisValidator* tidak memiliki atribut.

3.2.1.4.6.2 Deskripsi Operasi

Terdapat satu operasi yang ada pada kelas *MilisValidator* yaitu : *isValid() : bool*, operasi ini digunakan untuk melakukan proses validasi terhadap data milis, jika valid maka akan menghasilkan nilai *true*, jika tidak valid maka akan menghasilkan nilai *false*.

3.2.1.4.7 Kelas CrawlerListener

Kelas *CrawlerListener* merupakan kelas yang digunakan sebagai trigger pada saat akan melakukan proses penyimpanan data dan pengindeksan data. Seluruh atribut dan operasi yang terdapat pada kelas *CrawlerListener* dapat dilihat pada gambar 3.28.



Gambar 3.30 Kelas *CrawlerListener*

3.2.1.4.7.1 Deskripsi Atribut

Kelas *CrawlerListener* tidak memiliki atribut.

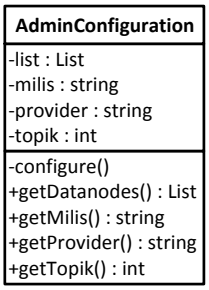
3.2.1.4.7.2 Deskripsi Operasi

Terdapat dua operasi yang ada pada kelas CrawlerListener, yaitu :

1. *onData(in topic : Topic)*, operasi ini dipanggil ketika pada saat proses penelusuran, saat kelas *Crawler* menemukan pesan, metode ini yang akan memberitahukan kelas *AdminProgram* untuk menyimpan data.
2. *onIndex(in topics : Topics)*, operasi ini dipanggil ketika pada saat proses penelusuran, saat kelas *Crawler* menemukan topik, metode ini yang akan memberitahukan kelas *AdminProgram* untuk mengindeks data.

3.2.1.4.8 Kelas AdminConfiguration

Kelas *AdminConfiguration* merupakan kelas yang digunakan untuk menampung seluruh data konfigurasi sistem administrator. Seluruh atribut dan operasi yang terdapat pada kelas *AdminConfiguration* dapat dilihat pada gambar 3.29.



Gambar 3.31 Kelas *AdminConfiguration*

3.2.1.4.8.1 Deskripsi Atribut

Tabel 3.12 Deskripsi Atribut Kelas *AdminConfiguration*

Nama	Tipe	Panjang	Data	Keterangan
list	List	-	Struktur Data List	Kumpulan data lokasi host DataNode
milis	String	-	Seluruh karakter, kecuali *white space*	Id milis
provider	String	-	yahoo, google	Nama provider milis
topik	Int	1-2147483647	-	No topik

3.2.1.4.8.2 Deskripsi Operasi

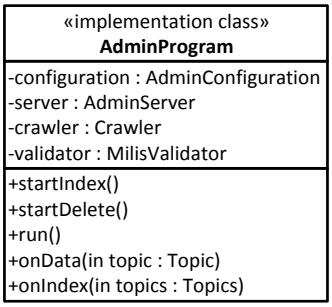
Terdapat lima operasi yang ada pada kelas *AdminConfiguration*, yaitu :

1. *configure()*, operasi ini digunakan untuk melakukan konfigurasi, operasi ini dijalankan saat pertama kali objek AdminConfiguration dibuat.

2. *getDatanodes() : List*, operasi ini digunakan untuk mengambil kumpulan lokasi host *DataNode*.
3. *getMilis() : string*, operasi ini digunakan untuk mengambil data milis.
4. *getProvider() : string*, operasi ini digunakan untuk mengambil data provider milis.
5. *getTopik() : int*, operasi ini digunakan untuk mengambil no topik,

3.2.1.4.9 Kelas AdminProgram

Kelas *AdminProgram* merupakan kelas yang digunakan sebagai program sistem Administrator. Seluruh atribut dan operasi yang terdapat pada kelas *AdminProgram* dapat dilihat pada gambar 3.30.



Gambar 3.32 Kelas *AdminProgram*

3.2.1.4.9.1 Deskripsi Atribut

Tabel 3.13 Deskripsi Atribut Kelas *AdminProgram*

Nama	Tipe	Panjang	Data	Keterangan
configuration	AdminConfiguration	-	Objek	Konfigurasi yang digunakan
server	AdminServer	-	Objek	*Server* yang digunakan
crawler	Crawler	-	Objek	Penelusur
validator	MilisValidator	-	Objek	Digunakan untuk melakukan validasi

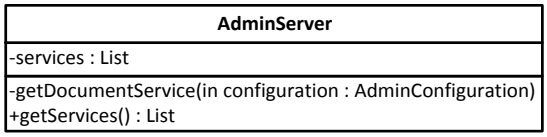
3.2.1.4.9.2 Deskripsi Operasi

Terdapat dua operasi turunan dari kelas *CrawlerListener* yaitu *onData(in topic : Topic)* dan *onIndex(in topics : Topics)*, dan tiga operasi yang dimiliki kelas AdminProgram yaitu :

1. startIndex(), operasi ini digunakan untuk memulai proses penambahan data milis.
2. startDelete(), operasi ini digunakan untuk memulai proses penghapusan data milis.
3. run(), operasi ini dipanggil saat program berhenti, berisikan proses penutupan akses ke jaringan.

3.2.1.4.10 Kelas AdminServer

Kelas AdminServer merupakan kelas yang digunakan sebagai perantara komunikasi antara sistem Administrator dengan DataNode melalui jaringan. Seluruh atribut dan operasi yang terdapat pada kelas *AdminServer* dapat dilihat pada gambar 3.31.



Gambar 3.33 Kelas *AdminServer*

3.2.1.4.10.1 Deskripsi Atribut

Tabel 3.14 Deskripsi Atribut Keals *AdminServer*

Nama	Tipe	Panjang	Data	Keterangan
services	List	-	Struktur data List	Kumpulan DocumentService

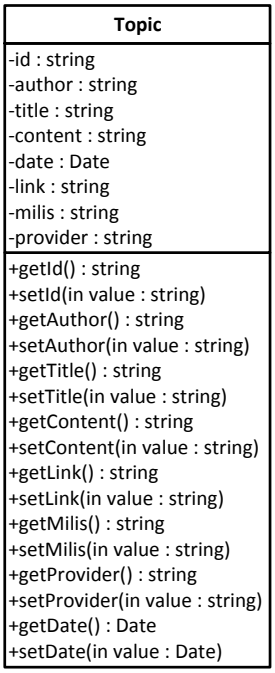
3.2.1.4.10.2 Deskripsi Operasi

Terdapat dua operasi yang ada pada kelas AdminServer yaitu :

1. *getDocumentService(in configuration : AdminCOnfiguration)*, operasi ini digunakan untuk mengambil *DocumentService* yang disharing oleh *DataNode* melalui jaringan.
2. *getServices() : List*, operasi ini digunakan untuk mengambil data *DocumentService*.

3.2.1.4.11 Kelas Topic

Kelas *Topic* merupakan kelas yang merepresentasikan pesan pada milis. Kelas *Topic* merupakan kelas data yang didalamnya terdapat atribut-atribut pesan pada milis. Seluruh atribut dan operas yang terdapat pada kelas *Topic* dapat dilihat pada gambar 3.32.



Gambar 3.34 Kelas *Topic*

3.2.1.4.11.1 Deskripsi Atribut

Tabel 3.15 Deskripsi Atribut Kelas *Topic*

Nama	Tipe	Panjang	Data	Keterangan
id	String	-	UUID	Id pesan
author	String	-	Semua karakter	Pemilik pesan
title	String	-	Semua karakter	Judul pesan
content	String	-	Semua karakter	Isi dari pesan
date	Date	-	Timestamp	Waktu pesan dibuat
link	String	-	Url	Lokasi Url pesan
milis	String	-	Semua karakter kecuali *white space*	Nama milis
provider	String	-	google, yahoo	Provider milis

3.2.1.4.11.2 Deskripsi Operasi

Terdapat enam belas operasi yang ada pada kelas Topic, yaitu :

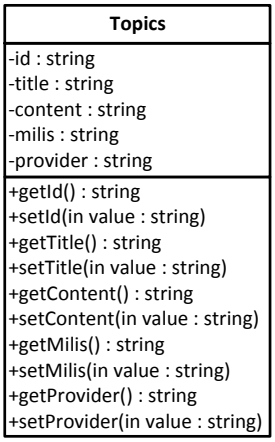
1. *getId() : string*, operasi ini digunakan untuk mendapatkan id pesan.
2. *setId(in id : string)*, operasi ini digunakan untuk mengubah id pesan.
3. *getAuthor() : string*, operasi ini digunakan untuk mendapatkan pembuat pesan.
4. *setAuthor(in author : string)*, operasi ini digunakan untuk mengubah pembuat pesan.
5. *getTitle() : string*, operasi ini digunakan untuk mendapatkan judul pesan.
6. *setTitle(in title : string)*, operasi ini digunakan untuk mengubah judul pesan.

7. *getContent() : string*, operasi ini digunakan untuk mendapatkan isi pesan.
8. *setContent(in content : string)*, operasi ini digunakan untuk mengubah isi pesan.
9. *getLink() : string*, operasi ini digunakan untuk mendapatkan lokasi url pesan.
10. *setLink(in url : string)*, operasi ini digunakan untuk mengubah lokasi url pesan.
11. *getMilis() : string*, operasi ini digunakan untuk mendapatkan nama milis pesan.
12. *setMilis(in milis : string)*, operasi ini digunakan untuk mengubah nama milis pesan.
13. *getProvider() : string*, operasi ini digunakan untuk mendapatkan provider pesan.
14. *setProvider(in provider : string)*, operasi ini digunakan untuk mengubah provider pesan.
15. *getDate() : Date*, operasi ini digunakan untuk mendapatkan waktu dibuat pesan.
16. *setDate(in date : Date)*, operasi ini digunakan untuk mengubah waktu pembuatan pesan.

3.2.1.4.12 Kelas Topics

Kelas Topics merupakan kelas yang digunakan sebagai representasi topik pada milis, dimana dalam setiap topik berisikan satu atau banyak pesan. Seluruh

atribut dan operasi yang terdapat pada kelas *Topics* dapat dilihat pada gambar 3.33.



Gambar 3.35 Kelas *Topics*

3.2.1.4.12.1 Deskripsi Atribut

Tabel 3.16 Deskripsi Atribut Kelas *Topics*

Nama	Tipe	Panjang	Data	Keterangan
id	String	-	UUID	Id topik
title	String	-	Semua karakter	Judul topik
content	String	-	Semua karakter	Isi dari topik
milis	String	-	Semua karakter kecuali *white space*	Nama milis
provider	String	-	google, yahoo	Provider milis

3.2.1.4.12.2 Deskripsi Operasi

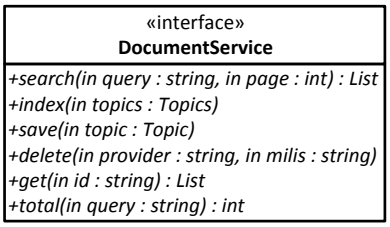
Terdapat sepuluh operasi yang ada pada kelas Topics, yaitu :

1. *getId() : string*, operasi ini digunakan untuk mendapatkan id topik.
2. *setId(in id : string)*, operasi ini digunakan untuk mengubah id topik.
3. *getTitle() : string*, operasi ini digunakan untuk mendapatkan judul topik.

4. *setTitle(in title : string),* operasi ini digunakan untuk mengubah judul topik.
5. *getContent() : string*, operasi ini digunakan untuk mendapatkan isi topik.
6. *setContent(in content : string),* operasi ini digunakan untuk mengubah isi topik.
7. *getMilis() : string*, operasi ini digunakan untuk mendapatkan nama milis topik.
8. *setMilis(in milis : string),* operasi ini digunakan untuk mengubah nama milis topik.
9. *getProvider() : string*, operasi ini digunakan untuk mendapatkan provider topik.
10. *setProvider(in provider : string),* operasi ini digunakan untuk mengubah provider topik.

3.2.1.4.13 Kelas DocumentService

Kelas *DocumentService* merupakan kelas yang digunakan untuk melakukan proses pencarian dan proses pengolaha data milis. *DocumentService* merupakan kelas *prototype* yang nantinya akan diimplementasikan sebagai kelas *DataDocumentService* di *DataNode* dan sibagikan ke klien lewat jaringan. Seluruh atribut dan operasi yang terdapat pada kelas *DocumentService* dapat dilihat pada gambar 3.34.



Gambar 3.36 Kelas *DocumentService*

3.2.1.4.13.1 Deskripsi Atribut

Kelas DocumentService tidak memiliki atribut.

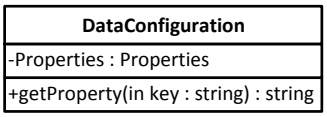
3.2.1.4.13.2 Deskripsi Operasi

Terdapat enam operasi yang terdapat pada kelas DocumentServices, yaitu :

1. *search(in query : string, int page) : List*, merupakan operasi yang digunakan untuk melakukan proses pencairian data, hasil dari operasi ini adalah kumpulan *Topics* yang disatukan dalam *List*.
2. *index(in topics : Topics)*, operasi ini digunakan untuk melakukan proses indeksasi data topik milis ke *DataNode*.
3. *save(in topic : Topic)*, operasi ini digunakan untuk melakukan proses penyimpanan data pesan ke *DataNode*.
4. *delete(in provider : string, in milis : string)*, operasi ini digunakan untuk melakukan proses penghapusan data dari *DataNode*.
5. *get(int uuid : string) : List*, operasi ini digunakan untuk mendapatkan pesan-pesan berdasarkan id pesan tersebut.
6. *total(in string : query) : int*, operasi ini digunakan untuk mendapatkan total halaman untuk sebuah query.

3.2.1.4.14 Kelas DataConfiguration

Kelas *DataConfiguration* merupakan kelas yang digunakan untuk menyimpan data konfigurasi sekaligus melakukan proses konfigurasi pada sistem *DataNode*. Seluruh atribu dan operasi yang terdapat pada kelas *DataConfiguration* dapat dilihat pada gambar 3.35.



Gambar 3.37 Kelas *DataConfiguration*

3.2.1.4.14.1 Deskripsi Atribut

Tabel 3.17 Deskripsi Atribut Kelas *DataConfiguration*

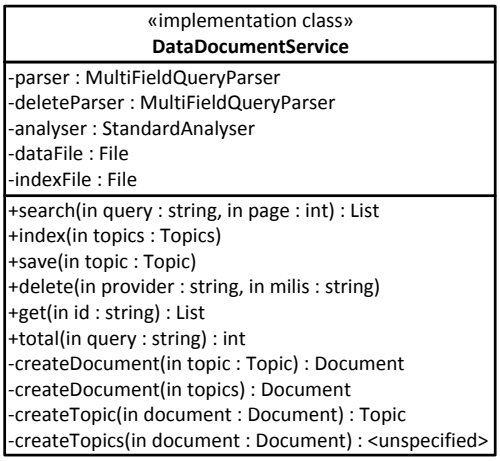
Nama	Tipe	Panjang	Data	Keterangan
properties	Properties	-	Objek	Kumpulan properti berupa *key-value*

3.2.1.4.14.2 Deskripsi Operasi

Terdapat satu operasi yang ada pada kelas DataConfiguration yaitu : *getProperty(in key : string) : string*, operasi ini digunakan untuk mendapatkan nilai properti dari berkas konfigurasi berdasarkan *key*.

3.2.1.4.15 Kelas DataDocumentService

Kelas *DataDocumentService* merupakan kelas implementasi dari kelas prototype *DocumentService*. Kelas ini berisikan seluruh implementasi dari proses pencarian dan pengolahan data untuk sistem *DataNode*. Kelas ini pula yang akan dibagikan lewat jaringan agar dapat diakses oleh sistem Administrator dan web sistem pencarian. Seluruh atribut dan operasi pada kelas *DataDocumentService* dapat dilihat pada gambar 3.36.



Gambar 3.38 Kelas *DataDocumentService*

3.2.1.4.15.1 Deskripsi Atribut

Tabel 3.18 Deskripsi Atribut Kelas *DataDocumentService*

Nama	Tipe	Panjang	Data	Keterangan
parser	MultiFieldQueryParser	-	Objek	Parser untuk proses pencarian
deleteParser	MultiFieldQueryParser	-	Objek	Parser untuk proses penghapusan
analyser	StandardAnalyser	-	Objek	Analyser untuk proses pencarian
dataFile	File	-	Objek	Lokasi berkas yang digunakan untuk menyimpan data
indexFile	File	-	Objek	Lokasi berkas yang digunakan untuk menyimpan indeks

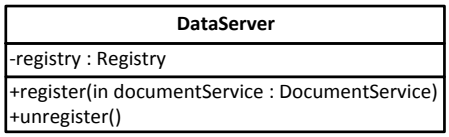
3.2.1.4.15.2 Deskripsi Operasi

Terdapat sepuluh operasi yang ada pada kelas *DataDocumentService*, dimana enam operasi pertama adalah operasi turunan dari kelas *DocumentService*, dan empat yang lainnya adalah operasi milik kelas *DataDocumentService*, yaitu :

1. *createDocument(in topic : Topic) : Document*, operasi ini digunakan untuk mengkonversi *Topic* menjadi *Document*.
2. *createDocument(in topics : Topics) : Document*, operasi ini digunakan untuk mengkonversi *Topics* menjadi *Document*.
3. *createTopic(in document : Document) : Topic*, operasi ini digunakan untuk mengkonversi *Document* menjadi *Topic*.
4. *createTopics(in document : Document) : Topics*, operasi ini digunakan untuk mengkonversi *Document* menjadi *Topics*.

3.2.1.4.16 Kelas DataServer

Kelas DataServer merupakan kelas yang digunakan untuk melakukan proses komunikasi NameNode dengan klien (sistem administrator dan web sistem pencarian) lewat jaringan komputer. Seluruh atribut dan operasi pada kelas *DataServer* dapat dilihat pada gambar 3.37.



Gambar 3.39 Kelas *DataServer*

3.2.1.4.16.1 Deskripsi Atribut

Tabel 3.19 Deskripsi Atribut Kelas *DataServer*

Nama	Tipe	Panjang	Data	Keterangan
registry	Registry	-	Objek	Objek yang

	digunakan sebagai server *Remote Procedural Call*

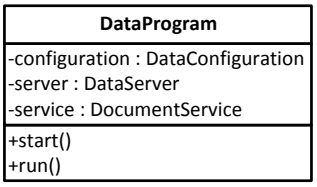
3.2.1.4.16.2 Deskripsi Operasi

Terdapat dua operasi yang ada pada kelas DataServer, yaitu :

1. *register(in documentService : DocumentService)*, operasi ini digunakan untuk melakukan proses berbasi data *DocumentServer* ke jaringan agar dapat diakses secara *Remote Procedural Call*.
2. *unregister()*, operasi ini digunakan menghapus data *DocumentService* dari jaringan, operasi ini dipanggil saat sistem berhenti.

3.2.1.4.17 Kelas DataProgram

Kelas *DataProgram* merupakan kelas yang digunakan sebagai program yang menjalankan sistem *DataNode*. Seluruh atribut dan operasi pada kelas *DataProgram* dapat dilihat pada gambar 3.38.



Gambar 3.40 Kelas *DataProgram*

3.2.1.4.17.1 Deskripsi Atribut

Tabel 3.20 Deskripsi Atribut Kelas *DataProgram*

Nama	Tipe	Panjang	Data	Keterangan
configuration	DataConfiguration	-	Objek	Konfigurasi yang digunakan
server	DataServer	-	Objek	*Server* yang digunakan
service	DocumentService	-	Objek	*Service* yang digunakan sebagai

	Rremote Procedural Call

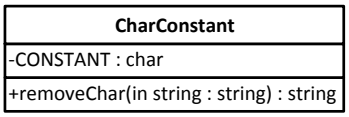
3.2.1.4.17.2 Deskripsi Operasi

Terdapat dua operasi yang ada pada kelas DataProgram, yaitu :

1. *start()*, operasi ini digunakan untuk menjalankan sistem.
2. *run()*, operasi ini dijalankan saat sistem mati.

3.2.1.4.18 Kelas CharConstant

Kelas *CharConstant* merupakan kelas yang berisikan utilitas dan konstanta karakter-karakter. Seluruh atribut dan operasi yang terdapat pada kelas *CharConstant* dapat dilihat pada gambar 3.39.



Gambar 3.41 Kelas *CharConstant*

3.2.1.4.18.1 Deskripsi Atribut

Tabel 3.21 Deskripsi Atribut Kelas *CharConstant*

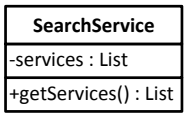
Nama	Tipe	Panjang	Data	Keterangan
CONSTANT	Char[]	-	Karakter Unique	Kumpulan karakter unique yang digunakan saat proses parsing

3.2.1.4.18.2 Deskripsi Operasi

Terdapat satu operasi yang ada pada kelas *CharConstant*, yaitu : *removeChar(in string : string) : string*, operasi ini digunakan untuk menghapus karakter unik pada string, operasi ini menghasilkan string yang sudah dihapus karakter uniknya.

3.2.1.4.19 Kelas SearchService

Kelas *SearchService* merupakan kelas yang digunakan untuk berkomunitaksi dengan *DataNode* lewat jaringan menggunakan *Remote Procedural Call*. Seluruh atribut dan operasi kelas *SearchService* dapat dilihat pada gambar 3.40.



Gambar 3.42 Kelas *SearchService*

3.2.1.4.19.1 Deskripsi Atribut

Tabel 3.22 Deskripsi Atribut Kelas *SearchService*

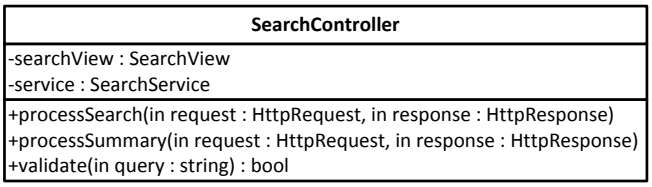
Nama	Tipe	Panjang	Data	Keterangan
service	List	-	Struktur data List	Berisikan *DocumentService* yang didapat dari *NameNode*.

3.2.1.4.19.2 Deskripsi Operasi

Terdapat satu operasi yang ada pada kelas *SearchService*, yaitu : *getServices() : List*, operasi ini digunakan mendapatkan *DocumentService* yang ada pada *DataNode*.

3.2.1.4.20 Kelas SearchController

Kelas *SearchController* merupakan kelas yang digunakan untuk menangani *request* yang diberikan oleh pengguna melalui *browser*. *SearchController* merupakan kelas yang menangani permintaan proses pencarian dan proses melihat topik. Seluruh atribut dan operasi kelas *SearchController* dapat dilihat pada gambar 3.41.



Gambar 3.43 Kelas *SearchController*

3.2.1.4.20.1 Deskripsi Atribut

Tabel 3.23 Deskripsi Atribut Kelas *SearchController*

Nama	Tipe	Panjang	Data	Keterangan
searchView	SearchView	-	Objek	Objek yang digunakan untuk menampilkan tampilan antarmuka
service	SearchService	-	Objek	Objek yang digunakan untuk berkomunikasi dengan *DataNode*.

3.2.1.4.20.2 Deskripsi Operasi

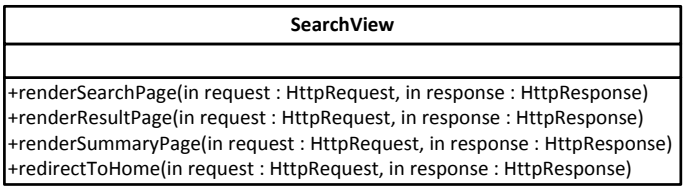
Terdapat tiga operasi yang ada pada kelas *SearchController*, yaitu :

1. *processSearch(in request : HttpRequest, in response : HttpResponse)*, operasi ini digunakan untuk menangani rekues yang diberikan pengguna untuk melakukan prose pencarian.
2. *processSummary(in request : HttpRequest, in response : HttpResponse)*, operasi ini digunakan untuk menangani rekues yang diberikan pengguna untuk melihat topik.
3. *validate(in query : string) : bool*, operasi ini digunakan untuk melakukan proses validasi terhadap data query yang dimasukkan pengguna untuk

melakukan proses pencarian. Jika validasi berhasil maka akan menghasilkan nilai true, jika gagal maka akan menghasilkan nilai false.

3.2.1.4.21 Kelas SearchView

Kelas *SearchView* merupakan kelas yang digunakan untuk menampilkan halaman antarmuka ke pengguna, baik itu antarmuka pencarian, hasil pencarian dan melihat topik. Seluruh atribut dan operasi yang terdapat pada kelas *SearchView* dapat dilihat pada gambar 3.42.



Gambar 3.44 Kelas SearchView

3.2.1.4.21.1 Deskripsi Atribut

Kelas *SearchView* tidak memiliki atribut.

3.2.1.4.21.2 Deskripsi Operasi

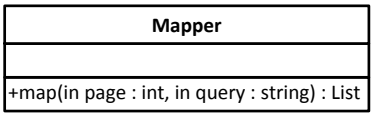
Terdapat empat operasi yang terdapat pada kelas SearchView, yaitu :

1. *renderSearchPage(in request : HttpRequest, in response : HttpResponse)*, operasi ini digunakan untuk menampilkan antarmuka halaman pencarian.
2. *renderResultPage(in request : HttpRequest, in response : HttpResponse)*, operasi ini digunakan untuk menampilkan antarmuka halaman hasil pencarian.
3. *renderSummaryPage(in request : HttpRequest, in response : HttpResponse)*, operasi ini digunakan untuk menampilkan antarmuka halaman topik.

4. *redirectToHome(in request : HttpRequest, in response : HttpResponse)*, operasi ini digunakan untuk me-*redirect* halaman ke halaman pencarian.

3.2.1.4.22 Kelas Mapper

Kelas Mapper merupakan melas yang merepresentasikan proses *Map* pada proses *MapReduce*. Seluruh atribut dan operasi yang terdapat pada keals *Mapper* dapat dilihat pada gambar 3.43.



Gambar 3.45 Kelas *Mapper*

3.2.1.4.22.1 Deskripsi Atribut

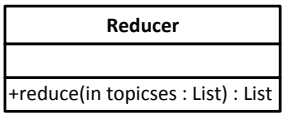
Kelas Mapper tidak memiliki atribut.

3.2.1.4.22.2 Deskripsi Operasi

Terdapat satu operasi pada kelas *Mapper* yaitu; *map(in page : int, in query : string) : List*, operasi ini digunakan untuk melakukan tahapan *Map* dalam proses *MapReduce*, dalam tahapan *Map* ini sistem pencarian memerintahkan seluruh *DataNode* untuk melakukan pencarian.

3.2.1.4.23 Kelas Reducer

Kelas *Reducer* merupakan kelas yang merepresentasikan tahapan *Reduce* pada proses *MapReduce*. Pada kelas *Reducer*, seluruh hasil pencaraian dari kelas *Map* akan digabungkan. Seluruh atribut dan operasi yang terdapat pada kelas *Reducer* dapat dilihat pada gambar 3.44.



Gambar 3.46 Kelas Reducer

3.2.1.4.23.1 Deskripsi Atribut

Kelas Reducer tidak memiliki atribut.

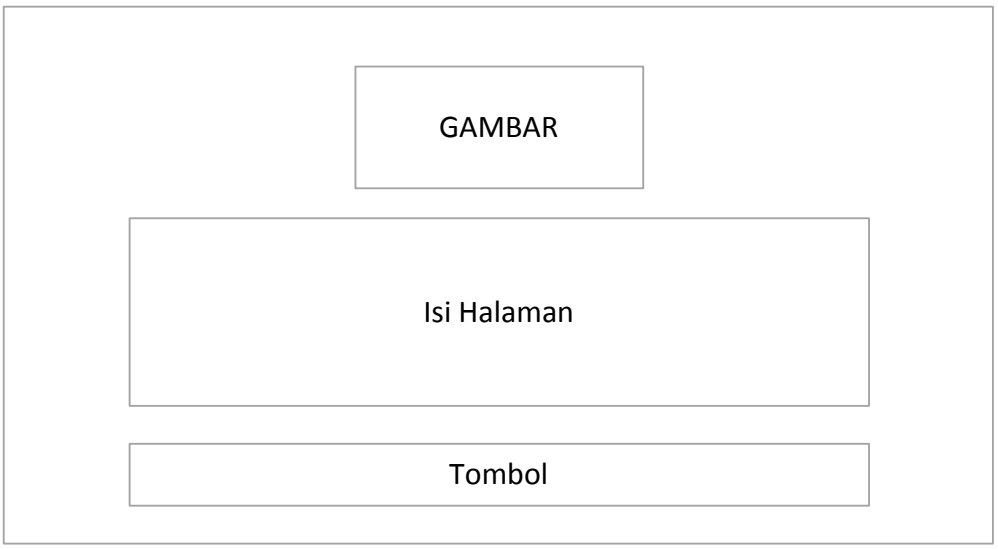
3.2.1.4.23.2 Deskripsi Operasi

Terdapat satu operasi pada kelas *Reducer* yaitu; *reduce(in topicses : List) : List*, operasi ini digunakan untuk melakukan tahapan *Reduce* pada proses *MapReduce*, pada tahapan ini seluruh *Topics* hasil pencarian dari proses *Map* digabungkan.

3.2.2 Perancangan Antarmuka

3.2.2.1 Deskripsi Layar

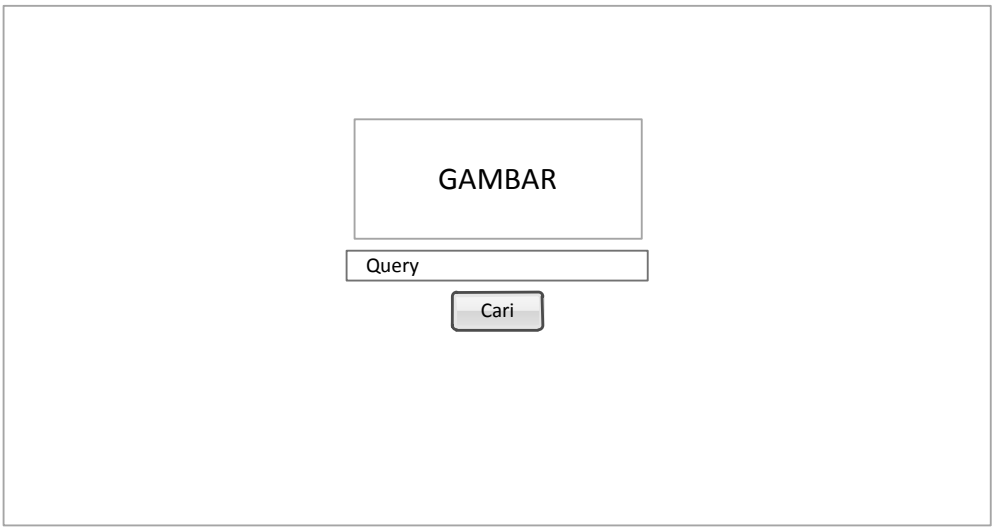
Secara umum, layout sistem pencarian yang akan dibangun akan seperti gambar 3.24.



Gambar 3.47 Deskripsi Layar Sistem Pencarian

Pada bagian *Gambar*, akan berisi gambar atau logo sistem. *Isi Halaman* akan berisikan isi dari halaman yang ditampilkan, seperti *form* pencarian data, ataupun hasil pencarian data. Bagian *Tombol* akan berisikan tombol-tombol yang digunakan dalam halaman sistem pencarian.

3.2.2.2 Antarmuka Halaman Pencarian



Gambar 3.48 Antarmuka Halaman Pencarian

3.2.2.2.1 Deskripsi Objek

Tabel 3.24 Deskripsi Objek Antarmuka Halaman Pencarian

Objek	Jenis	Keterangan
Gambar	Image	Gambar logo website
Teks	TextField	Masukkan untuk query pencarian
Cari	Button	Tombol untuk aksi pencarian

Jika pada bagian *Gambar* di klik maka, sistem akan menampilkan antarmuka *Halaman Pencarian*. Jika tombol *Cari* diklik, maka sistem akan melakukan proses pencarian, jika pada proses pencarian tersebut query yang dimasukkan dalam *Teks* valid, maka sistem akan menampilkan antarmuka

Halaman Hasil Pencarian, namun jika tidak valid, maka sistem akan menampilkan antarmuka *Halaman Kesalahan Pencarian*.

3.2.2.2.2 Algoritma

Jika tombol *Cari* di klik, maka sistem akan melakukan pencarian dengan menjalankan proses algoritma sebagai berikut :

```
// ambil data query
hasil query = request.getParameter("query");

// validasi query
if(query is valid){

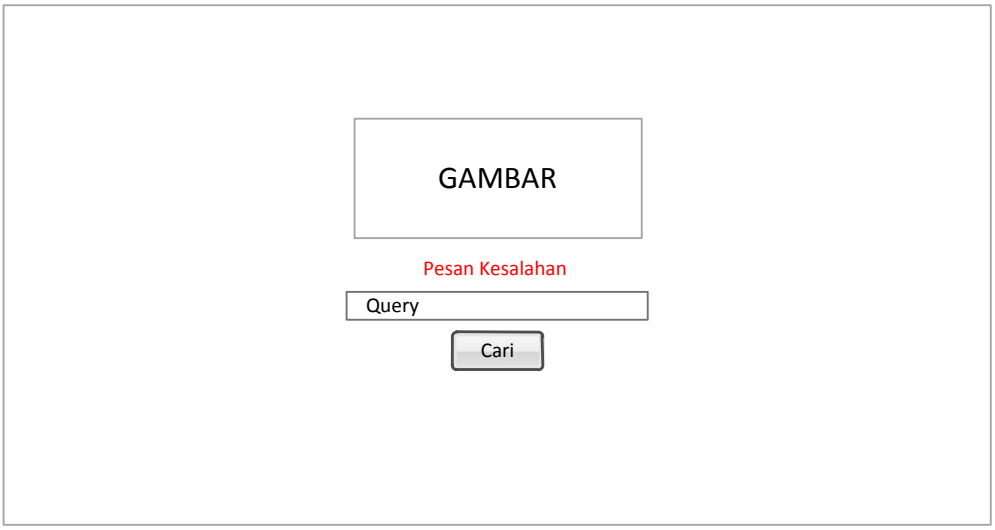
   // jika query valid, lakukan pencarian
   hasil = search(string query)
   // tampilkan hasil pencarian
   view.renderResult(hasil);

}else {

   // jika query tidak valid tampilkan error
   View.renderSearch()

}
```

3.2.2.3 Antarmuka Kesalahan Pencarian



Gambar 3.49 Antarmuka Kesalahan Pencarian

3.2.2.3.1 Deskripsi Objek

Tabel 3.25 Deskripsi Objek Antarmuka Kesalahan Pencarian

Objek	Jenis	Keterangan
Gambar	Image	Gambar logo website.
Teks	TextField	Masukkan untuk query pencarian.
Cari	Button	Tombol untuk aksi pencarian.
Pesan Kesalahan	Text	Pesan kesalahan dari sistem.

Jika pada bagian *Gambar* di klik, maka sistem akan menampilkan antarmuka *Halaman Pencarian*. Jika tombol *Cari* diklik, maka sistem akan melakukan proses pencarian, jika pada proses pencarian tersebut query yang dimasukkan dalam *Teks* valid, maka sistem akan menampilkan antarmuka *Halaman Hasil Pencarian*, namun jika tidak valid, maka sistem akan menampilkan antarmuka *Halaman Kesalahan Pencarian*.

3.2.2.3.2 Algoritma

Jika tombol Cari diklik, maka sistem akan melakukan proses pencarian dengan cara menjalankan algoritma sebagai berikut :

```
// ambil data query
hasil query = request.getParameter(“query”);

// validasi query
if(query is valid){

   // jika query valid, lakukan pencarian
   hasil = search(query);
   // tampilkan hasil pencarian
   searchView.renderInformation(hasil);

}else {

   // jika query tidak valid
   searchView.displayError(“pesan”, request, resp);

}
```

3.2.2.4 Antarmuka Hasil Pencarian



Gambar 3.50 Antarmuka Hasil Pencarian

3.2.2.4.1 Deskripsi Objek

Tabel 3.26 Deskripsi Objek Antarmuka Hasil Pencarian

Objek	Jenis	Keterangan
Gambar	Image	Gambar logo website.
Query	TextField	Text masukkan query pencarian
Cari	Button	Tombol yang digunakan untuk melakukan pencarian
Judul Topik	Link	Judul topik yang ditemukan sesuai dengan pencarian
Isi Topik	Text	Isi sebagian dari topik hasil pencarian
Halaman	Link	Link ke halaman selanjutnya atau sebelumnya

Jika pada bagian *Gambar* di klik, maka sistem akan menampilkan antarmuka *Halaman Pencarian*. Jika tombol *Cari* diklik, maka sistem akan melakukan proses pencarian, jika pada proses pencarian tersebut query yang dimasukkan dalam *Teks* valid, maka sistem akan menampilkan antarmuka *Halaman Hasil Pencarian*, namun jika tidak valid, maka sistem akan menampilkan antarmuka *Halaman Kesalahan Pencarian*. Jika pada bagian link *Halaman* di klik, maka sistem akan menampilkan antarmuka *Halaman Hasil Pencarian* dengan data seuai dengan halaman.

3.2.2.4.2 Algoritma

Jika tombol *Cari* diklik, maka sistem akan melakukan proses pencarian dengan cara menjalankan algoritma sebagai berikut :

```
// ambil data query
hasil query = request.getParameter(“query”);

// validasi query
if(query is valid){

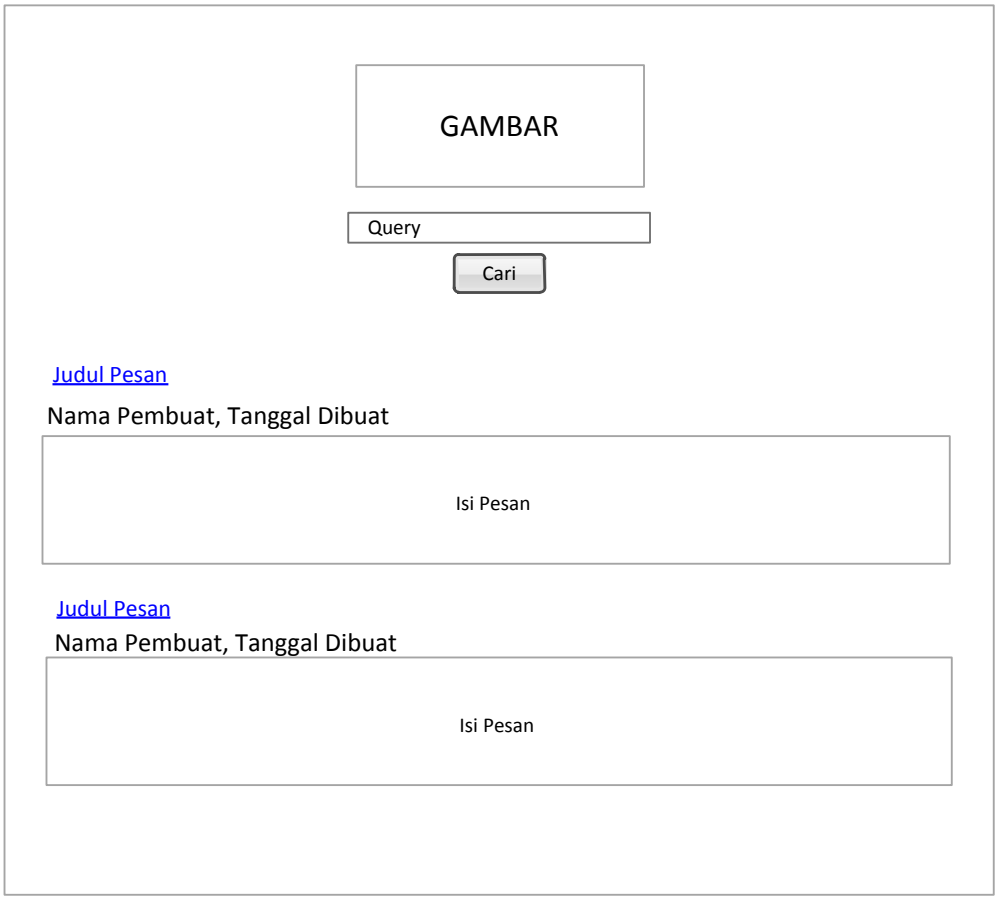
   // jika query valid, lakukan pencarian
   hasil = search(query);
   // tampilkan hasil pencarian
   view.renderResult(hasil);

}else {

   // jika query tidak valid
   View.renderSearch();

}
```

3.2.2.5 Antarmuka Halaman Topik



Gambar 3.51 Antarmuka Halaman Topik

3.2.2.5.1 Deskripsi Objek

Tabel 3.27 Deskripsi Objek Antarmuka Halaman Topik

Objek	Jenis	Keterangan
Gambar	Image	Gambar logo website.
Query	TextField	Text masukkan query pencarian
Cari	Button	Tombol yang digunakan untuk melakukan pencarian
Judul Pesan	Link	Judul pesan yang ada pada topik
Isi Pesan	Text	Isi pesan
Nama Pembuat	Text	Nama pembuat pesan
Tanggal Dibuat	Text	Waktu pembuatan pesan

Jika pada bagian *Gambar* di klik, maka sistem akan menampilkan antarmuka *Halaman Pencarian*. Jika tombol *Cari* diklik, maka sistem akan melakukan proses pencarian, jika pada proses pencarian tersebut query yang dimasukkan dalam *Teks* valid, maka sistem akan menampilkan antarmuka *Halaman Hasil Pencarian*, namun jika tidak valid, maka sistem akan menampilkan antarmuka *Halaman Kesalahan Pencarian*. Jika pada bagian *Judul Pesan* diklik, maka *browser* akan membuka lokasi url pesan tersebut berada (yaitu pada milis).

3.2.2.5.2 Algoritma

Jika tombol Cari di klik maka sistem akan melakukan pencarian dengan cara menjalankan algoritma sebagai berikut :

```
// ambil data query
hasil query = request.getParameter("query");

// validasi query
if(query is valid){

   // jika query valid, lakukan pencarian
   hasil = search(query);
   // tampilkan hasil pencarian
   view.renderResult(hasil);

}else {

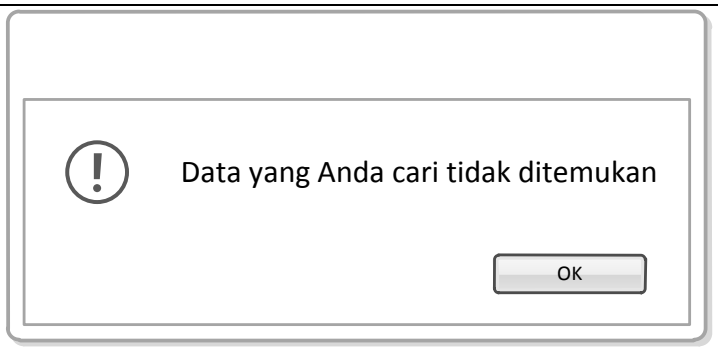
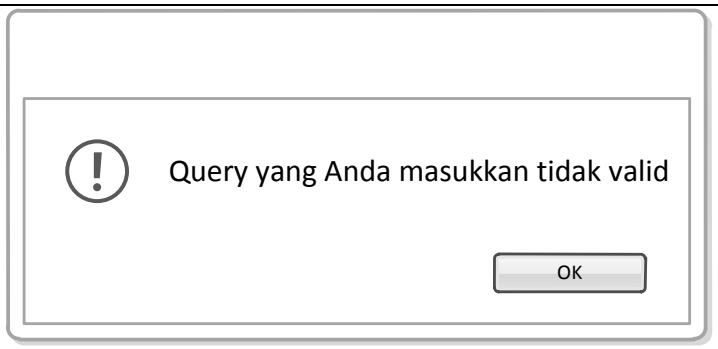
   // jika query tidak valid
   View.renderSearch();

}
```

3.2.2.6 Antarmuka Peringatan Kesalahan

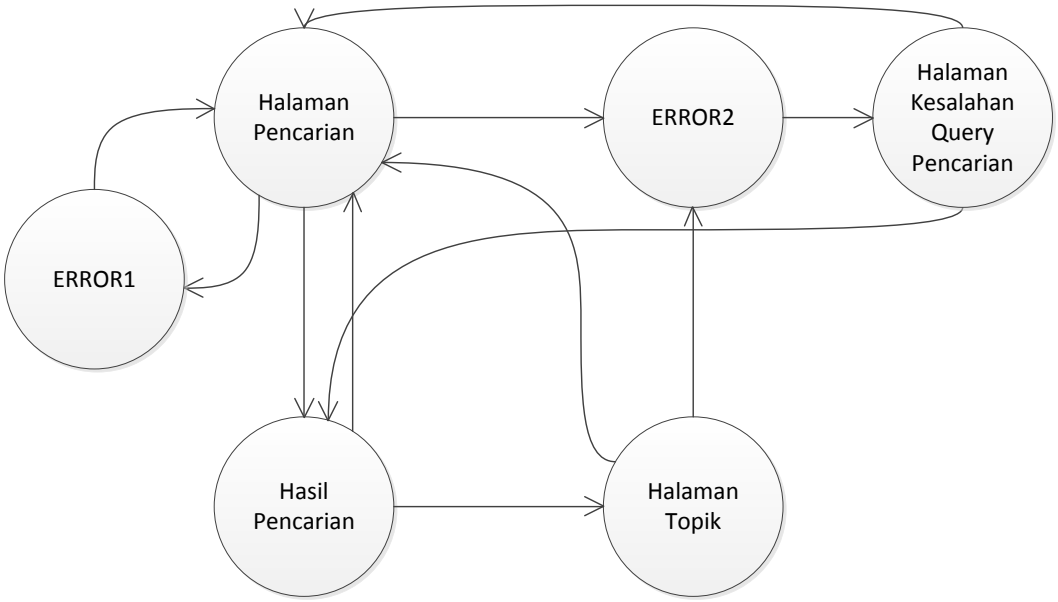
Terdapat beberapa tampilan antarmuka kesalahan yang akan ditampilkan pada sistem pencarian.

Kode Pesan	Antarmuka Kesalahan

ERROR1	 **Gambar 3.52 Antarmuka Kesalahan Data yang Anda cari tidak ditemukan**
ERROR2	 **Gambar 3.53 Antarmuka Kesalahan Query yang Anda masukkan tidak valid**

3.2.3 Jaringan Semantik

Alur antarmuka yang terdapat pada sistem pencaraian, dapat digambarkan dalam jaringan pada gambar 3.54.



Gambar 3.54 Jaringan Semantik Sistem Pencarian