

LAPORAN KEGIATAN MAGANG MAHASISWA

IMPLEMENTATION OF APACHE SOLR FOR DATA INGESTION IN BIG DATA

VISUALIZATION APPLICATION

Diajukan untuk Memenuhi Sebagian Persyaratan Memperoleh Kelulusan Matakuliah

Kegiatan Magang Mahasiswa



Disusun oleh :

LIA RISTIANA

NIM. M0513027

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SEBELAS MARET

2016

HALAMAN PERSETUJUAN

IMPLEMENTATION OF APACHE SOLR FOR DATA INGESTION IN BIG DATA

VISUALIZATION APPLICATION

Disusun Oleh :

Lia Ristiana

NIM. M0513027

Laporan Kegiatan Magang Mahasiswa ini disetujui untuk dipresentasikan pada
seminar KMM

Pada tanggal 23 Desember 2016

Dosen Pembimbing



Afrizal Doewes, S.Kom., M.Sc

NIP. 19850831 201212 1 004

Pembimbing KMM



Arief Dolants

Technical Leader & System

Analyst

Solusi 247, Jogja Branch

HALAMAN PENGESAHAN

Laporan Kegiatan Magang Mahasiswa yang dilaksanakan oleh :

Nama : Lia Ristiana

NIM : M0513027

Dengan Judul :

IMPLEMENTATION OF APACHE SOLR FOR DATA INGESTION IN BIG DATA

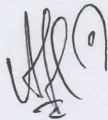
VISUALIZATION APPLICATION

Pada bulan Juli-Agustus 2016, diseminarkan dan disahkan pada :

Hari : Jumat

Tanggal : 23 Desember 2016

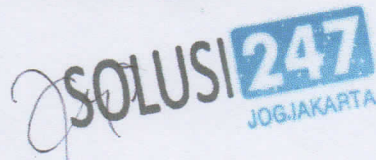
Dosen Pembimbing



Afrizal Doewes, S.Kom., M.Sc

NIP. 19850831 201212 1 004

Pembimbing KMM



Arief Dolants

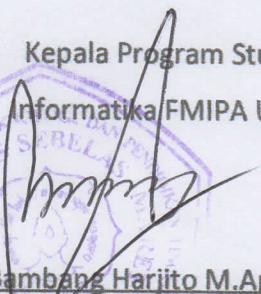
Technical Leader & System

Analyst

Solusi 247, Jogja Branch

Kepala Program Studi

Informatika FMIPA UNS



Drs. Bambang Harijito M.App.Sc, PhD

NIR. 19621130 199103 1 002

IMPLEMENTATION OF APACHE SOLR FOR DATA INGESTION IN BIG DATA VISUALIZATION APPLICATION

Lia Ristiana

NIM. M0513027

Abstract

In this data-driven era, the needs for data are not limited to only being stored, but also being accessed and analyzed for a variety of purposes. With the development of technology, data generated in the world everyday are getting bigger and more complex, thus the term big data. *Solusi247* as a leading big data company in Indonesia is motivated to develop an application called *BaciroGateway*, a dashboard to monitor and configure *BaciroBackend* service. In this internship, we add a new functionality to *BaciroGateway* called *Analytics*, a menu which deals with retrieving, storing, accessing, and visualizing big data by utilizing a number of big data technology such as *Hadoop*, *Solr*, *Flume*, etc. The development of the *Analytics* menu is done by employing *AngularJS* for the application interface and *NodeJS* for the application server. One of the sub menu of *Analytics* is *Ingestion*, developed for data ingestion to *Solr*, an open search platform built upon a Java library called Lucene. *Ingestion* is developed with the aim that users can utilize *Solr* functionalities such as creating core, creating schema for core, and submitting data to *Solr* through *BaciroGateway* interface. All the queries and operations sent to *Solr* are done by accessing *Solr HTTP API*.

Keyword: Apache Solr, Big Data

IMPLEMENTATION OF APACHE SOLR FOR DATA INGESTION IN BIG DATA VISUALIZATION APPLICATION

Lia Ristiana

NIM. M0513027

Abstrak

Di era yang berkaitan erat dengan data ini, kebutuhan akan data tidak terbatas pada disimpan saja, namun juga harus diakses dan dianalisis untuk berbagai macam keperluan. Dengan pesatnya perkembangan teknologi, data yang dihasilkan setiap harinya di dunia menjadi semakin besar dan kompleks, sehingga muncul istilah big data. *Solusi247* sebagai salah satu perusahaan big data terdepan di Indonesia termotivasi untuk membangun sebuah aplikasi bernama *BaciroGateway*, dashboard untuk memonitor dan mengkonfigurasi *BaciroBackend* service. Di kegiatan magang mahasiswa ini, penulis menambahkan fungsionalitas baru ke *BaciroGateway* yaitu *Analytics*, sebuah menu yang berhubungan dengan mengambil, menyimpan, mengakses, dan memvisualisasikan big data dengan menggunakan teknologi big data seperti *Hadoop*, *Solr*, *Flume*, dsb. Pengembangan menu *Analytics* dilakukan dengan menggunakan *AngularJS* untuk interface aplikasi dan *NodeJS* untuk server aplikasi. Salah satu sub menu *Analytics* adalah *Ingestion*, yang dibangun untuk melakukan submisi data ke *Solr*, *platform search engine* yang dibangun dari library *Java* bernama *Lucene*. *Ingestion* dikembangkan agar pengguna dapat memanfaatkan fungsionalitas *Solr* seperti membuat core, membuat skema untuk core, dan submisi data ke *Solr* melalui *interface BaciroGateway*. Semua query dan operasi yang dikirimkan ke *Solr* dilakukan dengan mengakses *Solr HTTP API*.

Kata kunci: Apache Solr, big data

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Tuhan Yang Mahaesa, pemilik segala ilmu dan pengetahuan yang ada di alam semesta, karena berkat rahmat dan ridha-Nya penulis dapat menjalankan kuliah magang di Solusi247 cabang Yogyakarta serta menyelesaikan laporan berjudul *“Implementation of Apache Solr for Data Ingestion in Big Data Visualization Application”* untuk memenuhi SKS Kuliah Magang Mahasiswa di program studi Informatika UNS.

Kegiatan Magang Mahasiswa merupakan salah satu mata kuliah wajib di program studi Informatika FMIPA UNS yang bertujuan untuk melatih mahasiswa mengembangkan dan menerapkan ilmu pengetahuan yang telah diperoleh selama di bangku kuliah. Sedangkan laporan Kegiatan Magang Mahasiswa ini merupakan salah satu prasyarat kelulusan mata kuliah Kegiatan Magang Mahasiswa itu sendiri.

Dengan selesainya laporan ini, penulis ingin mengucapkan terima kasih kepada pihak-pihak yang telah membantu penulis selama ini.

1. Bapak dan Ibu yang telah merestui dan mendukung penulis untuk terus belajar dan mencari pengalaman baru.
2. Bapak Mohammad Ibnu Abdissalam, kepala cabang Solusi247 Yogyakarta yang telah menerima penulis untuk belajar di sana.
3. Bapak Arief Dolants, pembimbing lapangan yang memberi banyak inspirasi dan bimbingan kepada penulis.
4. Teman-teman dari Amikom, Akakom, dan UGM selaku anggota tim magang yang juga menjadi teman belajar dan teman mencari pengalaman selama magang.
5. Segenap karyawan dan keluarga besar Solusi247 Yogyakarta.

6. Dosen pembimbing magang, Bapak Afrizal Doewes, S.Kom. M.Sc, yang telah memberikan banyak masukan dan bimbingan.
7. Keluarga besar Informatika FMIPA UNS.

Surakarta, 12 Desember 2016

Penulis

DAFTAR ISI

Abstract	iv
Abstrak.....	v
KATA PENGANTAR	vi
DAFTAR ISI	viii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah.....	2
1.3. Batasan Masalah.....	3
1.4. Tujuan dan Manfaat.....	3
1.5. Metodologi.....	3
1.6. Waktu dan Tempat	4
1.7. Sistematika Penulisan	4
BAB 2 GAMBARAN UMUM INSTANSI	5
2.1. Profil Solusi247	5
2.2. Visi dan Misi	6
2.3. Struktur Organisasi	6
2.4. Produk dan Layanan.....	6
BAB 3 LANDASAN TEORI	8
3.1. Konsep Dasar Big Data	8
3.1.1 Definisi Big Data	8
3.1.2 Trend dan Analisis Big Data	9
3.2. Teknologi Big Data yang Digunakan	10
3.2.1. Apache Solr	10

3.2.1.1. Definisi dan Sejarah Solr	10
3.2.1.2. Fitur dan Kegunaan Solr	12
3.2.1.3. Running Solr	13
3.2.1.3.1. Inisiasi Solr	13
3.2.1.3.2. Operasi Solr.....	15
3.2.1.3.3. Solr HTTP API	18
3.2.2. NodeJS.....	20
3.2.3. AngularJS.....	21
BAB 4 PEMBAHASAN	23
4.1. Gambaran Umum dan Arsitektur Sistem.....	23
4.2. Desain Aplikasi	25
4.3. Fungsionalitas Menu Ingestion to Solr	26
4.3.1. Create Core	26
4.3.2. Create Schema.....	29
4.3.3. Submit Data Form.....	33
4.4. Penggunaan Aplikasi	37
BAB 5 PENUTUP	47
5.1. Kesimpulan.....	47
5.1. Saran	47
DAFTAR PUSTAKA	48

DAFTAR TABEL

Tabel 4.1 Daftar Requirement	24
------------------------------------	----

DAFTAR GAMBAR

Gambar 3.1 Start Solr Service	14
Gambar 3.2 Tampilan Solr Admin Interface	14
Gambar 3.3 Solr Directory Folder	15
Gambar 3.4 Membuat Core di Solr melalui Terminal	16
Gambar 3.5 Core Selector di Solr Admin Interface	17
Gambar 3.6 Input Data ke Solr melalui Terminal	18
Gambar 4.1 Sistem Arsitektur BaciroGateway	24
Gambar 4.2 Tampilan Dashboard Aplikasi BaciroGateway	25
Gambar 4.3 Tampilan Submenu Ingestion pada Menu Analytics	26
Gambar 4.4 Form Create Core	27
Gambar 4.5 Source Code User Interface Create Core	27
Gambar 4.6 Controller Create Core	28
Gambar 4.7 Form Create Schema	29
Gambar 4.8 User Interface Create Schema	30
Gambar 4.9 Controller Create Schema	31
Gambar 4.10 Fungsi submitSchema()	32
Gambar 4.11 Response header	32
Gambar 4.12 Form Submit Data	33
Gambar 4.13 Source Code User Interface Submit Data	34
Gambar 4.14 Source code untuk Mengecek Tipe Dokumen pada Submit Data	35
Gambar 4.15 Salah satu contoh konversi dokumen dari CSV ke JSON	36
Gambar 4.16 Running server aplikasi melalui terminal	37
Gambar 4.17 Running php	38
Gambar 4.18 Pembuatan file configsets	39
Gambar 4.19 Notifikasi core berhasil dibuat	40
Gambar 4.20 Notifikasi field pada skema berhasil dibuat	41
Gambar 4.21 Query result pada Solr	42
Gambar 4.22 Pengisian Submit Data Form	43
Gambar 4.23 Pengisian Data Form bagian 2	44
Gambar 4.24 Notifikasi Data Form telah diproses	45
Gambar 4.25 Query result pada Solr yang telah menampilkan data yang diinput sebelumnya	46

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Di era internet sekarang ini ada banyak sekali data yang dihasilkan setiap harinya, mulai dari data transaksi online, media sosial, dan sebagainya. Secara konvensional, kebutuhan data hanya perlu disimpan ke database dan baru akan diakses jika dibutuhkan saja. Namun dengan banyaknya data yang dihasilkan saat ini, muncul istilah *big data* untuk mendefinisikan data yang sangat besar, bervariasi, dan memiliki struktur kompleks sehingga terdapat kesulitan untuk menyimpan, menganalisis, dan memvisualisasikan data untuk memperoleh proses lebih lanjut (Sagiroglu & Sinanc, 2013).

Dalam konteks *big data*, kebutuhan data tidak hanya perlu disimpan saja. Ketika data yang ada semakin banyak dan cepat bertambah, maka diperlukan penanganan khusus terhadap data ini. Oleh karena itu, untuk menangani kebutuhan pengolahan data yang semakin besar dan kompleks, diperlukan adanya visualisasi data, misalnya dalam bentuk *bar/pie chart*, *scatter chart*, *real time chart*, dan sebagainya. Dengan memvisualisasikan data, ada banyak manfaat yang dapat diperoleh, seperti membantu membuat keputusan bisnis (*business intelligence*) pada suatu perusahaan, memahami tingkah laku pengguna (*user intelligence*), dan sebagainya dengan lebih mudah. *Data visualization* atau visualisasi data telah terbukti memberikan hasil yang efektif tidak hanya dalam mempresentasikan informasi yang esensial pada ukuran data yang besar, namun juga dapat memberikan hasil analisis yang kompleks (Keim, Qu, & Ma, 2013).

Terinspirasi dari semakin berkembangnya big data sekarang ini, banyak developer yang mengembangkan teknologi baru untuk big data seperti *Hadoop*, *Lucene*, *Solr*, *Cassandra*, *Hive*, *Pig*, dan sebagainya (Sagiroglu & Sinanc, 2013). Dalam kegiatan magang mahasiswa ini, penulis dan tim melanjutkan pengembangan aplikasi *BaciroGateway* menggunakan beberapa teknologi big data, yaitu *Apache Hadoop*, *Solr*, dan *D3.JS* dan teknologi lainnya berupa *NodeJS*, *AngularJS*, dan *Express*. *BaciroGateway* merupakan sebuah aplikasi yang masih berada dalam tahap pengembangan oleh Solusi247 yang ditujukan sebagai dashboard untuk mengkonfigurasi dan memonitoring service *BaciroBackend*, salah satu produk Solusi247 yang menyediakan API untuk berbagai layanan seperti Hadoop.

Pada *BaciroGateway* terdapat menu *Analytics* yang dikembangkan oleh penulis dan tim, yang ditujukan untuk melakukan analisis dan visualisasi data. Untuk mengembangkan menu ini dipergunakan sejumlah teknologi seperti *Hadoop Distributed File System* (HDFS) yang berfungsi sebagai *data warehousing system*, *Solr* yang digunakan sebagai mesin pencari untuk melakukan indexing data, dan *D3JS* sebagai salah satu library *JavaScript* digunakan untuk membangun visualisasi data yang dinamis dan interaktif berdasarkan hasil query.

Pada laporan ini, penulis menekankan pengembangan aplikasi *BaciroGateway* menu *Analytics* yaitu pada bagian query data di *Solr* melalui HTTP API agar data dapat disimpan dan diakses melalui *interface* aplikasi *BaciroGateway*.

1.2. Rumusan Masalah

Bagaimana cara menyimpan dan mengakses data di *Solr* melalui *Solr HTTP API*?

1.3. Batasan Masalah

Batasan masalah pada penelitian ini adalah *Solr* dijalankan dalam mode *single-core* atau *stand-alone* yang hanya menggunakan satu node sebagai servernya.

1.4. Tujuan dan Manfaat

Tujuan dari kegiatan magang ini adalah mengimplementasikan *Apache Solr* pada aplikasi *BaciroGateway* agar fungsionalitas *Solr* seperti menyimpan dan mengakses data dilakukan melalui *user interface* pada aplikasi menggunakan *Solr* HTTP API. Manfaat dari aplikasi tersebut adalah agar penyimpanan dan pengaksesan data ke *Solr* dapat dilakukan dengan lebih efisien dan mudah melalui aplikasi *BaciroGateway* yang sudah terhubung ke *Solr* melalui HTTP API.

1.5. Metodologi

Metodologi yang dilakukan dalam dua tahap yaitu sebagai berikut.

1. Analisis Kebutuhan

Pada tahap ini dilakukan analisis aplikasi *BaciroGateway* beserta kebutuhannya yang salah satunya yaitu kebutuhan akan menu *Analytics*, terutama bagian yang berhubungan dengan akses data ke *Solr*.

2. Implementasi

Pada tahap ini dilakukan implementasi atas kebutuhan dari hasil analisis pada aplikasi *BaciroGateway*. Menu *Analytics* terdiri dari beberapa submenu, salah satunya yaitu *Ingestion* yang dibangun menggunakan teknologi berupa *Node.js* dan *AngularJS* serta menggunakan *Apache Solr* untuk mengindeks data.

1.6. Waktu dan Tempat

Penulis melaksanakan kegiatan magang mahasiswa ini di kantor Solusi247 cabang Yogyakarta yang beralamat di Jalan Cantel 352, Bacirow, Gondokusuman, Yogyakarta selama satu setengah bulan yang dimulai pada Senin, 18 Juli hingga Rabu, 31 Agustus 2016. Kegiatan magang mahasiswa ini dilaksanakan pada hari Senin sampai Jumat setiap minggunya pada jam kerja.

1.7. Sistematika Penulisan

Laporan Kegiatan Magang Mahasiswa ini ditulis dalam lima bab yang meliputi Bab 1 Pendahuluan, Bab 2 Gambaran Umum Instansi, Bab 3 Landasan Teori, dan Bab 4 Pembahasan, serta Bab 5 Penutup.

Bab 1 yaitu Pendahuluan yang menguraikan latar belakang kegiatan magang, rumusan masalah, batasan masalah, tujuan dan manfaat, metodologi, dan sistematika penulisan.

Bab 2 yaitu Gambaran Umum Instansi berisi mengenai profil instansi magang yang terdiri dari organisasi, produk dan jasa, dan manajemen dan teknologi informasi.

Bab 3 yaitu Landasan Teori yang menjelaskan dasar teori yang digunakan untuk menulis laporan.

Bab 4 yaitu Pembahasan berisi pembahasan atas proses implementasi dari kegiatan magang ini.

Bab 5 yaitu Penutup yang memuat kesimpulan dan saran.

BAB 2

GAMBARAN UMUM INSTANSI

2.1. Profil Solusi247

Solusi247 adalah sebuah perusahaan IT di Indonesia yang didirikan pada tahun 2000, yang memfokuskan pada pengolahan data berskala besar, sistem manajemen database relasional (RDBMS), dan *massive parallel flat file processing*. Kantor pusat Solusi247 adalah di Segitiga Emas Business Park Jalan Prof. Dr. Satrio KAV 6 Jakarta Selatan. Namun Solusi247 juga mempunyai sejumlah kantor cabang di berbagai kota di Indonesia, salah satunya yaitu di Yogyakarta yang beralamat di Jalan Cantel 352/GK IV 53 RW 14 Baciro, Gondokusuman, Yogyakarta.

Selama 16 tahun perjalanannya, Solusi247 telah banyak menangani proyek yang berhubungan dengan big data terutama pada bidang industri telekomunikasi. Fokus dari layanan dan produk Solusi247 meliputi *Big Data Solution*, *IT Consultion*, *Mobile Solution*, dan Penelitian dan Pengembangan. Salah satu produk andalan Solusi247 adalah *HGrid247*, tools dan aplikasi *MapReduce* yang menggunakan Hadoop yang ditujukan untuk melakukan *automatic code-generation* dan *real-time processing*. Kesuksesan Solusi247 terbukti pada keberhasilan implementasi kluster *Hadoop* yang melibatkan sekitar 200 node untuk perusahaan telekomunikasi di Indonesia.

Saat ini sebagai salah satu perusahaan big data yang ternama di Indonesia, Solusi247 telah memiliki sejumlah pelanggan setia seperti Telkomsel, Indosat, XL, Telkom Indonesia, Bakrie Telecom, dan sebagainya. Selain itu, untuk mendukung kapabilitasnya dalam bidang big data, Solusi247 juga telah menjalin kerjasama dengan perusahaan-perusahaan internasional seperti *Cloudera*, *Intel*, *Hortonworks*, *Oracle*, *IBM*, dan sebagainya.

2.2. Visi dan Misi

- *To become 'High End' & Middle End' Information Technology Consultant, Outsource Solution and Operational Support Provider, that proven guarantee the 'Customer Success'.*
- *To become Business Partner which win for both side and always ' Easy to do Business with'*
- *To become 'One of the Best Information Technology Company' in Indonesia.*

2.3. Struktur Organisasi

Solusi247 sebagai perusahaan Persero Terbatas di Indonesia dipimpin oleh Board of Directors yang beranggotakan Beno K. Pradekso selaku Chief Executive Officer, Aria Rahendra selaku Chief Marketing Officer, Luthfi Barani selaku Chief Finance Officer, dan Amin selaku Chief Operating Officer. Cabang Yogyakarta sendiri dipimpin oleh Mohammad Ibnu Abdissalam selaku Kepala Cabang Solusi247 Yogyakarta. Saat ini Solusi247 telah memiliki ratusan pegawai yang tersebar di berbagai kota di Indonesia.

2.4. Produk dan Layanan

Solusi247 telah membangun sejumlah produk yang berhubungan dengan big data seperti *Braja Big Data Appliance*, sistem dan ekosistem untuk menjalankan *Hadoop* yang telah memiliki komponen pendukung yang berhubungan dengan big data. Aplikasi ini berjalan di sistem operasi Linux dan JVM. Selain Braja, Solusi247 juga telah menghasilkan tool big data seperti YAVA dan *HGrid*.

Sementara pada bidang layanan, Solusi247 melayani application development seperti *data warehouse, data warehouse and business intelligence*,

customer relationship management, partner relationship management, dan sebagainya.

BAB 3

LANDASAN TEORI

3.1. Konsep Dasar Big Data

3.1.1 Definisi Big Data

Definisi big data yang paling sering dikutip adalah definisi dari Gartner, meskipun ia tidak secara eksplisit menyebutkan istilah 'big data' dalam tulisannya. Seperti dikutip oleh *Ward dan Barker, 2013* big data menurut definisi Gartner adalah data dengan 'tiga V' yaitu Volume, Velocity, dan Variety. Hal ini merujuk pada ukuran big data yang besar (*volume*), kecepatan bertambahnya data (*velocity*), dan variasinya (*variety*). Namun belakangan, definisi big data telah ditambah lagi oleh IBM yaitu *veracity* yang merujuk pada kepercayaan dan ketidakpastian hal-hal yang berhubungan dengan data dan hasil analisis dari data tersebut.

Sementara itu J. P. Dijcks dari Oracle sebagaimana dikutip oleh Ward dan Barker, 2013 mengajukan definisinya sendiri terhadap big data tanpa menggunakan istilah dari 'tiga V' ataupun 'empat V' seperti di atas. Oracle berpendapat bahwa big data adalah nilai hasil turunan dari hasil keputusan bisnis yang diperoleh dari database relasional tradisional, yang digabungkan dengan sumber baru berupa data tidak terstruktur (*unstructured data*). Sumber data baru bisa berbentuk blog, media sosial, sensor jaringan, gambar, dan semacamnya yang bervariasi dalam hal ukuran, struktur, format, dan faktor-faktor lainnya. Definisi Oracle cenderung menekankan pada infrastruktur yang berhubungan

dengan big data, yaitu berupa teknologi seperti NoSQL, Hadoop, HDFS, R, dan sebagainya.

Dari definisi-definisi yang ada ini, dapat ditarik suatu garis besar bahwa big data adalah istilah yang digunakan untuk mendeskripsikan *storage* atau tempat penyimpanan dan analisis dari sejumlah dataset yang besar dan kompleks menggunakan teknologi seperti NoSQL, MapReduce, dan *machine learning*.

3.1.2 Trend dan Analisis Big Data

Big data sering diasosiasikan dengan dua hal, penyimpanan data (*data storage*) dan analisis data (*data analysis*) (Ward dan Barker, 2013). Hal ini karena untuk menanggulangi ukuran data yang besar dan bervariasi penyimpanan data secara konvensional tidak cukup. Oleh karena itu, muncul berbagai macam teknologi baru seperti *Hadoop*, *HBase*, dan sebagainya untuk memenuhi kebutuhan akan penyimpanan data ini. Sedangkan dari segi analisis data, hal ini muncul sebagai kebutuhan untuk membuat keputusan bisnis yang tepat. Kemampuan untuk membuat keputusan bisnis berdasarkan data yang tersedia adalah kebutuhan krusial dalam kesuksesan bisnis.

Menurut *McKinsey Global Institute* sebagaimana dikutip oleh Sagioglu dan Sinanc, 2013, ada lima topik utama yang berpotensi besar dalam big data meliputi; 1) kesehatan: misalnya untuk analisis individual terhadap profil pasien; 2) sektor publik; 3) retail: misalnya untuk analisis tingkah laku pengguna atau optimisasi harga barang; 4) manufaktur; dan 5) data lokasi pribadi: misalnya untuk iklan berbasis geolokasi atau pembuatan model bisnis baru. Namun, secara umum, web telah

menyediakan banyak peluang untuk big data, seperti sosial media. Misalnya yaitu untuk analisis *user intelligence* untuk iklan bertarget khusus, analisis sentimen, dan sebagainya.

3.2. Teknologi Big Data yang Digunakan

3.2.1. Apache Solr

3.2.1.1. Definisi dan Sejarah Solr

Solr adalah suatu *framework search engine* berbasis web yang dibangun dari *Apache Lucene*. Dari library *Lucene* telah dibangun setidaknya dua *search engine* yang populer yaitu *Solr* dan *ElasticSearch*. *Lucene* sendiri adalah library *open source* berbasis *Java* yang ditujukan sebagai *information retrieval library*. *Information retrieval* dapat didefinisikan sebagai suatu proses pencarian material (biasanya dokumen) dari suatu data yang bersifat *unstructured* (seperti teks) yang dapat memenuhi kebutuhan informasi pada koleksi data yang besar. Namun perlu diperhatikan bahwa *Solr* bukanlah *web search engine* sebagaimana *Google* atau *Bing*.

Dengan mesin pencari seperti *Solr*, *information retrieval* dapat dilakukan dengan memasukan query yang akan menghasilkan informasi sebagaimana dibutuhkan. Sekilas, hal ini tidak terlalu berbeda dengan fungsi pencarian pada database relasional. Namun, perbedaan antara query *Lucene* dengan query database adalah pada hasilnya, di mana pada *Lucene*, hasil pencarian dapat diurutkan berdasarkan relevansinya. Dalam kata lain, mengurutkan dokumen berdasarkan relevansi adalah aspek utama dari *information*

retrieval yang membedakannya dengan jenis query lain (Manning, 2014).

Sejarah Solr dimulai dari tahun 2004 di CNET Networks (sekarang CBS Interactive) yang ketika itu membutuhkan search engine baru untuk menggantikan search engine komersial yang harus diberhentikan oleh vendornya. Pada akhir tahun 2005, Solr sudah digunakan untuk memfasilitasi kebutuhan pencarian di sejumlah situs milik CNET. Kemudian pada Januari 2016 Solr akhirnya dibuat open source dan dikontribusikan ke Apache Software Foundation sebagai subproject dari Lucene PMC. Hingga makalah ini ditulis kini Solr telah dikembangkan hingga mencapai versi 6.3.0 dan masih dikembangkan terus untuk memperbaiki dan menambah fitur-fitur yang dibutuhkan.

Sebagaimana kebanyakan proyek Apache yang berhubungan dengan big data, Solr dapat berjalan pada satu mesin atau beberapa mesin, bahkan hingga ribuan. Jika berjalan pada satu mesin, maka Solr hanya memiliki satu core atau single core. Sedangkan dengan beberapa mesin, maka Solr dapat berjalan sebagai SolrCloud dengan banyak core yang disebut collection.

Sebagaimana dikutip dari situs resminya (<https://wiki.apache.org/solr/PublicServers>), Apache Solr telah dikenal dan digunakan oleh banyak perusahaan besar seperti berikut ini.

- Netflix, menggunakan Solr untuk fitur pencarian situsnya.

- Instagram, menggunakan API Solr pada bagian geo-search.
- Reddit, menggunakan Solr untuk pencarian.
- SourceForge, menggunakan fitur faceted-search pada Solr untuk pencarian pada proyek.

3.2.1.2. Fitur dan Kegunaan Solr

Solr dirancang untuk memiliki feature yang menguntungkan baik untuk developer maupun end-user. Secara umum fitur Solr dapat dibagi menjadi dua jenis, yaitu fitur dari segi user experience dan dari segi data modeling.

a) User-experience features

Apache Solr memiliki sejumlah fitur untuk hal-hal yang berhubungan dengan pencarian dokumen, namun fitur-fitur ini hanya dapat diakses melalui HTTP REST-API, sehingga tidak benar-benar memiliki komponen graphic user interface (GUI). Developer harus membangun GUI sendiri agar dapat menggunakan fitur-fitur yang ada. Fitur-fitur tersebut meliputi sebagai berikut.

- i. Pagination and sorting
- ii. Faceting
- iii. Autosuggest
- iv. Spell-checking
- v. Hit highlighting

vi. Geospatial search

b) Data modeling feature

i. Result grouping/field collapsing

ii. Flexible query support

iii. Joins

iv. Document clustering

v. Multilingual support

3.2.1.3. Running Solr

3.2.1.3.1. Inisiasi Solr

Solr dapat berjalan di semua jenis OS yang dapat menjalankan Java Runtime Environment (JRE). Versi JRE minimum yang diharuskan untuk menjalankan Solr adalah 1.7.

Solr dapat diunduh di website resminya (<http://lucene.apache.org/solr/>) dalam bentuk .zip (Windows) atau .tgz (Linux/Unix/OSX). Pada sistem operasi Linux, file Solr dapat diekstrak di direktori `"/opt/"` dan langsung dijalankan dari direktori tersebut.

Untuk menjalankan Solr pada sistem operasi Linux, maka pertama masuk ke direktori `"/opt/solr"` dan jalankan perintah untuk memulai service. Secara default, Solr akan berjalan di port 8983, sehingga

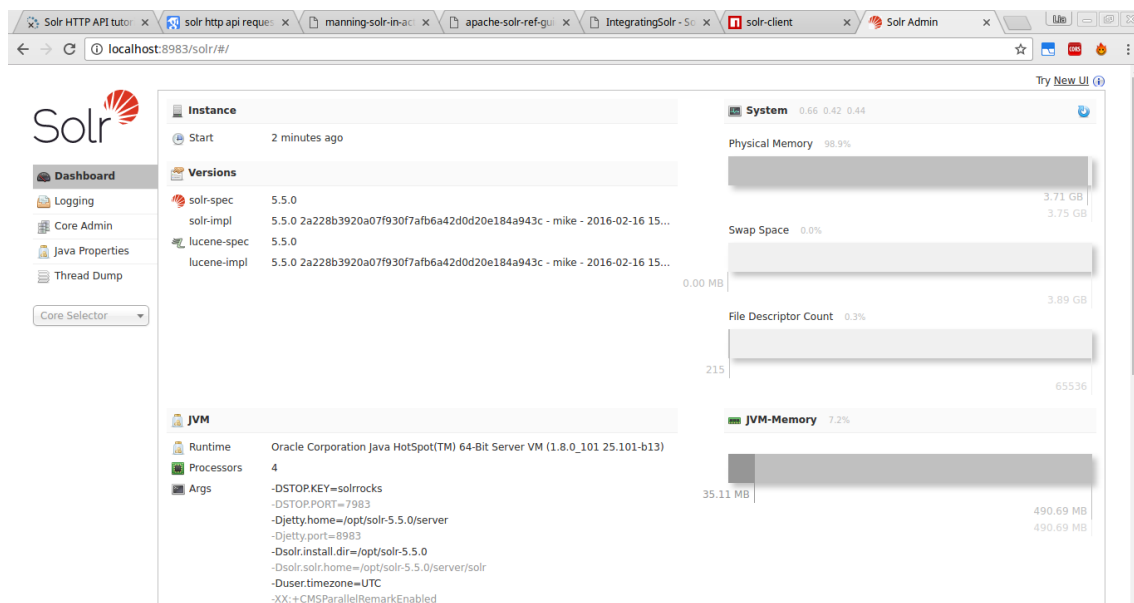
dapat diakses melalui browser pada alamat <http://localhost:8983/>. Perintah untuk menjalankan Solr adalah “bin/solr start” dengan tampilan sebagai berikut apabila perintah berhasil dijalankan.

```
lia@lia /opt/solr-5.5.0 $ sudo bin/solr start
[sudo] password for lia:
Waiting up to 30 seconds to see Solr running on port 8983 [/]
Started Solr server on port 8983 (pid=8476). Happy searching!

lia@lia /opt/solr-5.5.0 $
```

Gambar 3.1 Start Solr Service

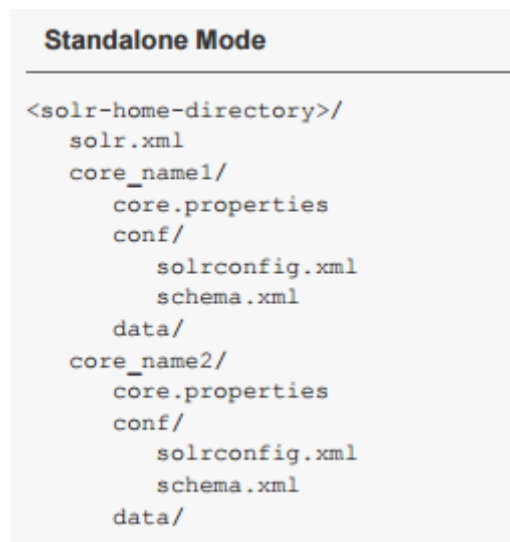
Tampilan yang diberikan ketika Solr berhasil dijalankan di browser adalah halaman Solr Admin Interface sebagai berikut.



Gambar 3.2 Tampilan Solr Admin Interface

Pada Solr yang berjalan dalam mode *standalone*, direktori utamanya adalah terdiri dari folder-folder masing-masing core yang pernah dibuat,

di mana pada masing-masing folder core terdapat file konfigurasi dari core tersebut.



Gambar 3.3 Solr Directory Folder

Direktori *Solr* yang berada pada server berisi folder-folder cores. Sebagaimana dilihat di atas terdapat beberapa core yaitu *cobacore*, *gettingstarted*, dan sebagainya. Direktori core *gettingstarted* terdiri dari folder *conf*, *core.properties*, dan *data*. Folder *conf* berisi konfigurasi core *gettingstarted* seperti *schema.xml* yang memuat skema dari data pada core tersebut (jika ada). Sedangkan folder *data* memuat data yang tersimpan pada core.

3.2.1.3.2. Operasi Solr

Jika *Relational Database Management System* (RDBS) memiliki database, maka *Solr* memiliki core. Core dibutuhkan untuk tempat penyimpanan indeks

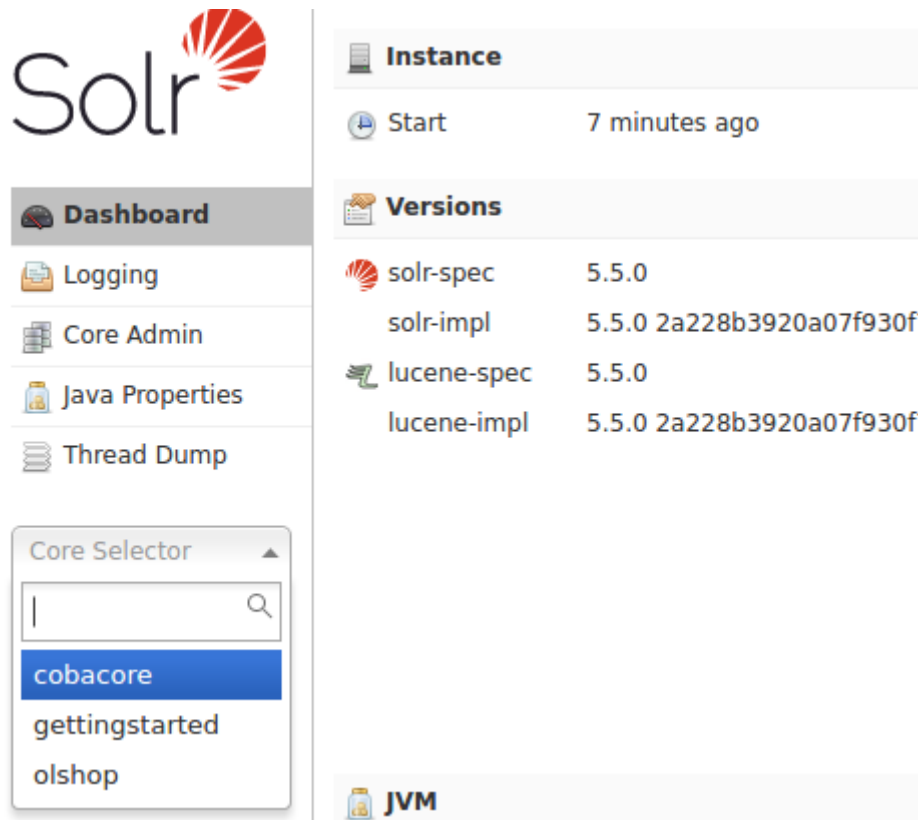
dokumen, sehingga sebelum memasukkan dokumen untuk indeks, hal pertama yang harus dilakukan adalah membuat core.

Operasi di *Solr* seperti pembuatan core ini dapat dilakukan melalui beberapa cara, yaitu menggunakan perintah melalui terminal, melalui *admin interface*, dan melalui *Solr HTTP API*. Pembuatan core menggunakan perintah melalui terminal dapat dilakukan dengan perintah “*bin/solr create -c <nama_core>*” sebagaimana berikut ini.

```
lia@lia /opt/solr-5.5.0 $ sudo bin/solr create -c gettingstarted
Copying configuration to new core instance directory:
/opt/solr-5.5.0/server/solr/gettingstarted
Creating new core 'gettingstarted' using command:
http://localhost:8983/solr/admin/cores?action=CREATE&name=gettingstarted&instanceDir=gettingstarted
{
  "responseHeader":{
    "status":0,
    "QTime":761},
  "core":"gettingstarted"}
lia@lia /opt/solr-5.5.0 $
```

Gambar 3.4 Membuat Core di Solr melalui Terminal

Semua core yang pernah dibuat dan masih tersimpan di *Solr* dapat dilihat melalui core selector pada *dashboard admin interface*.



Gambar 3.5 Core Selector di Solr Admin Interface

Setelah core berhasil dibuat, langkah selanjutnya adalah mengindeks dokumen ke core melalui perintah `"bin/post -c <nama_core> <nama_dokumen>"`. Instalasi *Solr* pada dasarnya telah menyediakan berbagai macam tipe dokumen yang tersedia di subdirektori `"example/"` sebagai contoh dokumen apa saja yang dapat diindeks oleh *Solr*. Contoh penggunaan perintah untuk mengindeks

dokumen ke core menggunakan file yang ada di subdirektori “example/” adalah sebagai berikut.

```
lia@lia /opt/solr-5.5.0 $ sudo bin/post -c gettingsstarted example/exampledocs/*.xml
[sudo] password for lia:
java -classpath /opt/solr-5.5.0/dist/solr-core-5.5.0.jar -Dauto=yes -Dc=gettingsstarted -Ddata=files org.apache.solr.
ple.xml example/exampledocs/hd.xml example/exampledocs/ipod_other.xml example/exampledocs/ipod_video.xml example/exan
ml example/exampledocs/money.xml example/exampledocs/monitor2.xml example/exampledocs/monitor.xml example/exampledocs
ampledocs/solr.xml example/exampledocs/utf8-example.xml example/exampledocs/vidcard.xml
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/gettingsstarted/update...
Entering auto mode. File endings considered are xml,json,jsonl, csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otf
POSTing file gbl8030-example.xml (application/xml) to [base]
POSTing file hd.xml (application/xml) to [base]
POSTing file ipod_other.xml (application/xml) to [base]
POSTing file ipod_video.xml (application/xml) to [base]
POSTing file manufacturers.xml (application/xml) to [base]
POSTing file mem.xml (application/xml) to [base]
POSTing file money.xml (application/xml) to [base]
POSTing file monitor2.xml (application/xml) to [base]
POSTing file monitor.xml (application/xml) to [base]
POSTing file mp500.xml (application/xml) to [base]
POSTing file sd500.xml (application/xml) to [base]
POSTing file solr.xml (application/xml) to [base]
POSTing file utf8-example.xml (application/xml) to [base]
POSTing file vidcard.xml (application/xml) to [base]
14 files indexed.
COMMITting Solr index changes to http://localhost:8983/solr/gettingsstarted/update...
Time spent: 0:00:02.163
lia@lia /opt/solr-5.5.0 $
```

Gambar 3.6 Input Data ke Solr melalui Terminal

Pada contoh di atas, dieksekusi perintah untuk mengindeks semua dokumen berekstensi .xml yang ada pada subdirektori “example/exampledocs” ke core *gettingsstarted* yang telah dibuat sebelumnya. Hasil dari eksekusi perintah ini adalah 14 file berhasil terindeks.

Ada lima operasi dasar yang dapat dijalankan oleh Solr yaitu *query*, *index*, *delete*, *commit*, dan *optimize*. Operasi-operasi ini dieksekusi dengan menciptakan URL yang mengandung parameter query sehingga dapat dieksekusi melalui Solr HTTP API yang akan dijelaskan pada subbab berikutnya.

3.2.1.3.3. Solr HTTP API

Pada dasarnya, *Solr* adalah sebuah *web application*, namun karena ia dibangun dengan *open protocol*, maka semua jenis aplikasi client tetap dapat menggunakan *Solr* melalui protokol HTTP yang menghubungkan aplikasi client dengan *Solr*. Aplikasi client dapat membuat sebuah *request* seperti *query* atau *indexing* dokumen ke *Solr* dan *Solr* akan memberikan respon berupa hasil atas *request* tersebut.

Solr menyediakan banyak client API untuk berbagai macam bahasa pemrograman dan *environment*, mulai dari *SolrRuby* dan *DelSolr* untuk *Ruby*, *Flare* untuk *Ruby on Rails*, *SolPHP* untuk *PHP*, *SolrJ* untuk *Java*, *Python API* dan *PySolr* untuk *Python*, dan sebagainya. Pada kuliah magang ini, penulis menggunakan *solr-client* yang merupakan salah satu *package* NPM untuk dijalankan di *NodeJS*.

Setelah berhasil membuat core dan mengindeks data ke core, hal yang dapat dilakukan selanjutnya adalah membuat *query* ke *Solr*. *Solr* menyediakan HTTP API untuk melakukan operasi seperti *query*.

Dengan menggunakan HTTP API yang telah disediakan oleh *Solr*, *query* terhadap suatu dokumen dapat dilakukan sebagai berikut. Misalnya pada core *gettingstarted* pengguna ingin menampilkan seluruh

data yang ada pada core, maka query yang digunakan adalah sebagai berikut.

http://localhost:8983/solr/gettingstarted/select?q=%3A*&wt=json&indent=true

Sedangkan jika user ingin mencari dokumen dengan keyword “video”, maka URL yang dihasilkan untuk mengeksekusi query tersebut adalah sebagai berikut.

<http://localhost:8983/solr/gettingstarted/select?q=video&wt=json&indent=true>

Query di atas memiliki parameter berupa *q=video* yaitu keyword yang digunakan pada query. Sedangkan *wt=json* dan *indent=true* berarti request ke Solr akan memberikan pengembalian berupa data dalam format json yang telah diindentasi.

3.2.2. NodeJS

Node.js adalah salah satu framework *JavaScript* yang bekerja sebagai *JavaScript runtime environment*. *Node* menggunakan model *input-output* yang bersifat *asynchronous* dan *event-driven*. *Node* sering disebut sebagai salah satu cara untuk menjalankan *JavaScript* di server.

Karena sifatnya yang *open-source*, *Node.js* dapat diunduh di situs resminya www.nodejs.org untuk dijalankan di sistem operasi *Windows*, *iOS*, maupun *Linux*. Sejak v0.6.3 pengunduhan *Node.js* juga disertai dengan *Node Package Manager* (NPM) yang berfungsi untuk dua

hal yaitu sebagai online repository untuk *node.js package/module* dan sebagai command line utility untuk menginstall package *Node.js*, manajemen versi dan dependency untuk *package Node.js*.

Package *Node.js* dapat ditemukan secara terbuka di situs-situs seperti *Github* dan *npmjs* (<https://www.npmjs.com/>) dan diinstall melalui *npm*. Beberapa package *Node.js* yang digunakan pada *BaciroGateway* di antaranya adalah sebagai berikut.

- A. *JSONStream* (<https://www.npmjs.com/package/JSONStream>) : digunakan untuk melakukan streaming data JSON.
- B. *Express* (<https://expressjs.com/>) : merupakan framework *Node.js* yang dapat melakukan routing yaitu melakukan definisi *end points* aplikasi dalam bentuk URI dan merespon request klien.
- C. *Solr-client* (<https://www.npmjs.com/package/solr-client>) : merupakan *Solr client library* untuk melakukan *indexing, adding, deleting, committing*, dan optimisasi serta pencarian dokumen pada *Apache Solr*.

3.2.3. AngularJS

AngularJS adalah framework *JavaScript* untuk aplikasi web yang dinamis. Framework ini memungkinkan pengguna untuk menggunakan HTML sebagai template dasar sekaligus menggunakan fitur-fitur tambahan dari *Angular* seperti *data binding* dan *dependency injection*, sehingga kode yang dihasilkan lebih jelas dan efisien.

Jika *Node.js* sering menjadi *server-side* pada aplikasi web, maka *AngularJS* sering disebut sebagai solusi *client-side* pada aplikasi. Hal ini

karena *AngularJS* dapat melakukan handling semua DOM dan AJAX pada *client-side*.

BAB 4

PEMBAHASAN

4.1. Gambaran Umum dan Arsitektur Sistem

BaciroGateway adalah aplikasi yang masih berada pada tahap pengembangan oleh Solusi247. Aplikasi ini ditujukan sebagai dashboard untuk mengkonfigurasi dan memonitoring service *BaciroBackend*, salah satu produk yang menyediakan API untuk berbagai macam layanan seperti *Hadoop*, *Flume*, dan sebagainya.

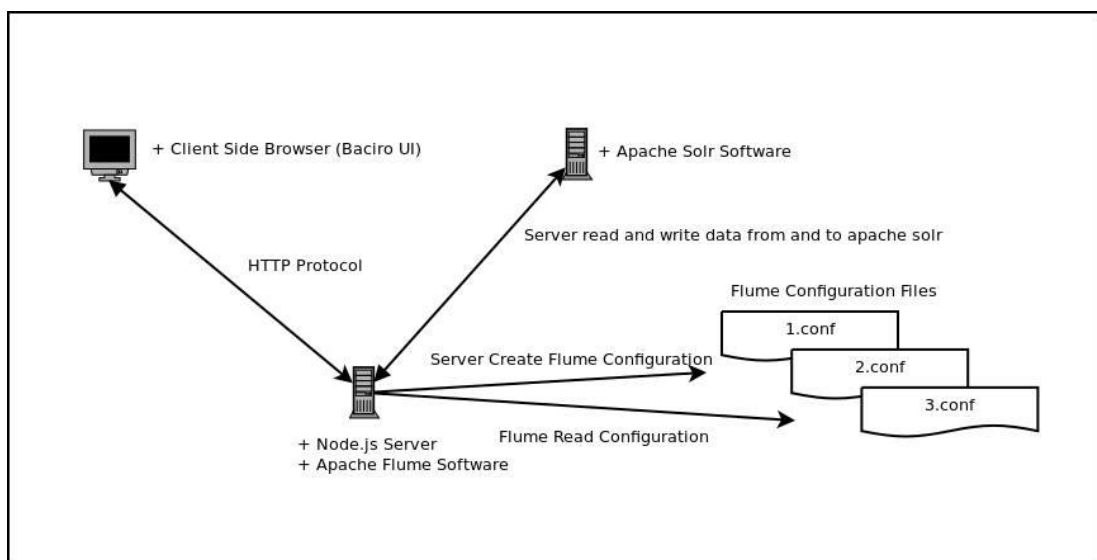
Secara umum, prototype aplikasi *BaciroGateway* ini dibangun dengan sejumlah teknologi dan framework yang meliputi *Apache Hadoop*, *Apache Flume*, *Apache Solr*, *D3.js*, *NodeJs*, *AngularJS*, dan *ExpressJS*. *System requirement* yang dibutuhkan untuk menjalankan *BaciroGateway* adalah sebagai berikut.

No	Requirement	Deskripsi
1	<i>Apache Solr 5.5.0</i>	Platform open source yang ditujukan sebagai mesin pencari yang mengindeks data, ditulis dalam Java, dan merupakan bagian dari proyek Apache Lucene.
2	<i>Apache Flume 1.6.0</i>	Framework untuk melakukan streaming data ke Hadoop.
3	<i>Node 4.5.0</i>	Environment berbasis JavaScript yang bersifat event-driven yang ditujukan sebagai server pada suatu kode JavaScript.

4	PHP 5	Bahasa pemrograman untuk aplikasi berbasis web.
---	-------	---

Tabel 4.1 Daftar Requirement

Secara arsitektur, aplikasi dibangun dengan model *client-server*. Server pada aplikasi ini diimplementasikan dengan menggunakan *NodeJS*, sedangkan client diimplementasikan oleh sebuah aplikasi bernama *BaciroUI* yang dibangun dengan *AngularJS*.

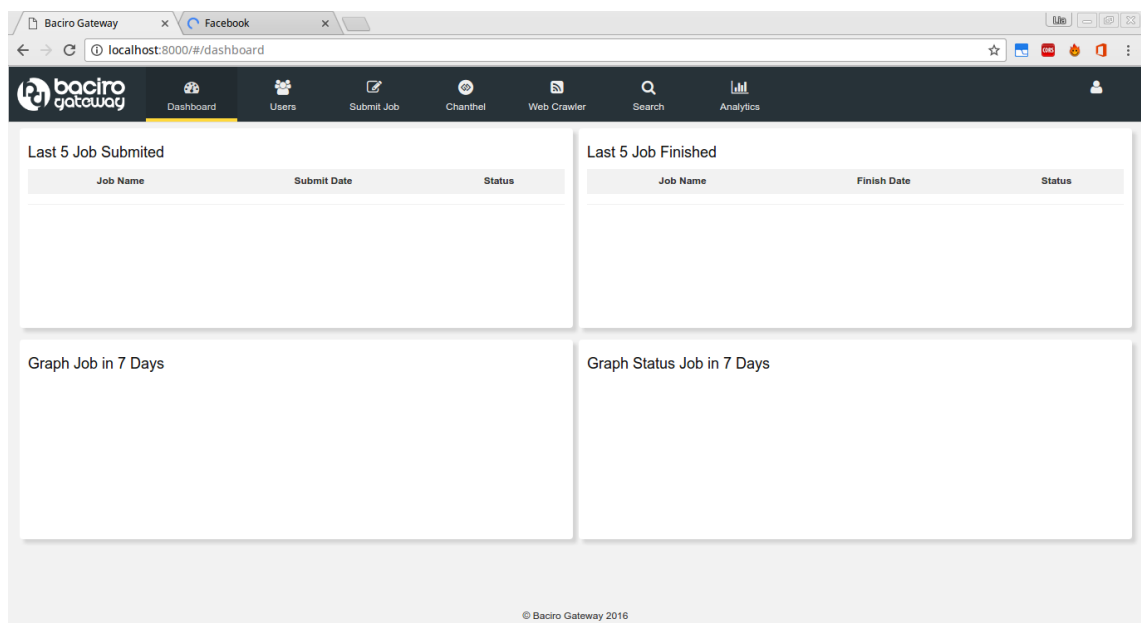


Gambar 4.1 Sistem Arsitektur BaciroGateway

Berdasarkan ilustrasi di atas, aplikasi *BaciroGateway* dapat dijalankan dari sisi klien melalui *web browser* yang mengakses ke server melalui protokol HTTP, di mana akan diakses server NodeJS dan Apache Flume software. Server akan membuat *flume configuration* dan Flume akan mengakses *file configuration* tersebut. Berdasarkan request user, server NodeJS juga akan melakukan akses read and write ke Apache Solr software dan mengembalikan hasilnya untuk ditampilkan ke client side di browser.

4.2. Desain Aplikasi

Aplikasi *BaciroGateway* terdiri dari beberapa menu seperti Dashboard, Users, Submit Job, Chantel, Web Crawler, Search, dan Analytics. Ketika dijalankan dan diakses dari browser, BaciroGateway memiliki tampilan sebagai berikut.



Gambar 4 2 Tampilan Dashboard Aplikasi BaciroGateway

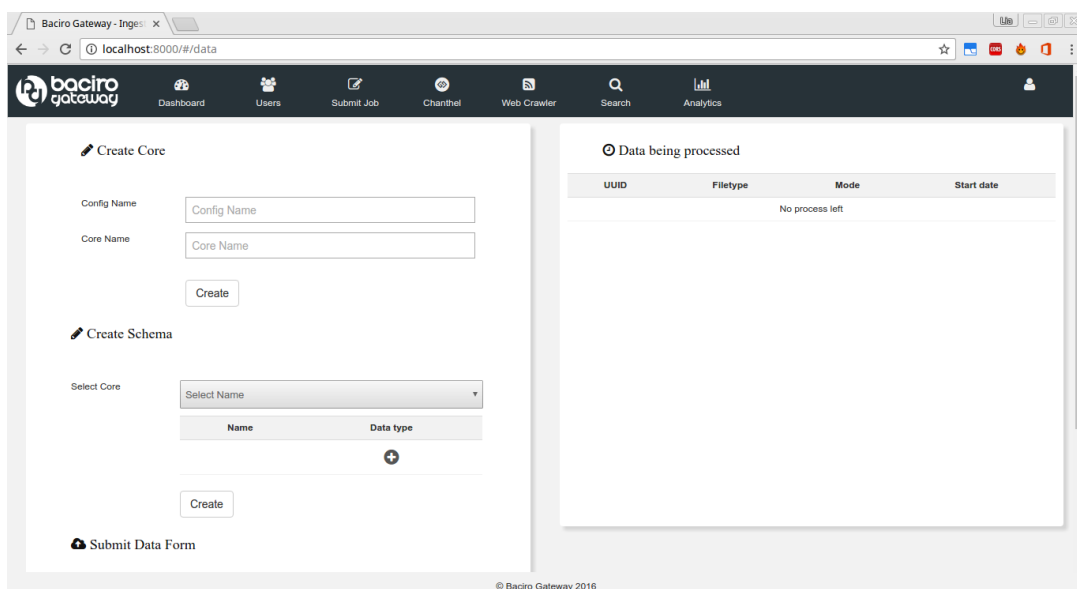
Pada kuliah magang mahasiswa ini, penulis dan tim menambahkan menu *Analytics* ke *BaciroGateway*. Menu *Analytics* dapat menjalankan sejumlah task seperti melakukan *fetching data* yang akan dimasukkan ke *Hadoop* menggunakan *Flume* atau langsung dimasukkan ke *Solr*. Di *Solr* data disimpan di core yang telah dibuat. Data yang disimpan tersebut dapat dibuat skemanya atau tidak menggunakan schema sama sekali (*schemaless*). Data juga dapat diindeks agar dapat ditampilkan secara grafis, yang dalam hal ini menggunakan framework *D3.js*.

Analytics terdiri dari beberapa submenu yang meliputi *Flume*, *Ingestion*, dan *Visual*. Submenu *Flume* berfungsi untuk melakukan *retrieve* data dari *data*

source. Submenu *Ingestion* adalah menu yang menghubungkan aplikasi ke fungsionalitas Solr. Sedangkan *Visual* adalah submenu yang memberikan hasil visualisasi berdasarkan request akses data ke *Solr*.

Akses aplikasi ke *Solr* dilakukan dengan menggunakan HTTP REST API dari *Solr*. Semua proses ini dilakukan dalam framework JavaScript yang menggunakan sejumlah package *NodeJS* di server dan menggunakan *AngularJS* di bagian front-end. *AngularJS* sendiri menganut arsitektur *Model-View-Controller (MVC)*.

Submenu *Ingestion* pada *Analytics* ini berhubungan dengan submit data ke *Solr*, yang meliputi *Create Core* (membuat core di *Solr*), *Create Schema* (membuat schema), dan *Submit Data Form* (memasukkan data ke core *Solr*).



Gambar 4 3 Tampilan Submenu Ingestion pada Menu Analytics

4.3. Fungsionalitas Menu Ingestion to Solr

4.3.1. Create Core

Create Core seperti namanya berfungsi untuk membuat core di Solr. Pengguna harus menspesifikasikan nama core dan nama file config

yang akan dipakai untuk membuat core. Tampilan menu Create Core adalah seperti Gambar 4.4.

The screenshot shows a web form titled "Create Core". It has two text input fields: "Config Name" and "Core Name". Below these fields is a "Create" button. The form is styled with a light gray border and a white background.

Gambar 4.4 Form Create Core

```

23 <div class="col-lg-6">
24   <div class="col-md-12 clear-padding">
25     <div class="panel-baciro" style="min-height: 510px">
26       <!-- SELECT OPTION -->
27
28       <!-- CREATE CORE NAME -->
29       <div class="col-md-12" ng-controller="coreCtrl">
30         <h3 style="padding: 5px 5px 10px 45px;" class="fa fa-pencil"> Create Core</h3>
31
32         <form ng-submit="processCreateCore()">
33
34           <div class="baciro-input-line">
35             <!-- CONFIG NAME -->
36             <div class="col-md-3">
37               <p>Config Name</p>
38             </div>
39
40             <div class="col-md-9">
41               <div>
42                 <input class="baciro-input" type="text" name="config_name" ng-model="coreForm.config_name" placeholder="Config Name" required />
43               </div>
44             </div>
45           </div>
46
47           <div class="baciro-input-line">
48             <!-- CORE NAME -->
49             <div class="col-md-3">
50               <p>Core Name</p>
51             </div>
52
53             <div class="col-md-9">
54               <div>
55                 <input class="baciro-input" type="text" name="core_name" ng-model="coreForm.name_chosen" placeholder="Core Name" required />
56               </div>
57             </div>
58           </div>
59           <div class="baciro-input-line" style="margin-top:30px;">
60             <div class="col-md-3"></div>
61             <div class="col-md-3">
62               <button type="submit" ng-click="submitted=true" class="btn btn-default">Create</button>
63             </div>

```

Gambar 4.5 Source Code User Interface Create Core

User Interface Create Core dibangun dengan AngularJS yang terdiri dari model, view, dan controller. Bagian view dibangun dengan HTML yang mendeskripsikan tampilan Create Core yang terdiri dari field dan tombol, yaitu field untuk menerima input nama core dan nama config

core serta tombol untuk membuat core yang direpresentasikan dalam potongan kode sebagaimana pada Gambar 4.5.

Sedangkan di bagian controller, data yang telah diperoleh dari user tadi akan diproses dengan code berikut.



```

1 app.controller('coreCtrl', function ($scope, $http, $sce, $compile, Data, |toaster| {
2
3     $scope.coreForm = {};
4     // var coreName = $scope.coreForm.name_chosen;
5     var instanceDir;
6     var configFileName;
7     var schemaFileName;
8
9     $scope.processCreateCore = function(){
10        console.log('pressed');
11        var configName = $scope.coreForm.config_name;
12        var coreName = $scope.coreForm.name_chosen;
13        var createCoreUrl = "admin/cores?action=CREATE"+
14        "&name="+coreName+
15        "&configSet="+configName+
16        "&wt=json";
17
18        Data.solrget(createCoreUrl).then(function(result) {
19
20            // $location.path('data');
21            // refresh this page?
22            console.log(result);
23
24            if(result.responseHeader.status==0){
25                window.location = window.location.href;
26                window.alert("Core has been successfully created.");
27                window.location.reload();
28                console.log('responded');
29            }else if(result.responseHeader.status==500){
30                toaster.pop("error", "", "Core name already existed", 3000, 'trustedHtml');
31            }else{
32                toaster.pop("error", "", "An error has occured.", 3000, 'trustedHtml');
33            }
34
35        });
36    };
37
38 }
39 });

```


Gambar 4.6 Controller Create Core

Pada potongan kode di atas proses pembuatan core di Solr dilakukan menggunakan HTTP API dengan URL berupa [/admin/cores?action=CREATE&name=<nama core>&configSet=<nama_c onfig>](#). Parameter `wt=json` di akhir URL digunakan agar Solr memberikan pengembalian hasil request dalam format JSON. Request ke Solr ini akan

memberikan status response header berupa 0 yang berarti request berhasil atau 500 yang berarti gagal karena nama core yang diinputkan sudah ada sebelumnya atau lainnya yang juga berarti request gagal dilakukan.


4.3.2. Create Schema

Data yang disimpan di core juga dapat didefinisikan melalui *schema* (skema). Skema menjelaskan data dengan mendefinisikan *field type* dan nama field.

 Create Schema

Select Core

Select Name ▼

Name	Data type
	

Create

Gambar 4.7 Form Create Schema

Pada bagian view Create Schema dibangun dengan HTML yang terdiri dari hal-hal yang berhubungan dengan front-end seperti form, tombol, dan field sebagaimana berikut.

```

28 <!-- CREATE CORE NAME -->
29 <div class="col-md-12" ng-controller="coreCtrl">
30   <h3 style="padding: 5px 5px 10px 45px;" class="fa fa-pencil"> Create Core</h3>
31
32   <form ng-submit="processCreateCore()">
33
34     <div class="baciro-input-line">
35       <!-- CONFIG NAME -->
36       <div class="col-md-3">
37         <p>Config Name</p>
38       </div>
39
40       <div class="col-md-9">
41         <div>
42           <input class="baciro-input" type="text" name="config_name" ng-model="coreForm.config_name" placeholder="
43             Config Name" required />
44         </div>
45       </div>
46     </div>
47
48     <div class="baciro-input-line">
49       <!-- CORE NAME -->
50       <div class="col-md-3">
51         <p>Core Name</p>
52       </div>
53
54       <div class="col-md-9">
55         <div>
56           <input class="baciro-input" type="text" name="core_name" ng-model="coreForm.name_chosen" placeholder="Core
57             Name" required />
58         </div>
59       </div>
60     </div>
61     <div class="baciro-input-line" style="margin-top:30px;">
62       <div class="col-md-3"></div>
63       <div class="col-md-3">
64         <button type="submit" ng-click="submitted=true" class="btn btn-default">Create</button>
65       </div>
66     </div>
67   </form>
68 </div>

```

Gambar 4.8 User Interface Create Schema

Sementara di bagian controller, dideskripsikan proses pengambilan data dari front-end untuk dihubungkan ke server Solr.

```
1 app.controller('schemaCtrl', function($scope, $http, Data, toaster){
2   // $scope.choices = [{id: 'choice1'}];
3   $scope.schemaForm = {};
4   $scope.schemaForm.params = [];
5   $scope.schemaForm.getCollectionsID = [];
6   $scope.schemaForm.getCollections = [];
7
8   $scope.schemaForm.dataType = [
9     { 'id': 0, 'type': 'string' },
10    { 'id': 1, 'type': 'float' },
11    { 'id': 2, 'type': 'int' },
12    { 'id': 3, 'type': 'double' },
13    { 'id': 4, 'type': 'date' },
14    { 'id': 5, 'type': 'boolean' }
15  ];
16
17  //add a row in the array
18  $scope.addField = function(){
19
20    $scope.schemaForm.params.push(
21      {
22        'col': '',
23        'type': $scope.schemaForm.dataType[0].type,
24      });
25  };
26
27  // remove the selected row
28  $scope.removeField = function(index){
29    // remove the row specified in index
30    if (confirm("Really want to remove?")) {
31      $scope.schemaForm.params.splice( index, 1);
32    }
33  };
34}
```

Gambar 4.9 Controller Create Schema

Pada Controller schema di atas dideklarasikan beberapa *field type* yang dapat dipilih oleh pengguna yaitu meliputi *string*, *float*, *int*, dan sebagainya. Create Schema terdiri dari beberapa fungsi, di antaranya yaitu *addField()* untuk menambah field dan *removeField()* untuk menghapus field.

```

78  //
79
80  $scope.submitSchema = function() {
81      console.log('clicked');
82      var schemaUrl = $scope.schemaForm.getCollections.select_collections+"/schema";
83      $scope.schemaForm.params.forEach(function(item, id){
84          var schema = {
85              "add-field":{
86                  "name":item.col,
87                  "type":item.type,
88                  "stored":true
89              }
90          };
91      });

```

Gambar 4.10 Fungsi submitSchema()

Pada controller Create Schema terdapat fungsi *submitSchema()* yang digunakan untuk melakukan pembuatan skema atas core yang dipilih ke ke Solr menggunakan API skema di mana nama field dan type yang diinputkan akan dikirim sebagai parameter request. Dari request ke Solr akan diperoleh hasil apakah request berhasil atau gagal sesuai kode di bawah ini.

```

105  Data.solrpost(schemaUrl, schema).then(function(res) {
106      if (res.responseHeader.status == 0) {
107          toaster.pop("success", "", "Field " + item.col + " successfully created", 3000, 'trustedHtml');
108      } else {
109          toaster.pop("error", "", "Field " + item.col + " failed to create", 3000, 'trustedHtml');
110      }
111  });
112  });
113
114  }
115


```

Gambar 4.11 Response header

Dari response header request seperti pada potongan kode di atas, jika statusnya 0, berarti field berhasil dibuat, sedangkan selain itu berarti

field gagal dibuat. Program akan menampilkan notifikasi per field kepada user.

4.3.3. Submit Data Form

 **Submit Data Form**

Mode

File No file chosen

Name	Type	Size (KB)	base64
------	------	-----------	--------

File Type

[Show/hide details Method](#)

Collections

[Show/hide details](#)

Gambar 4 .12 Form Submit Data

Submit Data Form digunakan untuk memasukkan data ke core Solr. Data yang dimasukkan ke Solr dapat berupa data offline (dokumen yang ada pada local storage) atau online (data dari internet). Pada form Submit Data, user memilih mode data yang akan dimasukkan yaitu online atau offline. Data yang ingin dimasukkan ke core harus diunggah (jika offline) atau user harus memasukkan URL yang berisi data (jika online). Setelah itu, user harus menspesifikasikan format data. Solr dapat mengindeks berbagai macam format dokumen, namun BaciroGateway sendiri membatasi jenis format dokumen berupa csv, txt, json, atau xml. Di akhir form user dapat memilih ke core mana data akan dimasukkan dan memilih *submit*.

```

137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
<!-- DATA CONFIG -->
<div class="col-md-12 clear-padding" ng-controller="dataCtrl">
  <h3 style="padding: 5px 5px 10px 45px;" class="fa fa-cloud-upload"> Submit Data Form</h3>
  <form ng-submit="processForm()">
    <div class="baciro-input-line">
      <div class="col-md-3">
        Mode
      </div>
      <div class="col-md-9">
        <div>
          <select class="baciro-input-select" ng-model="dataForm.mode_selected" ng-options="m as m.name for m in dataForm.modeList" ng-init="dataForm.mode_selected=dataForm.modeList[0]"></select>
        </div>
      </div>
    </div>
    <div class="kotak offline" ng-if="dataForm.mode_selected.id==0">
      <div class="baciro-input-line">
        <div class="col-md-3">
          File
        </div>
        <div class="col-md-9">
          <input class="baciro-input" type="file" ng-model="dataForm.mode_selected.file" placeholder="File Type"
            required base-sixty-four-input maxsize="800" accept=".json, .txt, .xml, .csv" />
        </div>
      </div>
      <div class="baciro-input-line">
        <div class="col-md-3">
          &nbsp;
        </div>
        <div class="col-md-9">
          <table class="table table-bordered table-striped">
            <tr>
              <th>Name</th>
              <th>Type</th>
              <th>Size (<i><small>KB</small></i></th>

```

Gambar 4.13 Source Code User Interface Submit Data

Sementara itu pada server-side Node.js digunakan untuk melakukan input data ke Solr. Pada server akan dilakukan pengecekan apakah dokumen yang dimasukkan berformat csv, json, xml, atau txt.

```
68 router.post('/add', function(req, res) {
69
70     var incomingParams,
71         reader,
72         converter,
73         intendedObj,
74         solrFields,
75         resp,
76         solr;
77
78     resp = {};
79     resp.code = 1;
80     resp.status = "success";
81     resp.messages = [];
82
83     res.header('content-type', 'application/json');
84
85     incomingParams = JSON.parse(JSON.stringify(req.body)); // make a copy
86
87     console.log(incomingParams.fileTypeSelect);
88
89     console.log(incomingParams.modeSelect.url);
90     // set the right converter
91     if (incomingParams.fileTypeSelect.name == 'csv') {
92         converter = new memoryStream(); //new csv2json.Converter({});
93     } else if (incomingParams.fileTypeSelect.name == 'json') {
94         converter = JSONStream.parse();
95     } else if (incomingParams.fileTypeSelect.name == 'xml'){
96         converter = new memoryStream();
97     } else if (incomingParams.fileTypeSelect.name == 'txt'){
98         converter = new memoryStream();
99     }
100
101     res.end('{"code": 1, "status" : "success", "message" : "Your data is being processed"}');
```

Gambar 4.14 Source code untuk Mengecek Tipe Dokumen pada Submit Data

Pengecekan format seperti di atas penting dilakukan karena agar dapat masuk ke Solr, format dokumen selain JSON perlu dikonversikan dulu ke JSON. Seperti pada potongan kode berikut di mana dokumen CSV harus dikonversikan dahulu ke JSON menggunakan package csv2json.

```
if (incomingParams.fileTypeSelect.name == 'csv') {  
    var fromStream;  
    var csvhead = [];  
    var cv = new csv2json.Converter({});  
    converter.on('finish', function() {  
        fromStream = converter.toString();  
        if (incomingParams.fileTypeSelect.header == 0) {  
            for (var i=1; i <= incomingParams.fileTypeSelect.params; i++) {  
                csvhead.push(i);  
            }  
            fromStream = csvhead.join() + '\n' + fromStream;  
        }  
        console.log(fromStream);  
        cv.fromString(fromStream, function(err, data) {  
            parse_and_send_to_solr(incomingParams, data, res);  
        });  
    });  
};
```

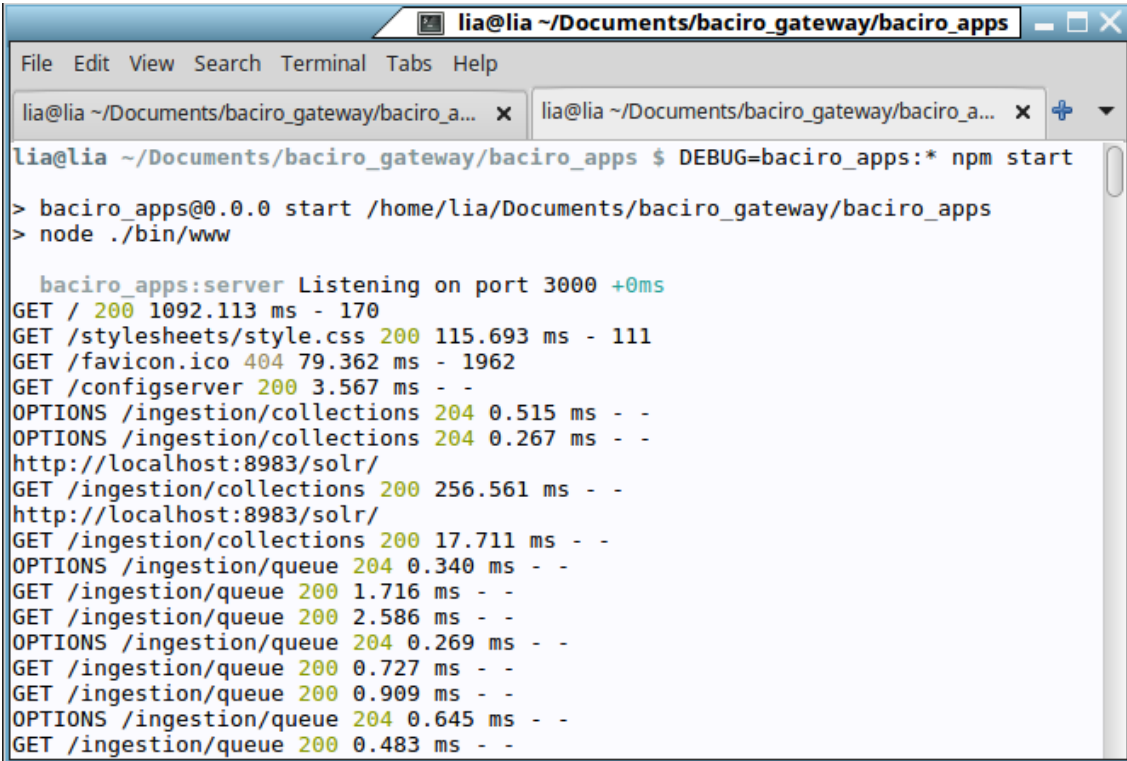
Gambar 4.15 Salah satu contoh konversi dokumen dari CSV ke JSON

Sedangkan dokumen yang sudah berformat JSON dapat langsung dimasukkan ke Solr.

4.4. Penggunaan Aplikasi

Untuk menjalankan aplikasi BaciroGateway ada beberapa langkah yang harus dijalankan. Untuk menjalankan menu Analytics submenu Ingestion, Solr harus dijalankan terlebih dahulu. Lalu server NodeJS harus dijalankan dengan command berikut.

```
DEBUG=baciro_apps:* npm start
```



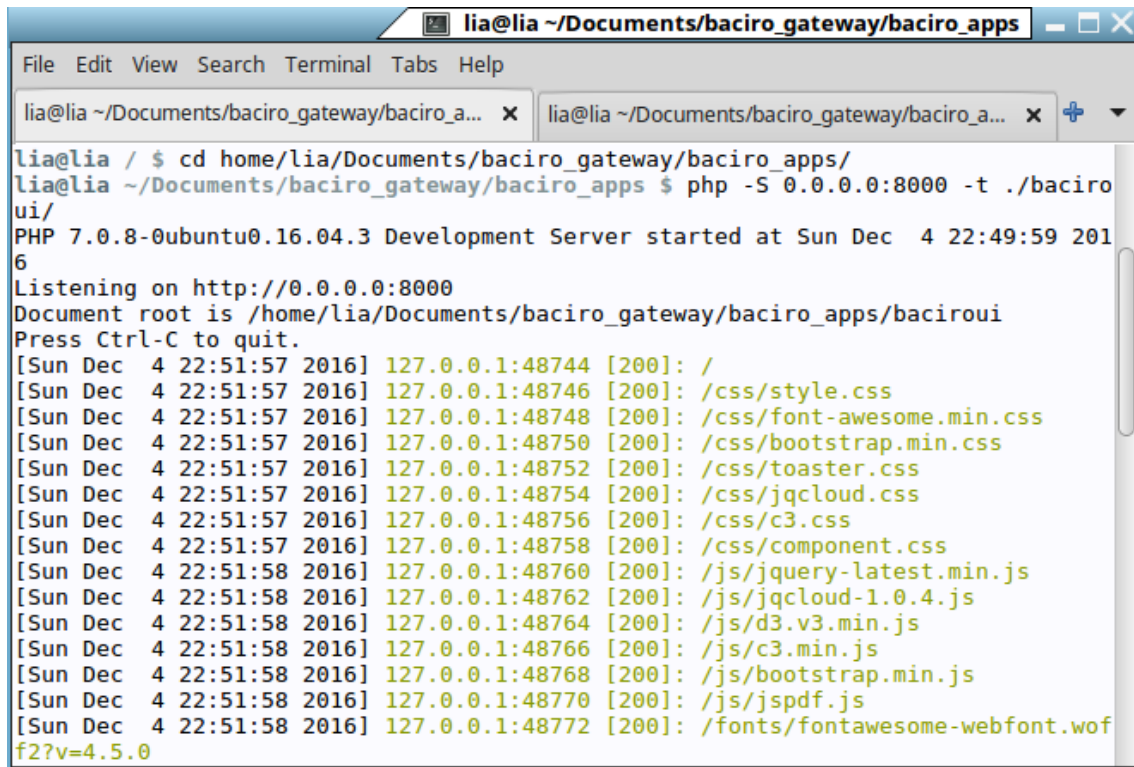
```
lia@lia ~/Documents/baciro_gateway/baciro_apps
File Edit View Search Terminal Tabs Help
lia@lia ~/Documents/baciro_gateway/baciro_a... x lia@lia ~/Documents/baciro_gateway/baciro_a... x +
lia@lia ~/Documents/baciro_gateway/baciro_apps $ DEBUG=baciro_apps:* npm start
> baciro_apps@0.0.0 start /home/lia/Documents/baciro_gateway/baciro_apps
> node ./bin/www

  baciro_apps:server Listening on port 3000 +0ms
GET / 200 1092.113 ms - 170
GET /stylesheets/style.css 200 115.693 ms - 111
GET /favicon.ico 404 79.362 ms - 1962
GET /configserver 200 3.567 ms - -
OPTIONS /ingestion/collections 204 0.515 ms - -
OPTIONS /ingestion/collections 204 0.267 ms - -
http://localhost:8983/solr/
GET /ingestion/collections 200 256.561 ms - -
http://localhost:8983/solr/
GET /ingestion/collections 200 17.711 ms - -
OPTIONS /ingestion/queue 204 0.340 ms - -
GET /ingestion/queue 200 1.716 ms - -
GET /ingestion/queue 200 2.586 ms - -
OPTIONS /ingestion/queue 204 0.269 ms - -
GET /ingestion/queue 200 0.727 ms - -
GET /ingestion/queue 200 0.909 ms - -
OPTIONS /ingestion/queue 204 0.645 ms - -
GET /ingestion/queue 200 0.483 ms - -
```

Gambar 4.16 Running server aplikasi melalui terminal

Sedangkan untuk aplikasi BaciroUI dijalankan dengan command berikut.

```
php -S 0.0.0.0:8000 -t ./baciroui
```



```
lia@lia ~/Documents/baciro_gateway/baciro_apps
File Edit View Search Terminal Tabs Help
lia@lia ~/Documents/baciro_gateway/baciro_a... x lia@lia ~/Documents/baciro_gateway/baciro_a... x + -
lia@lia / $ cd home/lia/Documents/baciro_gateway/baciro_apps/
lia@lia ~/Documents/baciro_gateway/baciro_apps $ php -S 0.0.0.0:8000 -t ./baciroui/
PHP 7.0.8-0ubuntu0.16.04.3 Development Server started at Sun Dec 4 22:49:59 2016
Listening on http://0.0.0.0:8000
Document root is /home/lia/Documents/baciro_gateway/baciro_apps/baciroui
Press Ctrl-C to quit.
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48744 [200]: /
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48746 [200]: /css/style.css
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48748 [200]: /css/font-awesome.min.css
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48750 [200]: /css/bootstrap.min.css
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48752 [200]: /css/toaster.css
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48754 [200]: /css/jqcloud.css
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48756 [200]: /css/c3.css
[Sun Dec 4 22:51:57 2016] 127.0.0.1:48758 [200]: /css/component.css
[Sun Dec 4 22:51:58 2016] 127.0.0.1:48760 [200]: /js/jquery-latest.min.js
[Sun Dec 4 22:51:58 2016] 127.0.0.1:48762 [200]: /js/jqcloud-1.0.4.js
[Sun Dec 4 22:51:58 2016] 127.0.0.1:48764 [200]: /js/d3.v3.min.js
[Sun Dec 4 22:51:58 2016] 127.0.0.1:48766 [200]: /js/c3.min.js
[Sun Dec 4 22:51:58 2016] 127.0.0.1:48768 [200]: /js/bootstrap.min.js
[Sun Dec 4 22:51:58 2016] 127.0.0.1:48770 [200]: /js/jspdf.js
[Sun Dec 4 22:51:58 2016] 127.0.0.1:48772 [200]: /fonts/fontawesome-webfont.woff
f2?v=4.5.0
```

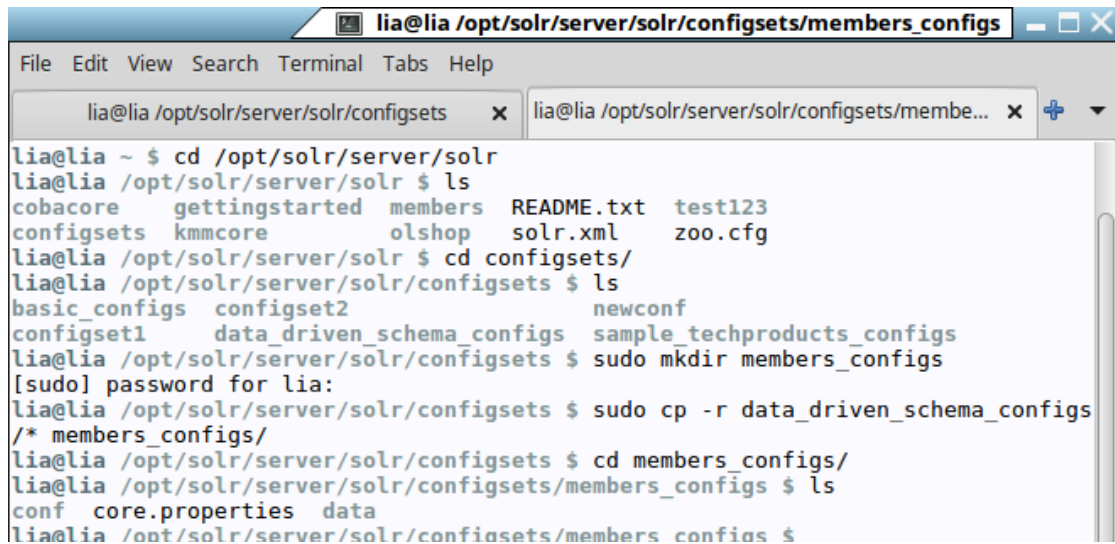
Gambar 4.17 Running php

Setelah aplikasi BaciroGateway berhasil dijalankan, maka langkah selanjutnya adalah mengakses aplikasi melalui browser. Aplikasi berjalan pada localhost dengan port 8000.

Untuk membuat core menu yang harus digunakan adalah Create Core. Secara default, Solr telah menyediakan file *basic_configs* sebagai config-nya. Namun pengguna juga dapat membuat configsets sendiri melalui terminal terlebih dahulu pada direktori `$SOLR_HOME/configsets`.

Misalkan user ingin membuat core bernama *members* dengan konfigurasi sendiri yang bernama *members_configs*. Maka terlebih dahulu

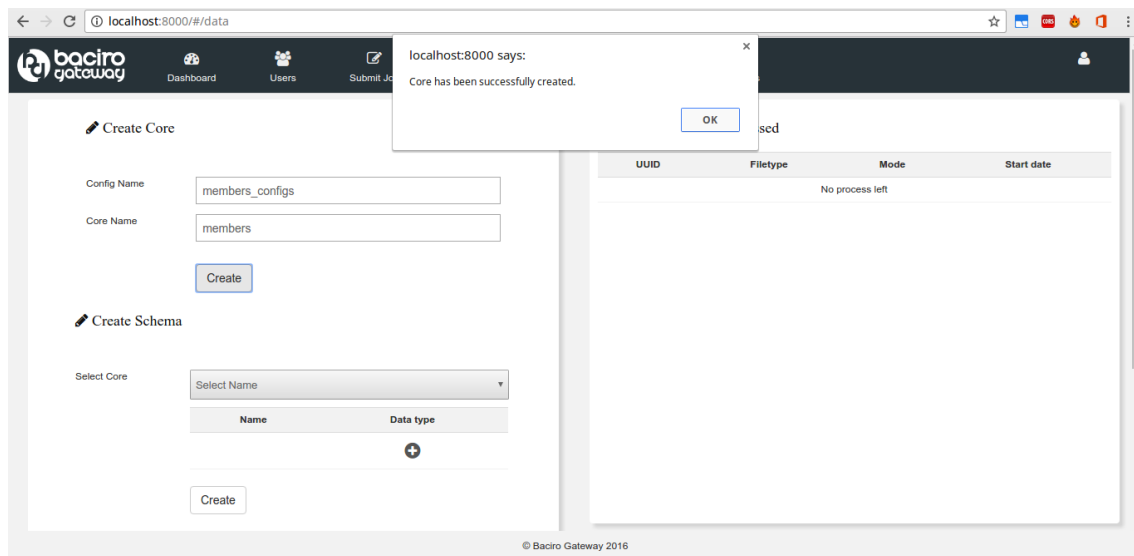
harus dibuat file konfigurasi dengan nama tersebut pada direktori `$SOLR_HOME/configsets` dengan menggunakan template folder dari Solr yaitu `data_driven_schema_configs`.



```
lia@lia /opt/solr/server/solr/configsets/members_configs
File Edit View Search Terminal Tabs Help
lia@lia /opt/solr/server/solr/configsets x lia@lia /opt/solr/server/solr/configsets/membe... x
lia@lia ~ $ cd /opt/solr/server/solr
lia@lia /opt/solr/server/solr $ ls
cobacore  gettingstarted  members  README.txt  test123
configsets  kmmcore        olshop   solr.xml    zoo.cfg
lia@lia /opt/solr/server/solr $ cd configsets/
lia@lia /opt/solr/server/solr/configsets $ ls
basic_configs  configset2  newconf
configset1     data_driven_schema_configs  sample_techproducts_configs
lia@lia /opt/solr/server/solr/configsets $ sudo mkdir members_configs
[sudo] password for lia:
lia@lia /opt/solr/server/solr/configsets $ sudo cp -r data_driven_schema_configs
/* members_configs/
lia@lia /opt/solr/server/solr/configsets $ cd members_configs/
lia@lia /opt/solr/server/solr/configsets/members_configs $ ls
conf  core.properties  data
lia@lia /opt/solr/server/solr/configsets/members_configs $
```

Gambar 4.18 Pembuatan file configsets

Seperti pada gambar di atas, langkah pertama yaitu masuk ke direktori `$SOLR_HOME/configsets` dan membuat direktori baru bernama `members_configs`. Lalu, isi direktori `data_driven_schema_configs` disalin ke direktori yang baru dibuat tadi. Setelah proses ini berhasil dilakukan, maka user dapat membuat core melalui aplikasi BaciroGateway sebagai berikut.



Gambar 4.19 Notifikasi core berhasil dibuat

Gambar di atas menunjukkan bahwa core *members* berhasil dibuat di Solr menggunakan file konfigurasi yang baru dibuat yaitu *members_configs*. Setelah core berhasil dibuat, langkah selanjutnya adalah membuat skema untuk core tersebut. Membuat skema dilakukan dengan memilih nama core dan mengisi form field pada bagian Create Schema.

Misalkan data yang akan dimasukkan oleh user adalah data yang memiliki field berupa age, name, email, dan gender.

localhost:8000/#/data

Create Schema

Select Core: members

Name	Data type
age	int
name	string
email	string
gender	string

Field gender successfully created

Field name successfully created

Field email successfully created

Field age successfully created

Mode: Offline

© Baciro

Gambar 4.20 Notifikasi field pada skema berhasil dibuat

Jika field skema berhasil dibuat maka akan muncul notifikasi yang menandakan proses berhasil.

Saat ini core members telah berhasil dibuat dan telah memiliki skemanya sendiri, namun core ini masih kosong karena belum ada data yang dimasukkan. Pada Admin Interface Solr dengan memanggil query melalui HTTP API dapat dilihat bahwa core masih kosong.



```
{
  "responseHeader": {
    "status": 0,
    "QTime": 0,
    "params": {
      "q": "*:*",
      "indent": "true",
      "wt": "json",
      "_": "1481201298785"
    }
  },
  "response": {
    "numFound": 0,
    "start": 0,
    "docs": []
  }
}
```

Gambar 4.21 Query result pada Solr

Oleh karena itu, core dapat diinput dengan data agar dapat diindeks oleh Solr. Memasukkan data ke core dapat dilakukan melalui Submit Data Form. Misalkan user ingin memasukkan data dari penyimpanan lokal yaitu file bernama generated-members.json dengan skema yang telah didefinisikan tadi.

Mode: Offline

File: Choose File generated-members.json

Name	Type	Size (KB)	base64
generated-members.json	application/json	2.35	Wwog

File Type: json

Show/hide details

Name	Data type
name	string
age	int
gender	string
email	string

+

Gambar 4.22 Pengisian Submit Data Form

ollections

members ▼

[Show/hide details](#)

Name	Type	Data Type
age	int	age ▼
email	string	email ▼
gender	string	gender ▼
id	string	▼
name	string	name ▼

Submit Reset

© Baciro


Gambar 4.23 Pengisian Data Form bagian 2


Collections

members

Show/hide details

Name	Type
age	int
email	string
gender	string
id	string
name	string

 Your data is being processed

 Please wait for message response

Submit

Reset

Gambar 4.24 Notifikasi Data Form telah diproses

Data yang berhasil dimasukkan dapat dicek melalui Solr Admin Interface dengan memanggil request query ke Solr untuk menampilkan seluruh data yang ada pada core *members* sebagai berikut.

```

http://localhost:8983/solr/members/select?q=%3A*&wt=json&indent=true

{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": ":*:*",
      "indent": "true",
      "wt": "json",
      "_": "1481201621397"
    }
  },
  "response": {
    "numFound": 20,
    "start": 0,
    "docs": [
      {
        "age": 49,
        "email": "barneswright@radiantix.com",
        "gender": "male",
        "name": "Barnes Wright",
        "id": "c8a04974-152d-4b7b-a7b6-2abe4b183b85",
        "_version_": 1553152452898848800
      },
      {
        "age": 18,
        "email": "lauraknowles@radiantix.com",
        "gender": "female",
        "name": "Laura Knowles",
        "id": "ee04c1f5-ce15-44c0-8692-194440579529",
        "_version_": 1553152452915626000
      }
    ]
  }
}

```

Gambar 4.25 Query result pada Solr yang telah menampilkan data yang diinput sebelumnya

BAB 5

PENUTUP

5.1. Kesimpulan

Berdasarkan kegiatan magang mahasiswa yang telah dilaksanakan oleh penulis di kantor Solusi247 cabang Yogyakarta sejak tanggal 18 Juli hingga 31 Agustus 2016 yang lalu, penulis menyimpulkan bahwa Apache Solr dapat diutilisasikan sebagai salah satu *tool* big data yang berfungsi untuk menyimpan dan mengindeks data dengan mudah dan efisien. Operasi dan query ke Solr dapat dilakukan dengan mengakses Solr HTTP API.

5.1. Saran

Berdasarkan pengalaman dan pengamatan penulis, untuk keperluan pengembangan berikutnya, penulis memberikan saran sebagai berikut.

1. Apache Solr telah menyediakan banyak API untuk berbagai macam platform dan bahasa pemrograman. Penggunaan API sebaiknya disesuaikan dengan kebutuhan environment.
2. Apache Solr menyediakan banyak fitur dan utilitas yang akan memberikan banyak keuntungan apabila dapat digunakan secara maksimal. Oleh karena itu, perlu pembelajaran dokumentasi Solr dan pemahaman aplikasi yang baik dan tekun agar pengembangan aplikasi yang memanfaatkan Solr dapat dilakukan dengan optimal.

DAFTAR PUSTAKA

Apache Solr Reference Guide

<https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide>

Grainger, T., Potter, T., & Seeley, Y. (2014). *Solr in action*. Manning.

Hortonworks: Apache Flume <http://hortonworks.com/apache/flume/>

Keim, D., Qu, H., & Ma, K. L. (2013). Big-data visualization. *IEEE Computer Graphics and Applications*, 33(4), 20-21.

Sagioglu, S., & Sinanc, D. (2013, May). Big data: A review. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on* (pp. 42-47). IEEE.

Ward, J. S., & Barker, A. (2013). Undefined by data: a survey of big data definitions. *arXiv preprint arXiv:1309.5821*.

Yonik: Solr 5.5 <http://yonik.com/solr-5-5/>